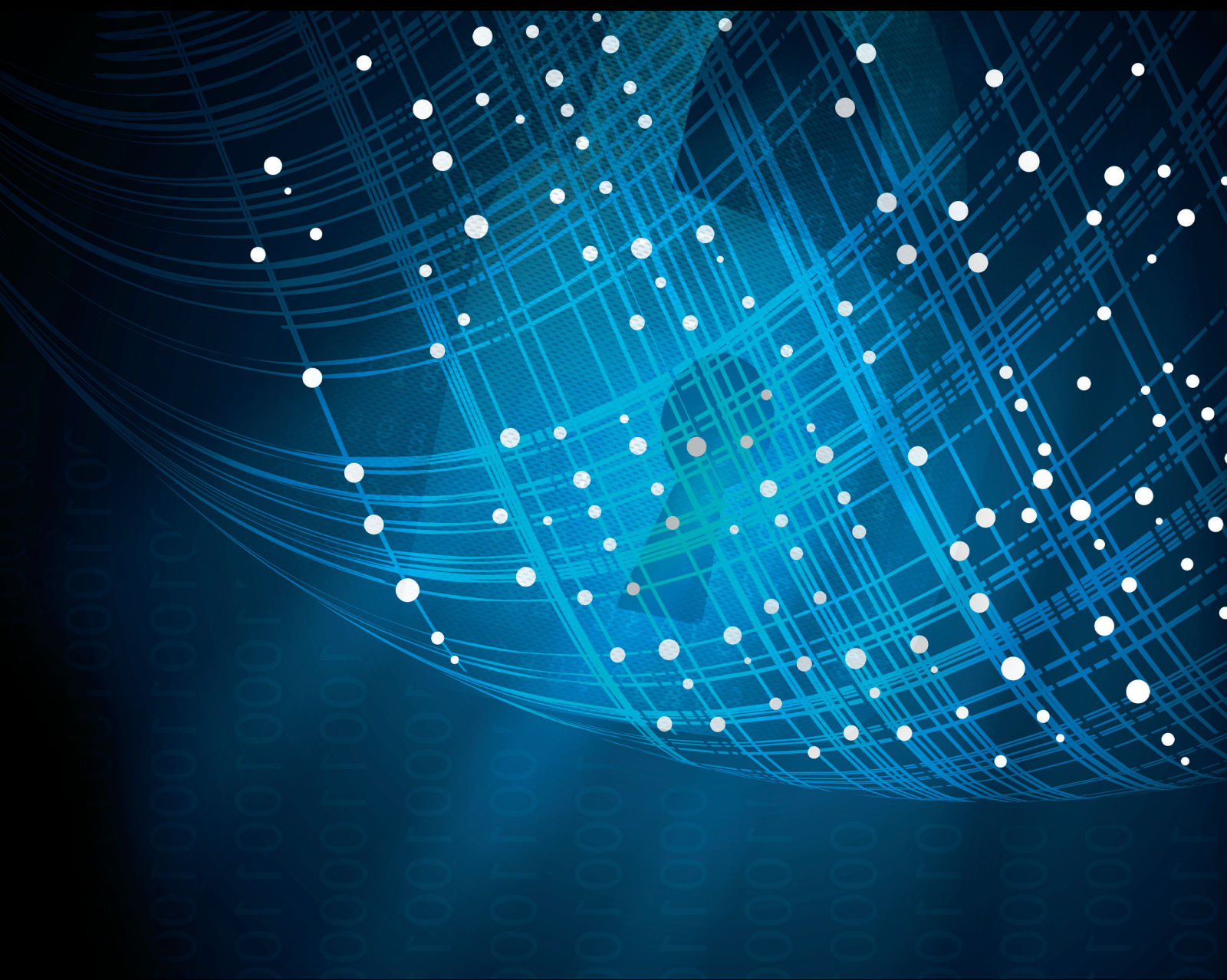
Security and Communication Networks

Emerging and Unconventional: New Attacks and Innovative Detection Techniques

Lead Guest Editor: Luca Caviglione
Guest Editors: Wojciech Mazurczyk, Steffen Wendzel, and Sebastian Zander



Emerging and Unconventional: New Attacks and Innovative Detection Techniques

Security and Communication Networks

Emerging and Unconventional: New Attacks and Innovative Detection Techniques

Lead Guest Editor: Luca Caviglione
Guest Editors: Wojciech Mazurczyk, Steffen Wendzel, and Sebastian Zander

Editorial Board

Contents

Hindawi
Security and Communication Networks
Volume 2018, Article ID 9672523, 1 page
https://doi.org/10.1155/2018/9672523

Editorial

Emerging and Unconventional: New Attacks and Innovative Detection Techniques

Luca Caviglione,[1] **Wojciech Mazurczyk**,[2] **Steffen Wendzel**,[3] **and Sebastian Zander**[4]

[1] *National Research Council of Italy, Genoa, Italy*
[2] *Warsaw University of Technology, Warsaw, Poland*
[3] *Worms University of Applied Sciences, Worms, Germany*
[4] *Murdoch University, Perth, WA, Australia*

Correspondence should be addressed to Luca Caviglione; luca.caviglione@ge.issia.cnr.it

Received 21 March 2018; Accepted 22 March 2018; Published 26 April 2018

Nowadays, security must face new and challenging scenarios, for instance, those exploiting cloud and fog computing, the Internet of Things (IoT), or complex frameworks for orchestrating botnets. Therefore, new attacks and innovative countermeasures should be investigated and this special issue focuses on how advancements provided by information and communication technologies influence modern cyberinfrastructures.

We received several high-quality submissions containing novel original research results. After a thorough review process, we accepted six articles that can be grouped into three different areas, each one offering insights on emerging and unconventional threats and detection techniques.

The first area deals with information hiding and covert channels, which are important aspects as many modern threats exploit a variety of methods to increase their stealthiness for remaining unnoticed for long periods or for limiting the efficiency of digital forensics techniques and detection tools. In this perspective, the article entitled "Leveraging KVM Events to Detect Cache-Based Side Channel Attacks in a Virtualization Environment" focuses on securing a virtualization environment by introducing a novel approach to detect covert communication attempts. Besides, the article entitled "Detecting Web-Based Botnets Using Bot Communication Traffic Features" introduces two metrics for the detection of command and control servers orchestrating botnets by means of HTTP commands and Webpages.

Detection is the second area. Novel forms of detection are mandatory to counteract sophisticated malware or to perform traffic analysis in emerging and complex scenarios. In this case, the article entitled "Leverage Website Favicon to Detect Phishing Websites" proposes a way to exploit favicon to reveal the identity of a Website and mitigate phishing attacks. Moreover, the article "Network Intrusion Detection through Stacking Dilated Convolutional Autoencoders" discusses how to approach the problem of automatically and efficiently extracting features from large amounts of unlabeled raw network traffic data using deep learning approaches.

The last area covered by this special issue deals with IoT and modern, interconnected, and smart systems. The article entitled "Remotely Exploiting AT Command Attacks on ZigBee Networks" addresses an IoT scenario and showcases how to remotely exploit AT commands to attack sensors. Lastly, the article entitled "Predictive Abuse Detection for a PLC Smart Lighting Network Based on Automatically Created Models of Exponential Smoothing" investigates statistical models to detect attacks targeting smart lighting infrastructures.

Summing up, we think that this special issue will improve the understanding of how modern communication and computing frameworks can be exploited and how they can be secured.

Acknowledgments

We would like to thank all the reviewers for reviewing the articles submitted to the special issue.

Luca Caviglione
Wojciech Mazurczyk
Steffen Wendzel
Sebastian Zander

Hindawi
Security and Communication Networks
Volume 2018, Article ID 7251750, 11 pages
https://doi.org/10.1155/2018/7251750

Research Article

Leverage Website Favicon to Detect Phishing Websites

Kang Leng Chiew, Jeffrey Soon-Fatt Choo, San Nah Sze, and Kelvin S. C. Yong

Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia

Correspondence should be addressed to Kang Leng Chiew; klchiew@unimas.my

Received 23 October 2017; Revised 10 January 2018; Accepted 30 January 2018; Published 6 March 2018

Academic Editor: Luca Caviglione

Phishing attack is a cybercrime that can lead to severe financial losses for Internet users and entrepreneurs. Typically, phishers are fond of using fuzzy techniques during the creation of a website. They confuse the victim by imitating the appearance and content of a legitimate website. In addition, many websites are vulnerable to phishing attacks, including financial institutions, social networks, e-commerce, and airline websites. This paper is an extension of our previous work that leverages the favicon with Google image search to reveal the identity of a website. Our identity retrieval technique involves an effective mathematical model that can be used to assist in retrieving the right identity from the many entries of the search results. In this paper, we introduced an enhanced version of the favicon-based phishing attack detection with the introduction of the Domain Name Amplification feature and incorporation of addition features. Additional features are very useful when the website being examined does not have a favicon. We have collected a total of 5,000 phishing websites from PhishTank and 5,000 legitimate websites from Alexa to verify the effectiveness of the proposed method. From the experimental results, we achieved a 96.93% true positive rate with only a 4.13% false positive rate.

1. Introduction

Phishing attacks can be defined as an act of deceiving victims via e-mail or a website to gain their trust to disclose their personal and financial information. With the advancement of information technology, many business agencies (e.g., banks, tourism, hotels, and airlines) can incorporate e-commerce, electronic payments, and social networking technologies into their businesses to increase sales. But this creates opportunities for phishers to gain illegal profits by disguising a wide range of services offered by financial institutions, social networking, and e-commerce websites. The Antiphishing Working Group (APWG) reported a total of 128,378 unique phishing websites detected in the second quarter of 2014 phishing activity trends report [1]. The report showed evidence that phishing activities are on the rise, which revealed that the existing antiphishing solutions were unable to resist phishing attacks efficiently.

The most common way to create a phishing website is through content replication of popular websites such as PayPal, eBay, Facebook, and Twitter. Phishing websites can be produced quickly and require little effort. This is because the phisher can simply clone the website with some modifications in the input tag to collect personal information. Furthermore, this process can be shortened by using a phishing kit [2] available on the black market. Inadvertently, advances in information technology also help phishers to develop high-profile phishing techniques to avoid phishing detectors. Figure 1 shows an example of a phishing website masquerading as PayPal. There are two flaws identified in the address bar (as shown by the red line box in Figure 1):

(i) The domain name is completely different from the genuine PayPal website.

(ii) It obfuscates the URL with HTTPS as part of the URL.

Although there are many solutions proposed to detect phishing websites, these solutions have some shortcomings. First, existing textual-based antiphishing solutions depend on the textual content of a webpage to classify the legitimacy of a website. Therefore, these solutions are incompetent to classify image-based phishing websites. A phisher can replace the textual contents with images to evade phishing detectors. Second, some phishers create phishing websites that are visually similar (e.g., webpage layout) to the legitimate website to phish potential victims. They preserve iconic images

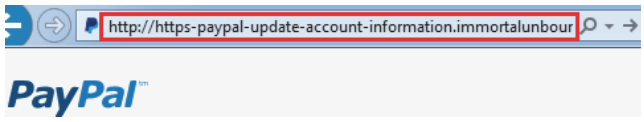


Figure 1: Example of a phishing website.

from legitimate websites to convince victims that the current webpage is benign. As a result, this type of phishing website may be incorrectly classified as legitimate by image-based antiphishing solutions that are based on similarity measurement. Third, most of the existing antiphishing solutions are unable to reveal the identity of targeted legitimate websites. Instead, they only notify the matching attributes of phishing. This can become a serious threat to Internet users if existing antiphishing solutions cannot identify the identity of new phishing website.

Our proposed method is called Phishdentity. It is driven by Chiew et al. [3], which is to find the identity of a website using the Google image search engine. In this work, we propose using the website favicon. The favicon is chosen because it represents the brand of the website. In addition, the favicon will not be affected by dynamic content (e.g., advertisement) displayed on the webpages. To determine the identity, we use the Google image search engine to return information about the favicon. The Google image search engine is chosen for this work because it allows the image (e.g., the favicon) to be used as a search query to find information. It also has the highest number of legitimate websites indexed [4]. Furthermore, using the Google image search engine can eliminate the need to maintain a database that may affect the effectiveness to detect a phishing website.

This paper is an extension of our previous paper work [5]. The previous work relies only on the usage of the favicon to identify the identity of a website. From the identity of the website, the legitimacy of the website-in-query can then be verified. However, if the website does not have a favicon, such approach will fail. In this paper, we proposed an enhanced version of the previous work with the introduction of Domain Name Amplification that further improves the accuracy of the detection and incorporation of five additional features. In particular, the additional features will be used to improve the performance in classifying websites without the presence of favicon. With the introduction of these new features, we present a more complete solution for the detection of the wide variation of phishing websites and not only the websites with the presence of a favicon as in the previous work. Furthermore, we have added a series of detailed experiments to analyze the proposed method.

The contributions of this paper are fourfold. First, we exploit the favicon extracted from the website and utilize the Google search by image engine to discover potential phishing attempts. Unlike current antiphishing solutions, our proposed method eliminates the need to perform intensive analysis on either text-based or image-based content. This has improved the detection speed. Second, the usage of mathematical equations has enabled the retrieval of the true identity from many entries of the search results. Third, we have proposed additional features to overcome the missing favicon issue. Fourth, our proposed method does not require a database of images or any presaved information from legitimate websites. Thus, it reduces the risk of having high false detection due to an outdated database.

The paper is structured as follows. The next section discusses related work. Section 3 presents the details of our approach. Section 4 describes the experiments conducted to verify the effectiveness of our solution. Finally, Section 5 concludes the paper.

2. Related Work

Many organizations, whether for-profit or nonprofit organizations, have joined forces to fight against phishing attacks. They collect and study the characteristics of phishing websites through different channels (e.g., PhishTank [6] and APWG) in order to develop effective solutions that can prevent Internet users from visiting phishing websites. In addition, these organizations also use delivery methods to educate the public about phishing websites. The delivery method involves an approach that uses posters, educational programs, campaigns, games, and so forth to convey information about phishing attacks. However, these efforts are not effective since phishing incidents are still increasing, as reported in [1, 7–9]. Therefore, it is necessary to understand the pros and cons of existing antiphishing solutions in order to develop a new solution that can overcome their limitations.

A list-based approach is a type of antiphishing solution that assesses the legitimacy of a website based on a presaved list. Normally, the list will be used as an extension of the web browser. These list-based approaches can achieve very high speeds in classification because they only compare the URL with the list. This list can be divided into whitelist and blacklist. The whitelist is a list of legitimate websites that are trusted by Internet users. Internet users can add or update the legitimate websites in the list. It is very unlikely for a whitelist to contain a phishing URL. However, Internet users can inadvertently whitelist a phishing website if they cannot distinguish between legitimate websites and phishing websites. Conversely, a blacklist contains a list of malicious websites. It is maintained and updated by the provider (e.g., Google developers). The blacklist can be very effective against phishing websites if the phishing URLs are already in the list. But the effectiveness is very much dependent on the update provided by the developer [10]. In addition, the study by Sheng et al. in [11] has shown that the blacklist is less effective against zero-hour phishing.

The image-based approach is another type of antiphishing solution that is based on the analysis of website images. Most of the time, this approach will store information of the website in a database. The information includes the visual layout and the captured images. Then, the information from the query website is compared with the database to determine the level of similarity. Image-based approaches have received considerable attention because of their ability to overcome the limitations imposed by the text-based antiphishing approaches [12, 13]. This approach includes utilizing the image processing tools (e.g., ImgSeek [14] and OCR technology [15]) and image

processing techniques (e.g., SIFT [16]). This approach is very effective against phishing websites that target legitimate websites whose information is already in the database. However, to be effective, this approach requires a comprehensive database. In other words, it may cause false alarms to legitimate websites that have not been registered in the database. The effectiveness of this approach also depends on the quality of the image. For example, OCR may extract incorrect data if the image is blurry. In addition, antiphishing solutions based on visual layout will fail if the phishing website contains dynamic content such as advertisements.

Another type of antiphishing solution is to utilize the search engine. This type of antiphishing solution will leverage the power of a search engine and perform further analysis based on the returned search results to determine the legitimacy of a website. Usually, this solution uses popular search engines like Google, Yahoo, and Bing. The results returned by these search engines are very sensitive to the search query. Each entry in the search results is usually listed in order of importance related to the search query. There are many studies that use search engines to check the legitimacy of a website. For example, Naga Venkata Sunil and Sardana [17] proposed a method that uses website ranking derived from the PageRank [18] to examine the legitimacy of the website. Huh and Kim [19] also proposed a similar method that uses website ranking to distinguish legitimate websites from phishing websites. Instead of using PageRank to check the ranking of a website, they compare the rates of growth in the ranking for new legitimate websites and new phishing websites. Similar methods can be found in [3, 4, 20]. This solution is effective against phishing websites because it is very rare for the search engines to include phishing entries in the search results. However, new legitimate websites may suffer because these websites usually do not have a high ranking in the search engine index.

3. Proposed Methodology

Typically, phishers like to imitate a legitimate website when developing phishing websites. They seldom make major changes to the content of phishing websites other than the inputs used to obtain personal credentials from potential victims. This is because major changes to the content of the website will only arouse user suspicion and increased workload. Besides, phishing websites usually have a short lifespan. The appearance of different phishing websites that are targeting the same legitimate websites will look similar to each other [14]. This appearance includes textual and graphic elements such as the favicon. Because the favicon is a representative of the website, this motivates us to use the favicon to detect phishing websites.

3.1. Website Favicon. The favicon is a shortcut icon attached to the URL that is displayed on the desktop browser's address bar or browser tab or next to the website name in a browser's bookmark list. Figure 2 shows an example of the Internet Explorer browser showing the PayPal favicon. The favicon represents the identity of a website in a 16 × 16 pixels' image

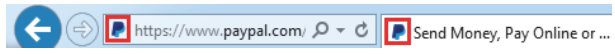


Figure 2: Example of PayPal's favicon displayed on the browser's address bar and tab.

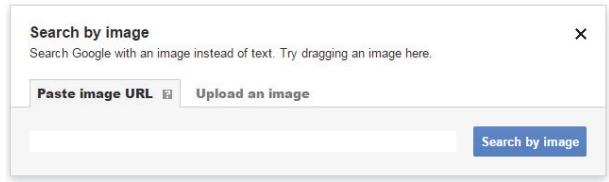


Figure 3: Example of a GSI interface.

file. It is also available in several different image sizes, such as 32 × 32, 48 × 48, or 64 × 64 pixels in size.

In order to access the favicon, we append the favicon.ico string to a website domain name. For example, given the PayPal website URL, https://www.paypal.com/, we extract the domain name (i.e., paypal.com) and append the favicon.ico to the end of the domain name, so that it becomes paypal.com/favicon.ico. The newly formed URL will be fed into the Google search by image engine to obtain information related to the favicon.

3.2. Google Search by Image. By default, the Google search engine allows text to be used as a search query to look up all sorts of images. In addition, Google also allows Internet users to search for information based on the content of an image. This mechanism of search by image content is essential for our proposed method to retrieve the correct information about an image. Figure 3 shows an example of the Google search by image (GSI) interface. Basically, there are two options to use GSI:

(i) Paste image URL: this option allows the user to directly use the image's URL found on the Internet.

(ii) Upload an image: this option allows the user to use images from local drives of computers. In addition, it also allows users to drag and drop images directly into the interface.

The Google search by image is an image query function with a Content-Based Image Retrieval (CBIR) approach and it returns a list of information specific to the query image. It extracts and analyzes the content (i.e., colours, shapes, textures, etc.) of the query image to find matching image data from the search engine database. The main difference between the search by image and normal image search is that the search by image utilizes image content to find matching image data while the normal image search uses metadata such as keywords, tags, or descriptions associated with the image to find matching image data. Figure 4 shows an example of the search results returned by GSI when the PayPal favicon is queried. We employ the *Paste image URL* option into our proposed method to feed the favicon into the GSI. To achieve this, we use a custom Application Program Interface (API) developed by Schaback [21], as shown in Figure 5. This API

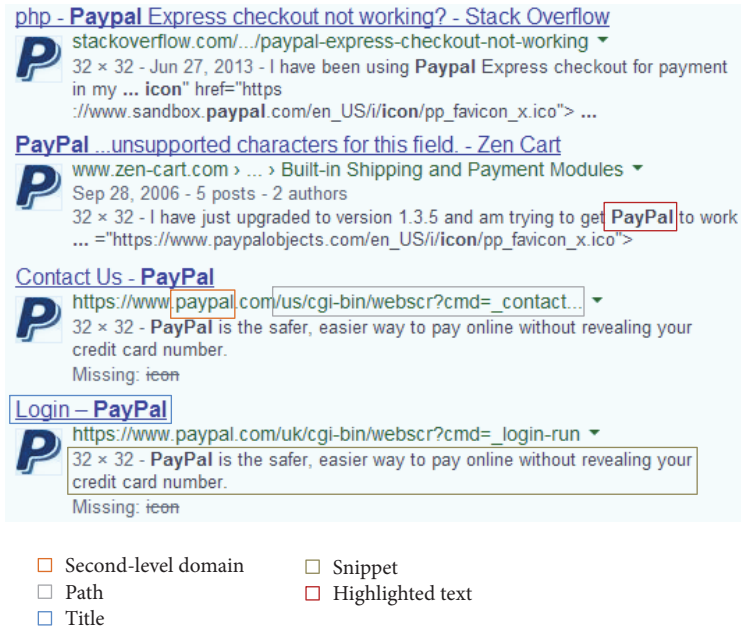


Figure 4: Example of GSI results when the PayPal favicon is queried.

```
https://www.google.com/searchbyimage?&image_url=<url>
```

Figure 5: Snippet code of GSI API.

utilizes the GSI to return a list of search entries relevant to the query image.

3.3. Proposed Features. We have a total of five features for this work. Four are extracted from the search result (refer to the red boxes outlined in Figure 4). They are as follows:

(i) Second-level domain (SLD): the SLD is the name by which a website is known. It is located directly next to the top-level domain (TLD). For example, in http://www.mydomain.com/, *mydomain* is the SLD of *.com* TLD. We propose using the SLD as part of our features because it has a high possibility of revealing the identity of a website based on the search results. Thus, the SLD is extracted from a list of entries returned by the GSI. In order to avoid confusion towards the counting, we use the term "unique term" to represent each unique SLD extracted from the entries. Therefore, the frequency of each unique term is computed from the number of occurrences of SLDs found from the search result.

(ii) Path in URL (path): this path is usually located after the top-level domain of a URL. For example, in https://www.domain.com/image/index.php, */image/index.php* refers to a unique location for a file named *index.php*. The path is used as part of our proposed features because the search results often contain terms in the path that are associated with the identity of the targeted legitimate website. While phishers can change the path of a URL in a different way, they must maintain the identity keyword in the URL in order to convince Internet users that they are visiting the correct destination. For this reason, we extract the full path of each URL from the search results returned by GSI. Then, we use the unique terms extracted in advance to find matching identities from all paths. Hence, in order to capture this property, the number of occurrences for every unique term found in the path is recorded.

(iii) Title and snippet (TNS): the title is the text that appears on top of the URL, and the snippet is the description that appears below the URL. We observed that the identity of the website does not always appear in the URL of the search entries. Instead, the identity of the website can also be found in the title or snippet of the search entries. Therefore, each unique term extracted beforehand is used to find a match in all the titles and snippets of search results. Hence, the number of occurrences for each unique term found in the titles and snippets is recorded.

(iv) Highlighted text (HLT): highlighted text is the bold text that appears in the title or snippet of an entry in the search results. The highlighted text indicates the most relevant and important keyword for the search favicon. This feature is very important and has a high tendency to reveal the true identity of the favicon. This is due to the fact that the content of a favicon must comply with the image content defined by GSI in order to have bold text appearing in the search results. Therefore, the proposed method extracts the bold text from each entry in the search results to find a match based on the unique terms extracted beforehand. The numbers of occurrence for each unique term found in all the highlighted text are recorded.

Table 1: Frequency of each unique term for each feature based on the entries in Figure 4.

Unique term	SLD	path	TNS	HLT
Stackoverflow	1	0	0	0
Zen-cart	1	0	0	0
PayPal	2	1	10	9
Total frequency count	4	1	10	9

Table 2: Weight assigned to the proposed features.

Feature	Notation	Weight
SLD	w_{sld}	20
Path	w_{path}	10
TNS	w_{tns}	30
HLT	w_{hlt}	40

To demonstrate the formation of each feature, we use the example shown in Figure 4 and show the frequency count of each unique term in Table 1.

We use the following equations to calculate the weighted frequency for each unique term across each feature (i.e., SLD, path, TNS, and HLT).

$$\begin{aligned} h_i^{\text{sld}} &= \frac{f_i^{\text{sld}} * w_{\text{sld}}}{F_{\text{sld}}}, \\ h_i^{\text{path}} &= \frac{f_i^{\text{path}} * w_{\text{path}}}{F_{\text{path}}}, \\ h_i^{\text{tns}} &= \frac{f_i^{\text{tns}} * w_{\text{tns}}}{F_{\text{tns}}}, \\ h_i^{\text{hlt}} &= \frac{f_i^{\text{hlt}} * w_{\text{hlt}}}{F_{\text{hlt}}}, \end{aligned} \tag{1}$$

h_i^{sld}, h_i^{path}, h_i^{tns}, and h_i^{hlt} refer to the weighted frequency of ith unique term for the four features. f_i^{sld}, f_i^{path}, f_i^{tns}, and f_i^{hlt} are the frequency count of ith unique term in which i is from the list of unique terms. F_{sld}, F_{path}, F_{tns}, and F_{hlt} are the total frequency count of all the unique terms under one feature. $w_{\text{sld}}, w_{\text{path}}, w_{\text{tns}}$, and w_{hlt} are the weight assigned for each feature (as shown in Table 2) and they are determined empirically. We assign the fourth feature the highest weight because the highlighted text usually has a high tendency to reveal the identity of the favicon. Based on preliminary experiments, we observed that the fourth feature has a very low frequency in the search results. It only occurs when the query favicon is truly matched with the image content stored in the Google image database. The first and third features are assigned with modest weight mainly because the frequency of identity depends on the Google search engine index. In other words, a false identity with a higher frequency can take over the real identity when the GIS cannot return enough information related to the favicon. We argue that these features are slightly lower in the level of importance compared with the fourth feature. The second feature is assigned with the lowest weight because phishers can always make changes to the path in a web address without affecting the whole website. In addition, we do not want Phishdentity to have a great effect if phishers exploit the path in a web address to avoid detection.

After obtaining the weighted frequency for each feature, we need to combine them to form the final frequency for each unique term. To do so, we use the following equation:

$$H_i = \frac{h_i^{\text{sld}} + h_i^{\text{path}} + h_i^{\text{tns}} + h_i^{\text{hlt}}}{w_{\text{sld}} + w_{\text{path}} + w_{\text{tns}} + w_{\text{hlt}}}. \tag{2}$$

3.3.1. Domain Name Amplification (DNA). A domain name is a unique name that is registered under the Domain Name System (DNS). It is used to identify an Internet resource such as a website. Typically, a legitimate website has a unique name differing from other websites. On the contrary, phishers are more likely to incorporate the name of a legitimate identity keyword into phishing URLs to confuse Internet users. Thus, this leads to the addition of the fifth feature, which is the DNA to the previous work.

Once we have computed the final frequency for all the unique terms, we need to amplify the final frequency of a unique term that corresponds to the SLD of a query website. To achieve this, we search through the list of unique terms to see if there is a match between the unique term and the SLD of the query website. If there is a match, we will increase the final frequency of the corresponding unique term by five percent.

$$H_i' = \begin{cases} 1.05 \times H_i, & \text{if } i\text{th unique term} = \text{SLD}_{\text{query}} \\ H_i, & \text{otherwise.} \end{cases} \tag{3}$$

Otherwise, we will append the SLD into the list of unique terms and count the frequency of all the terms for the three features (i.e., path, TNS, and HLT). This step is important because it can improve the detection performance. The rationale is that the GSI may not always include the entry for new and unpopular legitimate websites in the search results if they are not in the search index. Instead, the identity can be obtained from the other entries in the search results if the SLD of the query website is present in the list of unique terms. This is because the other entries might contain identity keywords in the path, TNS, or HLT. Next, we use the following equation to obtain a unique term with the highest final frequency:

$$\text{UT}_{\max} = \arg\max_i \left\{H_i'\right\}, \quad \text{where } i = 1, 2, 3, \ldots, n. \tag{4}$$

The unique term with the highest final frequency is deemed as the identity of the query website. If the identity of a query website does not match its SLD, we will assign a value of 1. Otherwise, we will assign a value of zero as below:

$$S_1 = \begin{cases} 1, & \text{if } \text{UT}_{\max} \neq \text{SLD}_{\text{query}} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where $\text{SLD}_{\text{query}}$ is the SLD of a query website.

3.4. Additional Features. We are aware that there may be some websites that do not have a favicon. Phishdentity will become suboptimal if the favicon is missing from the website. Therefore, we have incorporated additional features, which are based on the URL, to compensate for the missing favicon. While phishers can conceal malicious content on the website, they cannot hide the URL or the IP address. There are many studies conducted to detect phishing websites based on the URL [19, 20, 22]. These studies produce fairly good results in the classification. For this reason, we have adopted five additional features based on the URL to the proposed method for classifying websites. To achieve this, we will extract the URL from a query website. Then, we apply feature extraction on the URL to obtain the required features. The proposed additional features are as follows.:

(i) *Suspicious URL*. This feature examines the URL for at-sign (@) or dash (-) symbol. If the at-sign symbol is present in the URL, it forces the string to the left to be removed while the string to the right is considered to be the actual URL. Thus, the user will be redirected to access the URL that is located to the right of the at-sign symbol. We note that some of the latest browsers (i.e., Google Chrome) still experience this problem when the at-sign symbol is used in the URL. Another reason is that there are many Internet users still using legacy browsers and this makes them vulnerable to this type of phishing attack. Phishers use this technique to trick Internet users who rarely check the website URL when browsing the Internet. Likewise, the dash is also often used in phishing URLs. Phishers imitate legitimate domain names by inserting dashes into the URL to make an unsuspecting user believes it is the legitimate domain name. For example, the phishing domain, http://www.pay-pal.com/ is imitating PayPal domain name, https://www.paypal.com/. However, the use of dashes in a domain name is rarely seen on a legitimate website. This technique can easily deceive users who do not understand the syntax of the URL and cannot tell the difference in domain names. Thus, we will look for these symbols to identify suspicious URLs. In order to do that, we will tokenize the URL based on two delimited symbols (i.e., dot and slash). Next, we will find the at-sign and dash symbols by going through each token. If there is a matching symbol in the token, then we assign one for this feature. Otherwise, we assign zero.

(ii) *Dots in Domain*. During data collection, we note that it is very unlikely for a legitimate website to have more than five dots in the URL domain while most phishing websites have five or more dots in the URL domain. We reason that phishers use such tricks to obfuscate Internet users from perceiving the actual phishing URL. This feature is also mentioned in [4, 17]. Hence, we have adopted this feature in our proposed method to classify websites. In order to do that, we will count the number of dots that are present in a URL domain. We assign one for this feature if the domain has five or more dots. Otherwise, we assign zero for this feature.

(iii) *Age of Domain*. This feature examines the age of domain with the WHOIS service. Based on the experiments conducted, we observed that many phishing websites have a very short lifespan. Typically, they last from a few hours to a few days before disappearing from the Internet. CANTINA [4] proposed a similar feature to check the age of a website domain, but it is different than ours. Instead of using 12 months as the threshold to determine the legitimacy of a website, we proposed using 30 days to evaluate the query website. This is because there are a lot of new legitimate websites whose lifespan is less than 12 months. It is undeniable that there are some phishing websites that last longer than a week. However, the longest life expectancy ever recorded for phishing websites was 31 days according to the report published in APWG [23]. In addition, the report also showed that most of the phishing websites only have an average lifespan of a week. We assign one to this feature if the age of the query website is equal to or less than 30 days. Otherwise, we assign zero to this feature.

(iv) *IP Address*. The IP address is the numerical number separated by periods given by the computer to communicate with other devices via the Internet. During the data collection phase, we observed that there are some phishing websites using IP addresses in the URL (domain and path). However, we did not find any IP addresses on legitimate websites. It is very rare for legitimate websites to use an IP address as a website address for public access. This is because the IP address has no meaning apart from being an Internet resource identifier. Using an IP address is a cheap way to create a website because it can be done by using a personal computer as a web server. Therefore, phishers do not need to register a website address with any domain name registrar. For this reason, we extract the URL from the query website and look for the existence of an IP address. If the URL contains an IP address, we assign one to this feature. On the contrary, if the URL does not contain an IP address, this feature is assigned zero.

(v) *Web of Trust (WOT)*. WOT [24] is a website that displays the reputation of another website based on the feedback received from Internet users and information from third-party sources such as PhishTank and TRUSTe [25]. The Web of Trust has an API that can be used to inspect a website for its legitimacy. Figure 6 shows the WOT API snippet code used to retrieve the reputation of a query website. The *value* in the code is where we insert the query website, and the *api_key* is where we insert the API registration key to activate the API. Once the API has generated a reputation value for the query website, we compare the value based on the scale used by the WOT to evaluate the website. The reputation value used by the WOT is shown in Table 3 where a value of 80 and

http://api.mywot.com/version/interface?hosts=value&callback=process&key=api_key

Figure 6: Snippet code of the WOT API.

Table 3: Reputation rating of WOT.

Description	Very poor	Poor	Unsatisfactory	Good	Excellent
Reputation value	≥0	≥20	≥40	≥60	≥80

higher indicates that the website receives very good feedback from Internet users while a value of 19 and below indicates the website could endanger Internet users. To this end, this feature is assigned one if the reputation is rated less than 20. On the other hand, if the reputation of the query website is rated 20 and above, then this feature is assigned zero.

We use the following equation to formulate the calculation of additional features:

$$S_2 = \sum \left(w_j * h_j \right), \tag{6}$$

where S_2 is the score for additional features of a query website. h_j is the value obtained from jth additional feature. w_j refers to the weight assigned for each feature and it is determined empirically as shown in Table 4. The WOT feature is assigned with the highest weight mainly because it can display a website ranking. The ranking order can change based on votes received from the public and from the information obtained from third parties. The age of domain feature is assigned as the second highest weight because all website owners must register their websites with a hosting provider to obtain a meaningful domain name. Thus, phishers cannot easily fake the age of the website. We give a higher weight to the WOT than age of domain because it uses active information like the users' feedback and third-party listings to validate the legitimacy of the website. Suspicious URLs, dots in the domain, and the IP address are assigned the same weight. Mainly, they are the local features of the URL, and the data is not verified by any third parties. Therefore, we propose that the weight distribution of suspicious URLs, dots in domain, and IP address are a little lower than the WOT and age of domain.

3.5. Final Integrated Phishing Detection Scheme. To determine the legitimacy of a website, we use (7) to calculate the score. More precisely, we use scores derived from (5) and (6) as input to this equation. The score produced by this equation will be the final score for the website.

$$\text{FinalScore} = S_1 C_1 + S_2 C_2. \tag{7}$$

S_1 is the score obtained from (5) and S_2 is the score obtained from (6). C_1 is the weight given to S_1, while C_2 is the weight given to S_2. C_1 and C_2 are the optimum weights obtained from the experiments conducted in Section 4.2. The *FinalScore* is used to determine the legitimacy of the query website. If the *FinalScore* exceeds the threshold τ, then the query website is classified as phishing. Otherwise, the query website is classified as legitimate. Similarly, we have dedicated Section 4.2 to experiment with the different variants of this threshold. This experiment will provide the optimum threshold.

4. Experiments and Evaluation

We have implemented a prototype of Phishdentity. It is written in C# language using the Microsoft Visual Studio 2010 Professional Edition. To verify the effectiveness of Phishdentity, we have collected 5,000 phishing websites and 5,000 legitimate websites, from December 20 to December 26, 2017. The phishing websites are obtained from the PhishTank archive. In particular, we are only interested in collecting phishing websites that have not been verified and are still online during this period. For legitimate websites, we chose Alexa [26] to collect our data. We refined Alexa to return only the top 500 sites on the web by category. Categories include but are not limited to arts, business, health, recreation, shopping, and sports. During data collection, we found that there is a total of 165 websites (3.30%) from Alexa that did not have the presence of favicon. We also found that there is a total of 81 websites (1.62%) from PhishTank that did not have the presence of favicon. Figure 7 summarizes the distribution of the dataset.

We conducted three experiments to verify the effectiveness of the proposed method. Experiment 1 is designed to evaluate the detection performance of the proposed method without the additional features. Experiment 2 is conducted to assess the integration of the proposed method with additional features to classify websites with a missing favicon. Experiment 3 is conducted to benchmark the performance of the proposed method with other phishing detection methods. In experiments 1 and 2, we will use a total of 7,000 websites (3,500 legitimate and 3,500 phishing) for training and optimum parameters setup. In experiment 3, we use a total of 3,000 websites (1,500 legitimate and 1,500 phishing) to validate the proposed method. It is noteworthy to mention that the 3000 websites used in experiment 3 are a new set of samples that have not been used in experiments 1 and 2. The experimental results are displayed using the following measurement metric:

(i) True positive (TP): a phishing website is correctly classified as phishing.

(ii) True negative (TN): a legitimate website is correctly classified as legitimate.

(iii) False positive (FP): a legitimate website is misclassified as phishing.

(iv) False negative (FN): a phishing website is misclassified as legitimate.

(v) F-score (F_1): this shows the overall classification accuracy of the model and the equation was composed of $F_1 = 2\text{TP}/(2\text{TP} + \text{FP} + \text{FN})$.

Table 4: Weight for the additional features.

Feature, h	Suspicious URL	Dots in domain	Age of domain	IP address	WOT
Weight, w	0.1	0.1	0.3	0.1	0.4

Table 5: Phishdentity assessment results.

Test bed	TP (%)	TN (%)	FP (%)	FN (%)	F_1
(1) Phishdentity (DNA not included)	94.57	89.31	10.69	5.43	0.92147
(2) Phishdentity (DNA included)	97.00	95.54	4.46	3.00	0.96297

4.1. Experiment One: Evaluation of Phishdentity. Experiment one is designed to evaluate the performance of Phishdentity to classify the websites based on search results returned by Google. We used the default parameters specified in the GSI API to return the search results. We have designed two test beds for this experiment: (1) The first test bed is designed to classify the websites using SLD, path, TNS, and HLT. (2) We integrated a DNA feature in addition to all the features used in the first test bed to form the second test bed to amplify a unique term that corresponds to the SLD of query website.

Table 5 illustrates that the second test bed has shown improvement after integrating DNA into the test. More specifically, the use of DNA has increased the effectiveness of Phishdentity to find the identity of a website. In other words, the second test bed is able to locate the correct identity even if the search results may contain many false identities. This improvement is noticeable in classifying legitimate websites where there is a reduction of 6.23% in false positive numbers. However, the number of false positives is still considered high. There are a few factors that contribute to the high number of false positives. First, there are a total of 91 out of 3500 legitimate websites without the presence of a favicon. Therefore, this contributes a total of 2.6% to the false positive numbers. Second, the API can return incorrect information when feeding with a wrong version of a favicon. This happens when the query website has a newer version of the favicon than the one stored in the Google image database. Although this issue has contributed some false positives, we foresee that the new favicon will be soon available in the Google image database. It is also in accordance with the frequent update of the Google web crawler to the database. Third, there are 13 websites that have restricted the access to their favicons when we perform the test. On the other hand, the second test bed has shown some improvement in detecting phishing websites. It reduces 2.43% from 5.43% to 3.00% of false negatives. The decline in the number of false negatives is due to Google's ability to filter malicious entries from the search results. But it will take some time for Google to determine the legitimacy of new malicious entries because most phishing incidents usually occur in the early stage of the attack. We adopted the second test bed setup for subsequent experiments.

4.2. Experiment Two: Evaluation of Phishdentity with the Absence of Favicons. This experiment is designed to examine the performance of Phishdentity with the absence of a favicon. This experiment is important to our proposed method because it reveals the potential of Phishdentity in classifying websites. For this reason, we designed three series of experiments for this section. The first series will be used to determine the optimal weight of C_1 and C_2 in (7) using a threshold of 50 as the baseline. We recorded the number of correctly classified websites each time C_1 is increased by one and C_2 is reduced by one. We stopped the process when it produced the highest number of correctly classified websites. Once C_1 and C_2 have been determined, we used them to conduct the second series of experiments. The second series is designed to determine the optimal threshold, τ. To do so, we set τ to zero initially. Then, we observed the number of correctly classified websites each time τ is increased by one. We halted this process when it produced the highest number of correctly classified websites. The third series is used to demonstrate the performance difference with and without the integration of additional features into Phishdentity. Two test beds are designed for the third series. The first test bed was used to demonstrate the performance of Phishdentity without additional features. The second test bed integrated Phishdentity with additional features using τ obtained from the second series of experiments. Experiment two produced the final version of the Phishdentity and it was used in the next experiment.

Table 6: To find the optimum weight for C_1 and C_2 when τ is set to 50 as the baseline.

C_1	C_2	TP (%)	TN (%)	FP (%)	FN (%)	F_1
0	100	92.34	91.71	8.29	7.66	0.92050
20	80	94.77	93.20	6.80	5.23	0.94032
40	*60*	*97.14*	*95.89*	*4.11*	*2.86*	*0.96537*
60	40	97.0	95.54	4.46	3.0	0.96297
80	20	97.0	95.54	4.46	3.0	0.96297
100	0	97.0	95.54	4.46	3.0	0.96297

We observed that our Phishdentity with additional features achieved the lowest error rates (FP and FN) in classification when we allocated a weight of 40 to C_1 and a weight of 60 to C_2, as shown in Table 6. In addition, we were able to further reduce the error rate in the classification when τ is set to 60 using optimal weights obtained from the first series of experiments, as shown in Table 7. Based on our preliminary observations, the additional features can improve the weaknesses of a missing website favicon, as shown by the second test bed in Table 8. This reduced the false positive by 0.57%. However, the solution is subjected to a higher false negative where it increases the number of false negatives from 3.00%

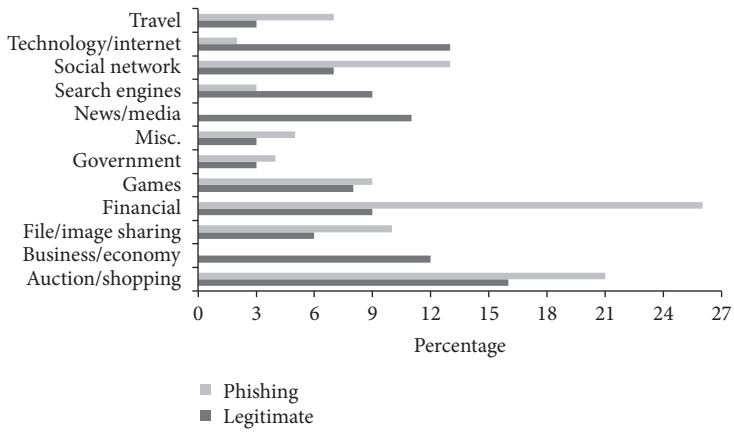


Figure 7: Categorization of legitimate and phishing websites.

Table 7: To find the optimum threshold, τ, for Phishdentity with additional features.

Optimal threshold, τ	TP (%)	TN (%)	FP (%)	FN (%)	F_1
0	99.80	90.23	9.77	0.20	0.95243
20	98.86	91.66	8.34	1.14	0.95425
40	97.34	94.40	5.60	2.66	0.95930
50	97.14	95.89	4.11	2.86	0.96537
60	*96.97*	*96.11*	*3.89*	*3.03*	*0.96555*
80	65.17	98.46	1.54	34.83	0.78184
100	23.34	100	0.00	76.66	0.37847

Table 8: Performance comparison for Phishdentity with and without additional features.

Test bed of Phishdentity	TP (%)	TN (%)	FP (%)	FN (%)	F_1
(1) Without additional features	97.00	95.54	4.46	3.00	0.96297
(2) With additional features	96.97	96.11	3.89	3.03	0.96555

Table 9: The score for each feature in Phishdentity.

Feature	Score
Favicon	40
Suspicious URL	6
Dots in domain	6
Age of domain	18
IP address	6
WOT	24
Total	100

to 3.03%. We argue that an increase of 0.03% in false negative is acceptable, given that the number of false positives was reduced by 0.57%. This showed that a solution based on the URL can be used to improve the detection results.

With the weight allocated to C_1 and C_2 as 40 and 60, respectively, the score for each feature is given in Table 9. Note that the weight of 60 for C_2 is a combination of scores from the suspicious URL, dots in domain, age of domain, IP address, and WOT feature. These scores are obtained and converted from the distribution shown in Table 4.

4.3. Experiment Three: Evaluation of Final Phishdentity. In order to show the effectiveness of Phishdentity in classifying websites, we perform benchmarking with other antiphishing methods. In this experiment, we chose CANTINA [4] and GoldPhish [15] for the benchmarking. CANTINA is a content-based antiphishing approach that utilizes the term frequency-inverse document frequency (TF-IDF) technique. It extracts five of the most important keywords from the website and feeds them to the Google search engine. Then, to determine the legitimacy of a website, it searches for a matching domain name from the search results. On the other hand, GoldPhish is an image-based antiphishing approach that utilizes the optical character recognition (OCR) technique. First, it uses a predefined resolution to capture a screenshot of the website. Then, OCR extracts the textual information from the captured screen and feeds this to the Google search engine. Next, it searches for a matching domain name from the search results to determine the legitimacy of a website.

Based on the experimental results in Table 10, CANTINA has falsely classified 5.87% of legitimate websites as phishing. We found that the TF-IDF does not work well for some of the legitimate websites. It will produce incorrect lexical signatures for the Google search engine if the query website

Table 10: Benchmarking results for final Phishdentity, CANTINA, and GoldPhish.

Antiphishing method	TP (%)	TN (%)	FP (%)	FN (%)	F_1
(1) Final Phishdentity	96.93	95.87	4.13	3.07	0.96419
(2) CANTINA	76.20	94.13	5.87	23.80	0.83704
(3) GoldPhish	98.07	80.13	19.87	1.93	0.89997

contains very little textual information that can describe the website. This issue is also discussed by Zhang et al. [4] where they planned to investigate for alternative approaches. From the evidence produced by this experiment, CANTINA does not work well for phishing websites. It falsely classified 23.8% of phishing websites as legitimate. We found that a relatively high number of misclassified phishing websites is due to the total weight assigned to additional features of the CANTINA. In other words, the total weight of the additional features can outweigh the TF-IDF in spite of the phishing websites that have been initially detected by the TF-IDF. Our proposed technique will not suffer this weakness because the final Phishdentity utilizes the favicon as the main input to GSI.

Table 10 shows that GoldPhish performs badly in classifying legitimate websites. The number of falsely classified legitimate websites is as high as 19.87%. We argue that GoldPhish faced several limitations when using the OCR tool to extract the textual information from the screenshot. First, GoldPhish is using a fixed size window to crop the screenshot. This will potentially exclude some important content that is located outside of the cropped region. Second, we noticed that most of the websites from Alexa have advertisements on the home page. There are some advertisements that are so large that they cover half of the screenshot. This has caused the OCR to extract incorrect messages. Third, some legitimate websites use many images on the homepage. This can cause the OCR to capture an image that does not contain important messages about the website. Fourth, the OCR does not work well with some of the uncommon typefaces and size of font. This makes the OCR unable to recognize the characters correctly. Among the three methods, GoldPhish performs the best in detecting phishing websites at 98.07%. This can be attributed to the ability of the Google search engine to return little or zero information about the phishing websites.

4.4. Limitations and Discussions. We discovered that our proposed technique has three limitations to the experiments. The first limitation is that phishers can slightly change the content of a favicon so that it is still familiar from the victim's point of view, but it does not retrieve legitimate websites. The phishers can also replace a favicon similar to another legitimate website. This can happen when Google has not yet crawled all the new updates for the parent company and its subsidiaries. However, this limitation is not vital. The GSI engine would extract different contents from the altered favicon and return information not related to the targeted legitimate website. In addition, phishing domains with altered favicons are less likely to appear in search results due to their relative young age. Thus, phishing websites with altered favicons can still trigger Phishdentity detection.

The second limitation is the false classification of the new or unpopular legitimate websites. New websites may not be indexed by Google and WOT may not have the information regarding these websites. Thus, the proposed method may falsely classify these websites as phishing. However, as time progresses, eventually the new websites will be listed in the WOT database. As for the unpopular websites, most likely they will not be targeted by the phishers. We believe that the missing data will be made available in the WOT database soon. This is because the WOT has a very large community which actively updates the information about the old and newly discovered websites. Furthermore, we have proposed an additional approach that is based on the website URL for classification. This additional approach is lightweight and can be used to offset Phishdentity's inability to classify the websites that do not have the presence of a favicon. Also, this additional approach will reduce the probability of such misclassification as evidenced by low false positive in experiment 3.

It is noteworthy to mention that the additional features are not performing well for some legitimate websites. First, the WOT categorizes pornographic websites listed in the Alexa ranked as dangerous websites. While this type of website has a lot of malicious advertising that will infect a computer with a virus, it is still classified as legitimate in the final calculation. The rationale is that we cannot deny that every legitimate website is harmless (i.e., pornographic websites), but our objective here is to determine the legitimacy and not the danger of the website. Second, WOT will give a low rating for e-commerce websites that do not use a secure connection for users to log in or make digital payments. The use of a secure connection on webpages can prevent eavesdropping, but it can be costly to implement, particularly for an e-commerce website targeting local businesses only. Nevertheless, this is unlikely to cause false alarms to legitimate websites because legitimacy is not solely determined by WOT.

5. Conclusion

In this paper, we have proposed a method known as Phishdentity. It is an approach that is based on the favicon to find the identity of a website. In order to retrieve information about the favicon, we use GSI API to return a list of websites that match the favicon. We have introduced simple mathematical equations to assist in retrieving the right identity from the many entries of search results. After that, we can reveal the identity of a website based on a unique term derived from the search results. We have also integrated additional features based on the URL to overcome Phishdentity's limitations. The additional features are used to address the scenario

where a favicon is missing from the website. In addition, we have conducted several experiments using a total of 10,000 websites obtained from Alexa and PhishTank. The experiments show that Phishdentity can achieve promising and reliable results.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The funding for this project is made possible through the research grant obtained from Universiti Malaysia Sarawak under the Special FRGS 2016 Cycle [Grant no. F08/SpFRGS/1533/2017] and Postdoctoral Research Scheme.

References

[1] APWG, http://docs.apwg.org/reports/apwg_trends_report_q2_2014.pdf.

[2] M. Cova, C. Kruegel, and G. Vigna, "There Is No Free Phish: An Analysis Of Free And Live Phishing Kits," *Proceedings of the Second USENIX Workshop on Offensive Technologies*, 2008.

[3] K. L. Chiew, E. H. Chang, S. N. Sze, and W. K. Tiong, "Utilisation of website logo for phishing detection," *Computers & Security*, vol. 54, pp. 16–26, 2015.

[4] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proceedings of the 16th International Conference World Wide Web (WWW '07)*, pp. 639–648, May 2007.

[5] J. C. S. Fatt, C. K. Leng, and S. S. Nah, "Phishdentity: Leverage website favicon to offset polymorphic phishing website," in *Proceedings of the 9th International Conference on Availability, Reliability and Security, ARES 2014*, pp. 114–119, Switzerland, September 2014.

[6] PhishTank, http://www.phishtank.com/.

[7] RSA, http://www.emc.com/collateral/fraud-report/online-fraud-report-1012.pdf.

[8] APAC, "Briefing on Handling of Phishing Websites in November 2016," http://en.apac.cn/Briefing_on_Handling_of_Phishing_Websites/201701/P020170112578956574716.pdf.

[9] "Malaysian Computer Emergency Response Team," http://www.mycert.org.my/en/services/advisories/mycert/2014/main/detail/955/index.html.

[10] S. Sheng, B. Magnien, and P. Kumaraguru, "Anti-Phishing Phil: the design and evaluation of a game that teaches people not to fall for phish," in *Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS '07)*, Pittsburgh, Pa, USA, July 2007.

[11] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *Proceedings of the 6th Conference on Email and Anti-Spam, CEAS 2009*, usa, July 2009.

[12] A. Herzberg and A. Jbara, "Security and identification indicators for browsers against spoofing and phishing attacks," *ACM Transactions on Internet Technology (TOIT)*, vol. 8, no. 4, pp. 16:1–16:36, 2008.

[13] Binational Working Group, https://www.publicsafety.gc.ca/cnt/rsrcs/pblctns/archive-rprt-phshng/archive-rprt-phshng-eng.pdf.

[14] M. Hara, A. Yamada, and Y. Miyake, "Visual similarity-based phishing detection without victim site information," in *Proceedings of the IEEE Symposium on Computational Intelligence in Cyber Security (CICS '09)*, pp. 30–36, IEEE, Nashville, Tenn, USA, April 2009.

[15] M. Dunlop, S. Groat, and D. Shelly, "GoldPhish: using images for content-based phishing analysis," in *Proceedings of the 5th International Conference on Internet Monitoring and Protection (ICIMP '10)*, pp. 123–128, Barcelona, Spain, May 2010.

[16] S. Afroz and R. Greenstadt, "PhishZoo: detecting phishing websites by looking at them," in *Proceedings of the 5th Annual IEEE International Conference on Semantic Computing (ICSC '11)*, pp. 368–375, Palo Alto, Calif, USA, September 2011.

[17] A. Naga Venkata Sunil and A. Sardana, "A PageRank based detection technique for phishing web sites," in *Proceedings of the 2012 IEEE Symposium on Computers and Informatics, ISCI 2012*, pp. 58–63, Malaysia, March 2012.

[18] Open SEO Stats, http://pagerank.chromefans.org/.

[19] J. H. Huh and H. Kim, "Phishing Detection With Popular Search Engine: Simple And Effective," in *Proceedings of the in Proceedings of the 4th Canada-France MITACS conference on Foundations and Practice of Security*, pp. 194–207, 2011.

[20] R. B. Basnet and A. H. Sung, "Mining web to detect phishing URLs," in *Proceedings of the 11th IEEE International Conference on Machine Learning and Applications, ICMLA 2012*, pp. 568–573, USA, December 2012.

[21] A. Schaback, http://skyzerblogger.blogspot.tw/2013/01/google-reverse-image-search-scraping.html.

[22] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pp. 1245–1253, July 2009.

[23] APWG, http://docs.apwg.org/reports/APWG_Phishing_Attack_Report-Jul2004.pdf.

[24] "Web of Trust," https://www.mywot.com/wiki/api.

[25] "True Ultimate Standards Everywhere," https://www.truste.com.

[26] "Alexa," http://www.alexa.com/topsites.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 4216240, 18 pages
https://doi.org/10.1155/2018/4216240

Research Article

Leveraging KVM Events to Detect Cache-Based Side Channel Attacks in a Virtualization Environment

Ady Wahyudi Paundu, Doudou Fall, Daisuke Miyamoto, and Youki Kadobayashi

Laboratory for Cyber Resilience, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara 630-0192, Japan

Correspondence should be addressed to Ady Wahyudi Paundu; ady.paundu.ak9@is.naist.jp

Received 25 September 2017; Revised 13 December 2017; Accepted 23 January 2018; Published 25 February 2018

Academic Editor: Wojciech Mazurczyk

Cache-based side channel attack (CSCa) techniques in virtualization systems are becoming more advanced, while defense methods against them are still perceived as nonpractical. The most recent CSCa variant called Flush + Flush has showed that the current detection methods can be easily bypassed. Within this work, we introduce a novel monitoring approach to detect CSCa operations inside a virtualization environment. We utilize the Kernel Virtual Machine (KVM) event data in the kernel and process this data using a machine learning technique to identify any CSCa operation in the guest Virtual Machine (VM). We evaluate our approach using Receiver Operating Characteristic (ROC) diagram of multiple attack and benign operation scenarios. Our method successfully separate the CSCa datasets from the non-CSCa datasets, on both trained and nontrained data scenarios. The successful classification also include the Flush + Flush attack scenario. We are also able to explain the classification results by extracting the set of most important features that separate both classes using their Fisher scores and show that our monitoring approach can work to detect CSCa in general. Finally, we evaluate the overhead impact of our CSCa monitoring method and show that it has a negligible computation overhead on the host and the guest VM.

1. Introduction

Virtualization technology has become a common utility in the current computation world. This technology has many advantages over traditional computing systems, such as lower cost, energy saving, faster provisioning, and application isolation. This isolation property is supposed to be one of the security properties of cloud computing systems. However, within the last decade, academicians and practitioners have discovered that this isolation is not impenetrable [1–4]. One well-known technique to break this isolation feature is a cache-based side channel attack (CSCa). This attack takes advantage of a main characteristic of the virtualization technique which shares physical hardware resources among multiple guest systems to improve server utilization. CSCa is known to be able to gather information such as cryptographic keys, keystroke sequences, coresidency, and website access across multiple CPUs, CPU cores, and even VMs. To help protect security in cloud computing systems, CSCa detection methods have become important.

There are many implementations of CSCa, but they all share one basic operation, which is timing a certain operation on the shared cache of the physical CPU. Two of the well-known CSCa techniques are Prime + Probe and Flush + Reload. In a nutshell, both techniques measure the time to read a certain location in the memory. The read operation will create either cache hit or cache miss events. Both events can be enumerated easily using the Hardware Performance Counter (HPC). Current CSCa detection methods utilize this cache hit and miss irregularity to detect the CSCa. However, the latest CSCa method called Flush + Flush employs an improved technique that does not require reading any memory locations. This improvement eliminates the cache hit-miss information and makes the Flush + Flush attack stealthier than the previous attacks.

On the other hand, to aid in virtualization security, methods for monitoring guest VM activity have also been proposed in many academic papers [5–12]. In general, those proposed VM monitoring methods can be categorized into three common techniques, which are computational metric

monitoring, system-call monitoring, and Virtual Machine Introspection (VMI). However, the effectiveness of a monitoring process in a public cloud is still limited due to the additional layer (virtualization layer) between the observer and the observation object. Furthermore, requirements in a public cloud limit the access of a cloud administrator to internal information from the guest system.

The motivation of this work is to introduce a Kernel Virtual Machine (KVM) events monitoring method for CSCa detection in the virtualization environment and to give proofs that the data features collected from the monitoring can give good detection results for three major variants of CSCa (Prime + Probe, Flush + Reload, and the latest more stealthy Flush + Flush) with a negligible computation overhead. To evaluate our monitoring method, we collected KVM events data inside the host from multiple emulated scenarios of CSCa and non-CSCa operation in the guest VM. Then we applied a Support Vector Machine (SVM) machine learning technique to analyze and classify the KVM events sequences and examined its accuracy using the AUC (Area Under the Curve of Receiver Operating Characteristic) unit.

The contribution of this paper is three-fold:

(1) First, we introduced a new method to monitor guest activity within a virtualization environment using KVM events data. This monitoring technique enables us to gather data with less performance overhead, without guest VM cooperation and without system components modification which are requirements in public cloud operation.

(2) Next, we showed that the proposed KVM events sequence data can be used to differentiate between non-CSCa operation data and CSCa operation data that includes Flush + Flush, the latest stealthier CSCa.

(3) Finally, we performed several empirical evaluations to measure the performance of our detection method. With our evaluation, we are able to answer the following questions:

   (a) Can the KVM events information be used to differentiate the CSCa and non-CSCa operations? How effective is this detection method to classify the trained scenarios and the new (untrained) scenarios?

   (b) Can the results in (a) be generalized, such that using this method with other scenarios or other microarchitectures is still able to give good detection results?

   (c) What is the effect of a noisy background or mimicry attempts by the adversary on the detection and the attack results?

   (d) How big is the impact given by the monitoring operation on the host and guest VM operation?

The remainder of this paper is organized as follows. In Section 2, we define the scope of our work by presenting the threat model and assumptions. In Section 3, we present several previous related works on cache-based side channel attacks, the evolution of the attack, and prevention and detection techniques. In Section 4, we give a theoretical background by providing a brief explanation on how KVM and the cache-based side channel attack work. In Section 5, we explain how our KVM event monitoring approach works. In Section 6, we present our empirical evaluation in detail. In Section 7, we discuss some more related issues and possibilities, and, finally, in Section 8, we conclude the paper.

2. Threat Model and Assumptions

In this section, we will define the scope of our work. In this work, we study the cache-based side channel attacks (CSCa). This set of attacks is a subset of two broader attack classes, side channel attacks, and microarchitecture attacks. We narrow this down to the three most well-known attack types, Prime + Probe, Flush + Reload, and Flush + Flush attack.

We focus further on attacks inside the virtualization environment. The resource sharing characteristic of virtualization technology makes this environment highly vulnerable to CSCa attacks. Since the virtualization environment is a vast and complex environment, it would be hard to cover it in full. To focus our study, within our threat model we assume that the cloud provider, its administrator, and its infrastructures are trusted. We also further assume that the Virtual Machine Monitor (VMM) is safe. However, we assume that one or more cloud tenants are not trusted and might have bad intentions to violate the privacy of the cloud by spying on a certain person's operation, either on their own VM or on the peer's VM. Moreover, our anticipated attack environments are public virtualization environments such as those using the Infrastructure as a Service (IaaS) Cloud model, where the host has limited-to-no authority over its guest system operations.

We set our defensive effort on a detection method. We base this choice on the assumption that the attackers do not know when the victim process will be executed; therefore the attackers have to put a constant probe on the cache before gathering any data from the victim. Furthermore, common CSCa techniques require many repeated bits of data from a victim to be able to extract any useful information. Hence, a CSCa spends most of its time in a loop observing the cache. We propose a detection method for this CSCa probing phase, which can then stop the attack from actually gathering its target information.

3. Related Work

The threat of CSCa attacks, especially in the virtualization environment, and methods of defense against these attacks have been researched since the early 2000s. This section first examines some of the work on CSCa attacks and then focuses on such attacks in the virtualization environment, before looking at research on defense and prevention. Since we propose a new VM monitoring technique, in this section we also describe the previous related works on guest VM observation method. These related studies provide the background for our study.

The idea of observing the cache access time as a side channel medium to spy on the victim process has been around since the early 2000s [13, 14]. The first application of this cache-based timing attack was demonstrated by Osvik et al. in 2006 [15]. The authors introduced two methods called Evict + Time and Prime + Probe. Both methods observe the state of the CPU's memory cache to reveal memory access patterns that later can be used in a cryptanalysis process. The Flush + Reload attack was introduced by Yarom and Falkner in 2014 [16]. This method took advantage of a memory deduplication technique [17] and improved the previous CSCa methods by increasing the speed and granularity of the attack to the cache-line level using the clflush function in the microarchitecture API. The CSCa not only has been proven to work for cryptanalysis purposes, but also can be used to spy on many other daily applications, such as a javascript browser [18], user interface [19], and even a mobile application [20].

In particular in the virtualization environment, an attack on a coresident VM was demonstrated by Ristenpart et al. in 2009 by recovering the keystrokes from a coresident VM in commercial clouds [21]. In 2012, Zhang et al. showed how to recover an El-Gamal decryption key from a coresident VM [22]. Later, the same authors presented ways to use CSCa to attack a peer VM within a Platform using a Service (PaaS) cloud model [23]. İnci et al. in 2016 presented a cache attack to enable bulk key recovery in a commercial cloud [24].

Many research studies have also been conducted on defense against CSCa attacks. One defense idea is to make the attack measurement process more difficult by introducing random variables. Such random variables include random memory-cache mapping, the use of prefetches, random timers, and random cache states [25–28]. Other proposals aimed to strengthen the victim application code to make it less vulnerable to CSCa attacks. This technique can be applied at the Operating System (OS) level [29, 30] or at the application level using sanity verification frameworks [31, 32]. Other approaches prevented cache sharing by distributing the VMs to different partitions in the cache, using either hardware [29, 33] or software [34, 35]. For CSCa in the cloud, the common protection idea is to change the new VM placement policies to reduce the probability of having the attacker VM and the victim VM stay in the same physical host [36–38]. However, cloud providers might find all these approaches less attractive because they require significant modifications to the cloud infrastructure.

Contrary to the many prevention techniques for CSCa attacks, detection methods have not been as widely studied. CSCa techniques are well-known to be very noisy and therefore can be easily detected using the Hardware Performance Counter (HPC). Chiappetta et al. used the HPC data and coupled it with a neural network method to detect CSCa in real time [39]. Zhang et al. went further by implementing CSCa detection in a virtualization environment [40]. They created a handshake system that correlates the signature-based detection of the cryptographic application in the victim VM with the anomaly detection system in the attacker's VM. This method requires cooperation from the victim VM to provide signatures of their cryptographic operation. Other detection methods were presented by Payer [41] and Herath and Fogh [42]. However, the latest development in CSCa introduced a new stealthier variant called Flush + Flush [43]. Since this method does not try to read the memory, no hit and miss events will happen; thus its existence cannot be detected using the HPC. As of the writing of this paper, we have not heard yet any academic paper presented to detect such attacks.

On the aspect of guest VM monitoring, many methods have been studied. Some of the common techniques to monitor the guest VM in a nonintrusive way are computation metric observation [5, 6], system-call observation [7–9], and Virtual Machine Introspection [10–12].

(i) The computation metric observation approach analyzes metrics such as CPU utilization, memory utilization, and the volume of block device read and write operations inside the guest VM. The main assumption of this approach is that a malicious activity will likely change some considerable amount of computing resources. The shortcoming of this approach is that it is hard to map the workload data to a specific process target inside the guest VM.

(ii) System-call is a set of interfaces that enable user processes to access the services that are provided by the Operating System (OS) kernel. By observing the system-call invocations from user processes to the underlying kernel system, a security agent can try to infer whether the user processes constitute a normal operation or not. However, the addition of a virtualization layer in the system makes this observation method less effective.

(iii) Virtual Machine Introspection (VMI) was first introduced by Garfinkel and Rosenblum (2003) [44]. It works by capturing a snapshot of the memory space used by the guest VM and uses it to reconstruct an exact same picture of the situation inside the guest VM. One minor limitation of the current VMI implementations is their dependency on information from the guest OS to correctly interpret the memory snapshots. Examples of such data are the debugging symbols information file for Windows systems or memory offset information file for Linux systems. Although this requirement is easy to satisfy in a private environment, in other arrangements such as a public IaaS, this approach could be hard to implement.

In this study, we move the research on CSCa detection forward by proposing a monitoring method that can detect even the latest Flush + Flush attack. This monitoring system can also be seen as an improvement over previous guest VM monitoring methods, as it can give clearer information on VM operation without the need of the guest VM operator participation.

4. Background

This section provides a brief explanation of how KVM and cache-based side channel attacks (CSCa) operate.

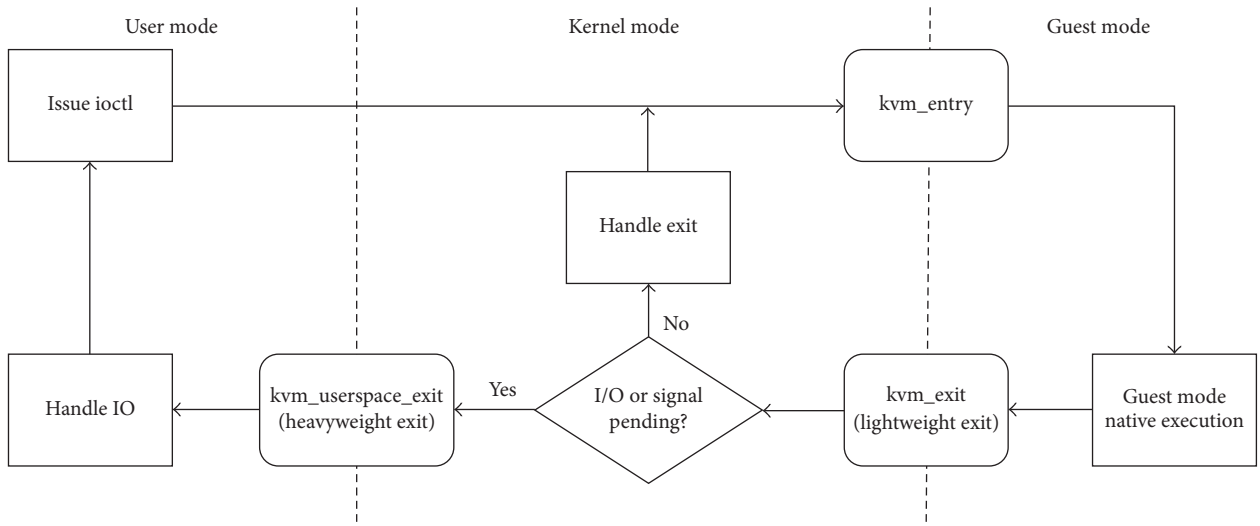


Figure 1: Guest execution loop.

4.1. Kernel Virtual Machine. A Kernel Virtual Machine (KVM) [45] is a virtualization solution that is embedded as a kernel module inside the Linux Operating System. This module enables the Linux system to act as a bare metal Virtual Machine Monitor (VMM) system (also usually being referred as type 1 virtualization). A VMM (or a Hypervisor) is a software that manages the virtualization environment operation, which includes the management of Virtual Machines (VM).

A KVM provides a set of Application Programming Interfaces (API) to utilize the hardware-assisted virtualization functions from the latest CPU architectures, such as Intel VT-x or AMD-V. Even though the hardware-assisted virtualization extensions are not standardized (both Intel and AMD processors have different instruction sets and capabilities), the basic operations are similar:

(i) The processors provide a new operating mode called guest mode, in addition to the previous two modes, the userspace mode and the kernel mode (the basic scheme of guest system operation is given in Figure 1). The guest mode enables the guest system to have all the regular privilege levels of the normal operating modes of a single Operating System. The exceptions of the privileges are several critical operating modes such as the control-sensitive IO operations (operations that have to change the state of system resources) and the handling of external interrupts, exception, and time-outs (scheduling operations are still performed by the host). These exceptions need to be performed by the host.

(ii) The operation switches between the kernel mode and guest mode, which include control registers, segment registers, and instruction pointers are performed by the hardware.

(iii) The hardware reports every exit reason (changes from the guest mode to the kernel mode) so the software can take proper action for the switch.

When it is time to run the guest system, the VMM calls ioctl() to instruct the KVM module to start up the guest system. The KVM then performs the VM entry and lets the guest system directly interact with the processor. If later the guest system is required to perform a critical instruction, it transfers the control to the kernel mode through VM exit (lightweight exit). If VMM intervention is required to execute an IO task, control is further transferred to the VMM userspace mode through KVM exit (heavyweight exit). On the completion of the VM exit handling, control is then given back to the guest mode through the VM entry process.

4.2. Cache-Based Side Channel Attack. A side channel attack is a method to gain information from a victim by eavesdropping through a nonconventional channel. An analogy would be that it is like trying to count the number of people in another room by hearing footsteps on the floor. In the case of a cache-based side channel attack, the floor is analogous to the CPU cache. An attacker measures the time to access a certain memory address to find out if those locations have been accessed (and henceforth cached) by the victim. The access information can then be translated into information about whether a certain operation has been executed or not by the victim. Since all the VMs inside a host share the same set of CPU caches, this technique can be used in the virtualization environment by an adversary to spy on its peer VM. For example, an attacker can spy on his neighbor VMs to detect if a certain user exists [21], or the attacker can spy any key-press on his peer tenant applications [19].

There are three common methods being used for cache-based side channel attacks, Prime + Probe, Flush + Reload, and Flush + Flush attack.

4.2.1. Prime + Probe. As the name suggests, this technique is comprised of two stages. In the Prime stage, the attacker evicts all the victim's data from the targeted cache set by allocating an array of memory blocks into that set. The attacker then

```
(1) procedure PrimeProbe (addr, thr)
(2)    accessed = []
(3)    access (addr)
(4)    while (true) do
(5)        wait ()
(6)        t0 = time ()
(7)        access (addr)
(8)        tx = time () − t0
(9)        if tx > thr then
(10)           accessed.append (1)
(11)       else
(12)           accessed.append (0)
(13)       end if
(14)   end while
(15)   return accessed
(16) end procedure
```

Algorithm 1: Prime + Probe.

```
(1) procedure FlushReload (addr, thr)
(2)    accessed = []
(3)    while (true) do
(4)        flush (addr)
(5)        wait ()
(6)        t0 = time ()
(7)        access (addr)
(8)        tx = time () − t0
(9)        if tx < thr then
(10)           accessed.append (1)
(11)       else
(12)           accessed.append (0)
(13)       end if
(14)   end while
(15)   return accessed
(16) end procedure
```

Algorithm 2: Flush + Reload.

waits for an interval before performing the next step. In the Probe stage, the attacker again reads the memory array and measures the access time. If the access time took longer than a certain time threshold, the attacker assumes that the cache set has been accessed by the victim during the interval. The attacker keeps repeating these Prime and Probe actions to collect the pattern of cache access by the victim which can be used later to extract information about the victim's operation. The method's operation is depicted as pseudocode in Algorithm 1.

4.2.2. Flush + Reload. This method requires that multiple identical processes using different virtual addresses be mapped into the same physical addresses. This mapping mechanism is intended to augment memory density. Two well-known implementations of this mechanism are Kernel Same-Page Merging (KSM) [46] and Transparent Page Sharing (TPS) [47].

The attacker first runs the process he wants to spy for so the process occupies the physical memory and the cache. Henceforth, anytime the victim runs the same process, the Operating System will map the process to the same location used by the attacker. The attacker then selects some specific cache line from the shared pages to be monitored. In the Flush stage, the attacker flushes his targeted cache lines. The attacker then waits for an interval before performing the Reload stage. In the Reload stage, the attacker reloads the memory blocks into the cache and measures the access time. If the access time is shorter than a predefined time threshold, it indicates a cache hit and the attacker will assume that the victim has performed the same instruction during the waiting time. As with the Prime + Probe, the attacker keeps repeating the Flush + Reload stages to collect the victim's instruction execution patterns.

Flush + Reload utilizes the assembly mnemonic *clflush*() that enables the cache flush to operate at the granularity of cache lines. To perform time measurement, this method uses the processor's hardware API, the *rdtsc*(). This Flush + Reload method has higher granularity information compared to the Prime + Probe since the Flush + Reload works at the level of cache lines. This method's operation is depicted as pseudocode in Algorithm 2.

4.2.3. Cache-Based Side Channel Attack Detection. Both Prime + Probe and Flush + Reload measure the access time of the cache. The access time of the cache is highly affected by the existence of the accessed data in the cache. The access time will be shorter if the data already exists in the cache. This is usually called a cache hit situation. In comparison, a cache miss means that the data being accessed is currently not in the cache and needs to be copied from memory, hence the longer access time. Fortunately, both events, the cache-hit and cache-miss, are observable from the processor. Modern microprocessors are equipped with a set of special purpose registers called Hardware Performance Counters (HPC). The HPCs are used to count all the CPU processing events and activities inside the computer system. Therefore, based on the HPC readings, previous CSCa detection methods can spot any CSCa attempts if they read an unusual number of cache-hits or cache-misses. As an example, a Flush + Reload probing process will create a constant high number of cache-miss that can easily be spotted.

4.2.4. Flush + Flush. The Flush + Flush method [43] is the latest variation of the Flush + Reload attack. It enhances the attack by removing the Reload stage of the spy process. Instead of measuring the time needed for the Reload stage, this method simply measures the time needed to execute the clflush(). The idea is that a flushing process will require less time if the address that needs to be flushed is not in the cache. Since there is no memory access in this attack, there is no cache miss which makes the previous detection technique almost impossible. Another advantage of Flush + Flush is that it gives higher resolution information because it works faster than the Flush + Reload attack. The Flush + Flush operation is depicted as pseudocode in Algorithm 3.

```
(1) procedure FlushFlush (addr, thr)
(2)    accessed = []
(3)    while (true) do
(4)        t0 = time ()
(5)        flush (addr)
(6)        tx = time() − t0
(7)        if tx > thr then
(8)           accessed.append (1)
(9)        else
(10)          accessed.append (0)
(11)       end if
(12)       wait ()
(13)   end while
(14)   return accessed
(15) end procedure
```

Algorithm 3: Flush + Flush.

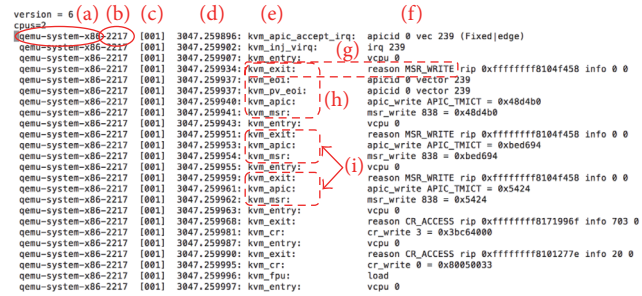


Figure 2: A snapshot example of trace-cmd output for KVM events. The preprocessing procedure to transform the text format into a vector input is explained in Section 5.2. (a) Process name. (b) Process/thread ID. (c) CPU ID. (d) Timestamps. (e) KVM event name. (f) KVM event information. (g) An example of one KVM_exit event and its exit reason. In this case we log the reason attribute. (h) An example of one KVM exit session that we used as one data (sequence) type. (i) An example of two sequences that belong to one sequence type.

We performed a simple test using *perf* tool ("*perf kvm stat -e cache-misses, cache-references -p* PID") on a VM running each of Prime + Probe, Flush + Reload, Flush + Flush, and a VM running web application. The average over 10 tries were 94%, 97%, 12%, and 18% (the percentage represents the ratio of cache misses over cache references) for Prime + Probe, Flush + Reload, Flush + Flush, and web application, respectively.

5. Monitoring System Design

This section describes our approach to detecting CSCa.

5.1. KVM Events. In computing world terms, an event can be defined as "a change of state." The same definition will be used in this paper, where KVM events are the changes of states inside the KVM module during kernel mode operation (see Figure 1). In our implementation, we introspected the KVM events that are instrumented by a standard Linux kernel tracing utility called *ftrace* [48]. Ftrace was built directly into the Linux kernel and thus brings the ability to see what is happening inside the kernel. We have three reasons to utilize this default Linux KVM instrumentation instead of creating our own user defined instrumentation. First, it allows us to target the generic hardware environment. Microarchitecture attacks depend on the type of hardware being used. To add a new probe, we would have to consider every possible hardware combination, which would increase the complexity of our study. Therefore, we decided to utilize the default set of probes that are provided by Linux and use a machine learning process to decide which events should be used for the classification process. Second, by not changing the default set of trace points, we wanted to ensure ease of implementation and make it applicable in a production environment. Finally, by using the built-in Linux function, we expected a lesser cost in computation. To ease the ftrace tracing process, we used the *trace-cmd* tool. Trace-cmd is a user-space front-end for ftrace that automates the process of accessing multiple files when directly working with ftrace itself.

The basic trace-cmd command that we used to capture KVM events from the host is *"trace-cmd record -e kvm -P xxx"* (where *xxx* is the process/thread ID of the guest KVM VCPU). This command pins data collection to one specific process/thread that represents the VCPU of the VM, thus enabling us to specify which guest VM to observe. An example of the output of this tool is given in Figure 2. It gives us the list of KVM events sequences that occurred during kernel mode operation (Section 4.1). The information gathered from this tool is the process name, process or thread ID, CPU ID, time information, KVM event name, KVM event information, and the sequence of the events.

5.2. Data Transformation. The raw data format is a text file that contains a list of KVM operation events in a chronological order. This raw data also gives additional information such as the name and parameters of each event. Figure 2 shows an example of the raw data.

We defined our data unit as the number of KVM event sequences within one monitoring time unit (e.g., 1 second). A KVM event sequence is a list of ordered KVM events that occurred between one VM exit to the next VM entry (one kernel mode session). For each KVM event, we only captured its name, with an exception for VM exit events where we captured its exit reason information. Having more features from the KVM events might increase the detection results; however, to minimize complexity, we decided to start simple and only increase the information level if it is deemed as necessary.

We formalize a data unit $X = \{Y_1, Y_2, Y_3, \ldots, Y_n\}$, where Y_i is the number of ith KVM event sequences in observation X and n is the total number of unique KVM event sequences in the dataset.

For an illustration, the observation example in Figure 2 contains five KVM event sequences:

(1) MSR_WRITE - kvm_eoi - kvm_pv_eoi - kvm_apic - kvm_msr

(2) MSR_WRITE - kvm_apic - kvm_msr

(3) MSR_WRITE - kvm_apic - kvm_msr

(4) CR_ACCESS - kvm_cr

(5) CR_ACCESS - kvm_cr - kvm_fpu

We simplified the data presentation by converting them into sequence IDs. The observation example in Figure 2 gives four sequence IDs (note that sequences which are pointed to by (i) belong to the same ID):

(i) ID1: MSR_WRITE - kvm_eoi - kvm_pv_eoi - kvm_apic - kvm_msr

(ii) ID2: MSR_WRITE - kvm_apic - kvm_msr

(iii) ID3: CR_ACCESS - kvm_cr

(iv) ID4: CR_ACCESS - kvm_cr - kvm_fpu

After having transformed all the sequences into IDs, we then counted how many times each ID showed up in an observation. Again, for illustration, having an input of Figure 2, the output would be freq(X) = freq(ID1, ID2, ID3, ID4) = (1, 2, 1, 1). We use this bag of KVM event sequence data as the input for the machine learning process to detect a CSCa attack.

6. Evaluation

6.1. Setup

6.1.1. Computation Environment. We setup one host on a Dell Poweredge 860. This machine was equipped with one Intel Xeon Dual core 3040 1.86 GHz (Conroe), 64 KB L1 (32 KB L1d + 32 KB L1i), 2 MB L2, and 8 GB system memory. Inside the host we setup eight VMs (for the scalability evaluation later). All the VMs had one virtual CPU, 512 MB memory, and 20 GB disk size. For the OS in the host and guest VM we used Ubuntu LTS 14.04 Linux (kernel version: 3.13.0-24-generic). We also setup one external computer as the web workload generator.

6.1.2. Scenarios. We collected data from multiple scenarios that represent the cache-based side channel attacks and common operations in the public cloud. We categorized the scenarios into two main classes, a positive class which contains all CSCa scenarios and a negative class which contains all non-CSCa scenarios (Table 1).

For the positive class, we collected five datasets of CSCa attacks:

(1) Three CSCa implementations from Gruss [43] (https://github.com/IAIK/flush_flush/tree/master/sc). These are Prime + Probe, Flush + Reload, and Flush + Flush attacks to eavesdrop for function calls of key-press on a Linux User Interface that utilized the libgdk library.

(2) The original Flush + Reload implementation from Yarom that spies on GnuPG's RSA implementation [16] (https://github.com/defuse/flush-reload-attacks/tree/master/flush-reload/original-from-authors).

(3) Another Flush + Reload implementation from Hornby that spies on the victim's browsing destinations [19] (https://github.com/defuse/flush-reload-attacks).

Table 1: List of all collected scenarios for evaluation.

Positive class	Negative class	
	Standard Op.	CPU Intensive Op.
Prime + Probe (Gruss)	Idle	Stress CPU
Flush + Reload (Gruss)	RUBiS 20 clients	Stress memory
Flush + Flush (Gruss)	RUBiS 200 clients	Binary tree
Flush + Reload (Yarom)	RUBiS 2000 clients	Lucas-Lehmer
Flush + Reload (Hornby)	Mail server	Urandom generator

For the negative class, we collected ten datasets of non-CSCa operation:

(1) Idle scenario: in this scenario, the VM just did nothing (with the exception of standard Linux daemons in the background). We needed to include this in our evaluation since every guest VM would go through this scenario at some time in its life-cycle.

(2) Web application scenario: we decided to use web scenario workloads under the assumption that web operations are being run the most in the public cloud system. Approximately 25% of IP addresses in Amazon's EC2 address space hosted a publicly accessible web server [21]. Web server operations also allowed us to experiment with multiple normal workload profiles for our evaluation purpose. We used RUBiS application [49] to emulate this web application scenario. RUBiS is a prototype of an auction site that was built to evaluate web application server scalability. RUBiS allowed us to easily scale the workload and generate dynamic web traffic. We used the *workload_number_of_clients_per_node* attribute to control the application workload. We collected the KVM events for three web application scenarios with different workloads, which are 20, 200, and 2000 clients.

(3) Mail server scenario: we set up a Postfix mail server system in a VM with 100 dummy users. We generated the load data from an external machine using the *postal* application (https://doc.coker.com.au/projects/postal/). For this scenario, the options that we used were as follows:

(i) Maximum size of message body: 10 Kilobytes

(ii) Number of threads that should be created for separate connection attempts: 10

(iii) Number of messages per SMTP connection: 100

(iv) Maximum number of messages per minute: 1000.

(4) CPU and memory stress test scenario: our decision to include this scenario class was intended to possibly maximize the number of false positives that our test scenarios can generate. The high intensity usage on the CPU and memory by the CSCa might not be observed in a standard VM operation (such as a web server). Therefore, we needed to introduce

several scenarios that uniformly and highly utilized the computer's CPU or memory to give a good upper false positive threshold. We collected five datasets for this scenario:

(a) Linux CPU and memory stress test: we used the standard *stress* tool from the Linux.

(b) Standard Linux random number generator: we chose the *urandom* device from Linux that use "unlimited" nonblocking random source. We performed the following: cat /dev/urandom > /dev/null. This operation is another well-known stress test for CPU.

(c) Another two mathematical operations.

(i) A python operation to solve Lucas-Lehmer prime test equation. This problem is used by many benchmark tools for stressing the CPU operation.

(ii) A binary tree operation to fully create perfect binary trees. This program stretched memory utilization by allocating, walking, and then deallocating nodes of a big binary tree. The process of allocating and deallocating memory page will mimic the cache access operation of a CSCa.

We collected data from all the scenarios exclusively. This means that there were no other operations being run at the same time we executed and collected each scenario's data. The adversary also will try to operate in an exclusive environment as much as possible to increase the CSCa effectiveness. Our evaluations on the obfuscation attempts by an attacker are given in a separate section (Section 6.4).

We evaluated our data in batches. That means, instead of evaluating them one by one in real time as the data came in, we collected the data in groups and evaluated them all together (offline). One observation data unit is a collection of all KVM events that were captured in one second. We ran each of the scenarios in turn inside the guest while collecting the KVM events inside the host. For each scenario, we collected 500 units of observation data.

For further research on this topic, our dataset can be accessed at http://iplab.naist.jp/research/CSCaD.

6.1.3. Machine Learning Setup. We applied a machine learning approach for the classification process. Microarchitectural and Operating System domain data consist of a high number of variables and parameters which are hard to observe manually. Furthermore, not all information about those variables and parameters is available to the virtualization operators. Therefore, we believe that a machine learning approach is the best option for a real world detection operation. In the evaluation phase we used the Support Vector Machine method [50] with a Radial-Based Function (RBF) to perform binary classification (CSCa or non-CSCa). We chose this supervised approach for its ease of use, while allowing us to observe in detail the differential aspect of the monitoring data between the benign scenarios and the CSCa scenarios. We utilized Scikit-learn libraries [51] for the machine learning implementation.

It is important to emphasize that our evaluation was not meant to benchmark the machine learning engine. Our chosen machine learning algorithm was selected only by its common use in classifying high dimensional data. Instead, we wanted to benchmark the ruleset, which in this case describes the characteristics and formats of the KVM event sequences from our monitoring data. Therefore, the identification of false positives and false negatives is still required even though we only used one machine learning method in our evaluation.

To avoid any confusion, we define the following quantities:

(i) True positive, CSCa data classified into the CSCa class

(ii) False positive, non-CSCa data classified into the CSCa class

(iii) True negative, non-CSCa data classified into the non-CSCa class

(iv) False negative, CSCa data classified into the non-CSCa class.

We conducted a small scale Grid Search experiment to find the best γ value for our SVM function. We found the value of 0.0003 for γ and used this value throughout this evaluation process.

For preprocessing the data, we first applied a standardization process that converted the data into standard normally distributed data: Gaussian with zero mean and unit variance. The second preprocessing step was simple removal of all the features with low variance. This second step was needed because there were a lot of sequences that appear only rarely (most of its occurrence value was 0) and can be seen as exceptions. Our filter was arbitrarily set up at 0.9, such that we removed all features that contained at least 90% similar values. The initial number of features (unique sequences of events) in the raw data was 271. After the preprocessing stage, the number of features was reduced to 69.

It is preferable to have multiple pairs of learning-test datasets to make sure that the results are not dependent on one particular random choice of learning datasets. One way to create multiple learning and test datasets is by applying a *k-fold cross validation*. In this study, as we have 500 data units for each dataset, we applied a 5-fold cross validation. In our evaluation, we calculated the average score of the 5-fold results as the final detection score.

For the detection measurement unit, we used the Area Under the Curve (AUC) value of the Receiver Operating Characteristic (ROC). The AUC value can be interpreted as the expectation that a uniformly drawn random positive sample is consistently ranked before or after a uniformly drawn random negative samples. Thus, the AUC can be seen as the separation score between two sample classes, which ranges from .50 (both classes datasets cannot be separated, fully random) to 1.00 (both classes datasets are fully separated). For the binary classification process, ROC has the advantage of being able to show the outputs from all possible positive-negative discrimination thresholds and therefore has the ability to depict relative trade-offs between

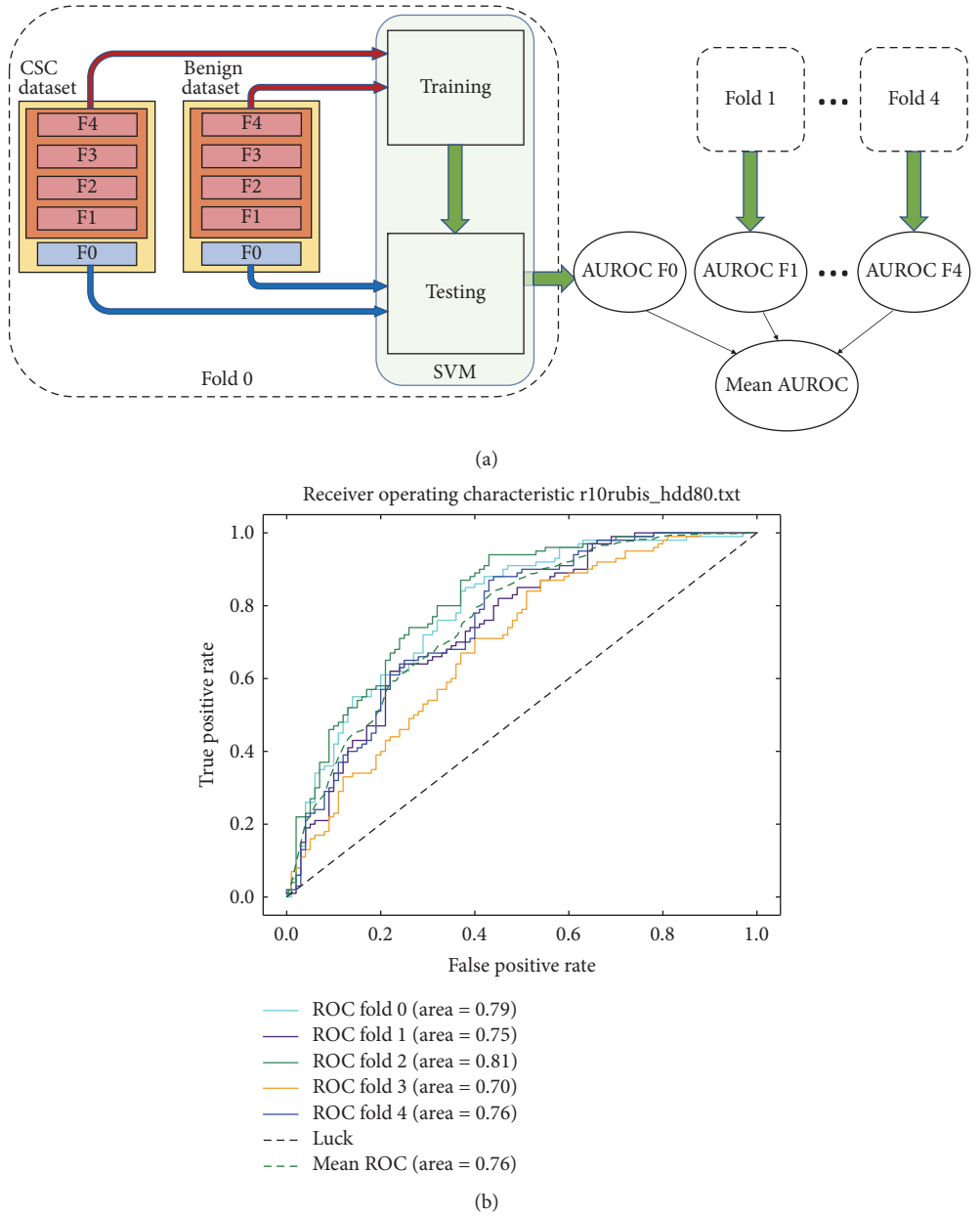


Figure 3: (a) Dataset distribution using the k-fold cross-validation technique to find an AUC value from a pair of CSCa and non-CSCa datasets for the SVM classification. (b) An example of a 5-fold cross-validation ROC graph.

the number of true positives (benefits) and false positives (costs). To create the ROC graph, instead of using the binary nonprobabilistic output of the SVM model, we used the distance of data point to the SVM model decision boundary as the input for ROC.

The scheme for dataset treatment and an illustration of its outputs are shown in Figure 3.

6.2. The Binary Class Classification for the CSCa Detection. The reason for a machine learning implementation is to use all the information one can get in the learning process. A server in the cloud is most likely performing only a small set of tasks, such as a web server, file server, or mail server. This means that having training data samples for the negative class (non-CSCa scenario) in real life is not difficult. We used this assumption to evaluate our dataset in a binary class classification approach by providing both positive class and negative class datasets for the training stage.

To evaluate this approach, we created two superset classes called the trained class and the untrained class. The trained class was the set of scenarios that were already known by the system and would be used for the training phase. The untrained class was the collection of scenarios that were not known previously by the system; therefore they were not used in the training process and would only be used in the test phase. We divided the scenarios of the positive class into

Table 2: The arrangement of scenario datasets for the binary SVM evaluation.

Trained class		Untrained class	
Positive class	Negative class	Positive class	Negative class
Prime + Probe (Gruss) and Flush + Reload (Gruss) and Flush + Reload (Yarom) scenarios are combined into one dataset	Idle and RUBiS 20 clients and RUBiS 200 clients and stress CPU and stress memory scenarios are combined into one dataset	Flush + Flush (Gruss) Flush + Reload (Hornby)	RUBiS 2000 clients Mail server Urandom generator Lucas-Lehmer Binary tree

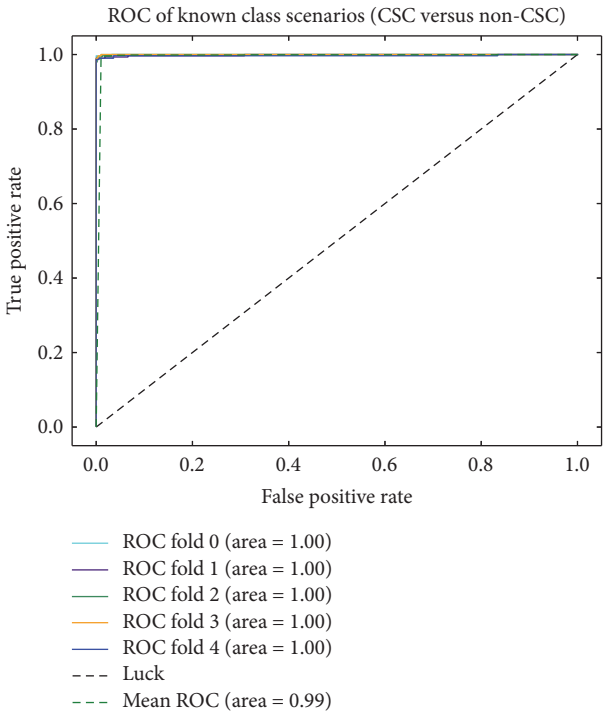


Figure 4: The classification ROC of the trained CSCa scenario test data and the trained non-CSCa scenario test data.

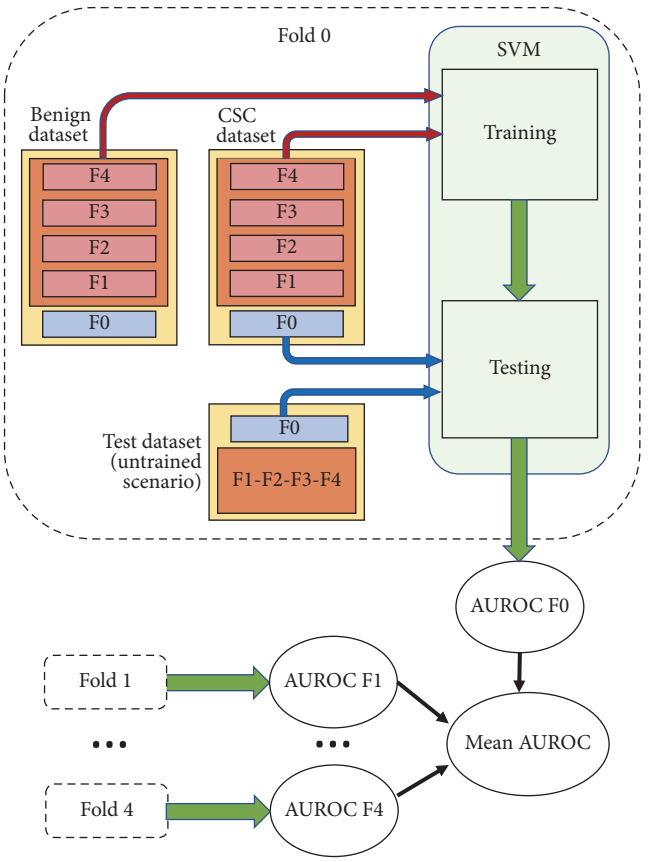


Figure 5: The evaluation scheme for each of the untrained scenario.

the trained-positive class and the untrained-positive class. We also divided the scenarios in the negative class into two, the trained-negative class and the untrained-negative class. The arrangement of all collected scenarios for use in this evaluation process is given in Table 2.

6.2.1. Test of the Trained Scenario. Our first test deals with the data that belong to the trained scenario class but not included in the training process. The aim is to see if the trained model was able to represent the trained scenario class in general. The procedure of the test is given in Figure 3(a). In this test, we do not yet use the untrained class scenarios of Table 2. The results of this test are given in Figure 4.

The results show that the detection system can successfully classify the data from all the scenarios that have been trained into CSCa and non-CSCa classes (0.99 AUC). This further shows that there are differentiable patterns of KVM event sequences between the trained CSCa scenarios and non-CSCa scenarios.

6.2.2. Test of the Untrained Scenario. In this second test, we wanted to see if the trained model was able to represent both classes, the positive class and negative class, in general. Therefore, we used the scenarios from the untrained class for the test phase. To achieve the concept of a signature-based detection system, in the test phase, the untrained class scenarios were compared against the trained-positive class dataset. The procedure of this test is given in Figure 5. The expected results should give a low AUC value (around 0.50 AUC) for the untrained-positive class scenarios and high

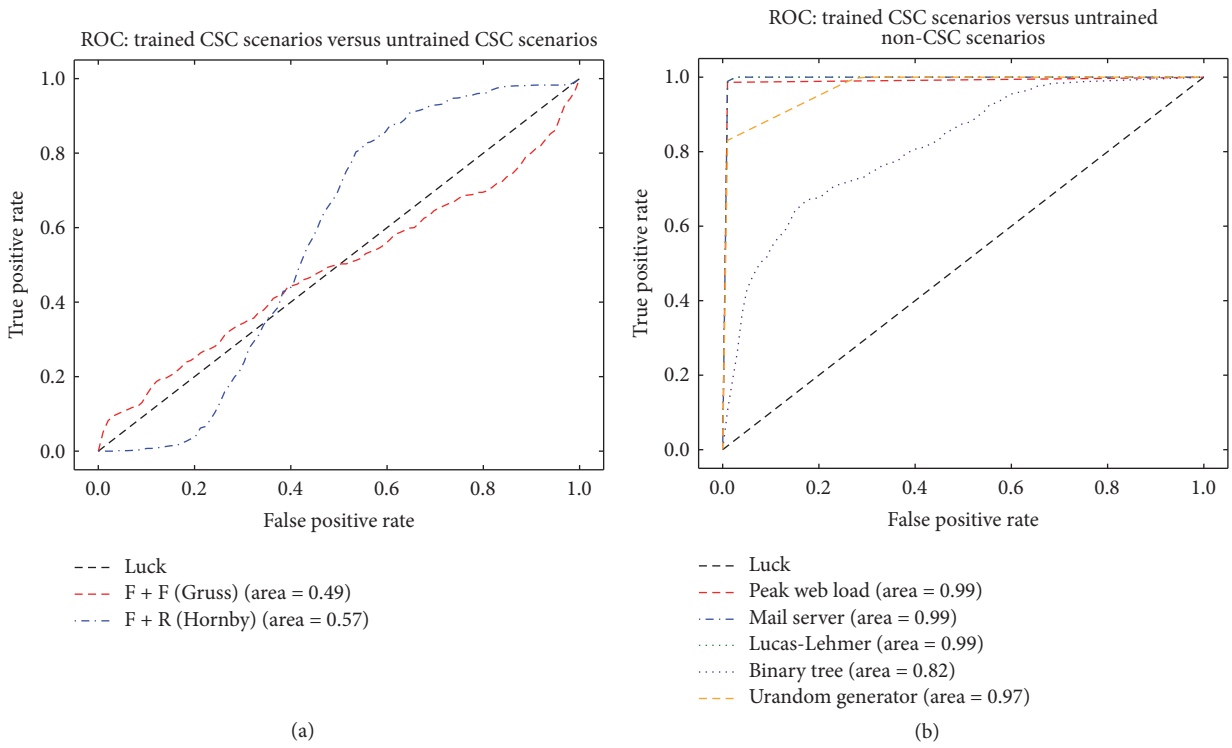


Figure 6: (a) Binary classification results for the Trained Positive Class versus each of the Untrained Positive Class (CSC) test scenarios. (b) Binary classification results for the Trained Positive Class versus each of the Untrained Negative Class (non-CSC) test scenarios.

AUC value (around 0.99 AUC) for the untrained-negative class scenarios. The actual results are given in Figure 6.

Figure 6(a) shows near to 0.50 AUC score for Flush + Flush scenario data (=0.49) and Flush + Reload Hornby scenario data (=0.57). This shows that the machine learning trained model cannot differentiate between the known CSCa scenario dataset and the unknown CSCa scenario dataset. This is the expected result as it means that the model created by the SVM training process was able to capture the common features of all CSCa and therefore will have low false negative rate detections.

On the other hand, Figure 6(b) shows near to 0.99 AUC score for the peak web workload scenario data (=0.99), mail server operation scenario data (=0.99), Lucas-Lehmer Test scenario data (=0.99), Binary Tree Operation scenario data (=0.82), and urandom generator scenario data (=0.97). This shows that the detection system was able to differentiate between the known CSCa scenarios and the unknown non-CSCa scenarios. This further means that the KVM event sequence training model was able to capture the generic differentiable features between CSCa operation and non-CSCa operation, which leads to low false positive rate detection.

In our test case, the binary tree scenario gave a smaller separation score in comparison to the other non-CSCa scenarios. We believe the reason for this score is the lesser number of arithmetic operations within the binary tree program. An in-depth explanation of the generic differentiable features between CSCa operation and non-CSCa operation is given in the next section.

6.3. Generalizing the Classification Results. In the previous sections, we showed that our monitoring system worked successfully against the scenarios that we prepared. Even though we showed that our system also works for the scenarios that were not yet trained, we still need to show that our solution can work in general for all other possible scenarios. To explain the separation between the CSCa class scenarios and the non-CSCa class scenarios, we made the effort to identify the exact KVM event sequences that separate CSCa operation and non-CSCa operation. First, we divided the non-CSCa scenarios into three different operation types: regular operations, CPU intensive operations, and Memory intensive operations. Then, we used the Fisher Score [52] approach to look for the most important features that separated each non-CSCa operation type dataset from the CSCa dataset. Fisher score comparison is a well-known method to find the optimal features, so that the distances between data points in the same class are minimized and the distances between data points of different classes are maximized. Even though the discrimination process between the SVM method and the Fisher score are not the same, we believe the results of this Fisher Score evaluation can give basic insight on the class discriminatory features. The results of this evaluation are given in Table 3.

Table 3 lists only five of the highest Fisher score features for each non-CSCa operation type dataset when compared to the CSCa dataset. Besides the Fisher scores, we also listed the median, average and standard deviation value of each feature to give a basic statistical perspective of the separation.

Table 3: Five features with the highest Fisher score for each non-CSCa scenario operation type compared to the CSCa scenarios. Note: ":" symbol represent delimiter.

Vs CSCa	KVM event sequence	Fisher score	Non-CSCa Seq. Stat.			CSCa Seq. Stat.		
			Med.	Mean	St. dev.	Med.	Mean	St. dev.
Regular operation	MSR_WRITE:kvm_apic:kvm_msr:	28.4	132	133.5	20.7	1	1.3	4.5
	HLT:kvm_eoi:kvm_pv_eoi:kvm_apic_accept_irq: kvm_inj_virq:	26.7	45	45.1	8.6	0	0	0
	HLT:kvm_inj_virq:	23.8	87	87.3	16.1	0	0	0
	MSR_WRITE:kvm_apic:kvm_apic:kvm_apic_ipi: kvm_apic_accept_irq:kvm_msr:	19.5	109	108.2	21.6	0	0.1	2.8
	HLT:kvm_eoi:kvm_pv_eoi:kvm_inj_virq:	17.2	312	305.8	67.3	0	0	0
CPU-intensive operation	EXCEPTION_NMI:kvm_fpu:	9.3	9	9.2	2.2	0	0.5	0.6
	EXTERNAL_INTERRUPT:kvm_fpu:	4.7	7	7.7	3.1	2	1.6	0.9
	CR_ACCESS:kvm_cr:kvm_fpu:	0.8	0	0.4	2.4	3	2.6	0.9
	PENDING_INTERRUPT:kvm_inj_virq:	0.4	2	3.2	6.6	167	106.5	84.3
	EXTERNAL_INTERRUPT:kvm_apic_accept_irq: kvm_inj_virq:	0.3	255	255.4	6.1	94	152.8	82.6
Memory-intensive operation	EXCEPTION_NMI:kvm_page_fault:kvm_emulate_insn:	434.9	1004	1002.1	15.7	0	0.7	18.3
	EXCEPTION_NMI:kvm_page_fault:kvm_inj_exception:	87.1	5058	4949.2	233.6	0	5	189.8
	EXCEPTION_NMI:kvm_page_fault:kvm_apic_accept_irq: kvm_inj_virq:	22.9	22	22.2	4.2	0	0	0.4
	EXCEPTION_NMI:kvm_page_fault:	19.3	5128	5630.6	983.2	0	7.9	283.8
	EXCEPTION_NMI:kvm_page_fault:kvm_emulate_insn: kvm_apic_accept_irq:kvm_inj_virq:	9.5	11	11.2	3.6	0	0	0

6.3.1. Regular Workload. In the case of a regular workload, such as web server operation and mail server operation, Table 3 shows there were a high number of VM exits on the Model Specific Register (MSR) writing operation to access the Advanced Programmable Interrupt Controller (APIC) chip in the non-CSCa scenario. This shows that, in comparison with the CSCa operation, the regular workload scenario operation produced more software and hardware interrupts. Another important VM exit shown in the table is HLT. *hlt* is an instruction to halt the CPU until it receives the next external interrupt requesting its service. The table shows that the regular scenario operations in the guest were not using the CPU intensively and therefore fired more *hlt* instructions to save the CPU power usage and heat output. The CSCa, on the other hand, were using the CPU extensively, hence the rare *hlt* calls.

However, a quick look at the entire raw data of the regular workload operation scenario is enough to easily discriminate the CSCa and non-CSCa data. There are several other features (KVM event sequences) besides the five listed in Table 3 that can be used to differentiate CSCa and non-CSCa operation. We believe this is because the regular non-CSCa operation works with diverse workload types and resources and therefore creates many different KVM event sequences, while the CSCa operations work uniformly with only a small set of suboperations (timing operation, read or write specific memory addresses and cache flushing). With knowledge of the difference in patterns of KVM event sequences between our regular operation scenario and the CSCa, we can safely extrapolate that the classification results would be the same for other regular operations within the public guest VM.

6.3.2. CPU Intensive Workload. Manual observation of the raw data shows an almost similar pattern between the CSCa scenarios and the CPU intensive non-CSCa scenarios. Table 3 for CPU-intensive operation shows that only two of the five features listed (EXCEPTION_NMI - kvm_fpu and EXTERNAL_INTERRUPT - kvm_fpu) can actually be useful for classification (the Fisher Scores are higher than 1). Both of these sequences are related to the use of the Floating Point Unit (FPU). In comparison to the CSCa attack, common CPU intensive non-CSCa operations are usually related to complex mathematical-related operations. On the other hand, CSCa does not need any complex mathematical operations and therefore can be discriminated from the CPU intensive non-CSCa operation using the sequence of FPU utilization. Examples of CPU intensive workload are cryptography operations.

6.3.3. Memory Intensive Workload. We also checked the discriminatory features between the CSCa scenarios and the memory intensive non-CSCa scenarios. All the features in Table 3 on memory intensive operation show high Fisher scores, which means that the CSCa operations can easily be separated from the non-CSCa memory intensive operation. The table shows that the memory intensive non-CSCa scenarios create a lot more page fault exceptions than the CSCa operations. Page fault exceptions may happen for

Table 4: Fisher score for the evaluation on another host with different microarchitecture.

	Sequence	Fisher score
Regular load	HLT:kvm_inj_virq:	26.74
	EXCEPTION_NMI:kvm_page_fault: kvm_inj_exception:	26.18
	MSR_WRITE:kvm_apic:kvm_msr: kvm_apic_accept_irq:	24.43
	HLT:kvm_eoi:kvm_pv_eoi:kvm_inj_virq:	20.65
	CR_ACCESS:kvm_cr:	18.79
CPU-intensive load	EXCEPTION_NMI:kvm_fpu:	9.13
	EXTERNAL_INTERRUPT:kvm_fpu:	6.76
	EXTERNAL_INTERRUPT: kvm_apic_accept_irq: kvm_inj_virq:	0.52
	EXTERNAL_INTERRUPT: kvm_apic_accept_irq:	0.51
	PENDING_INTERRUPT:kvm_inj_virq:	0.29
Memory intensive load	EXCEPTION_NMI:kvm_page_fault: kvm_inj_exception:	91.42
	EXCEPTION_NMI:kvm_page_fault: kvm_emulate_insn:	75.46
	EXCEPTION_NMI:kvm_page_fault:	43.69
	EXCEPTION_NMI:kvm_page_fault: kvm_inj_exception:kvm_apic_accept_irq: kvm_inj_exception:	15.31
	EXCEPTION_NMI:kvm_page_fault: kvm_apic_accept_irq:kvm_inj_virq:	12.84

two reasons, either because there is no translation for the memory address or because there is no access right for the specified address. In short, the high number of page fault exceptions in the memory intensive non-CSCa scenarios points to diverse memory address access, in contrast to the CSCa scenarios that focused on accessing only a small set of memory addresses.

6.3.4. Evaluation on Different Microarchitecture. As a part of the microarchitecture attack class, CSCa are characterized by the type of the CPU architecture of the physical host. To check the impact of different types of CPU architecture on the results of our monitoring method, we performed the same Fisher score evaluation as above on a host with a different microarchitecture. We set up the host on a Dell Poweredge R910 machine which is equipped with two Intel Xeon Quad-Core E7520 1.86 GHz (Nehalem), 36 MB L3 cache, and 32 GB system memory. We choose this specification as it has a different Last Level Cache (LLC) layer and different chipset architecture compared to our main evaluation setup (Section 6.1.1). Table 4 lists the five highest Fisher score features from each of the non-CSCa operation type datasets compared to the CSCa dataset on the Nehalem-based host.

Table 3 (Conroe setup) and Table 4 (Nehalem setup) show that the two highest Fisher score features that differentiate the CSCa scenario dataset and non-CSCa CPU intensive scenario dataset in the Conroe setup and Nehalem setup are the same. The similarity of the higher Fisher score set also happened in the case of the non-CSCa memory intensive dataset differentiation (4 out of 5 similar features). This shows that the operational characteristics of the non-CSCa CPU-intensive scenario and memory intensive scenario on both microarchitectures are similar and thus can be captured through KVM events observation.

On the other hand, for the regular operation datasets in the Conroe and the Nehalem setups, there were four out of five different features in the set of the five highest Fisher score features that differentiated between the CSCa scenario and the non-SCSa regular operation datasets. We believe this result could be expected since there are many features that can be used to differentiate these operations and their Fisher scores might change slightly with each evaluation, thus changing the Fisher score ranking. However, the high Fisher scores show that even though the order of ranking is different, the regular non-CSCa operation scenario and the CSCa scenario can still be easily differentiated.

6.4. On the Case of Noisy Environments and Mimicry Attempts. In this evaluation part, we examine the performance of our approach against two types of attack evasion scenarios. First is having to detect CSCa within a noisy environment. In this scenario, the adversaries try to run their CSCa attack, while, either intentionally or unintentionally, there are other benign operations running in the VM (e.g., web server transactions). Second is having to detect a modified CSCa process that tries to mimic benign operation to evade any detection process.

(1) Noisy environment: we collected another dataset of the positive class (CSCa class). This time, we ran the CSCa in the guest VM while at the same time processing a significant web application workload.

(2) Mimicry attempt: we collected several new datasets from a modified CSCa that slightly altered its behavior to obfuscate its signature characteristics.

   (a) We reduced the spy frequency by increasing the waiting interval between cache access timing. We modified the Gruss's Flush + Reload implementation by increasing the number of yield operations between each timing process (Algorithm 4). We tried 100 and 1000 yield repetition values.

   (b) We added a diversion function inside the real CSCa code. We added a read and write file operation between cache access timing operations

```
...
start = rdtsc ();
While (1){
      flush_flush (addr + offset);
      for (int i=0; i<1000, ++i)
            sched_yield ();
}
...
```

Algorithm 4: An example of a mimicry attempt by reducing the spy frequency.

```
...
start = rdtsc ();
While (1) {
      flush_flush (addr + offset);
      diversion_func ();
      sched_yield ();
}
...
```

Algorithm 5: An example of a mimicry attempt by introducing a diversion function.

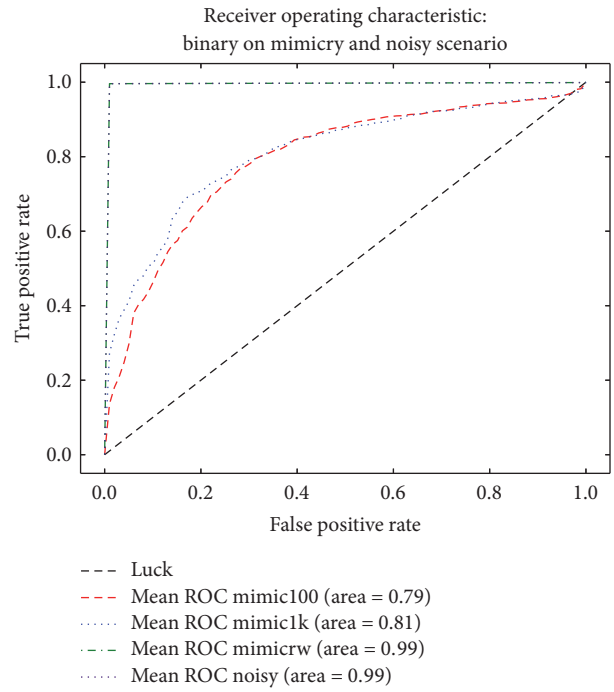


Figure 7: ROC of several mimicry attack and noisy case scenario.

in the Gruss's Flush + Reload implementation (Algorithm 5).

Using the previous SVM Binary Class Classification, the results are given in Figure 7. We can see that, in both cases, the noisy environment and mimicry attempts, the AUC values are high (0.79 and 0.81 for frequency alteration and 0.99 for both noisy environment and R/W mimicry attempt). These results point to high false negative detections. This shows that our detection method is still vulnerable to the scenarios of a noisy environment or mimicry attack. The poor results on detecting the mimicry CSCa are actually a common consequence for any indirect observation. Since we are not directly observing the target, the adversary can always create a diversion to hide their true acts.

However, looking from a different perspective, we believe that working in a noisy environment will also significantly decrease the CSCa effectiveness, making it impractical, and therefore would be avoided by the attacker. The same thing would happen in the mimicry attack. CSCa is actually a highly focused operation and requires a high level of information granularity. An attempt to obfuscate its procedure will highly reduce the granularity of the collected information. This is especially true for the Flush + Flush attack where the timing differences of clflush() hits and misses are very small. These requirements will limit the type and amount of obfuscation an adversary can use [53–56].

To evaluate the impact of a noisy environment and mimicry attempts on the CSCa output, we performed a Flush + Reload attack against an AES implementation of OpenSSL (as attempted in [43]) with four conditions: clean implementation, noisy environment, R/W mimicry attempt, and reduced probing frequency scenarios. Figure 8 shows the comparison of the cache lines visible pattern in the case of k_0 = 0xf_ between a clean Flush + Reload implementation and a frequency-reduced Flush + Reload implementation. We highlighted all cache line entries that were hit at least 99% times the number of encryptions. The number of encryptions that were required to produce less than 2% pattern errors are given in Table 5.

Table 5 shows that a noisy environment, R/W mimicry attempt, or reduced probing frequency will decrease the effectiveness of the CSCa attacks. In our case, the noisy environment and mimicry attack scenarios reduced the accuracy to 25% and 20%, respectively. In the case of reduced probing frequency, we could not capture the cache-line pattern with less than 2% error after up to 10000 trial encryptions. The high standard deviation for the Noisy scenario shows that the load fluctuation in the background will affect detection accuracy. Finally, the mimicry attack will add to the computational load of the spy process and lead to some additional processing time, reducing the CSCa resolution timers and increasing the probability of missing the real encryption events from the victim.

Basically, while noisy environments and mimicry may obfuscate the CSCa signatures, these also make the CSCa less effective. We did not study the way to tackle this noisy and mimicry problem within this work as we believe this problem is quite big and challenging for another future work of its own.

6.5. Performance Impact of the Monitoring Process to the Host and Guest VM. We also tested the scalability of our monitoring approach by increasing the number of monitored VMs from 1 up to 8 guest VMs and measured the time needed

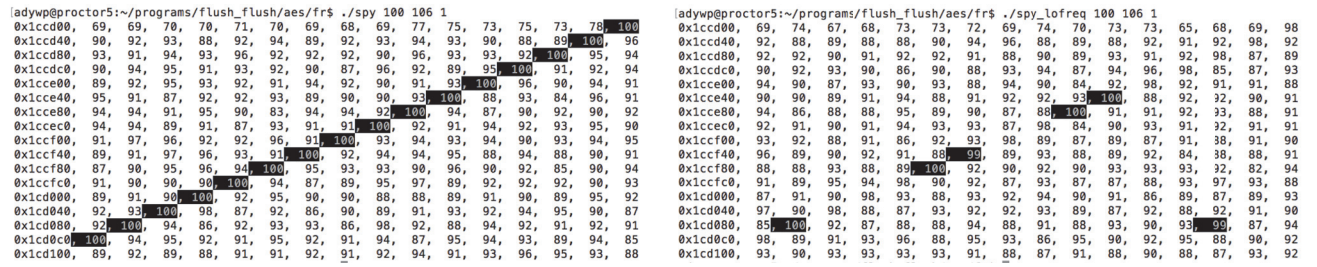


Figure 8: Cache line pattern of k_0 = 0xf_ (a) for clean CSCa implementation (b) for the CSCa with a reduced probe frequency.

Table 5: Comparison between clean, noisy, mimicry, and reduced probe frequency scenarios.

Scenario	(a)	(b)	(c)
Clean	44	7.8	75.93
Noisy (200 CU)	178	82.3	78.86
Mimicry R/W	209	10.5	134.22
Reduced Probing freq.	NA	NA	761.83

(a) Number of encryptions needed to create a cache line pattern of the upper 4 bits of k_0 with less than 2% error (average of 10 attempts); (b) standard deviation of (a); (c) CPU task-clock needed to find the pattern for 100 encryptions (average of 10 attempts).

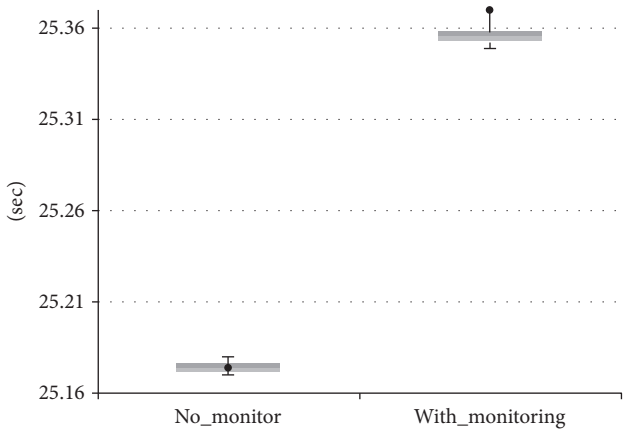


Figure 9: The comparison of time that was needed to calculate 10000 first prime numbers in the guest VM when the KVM events at the host was monitored and when it was not monitored.

to collect 1 unit of observation data. We used Linux's *perf* tool and collected the task-clock data (the CPU time). We found out that the trace-cmd KVM tracing process did not increase CPU utilization even if the number of monitored VMs was increased (at least up to 8 guest VMs in our experiment). The task clock for collecting data remained constant with an average of 0.0443 msec and standard deviation of 0.00176 msec.

Next, we compared the CPU performance of a guest VM with no monitoring and when it is being monitored by the host. For this measurement process, we used the sysbench tool. For this benchmark, we recorded the total execution time of one thread to calculate the first 10000 prime numbers. Figure 9 presents the averages from twenty benchmarking results.

The boxplot shows that the monitoring process in the host had a small impact on the computation performance of the guest VM. In this experiment, there was an increase of 0.7% in the time to complete the task in the guest system when it was monitored from the host using our approach (KVM event observation).

7. Discussion

7.1. Considerations for Implementation in Operational Environment. The procedures used in this study were set up for experimentation purpose. To have a working operational system, we need to determine the explicit threshold for positive-negative decision and the explicit number of positive results threshold to decide when to fire the alarm.

7.1.1. Positive-Negative Discrimination Threshold. The Receiver Operating Characteristic (ROC) curve shows the whole spectrum of possible discrimination thresholds and therefore is useful for selecting the optimal criterion (maximize the true positive rate and minimize the false positive rate). Theoretically, the optimum threshold that maximizes the trade-off between the true positive rate and false positive rate can be derived from the ROC using the Youden Index [57]. The Youden Index J is formulated as

$$J = \text{Sensitivity} + \text{Specificity} - 1, \quad (1)$$

where Sensitivity refers to the true positive rate and Specificity refers to the true negative rate. Graphically, the index can be explained as a single operating point of the ROC with the maximum distance from the chance (diagonal) line.

In practice, the optimum threshold from the Youden Index is not always applicable. That is because the Youden Index gives both false positive and false negative the same

weight (cost). In real-life operation, the operator might apply a different weight to the false positives and false negatives. If we apply a different weight to FP and FN as α and β, respectively, we can write a cost function C as follows:

$$\begin{aligned} C &= \mathrm{FPR}\alpha\left(1-p\right) + \mathrm{FNR}\beta p \\ &= \mathrm{FPR}\alpha\left(1-p\right) + \left(1-\mathrm{TPR}\right)\beta p, \end{aligned} \tag{2}$$

where p is the ratio of positive events from the total events:

$$\begin{aligned} p &= \frac{\mathrm{TP}+\mathrm{FP}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}} \\ \mathrm{FPR} &= \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} \\ \mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \end{aligned} \tag{3}$$

The gradient of the cost function C is any line with coefficient $c = \alpha(1-p)/\beta p$. The optimal threshold that produces a minimum cost is the intersection of the line with coefficient c and the ROC.

7.1.2. Alarm Threshold. Raising an alarm based on only one unit of observation is not suggested as it has a high probability of introducing false alarms from outlier events. We argue that, by assessing the detection status in groups of sequenced observation data (a decision window), the accuracy can be increased. For example, having a decision window of 10, we might choose to raise the alarm anytime it contains 7 positive observations. The proper value for the window size and positive data threshold can be varied for different types of implementations. Finding the optimum value of the observation window size and the threshold value for positive data is a good new research subject.

7.2. Potential Use of KVM Event Data. In the evaluation section, we have shown that even though the KVM event sequences are not directly related to internal CSCa functions, this dataset can still be used to differentiate between CSCa operations and non-CSCa operations. This leads us to believe that the KVM event sequence information can also be used for other more generic monitoring functions, such as an Anomaly Detection System. We can use the same approach we used in Section 6.4, but instead of comparing the incoming data (or the test data) with a specific attack patterns, we can compare it with the benign class scenario which will make this system work as an anomaly detection system.

8. Conclusion

This work is a feasibility study of using KVM events information to detect the cache-based side channel attacks (CSCa). Within this paper, we have shown that CSCa create several unique patterns of KVM event sequences. These patterns can be used to detect the existence of any CSCa variants, including the Flush + Flush attack, within a guest VM. The monitoring system which collected the KVM events does not need any host or guest VM modification. It can work inside the host without guest participation. Furthermore, it only has a small impact on the guest performance and almost zero impact on the host performance which can lead to a highly scalable monitoring system.

We showed that, by using the KVM event sequences for the Support Vector Machine classification method, the separation score of our trained CSCa scenarios and trained non-CSCa scenarios was 0.99 AUC (Area Under the Curve of Receiver Operating Characteristic). The separation score between the trained CSCa scenarios and the untrained CSCa scenarios, which includes the Flush + Flush attack, was close to 0.50 AUC, while the separation score between the trained CSCa scenarios and the untrained non-CSCa scenarios was close to 0.99 AUC. These results show that the KVM events monitoring method can provide low false negatives and low false positives for a CSCa detection system. To strengthen our claim, we performed Fisher score evaluation and successfully identified the KVM event sequences that generalize the separation of the CSCa and non-CSCa operation dataset.

Our further investigation on false negatives showed that our detection method still did not address evasion techniques such as the noisy environment and mimicry attack scenarios. However, we also showed that both scenarios negatively affected the CSCa effectiveness, thus limiting these options for the adversary.

Finally, we evaluated the computation overhead impact of our CSCa monitoring approach and showed that it has a negligible overhead on the host and the guest VM operations.

We believe the results of these experiments are useful to broaden the understanding of CSCa in particular and the operation of CPU caches in general. Our findings can benefit future research in this field to help identify ways to detect CSCa.

We identify several research direction to move forward:

(i) We would like to design an operational version of this detection system. This is not a trivial task because there are many different functions to adapt from the current experimental implementation, such as real-time data collection, preprocessing, and analysis, along with developing a process to find the proper threshold for a positive or negative detection decision.

(ii) An interesting case is to find the solution of CSCa monitoring for other processor architecture, such as the ARM processors which has gained more popularity recently.

(iii) Another challenging problem to be solved is detecting any effort to obfuscate the CSCa in noisy environments or with mimicry operations. A potential approach would be by using the combination of multiple monitoring techniques such as Hardware Performance Counter (HPC), KVM events, and another probing point available from the VMM.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," *Journal of Cryptographic Engineering*, 2016.

[2] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and defenses. Cryptology ePrint Archive," Report 2016/479, 2016.

[3] A. K. Biswas, D. Ghosal, and S. Nagaraja, "A survey of timing channels and countermeasures," *ACM Computing Surveys*, vol. 50, no. 1, article no. 6, 2017.

[4] Security aspects of virtualization, "The European Union Agency for Network and Information Security," Technical Report, February 2017.

[5] D. J. Dean, H. Nguyen, and X. Gu, "UBL: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems," in *Proceedings of the 9th ACM International Conference on Autonomic Computing, (ICAC '12)*, pp. 191–200, USA, September 2012.

[6] F. Doelitzscher, M. Knahl, C. Reich, and N. Clarke, "Anomaly detection in IaaS Clouds," in *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science, (CloudCom '13)*, pp. 387–394, UK, December 2013.

[7] S. S. Alarifi and S. D. Wolthusen, "Detecting anomalies in IaaS environments through virtual machine host system call analysis," in *Proceedings of the 7th International Conference for Internet Technology and Secured Transactions, (ICITST '12)*, pp. 211–218, December 2012.

[8] A. S. Abed, T. C. Clancy, and D. S. Levy, "Applying bag of system calls for anomalous behavior detection of applications in linux containers," in *Proceedings of the IEEE Globecom Workshops, (GC Wkshps '15)*, IEEE, USA, December 2015.

[9] W. Sha, Y. Zhu, M. Chen, and T. Huang, "Statistical learning for anomaly detection in cloud server systems: A multi-order Markov chain framework," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, 2015.

[10] B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin, and W. Lee, "Leveraging forensic tools for virtual machine introspection," Technical Report GT-CS-11-05, Georgia Institute of Technology, 2011.

[11] B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin, and W. Lee, "Virtuoso: Narrowing the semantic gap in virtual machine introspection," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy, (SP '11)*, pp. 297–312, USA, May 2011.

[12] Y. Fu and Z. Lin, "Bridging the semantic gap in virtual machine introspection via online kernel data redirection," *ACM Transactions on Information and System Security*, vol. 16, no. 2, article no. 7, 2013.

[13] W.-M. Hu, "Lattice scheduling and covert channels," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 52–61, May 1992.

[14] D. J. Bernstein, "Cache-timing attacks on aes," Technical Report, The University of Illinois at Chicago, 2005.

[15] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: the case of AES," in *Proceedings of the 2006 The Cryptographers' Track at the RSA conference on Topics in Cryptology, (CT-RSA'06)*, vol. 3860 of *Lecture Notes in Computer Science*, pp. 1–20, Springer, Berlin, 2006.

[16] Y. Yarom and K. Falkner, "Flush + reload: a high resolution, low noise, l3 cache side-channel attack," in *Proceedings of the 23rd USENIX conference on Security Symposium, (SEC '14)*, pp. 719–732, August 2014.

[17] K. Suzaki, K. Iijima, T. Yagi, and C. Artho, "Memory deduplication as a threat to the guest OS," in *Proceedings of the 4th Workshop on European Workshop on System Security, (EUROSEC '11)*, Austria, April 2011.

[18] Y. Oren, V. P. Kemerlis, S. Sethumadhavan, and A. D. Keromytis, "The spy in the sandbox: Practical cache attacks in JavaScript and their implications," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, (CCS '15)*, pp. 1406–1418, USA, October 2015.

[19] T. Hornby, *Side-channel attacks on everyday applications*, Black hat, USA, 2016.

[20] M. Lipp, D. Gruss, M. Schwarz, D. Bidner, C. Maurice, and S. Mangard, "Practical Keystroke Timing Attacks in Sandboxed JavaScript," in *Proceedings of the 25th USENIX Security Symposium*, vol. 10493 of *Lecture Notes in Computer Science*, pp. 549–564, Springer International Publishing.

[21] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, pp. 199–212, November 2009.

[22] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-vm side channels and their use to extract private keys," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security, (CCS '12)*, pp. 305–316, USA, October 2012.

[23] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-tenant side-channel attacks in PaaS clouds," in *Proceedings of the 21st ACM Conference on Computer and Communications Security, (CCS '14)*, pp. 990–1003, USA, November 2014.

[24] M. S. İnci, B. Gulmezoglu, G. Irazoqui, T. Eisenbarth, and B. Sunar, "Cache attacks enable bulk key recovery on the cloud," in *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, (CHES '16)*, vol. 9813, pp. 368–388, August 2016.

[25] Z. Wang and R. B. Lee, "A novel cache architecture with enhanced performance and security," in *Proceedings of the 2008 - 41st Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-41*, pp. 83–93, IEEE, Italy, November 2008.

[26] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *Proceedings of the 34th Annual International Symposium on Computer Architecture, (ISCA '07)*, pp. 494–505, ACM, USA, June 2007.

[27] F. Liu and R. B. Lee, "Random Fill Cache Architecture," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture, (MICRO '14)*, pp. 203–215, UK, December 2014.

[28] Y. Zhang and M. K. Reiter, "Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, (CCS '13)*, pp. 827–837, ACM, November 2013.

[29] F. Liu, Q. Ge, Y. Yarom et al., "CATalyst: Defeating last-level cache side channel attacks in cloud computing," in *Proceedings of the 22nd IEEE International Symposium on High Performance Computer Architecture, (HPCA '16)*, pp. 406–418, IEEE, Spain, March 2016.

[30] Z. Zhou, M. K. Reiter, and Y. Zhang, "A software approach to defeating side channels in last-level caches," in *Proceedings of the 23rd ACM Conference on Computer and Communications Security, (CCS '16)*, pp. 871–882, Austria, October 2016.

[31] G. Irazoqui, T. Eisenbarth, and B. Sunar, "Mascat: Stopping microarchitectural attacs before execution," *Cryptology ePrint Archive*, 2016.

[32] G. Doychev, D. Feld, B. Köpf, L. Mauborgne, and J. Reineke, "Cacheaudit: a tool for the static analysis of cache side channels," in *Proceedings of the 22nd USENIX conference on Security, (SEC '13)*, vol. 18, pp. 431–446, 2013.

[33] L. Domnitser, A. Jaleel, J. Loew, N. Abu-Ghazaleh, and D. Ponomarev, "Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, no. 4, article no. 35, 2012.

[34] T. Kim, M. Peinado, and G. Mainar-Ruiz, "Stealthmem: system-level protection against cache-based side channel attacks in the cloud," in *Proceedings of the 21st USENIX conference on Security symposium*, August 2012.

[35] J. Shi, X. Song, H. Chen, and B. Zang, "Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring," in *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops, (DSN-W '11)*, pp. 194–199, China, June 2011.

[36] Y. Han, T. Alpcan, J. Chan, and C. Leckie, "Security games for virtual machine allocation in cloud computing," in *Proceeding of the 4th International Conference on Decision and Game Theory for Security, (Gamesec '13)*, vol. 8252 of *Lecture Notes in Computer Science*, pp. 99–118, November 2013.

[37] S.-J. Moon, V. Sekar, and M. K. Reiter, "Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, (CCS '15)*, pp. 1595–1606, USA, October 2015.

[38] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang, "Incentive compatible moving target defense against VM-colocation attacks in clouds," in *Proceedings of the IFIP International Information Security Conference on Information Security and Privacy Research, (SEC '12)*, pp. 388–399, Springer, 2012.

[39] M. Chiappetta, E. Savas, and C. Yilmaz, "Real time detection of cache-based side-channel attacks using hardware performance counters," *Applied Soft Computing*, vol. 49, pp. 1162–1174, 2016.

[40] T. Zhang, Y. Zhang, and R. B. Lee, "Cloudradar: A real-time side-channel attack detection system in clouds," in *Proceedings of the Research in Attacks, Intrusions, and Defenses. (RAID '16)*, vol. 9854 of *Lecture Notes in Computer Science*, pp. 118–140, Springer.

[41] M. Payer, "HexPADS: A platform to detect "stealth" attacks," in *Proceedings of the 8th International Symposium on Engineering Secure Software and Systems, (ESSoS '16)*, vol. 9639, pp. 138–154, Springer-Verlag.

[42] N. Herath and A. Fogh, *These Are Not Your Grand Daddy'S cpu Performance Counters*, Black hat, USA, 2015.

[43] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush + Flush: A fast and stealthy cache attack," in *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, (DIMVA '16)*, vol. 9721, pp. 279–299, Springer-Verlag, July 2016.

[44] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *Proceedings of The 10th Annual Network and Distributed System Security Symposium*, 2003.

[45] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Ligauri, "Kvm: the linux virtual machine monitor," in *Proceedings of the 2007 Ottawa Linux Symposium - OLS07*, 2007.

[46] A. Arcangeli, I. Eidus, and C. Wright, *Increasing Memory Density by Using Ksm*, Red Hat Inc, 2009.

[47] G. Venkitachalam and M. Cohen, "Transparent page sharing on commodity operating systems," *Patent US7500048 B1*, 2009.

[48] S. Rostedt, "Ftrace kernel hooks, more than just tracing. LINUX Plumbers Conference".

[49] E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite, and W. Zwaenepoel, "Performance comparison of middleware architectures for generating dynamic web content," in *Proceedings of the 4th ACM/IFIP/USENIX International Middleware Conference*, June 2003.

[50] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "Training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (COLT '92)*, pp. 144–152, ACM, July 1992.

[51] F. Pedregosa, G. Varoquaux, and A. Gramfort, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[53] G. Irazoqui, M. S. IncI, T. Eisenbarth, and B. Sunar, "Know Thy neighbor: crypto library detection in cloud," in *Proceedings on Privacy Enhancing Technologies*, vol. 2015.

[54] A. Fogh, "Cache side channel attacks: Cpu design as a security problem. Hack In The Box".

[55] G. Irazoqui, M. S. Inci, T. Eisenbarth, and B. Sunar, "Wait a minute! A fast, cross-VM attack on AES," in *Proceeding of the Research in Attacks, Intrusions and Defenses, (RAID '14)*, vol. 8688, pp. 299–319, Lecture Notes in Computer Science, 2014.

[56] G. Irazoqui, T. Eisenbarth, and B. Sunar, "S$A: A shared cache attack that works across cores and defies VM sandboxing - And its application to AES," in *Proceedings of the 36th IEEE Symposium on Security and Privacy, (SP '15)*, pp. 591–604, USA, May 2015.

[57] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.

Hindawi
Security and Communication Networks
Volume 2017, Article ID 5960307, 11 pages
https://doi.org/10.1155/2017/5960307

Research Article

Detecting Web-Based Botnets Using Bot Communication Traffic Features

Fu-Hau Hsu,[1] Chih-Wen Ou,[1] Yan-Ling Hwang,[2] Ya-Ching Chang,[1] and Po-Ching Lin[3]

[1]*Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan*
[2]*School of Applied Foreign Languages, Chung Shan Medical University, Taichung, Taiwan*
[3]*Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan*

Correspondence should be addressed to Chih-Wen Ou; chihwen.frankou@gmail.com

Received 28 March 2017; Revised 18 June 2017; Accepted 25 September 2017; Published 3 December 2017

Academic Editor: Steffen Wendzel

Web-based botnets are popular nowadays. A Web-based botnet is a botnet whose C&C server and bots use HTTP protocol, the most universal and supported network protocol, to communicate with each other. Because the botnet communication can be hidden easily by attackers behind the relatively massive HTTP traffic, administrators of network equipment, such as routers and switches, cannot block such suspicious traffic directly regardless of costs. Based on the clients constituent of a Web server and characteristics of HTTP responses sent to clients from the server, this paper proposes a traffic inspection solution, called Web-based Botnet Detector (WBD). WBD is able to detect suspicious C&C (Command-and-Control) servers of HTTP botnets regardless of whether the botnet commands are encrypted or hidden in normal Web pages. More than 500 GB real network traces collected from 11 backbone routers are used to evaluate our method. Experimental results show that the false positive rate of WBD is 0.42%.

1. Introduction

A botnet is a group of compromised computers, namely, bots, controlled by one or multiple controllers [1–3]. These botnet controllers, also named bot masters, provide commands to their bots through C&C (Command-and-Control) servers so that the bots can perform actions for their bot masters. There are several criteria to categorize botnets, including the attacking behavior, C&C model, communication channel, rallying mechanism, and the evasion technique. We firstly focus on the centralized C&C model and discuss details about it in this study.

For a botnet with a centralized C&C model, each bot connects to its C&C server to retrieve commands or to deliver data. There are many advantages to use such an architecture to organize C&C servers and their bots compared to the decentralized and randomized models. The first advantage is the low cost to construct such a botnet, because bot masters can easily create this kind of botnets using many off-the-shelf open resources and applications. Meanwhile, the centralized model allows a bot master to quickly rally a large number of its bots by commanding few C&C servers. Such efficiency obviously facilitates cybercriminals to use botnets to conduct malicious activities, such as DDoS attacks and spamming [4].

According to the communication protocols used by botnets, botnets can be classified into several categories. These categories include the IRC- (Internet Relay Chat-) based botnet, the IM- (Instant Message-) based botnet, and the Web-based botnet. This paper focuses on the Web-based botnet, also named *HTTP botnet*, whose communication channel between the C&C server and its bot clients is via the HTTP. A C&C server of a Web-based botnet works like a normal Web server, and bot clients of a Web-based botnet work as normal Web clients. We call the C&C server of a Web-based botnet a *botnet Web server* hereafter. Two botnets, Spyeye [5] and Zeus [6], are well-known HTTP-based botnets. According to previous studies on these two botnets, there are several reasons why the HTTP is attractive to botnet owners. First, HTTP traffic is the most popular Internet traffic nowadays so that Web-based botnet traffic can be easily disguised as normal HTTP traffic, making the botnets more difficult to be discovered than those that use less popular protocols. Second, most network firewalls/proxies allow hosts behind them to access Internet via the HTTP. As a result,

Web-based botnets can easily provide stable and qualified client-to-server connectivity. Third, many promising solutions [1, 7–9] have been developed to precisely detect traditional IRC-based botnets instead of Web-based botnets. Therefore, the HTTP gradually becomes an ideal alternative protocol for botnet owners to use as the communication channel in recent years, and our study focuses on this kind of botnets.

1.1. Web-Based Botnet Detection. Bot clients are trojans, executable programs, or scripts running on compromised hosts. Hence, their behavior is different from human user behavior. Besides, their activity pattern, sizes, and transferred content are also different from human users. As programs generate the communication traffic automatically, the Web-based botnet communication has some prominent characteristics. According to our preliminary survey on a botnet taxonomy study [2], a typical bot client of a centralized C&C botnet often needs to synchronize with its botnet Web server to retrieve commands or deliver execution results. Such synchronization is often scheduled when bot clients are effectively controlled by botnet Web servers. Hence, we think that this phenomenon of synchronization can be utilized as a hint to indicate whether Web clients are controlled by human users or by bot clients. Besides, we also found that if a group of Web clients associated with a Web server consists of human users, each of them often has a different access pattern to the Web server. For example, these human clients may visit the Web server at different times of a day, or these clients may visit the Web server different numbers of times each day. On the contrary, if these Web clients are bot clients, which run programs or scripts, they may act together and behave similarly. Therefore, they may contact their botnet Web server repeatedly according to a predefined time interval to access commands from their botnet Web server. Such repeated contact to certain botnet Web servers may continue for several days, which is apparently different from normal human behavior. In addition, the same group of bot clients usually tends to communicate with the same botnet Web server. Based on the long-term repeated contact phenomenon and similar access pattern of the clients of a Web server, we use a metric named *Total Host Repetition Rate*, or *THR* in short, as one of our criteria to examine whether a Web server is a suspicious botnet C&C server.

Instead of THR, we also found that the payload inside the traffic between bot clients and their botnet Web server usually contains short and simple commands. Furthermore, all bot clients commanded by a certain C&C server tend to receive commands at the same time. This similarity of payloads among the bot clients controlled by the same botnet Web server is also described as the command-response pattern by BotProbe [7]. A normal Web server usually contains many Web pages and different users accessing different Web pages. Hence, unless the Web server contains only one Web page, the probability that its users retrieve the same Web page from the Web server simultaneously is low, and different Web pages usually have different sizes. As a result, during a period of time, the sizes of responding payloads of different Web clients accessing the same Web server are supposed to be different, while a botnet Web server often dispatches similar commands to its bot clients at the same time. Thus, we utilize the payload size difference as a metric called the *payload size similarity*, or *PSS* in short, to judge whether a Web server is a suspicious botnet C&C server or not. The formalization for these two metrics will be discussed later in Section 2.2.

In our prototype implementation, we designed an automatic mechanism based on the above two metrics and integrated this mechanism into our prototype system, named *Web-based Botnet Detector*, or *WBD* in short, to perform the inspection. WBD is attached to a network traffic monitoring system which is able to generate traffic logs from the online network stream and analyzes these logs simultaneously. Only few arithmetic calculations are required by WBD while performing runtime inspection on those monitored traffic logs. Such calculation brings significant overhead to other similar approaches during traffic inspection.

1.2. Contributions. The solution of this paper contains the following characteristics: (1) Compared to mainstream machine learning approaches which often rely heavily on tens or even hundreds of features, an approach with only few features can reduce notable overhead. WBD requires only several deterministic calculations which are easily extracted and calculated from monitored network traffic. (2) WBD inspects traffic of backbone networks. It does not require any program installed on network end-hosts and servers. (3) WBD does not use features based on traffic content mining. It does not rely on particular protocol-parsing as well. In summary, the contributions of WBD include the following.

(1) WBD requires only several deterministic calculations, which means that it is ideal to cooperate with heavy-loading backbone equipment.

(2) We conducted large-scale backbone data inspection for this study. It reveals those IP addresses and timestamps of Web servers that generate suspicious Web-based botnet communications across the global Internet.

(3) Due to the low correlation between the content itself, our solution can target the HTTPS protocol theoretically. Also, botnet owners may deliberately embed their botnet commands into some normal traffic, such as universal Web contents, to bypass the potential inspection along the traffic path. Our solution can work for this situation, because the calculation on THR and PSS requires the source and destination IP addresses of the packets in the traffic, which are not encrypted by most secure protocols.

This paper includes six sections. Section 2 will explain how we use features calculated from these criteria for the datagram-like network traffic logs. Sections 3 and 4 evaluate our approach and discuss issues including comparison with other similar approaches and the accuracy. Section 5 describes previous studies aiming at botnet related issues. Section 6 summaries this study.

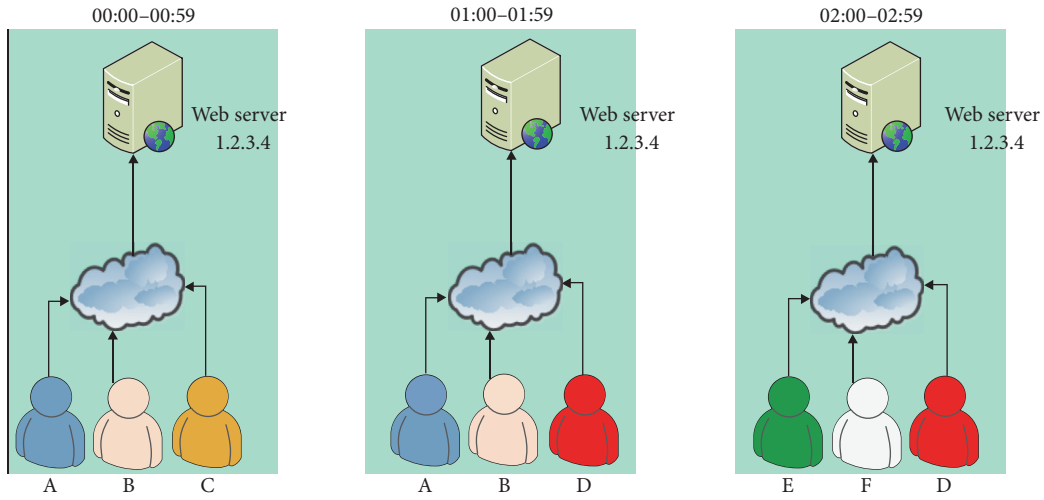


Figure 1: A communication pattern between 6 Web clients and a Web server during a three-hour monitoring period.

2. Methodology

In order to develop a traffic inspection approach, several issues have to be considered. These issues include the involved scale of monitoring, the volume of the traffic, and the feasibility to obtain such traffic. If an approach has to monitor the user-side traffic, an appropriate inspecting location may be at a gateway, a router, or a proxy (if it is mandatory for each network user) of the target user-side network. If an approach requires to monitor the server-side network, a possible location to do this work may be located at the intrusion detection equipment or firewall equipment of the target server-side network. These two deployments are commonly selected by many traffic inspection approaches because of their deployment feasibility and the affordable traffic volume. Different from these two categories, our approach aims at monitoring the global Internet as much as possible. The possible inspecting locations for such kind of approach should include backbone equipment that routes and processes large amount of IP packets. In our study, we are allowed to obtain the traffic from several online backbone routers in Taiwan so that we can develop a solution that is not specifically restricted by user-side or server-side networks.

Even though we are able to obtain logs from actual backbone routers, these routers are so important for our Internet service provider and they are always full-loaded. Hence, directly running inspecting procedures on them is certainly impractical so we adopted offline analyzing-after-recording method to make our experiments. We collected log samples from these backbone routers for several times and analyzed them. The detailed information about this will be described in Section 3. Due to many security and privacy concerns, all these actions were conducted and completed in an office of an Internet service provider. We cannot see the content of IP packet payloads and we cannot take any log-out from the office. Information about the recorded data from the traffic will be described later in Section 3.1.

2.1. A Case Study. To provide a clear understanding of our approach, considering an input case extracted from our logs depicted in Figure 1, there are six Web clients named from A to F requesting, respectively, a Web server with an IP address denoted as 1.2.3.4 hereafter. The three-hour long monitoring period is separated into three consecutive time intervals, as shown in Figure 1, and the length of each time interval is one hour. If clients A, B, and C make requests to the Web server in the first time interval, there will be arrows connecting them to the Web server, as shown in the left time interval marked with 00:00–00:59. Similarly, clients A and B repeat requesting and D makes the request in the second time interval. Client D repeats requesting, and clients E and F make requests in the third time interval in this case. A graphic representation of this case is shown in Figure 2. A Web server is denoted as an S-vertex, and a Web client is denoted as a C-vertex. The communication between a Web server and a client is denoted as an undirected edge connecting these two vertexes, as an example shown in Figure 2. There is no edge between two different C-vertexes because we do not need to consider the case when a Web client also runs a Web service. After all, we only focus on centralized botnets. We can also ignore edges between two S-vertexes because we only focus on communication made by Web clients. A graph is used to describe the communication patterns between Web clients and a Web server in a time interval. These graphs will not be used directly for graph computation. How these graphs are used will be described in Section 2.2.

2.2. Features Formulation. We use graphic representation only for conceptive discussion. For actual calculation conducted by WBD, equivalent formulas calculation is adopted after obtaining traffic logs. Such a design ensures that related calculation is theoretically light-weight, and such an approach is suitable for working with existing Internet backbone equipment. As we have mentioned in Section 1.1, two metrics are used to determine whether there exists botnet

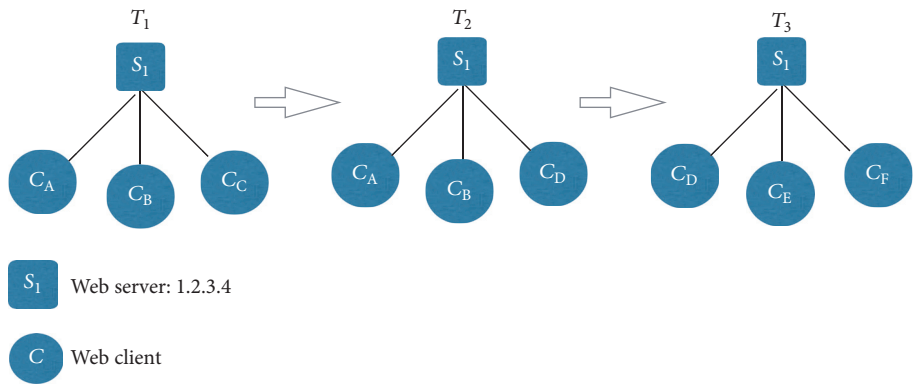


FIGURE 2: Graph representations of communication patterns between Web clients and their Web server at different monitoring time intervals.

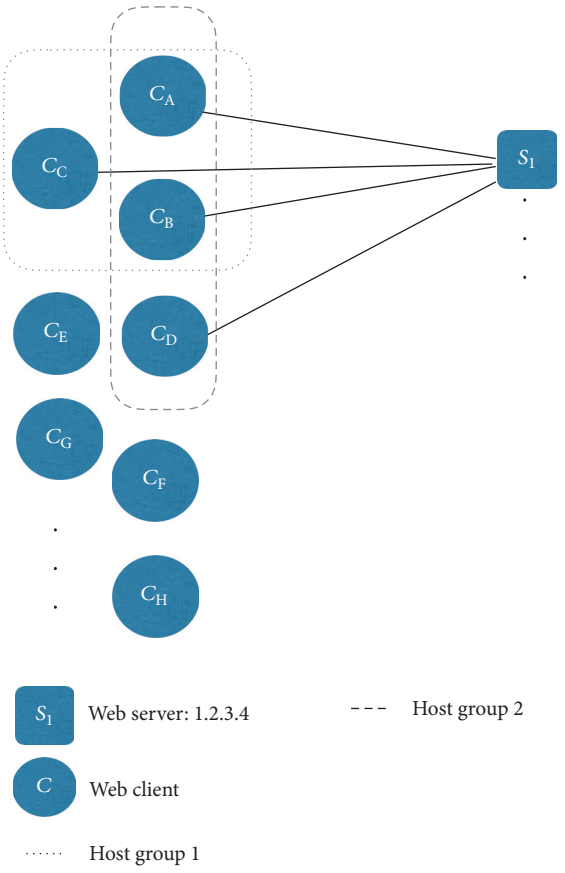


FIGURE 3: Two host groups related to Web server S_1 at two consecutive time intervals T_1 and T_2.

communication in the monitored HTTP traffic. In this paper, we call the group of Web clients that communicate with a Web server in a time interval a *host group*, as two groups of this case shown in Figure 3 show two host groups appearing at two different time intervals. If a Web server is a botnet Web server, the associated host groups are called *bot groups*. According to the previous studies [2, 10] and observation, the hosts of a bot group tend to communicate with the same botnet Web server all the time. Even though the constituent members of a bot group may change due to some technical issues or management reasons, such a change does not occur dramatically in a short period of time.

Two host groups, which are associated with the same Web server appearing in successive time intervals, are called *adjacent host groups*. We use two scores, *Access (AC)* score and *Total Host Repeat (THR)* score, to evaluate the THR feature of a Web server. Equation (1) defines these two scores, respectively. Score AC_s represents the number of total hosts communicating with Web server s in these n time intervals. For a certain time interval, the *HR* score is defined as the

proportion of the number of hosts appearing in both the current host group and its previously adjacent host group to the number of hosts in the current host group. The intersection of $\text{hostgroup}_{s_{t-1}}$ and hostgroup_{s_t} denotes the set of hosts appearing in both the adjacent host groups. Score THR_s is an average of n *Host Repeat (HR)* scores, denoting the similarity of the host groups of Web server s in n time intervals.

$$\begin{aligned} \text{AC}_s &= \sum_{t=1}^{n} \left|\text{hostgroup}_{s_t}\right| \\ \text{THR}_s &= \frac{1}{n}\sum_{t=1}^{n} \frac{\left|\text{hostgroup}_{s_{t-1}} \bigcap \text{hostgroup}_{s_t}\right|}{\left|\text{hostgroup}_{s_t}\right|}. \end{aligned} \tag{1}$$

Assume the host group of Web server s is hostgroup_{s_t} in time interval t and there are k_t different total responding payload sizes, $\text{PS}_{t_{k_1}}, \text{PS}_{t_{k_2}}, \ldots, \text{PS}_{t_{k_t}}$, for the $|\text{hostgroup}_{s_t}|$ hosts. $|\text{PS}_{t_{k_i}}|$ represents the number of hosts whose payload size is $\text{PS}_{t_{k_i}}$ in time interval t. Score *payload size similarity (PSS)* defined in (2) is used to evaluate the payload similarity feature of a Web server. Equation (2) gives its definition.

$$\text{PSS}_s = \frac{\sum_{t=1}^{n} \max_{i=1}^{k_t} \left(\left|\text{PS}_{t_i}\right|\right)}{\sum_{t=1}^{n} \left|\text{hostgroup}_{s_t}\right|} = \frac{\sum_{t=1}^{n} \max_{i=1}^{k_t} \left(\left|\text{PS}_{t_i}\right|\right)}{\text{AC}_s}. \tag{2}$$

2.3. WBD Classifier. After quantification of features for each input Web server, we can distinguish between normal Web servers and suspicious botnet Web server by comparing their THR, AC, and PSS scores to thresholds. In order to find the proper thresholds, we extracted the HTTP traffic traces of Alexa top 20 Websites in Taiwan from 11 backbone routers. The traffic related to these popular Websites is supposed to be nonbotnet traffic. However, when botnet traffic is hidden in the traffic of a social network Website, the social network traffic may contain botnet traffic. Hence, when collecting nonbotnet traffic, we can filter out social network related traffic first to avoid the above problem. However, in our traces, except Facebook which continues detecting and removing fake or malicious accounts, almost all of the top 20 Websites are nonsocial network Websites. Therefore, we consider the traffic associated with these popular Websites as benign. We calculated the THR score and AC score of each Web server and then selected the maximum THR score and AC score as the thresholds. Equation (3) shows the equations. THR_i represents the THR score of Web server i, and AC_i represents the AC score of Web server i.

$$\begin{aligned} \text{THR}_{\text{thershold}} &= \max_{i \in \text{top\_20\_benigh\_servers}} \left(\text{THR}_i\right) \\ \text{AC}_{\text{thershold}} &= \max_{i \in \text{top\_20\_benigh\_servers}} \left(\text{AC}_i\right). \end{aligned} \tag{3}$$

WBD uses the thresholds to examine Web servers appearing in the HTTP traffic traces and uses (4) to check whether Web server i exhibits the THR feature denoted by $\text{HR}_{\text{Susp\_Server}}(i)$.

$$\text{HR}_{\text{Susp\_Server}}(i) = \begin{cases} \text{True}, & \text{THR}_i > \text{THR}_{\text{thershold}} \wedge \text{AC}_i < \text{AC}_{\text{thershold}} \\ \text{False}, & \text{otherwise.} \end{cases} \tag{4}$$

WBD uses (5) to check whether Web server i exhibits the similar payload size feature denoted by notation $\text{PSS}_{\text{Susp\_Server}}(i)$. $\text{PSS}_{\text{threshold}}$ in (5) is defined as 0.5 because if in average the payload sizes of 50% hosts of each host group of a Web server during a time interval are similar to each other, the Web server is unlikely to be a normal Web server.

$$\text{PSS}_{\text{Susp\_Server}}(i) = \begin{cases} \text{True}, & \text{PSS}_i > \text{PSS}_{\text{threshold}} \\ \text{False}, & \text{otherwise.} \end{cases} \tag{5}$$

Based on the values determined by (4) and (5) for Web server i, WBD uses (6) to determine whether Web server i is a suspicious botnet Web server. All hosts which connect to it more than once are supposed to be its bot clients.

$$\text{Susp\_C\&C}(i) = \begin{cases} \text{True}, & \text{HR}_{\text{Susp\_Server}}(i) == \text{True} \vee \text{PSS}_{\text{Susp\_Server}}(i) == \text{True} \\ \text{False}, & \text{otherwise.} \end{cases} \tag{6}$$

WBD is built based on the above equations. There are four components in our prototype system. The first is to collect the raw data. The second is a module able to calculate THR and AC. The third is a module able to calculate PSS. The last is a combination of a report generator and a classifier operating according to the output from the second and the third modules.

3. Evaluation

The evaluation of WBD has two purposes. The first purpose is to discover appropriate thresholds of our solutions. The second purpose is to estimate the effectiveness of WBD. In order to evaluate the effectiveness of WBD, we need to know the number of botnet Web servers whose network traffic is recorded in our datasets. However, according to the phishing domain survey reports made by McGrath et al. [9] and Aaron et al. [11], attackers usually do not use a compromised host for more than a couple of days. Hence, we are not able to check all the Web servers in our collected datasets before attackers stop using some botnet Web servers that are hidden inside these large numbers of Web servers. Instead of checking all Web servers for entire datasets, we can only perform manual check on malicious hosts identified by WBD to determine whether they are truly malicious, so that we can at least calculate the false positive rate of WBD in our evaluation.

We collected network traces three times from 11 backbone routers in Taiwan. These routers belong to one of the three largest Internet service providers in Taiwan. Each collection generates a dataset. Each collection lasts for 48 hours to generate a dataset. Hence, we obtained 3 datasets. These routers

Table 1: Fields of a NetFlow V5 record.

Content	Bytes offset	Description
srcaddr	0–3	Source IP address
dstaddr	4–7	Destination IP address
dPkts	16–19	Packets in the flow
srcport	32-33	Source port number
dstport	34-35	Destination port number
prot	38	Protocol (6 = TCP, 17 = UDP)

Table 2: Information of our training phase dataset.

Time period	Size of the raw file	Number of Web servers
2013/03/16 00:00–2013/03/17 23:59	About 200 GB	9933

Table 3: Information of our testing phase datasets.

Index	Time period	Size of the raw file	Number of Web servers
1	2013/06/01 00:00–2013/06/02 23:59	About 140 GB	156294
2	2013/01/18 00:00–2013/01/19 23:59	About 160 GB	170920

Table 4: Results of the testing phase.

Index	False positive	Number of suspicious Web servers
1	3 (0.28%)	1047
2	9 (0.83%)	1085
Total	12 (0.42%)	2132

are Cisco routers equipped with NetFlow [12]. Therefore, these datasets were recorded in the NetFlow V5 compatible format. Our experiments include two phases. The first is the training phase which is used to determine the thresholds using the first dataset. The second is the testing phase which uses the other two datasets.

3.1. NetFlow. NetFlow is able to record all traffic passing through a Cisco router. It fetches data from IP packets and generates flow records. Those flow records can be transferred to other devices for further analysis. The source address field *srcaddr*, destination address field *dstaddr*, source port field *srcport*, destination port field *dstport*, and protocol field *prot* of a NetFlow V5 record specify a session between a certain source host and a destination host via the HTTP, as shown in Table 1. The *dPkts* field contains the raw packet data, so that we can calculate the payload size of a packet and the total payload size of an HTTP session.

3.2. Threshold and Training Phase. The first part of our experiments is to calculate the thresholds and perform training. The training data were collected from March 16 to 17 in 2013. The number of backbone routers involved in this phase is less than the number of routers in the testing phase because we chose the routers which forward packets to popular Web servers in this phase. As shown in Table 2, the total raw data size in this phase is about 200 GB which consists of the IP addresses of 9,933 Web servers. The THR scores of Alexa top 20 popular Websites in Taiwan are all less than 0.521. We also used a browser to manually connect to the 9,933 Web servers to check which of them are normal Web servers and which of them are abnormal. The THR scores of the above normal Web servers are almost all less than 0.521. In contrast, the THR scores of the above abnormal Web servers are almost all greater than 0.521. Therefore, we set $\mathrm{THR}_{\mathrm{threshold}}$ as 0.521 and set $\mathrm{AC}_{\mathrm{threshold}}$ as 12,000. Besides, $\mathrm{PSS}_{\mathrm{threshold}}$ is set to 0.5 as described in previous subsection. We also calculated the average Web page size for these top 20 Websites, and the size is 47,087 bytes.

3.3. Testing Phase. In the testing phase, two datasets were used. Table 3 shows the information of these samples. More than 300 GB data were used in our analysis. These two datasets contain network traces of 156,294 and 170,920 Web servers, respectively. Among these Web servers, WBD found 1,047 suspicious botnet Web servers from testing dataset 1 and 1,085 suspicious servers from testing dataset 2. For each of these 2,132 suspicious botnet Web servers, we use a browser to manually check their content. If a suspicious botnet Web server replies to a normal Web page, we treat this case as a false positive case. Besides, bot clients usually retrieve commands from their botnet Web server, and the sizes of the commands are supposed to be smaller than the size of normal Web pages. Therefore, if the size of data returning from a Web server is greater than 47,087 bytes, we will deem the Web server as a normal one and also treat this case as a false positive case. The result of the testing phase is shown in Table 4. To calculate the false negative rate, we need to

Table 5: Features and classifiers used by four similar approaches.

Approaches	Features	Classifier
Venkatesh and Nadarajan	ORT, RIO, PT, SYN, FIN, PSH	Neural network
Zhao et al.	PX, PPS, NR, APL, FPS, PV, FPH, TBP	Decision tree
Cai and Zou	SHH, CC, SCL, BIC, PR, DWS	Multilayer filter
WBD	AC, THR, PSS	Decision tree

Table 6: Comparisons of features used by four approaches.

Calculations	Venkatesh and Nadarajan	Zhao et al.	Cai and Zou	WBD
Counting specific packets	ORT, RIO, PT, SYN, FIN, PSH	PX, PPS, NR	—	—
Arithmetic based on packet size	—	APL, FPS, PV	SHH, CC, SCL	PSS
Arithmetic based on numbers of hosts	—	FPH	BIC	AC, THS
Arithmetic based on interpacket timing	—	TBP	PR	—
Host fingerprinting	—	—	DWS	—

manually check 327,214 Web servers to confirm the botnet Web servers within them. However, according to the phishing domain survey reports made by McGrath et al. [13] and Aaron et al. [14], attackers usually do not use a compromised host for more than a couple of days. Because we are not able to check all the 327,214 Web servers before attackers stop using some botnet Web servers that are hidden inside these 327,214 Web servers, currently we are not able to calculate the false negative rate of WBD. However, when comparing with a malicious IP list provided by ICST [15], we found that the majority of the botnet Web servers we found are not in the list, which shows that WBD provides a list of originally unknown botnet Web servers to system administrators.

4. Discussion

Some approaches aiming at detecting HTTP botnets were also proposed in recent years. These approaches use various features to inspect network traffic to detect HTTP botnets. Three of such approaches are selected and compared with WBD. Table 5 lists these approaches with their features and classifiers. Venkatesh and Nadarajan [16] proposed a multilayer feedforward neural network solution with six features, including one-way ratio of TCP packets (ORT), ratio of incoming to outgoing TCP packets (RIO), the proportion of TCP packets in the flow (PT), and TCP flags counting on SYN, FIN, and PSH flags. These features require only counting specific packets, so that they increase relatively slight performance overhead compared to other complex features used by the rest of the approaches. Such counting-based features are simple and can be manipulated by communicators, so that botnet owners who are aware of such features can bypass the detection by specifically changing their forms of communication packets. Zhao et al. [17] proposed a solution with eight features. Three of them are related to counting-based features including the number of packets exchanged (PX), the number of packets exchanged per second in short time interval (PPS), and the number of reconnections (NR). Three of them are related to arithmetic operations based on the packet payload size, including the average payload packet length (APL), the variance of the payload packet length (PV), and the size of the first packet (FPS). One of the two remaining features involves arithmetic calculations for the number of flows from this address over the total number of flows generated per hour (FPH), and the other feature calculates the average time interval between two consecutive packets (TBS). This approach has higher accuracy than the previous study of Venkatesh and Nadarajan, and its performance overhead is certainly increased due to involving more complicated features compared to the previous study. Cai and Zou [10] proposed a solution with six features. Three of them require arithmetic operations based on the packet payload size, including short HTTP header (SHH), constant content (CC), and short content length (SCL). One feature is related to the bot IP clustering (BIC), one focuses on the periodical request (PR), and the last one requires the host fingerprinting among Web servers to estimate the extent of diversified Web services (DWS). Although this approach has the comprehensive discussion about the features of HTTP botnet and comes up with a set of complicated features that is suitable for determining the existence of botnet communication precisely, the performance overhead is still a significant issue. Many complex features, especially the DWS feature, are involved in this approach for traffic inspection.

Based on the above discussion, we discovered that some kinds of calculations are commonly required by some of these four approaches. Table 6 describes the summarization. Both the study of Venkatesh and Nadarajan and the study of Zhao et al. count specific packets. Both the study of Zhao et al. and the study of Cai and Zou have features which require performing arithmetic operations based on the packet payload size or based on interpacket timing. Three approaches, including WBD, have features requiring execution of arithmetic operations based on the numbers of hosts and the packet size. However, WBD uses only three features requiring execution of arithmetic operations based on numbers of hosts and the packet payload size. Compared to the study of Venkatesh and Nadarajan and the study of Zhao et al., WBD does not need to count specific packets, so that botnet owners have fewer opportunities to bypass WBD. Besides,

Table 7: Comparison among false positive rates of four similar approaches.

Approaches	False positive rates of various test datasets
Venkatesh and Nadarajan	Spyeye-1 (0.97%), Spyeye-2 (0.98%), Zeus-1 (0.99%), Zeus-2 (0.96%)
Zhao et al.	BlackEnergy (0%), Weasel (82%)
Cai and Zou	SJTU1 (17.6%), SJTU2 (26.3%), QingPu (13.6%)
WBD	0.42%

unlike the study of Cai and Zou, WBD does not apply time consuming features such as features of interpacket timing and host fingerprinting so that WBD has limited performance overhead and is able to complete classification in time.

To evaluate the effectiveness of their approaches, each of these four similar approaches used their own datasets to obtain the false positive rates of the chosen datasets. Table 7 lists test datasets and respective false positive rates of these four similar approaches. Four datasets were used by Venkatesh and Nadarajan, and all false positive rates of these datasets are under 1%. For the false positive rates of Zhao et al., the false positive rate of test dataset BlackEnergy is 0%. Dataset BlackEnergy is a pure botnet traffic dataset. The false positive rate of test dataset Weasel is 82%. Dataset Weasel contains normal traffic. The authors analyzed these 82% false positives (2902 false alerts) and discovered that all of these false alerts belong to six applications. They claimed that once a whitelist is adopted for their approach, these false positives would be reduced. The study of Cai and Zou used three datasets to test their approach. The false positive rates range from 13.6% to 26.3%. WBD used logs directly captured from backbone routers and the false positive rate is 0.42%. Compared to other three approaches, WBD is better than the study of Venkatesh and Nadarajan and the study of Cai and Zou. WBD does not need a prebuilt whitelist to remove normal applications before detection.

4.1. False Positives. The total false positive rate of our study is 0.42%. This excellent accuracy results from the adoption of THS and PSS. In fact, many existing front-end Web applications may repeat contacting a Web server. The Web-based instant messenger is one of the typical examples where Web clients contact their Web servers repeatedly. However, as mentioned in previous paragraph, other related approaches may not distinguish the differences between a botnet and a Web server functioning as a Web-based instant messenger.

4.2. False Negatives. Due to the reasons described in this subsection, currently we are not able to discuss the false negative of our work. According to the phishing domain survey reports made by McGrath et al. [13] and Aaron et al. [14], attackers usually do not use a comprised host for more than a couple of days. Apparently, we are not able to check all the Web servers classified as benign in our datasets in time before most attackers stop using botnet Web servers that are hidden inside these large amounts of Web servers. Compared to several previous similar studies [10, 16, 17], most of them evaluate their solution by the datasets containing specific botnets of Spyeye [5] and Zeus [6] instead of real live traffic from Internet. This means that these similar approaches may be accurate when they are applied for detecting those botnets which have similar characteristics to Spyeye and Zeus, but the accuracy is not evaluated for other Web-based botnets. However, most botnet owners keep changing attributes and characteristics of their botnets to avoid being detected. Another reason why false negatives sometimes are impractical is that the Web-based botnets may provide legal online Web services simultaneously. Mostly they may act like normal Web services, and it is very difficult, if not impossible, to enumerate all Internet Web servers having such a characteristic. All issues listed here lead to the uncertainty of the discussion about the false negatives. We will keep discussing this issue in our future work.

4.3. Detection Evasion. Experimental results show that WBD is an ideal solution for Web-based botnet detection. However, current Web-based botnets may change their designs to bypass the detection of WBD. For example, bot clients may connect to their C&C server at nonadjacent time intervals, or various lengths of gibberish bytes may be added to the response payloads of different bot clients to diversify the response lengths. However, such evasion methods may create several drawbacks in the modified botnets. First, this makes the design and operation of a botnet much more complicated because a botnet needs to coordinate the action of each bot client. Second, gibberish bytes increase network traffic. Besides, if different bot clients use the same URL but get Web pages with different lengths, this may be a sign that the related server is not a normal Web server. Besides, C&C servers may apply fast-flux domain technique to change their IP addresses frequently in a very short period of time. Botnets with such ability theoretically possibly bypass WBD deliberately with the price that all bots need to connect and disconnect different hosts frequently, which makes them much more detectable by system/network administrators. In the literature of fast-flux research, an approach proposed by Hsu et al. [18] has been developed to detect fast-flux domains from a single host without using router traces. Hence, by integrating both kinds of approaches, we can create an effective method to detect various Web server-based botnets.

The goal of this paper is to find the botnet Web servers. However, during the detection, we can also obtain the hosts, that is, bot clients, that connect to the botnet Web servers. Hence, in our future work, we will make more detailed survey to find the properties of bot clients. Moreover, this study also works for detecting botnet Web servers communicating with their bot clients via the HTTPS channel because the detection relies only on unencrypted parts of IP packets instead of inspecting the payload content. The unencrypted parts include the information of the source host and

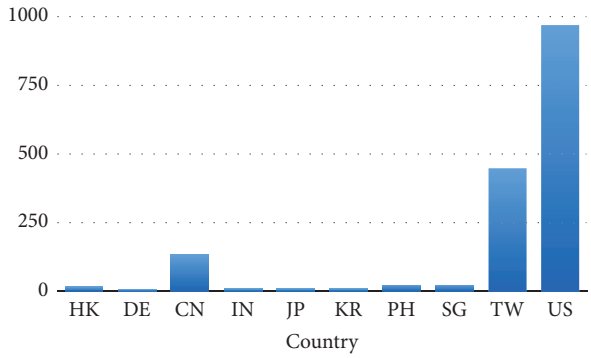


Figure 4: C&C server locations.

destination host and the payload size. Furthermore, the THR feature and the PSS feature will not be changed by modifying the content that a botnet Web server sends to its bot clients. Hence, even if a bot master hides its command inside a normal-looking Web page, WBD is still able to detect it. After detecting a list of C&C servers, we also survey the distribution of the locations of these C&C servers. Figure 4 shows the locations of the 1,085 C&C servers that WBD detects from testing dataset 2. The majority of them are located in the USA and Taiwan. Some of them are located in China, Singapore, Philippines, and so on.

5. Related Work

Most previous studies aim at generic botnet detection. Gu et al. proposed several correlation-based detection solutions: BotMiner [19], BotSniffer [3], BotProbe [7], and BotHunter [20]. BotMiner is a well-known network level correlation-based and protocol-structure independent solution. It performs the connection behavior (C-Plane) and attack behavior (A-Plane) clustering and then performs cross-plane correlation to build a model for botnet C&C servers. BotMiner requires some real-world C&C server network traces in the training phase. However, such reliable C&C traces are not always available in practice. BotSniffer is also a correlation-based solution able to detect C&C servers in a port-independent manner. It is composed of a protocol matcher, an activity/message response detector, and a correlation engine. The correlation engine runs group activity/message response analysis based on the outputs from this protocol matcher and response detectors without requiring other prior knowledge of these botnet C&C servers. Only few packets are needed for training BotSniffer, and it also works well at detecting small botnets. BotProbe is a behavior-based solution, specifically focusing on the command-response pattern of the botnet and its deterministic behavior (for the stateless bot client). BotMiner, BotSniffer, and BotProbe have some problems when the botnet attempts to avoid such detection. The possible evasions include using strong encryption, using atypical response, and injecting random noise packets. Especially for BotMiner, the botnet can create a specific evasion for bypassing the C-plane and A-plane clustering. BotProbe has assumptions that the input has to be perspective and the chat protocol between bots has to be available for the detection engine. BotHunter detects botnets based on the bot-specific heuristics and the IDS dialog-based correlation. The IDS dialogs represent different stages of a botnet life cycle. Such correlation can produce signatures for IDS systems. This IDS-driven strategy has a problem when detecting encrypted botnet communications. Furthermore, this solution also has weaknesses similar to IDS, and the signature generation and update problems must be overcome to reach ideal detection performance.

Yu et al. proposed the SBotMiner [21], an approach based on large-scale network traffic filtering aiming at detecting search bots, which often perform suspicious search activities on the Internet, and SBotMiner uses PCA (Principle Component Analysis) to separate the bot traffic from the benign user traffic. This approach suffers from noise-queries because the search bots can generate lots of meaningless search activities to decrease the detection performance considerably. Karasaridis et al. proposed a wide network traffic correlation solution [9]. However, it only focuses on IRC-based botnet and needs many kinds of prior knowledge before performing the correlation. Zand et al. proposed an approach [22] to automatically extract Command-and-Control signatures for detecting botnets. Since the signature generation is based on the extraction of frequent communication patterns, it is also not applicable to encrypted communication. Wang et al. proposed a fuzzy pattern-based filtering algorithm [23]. This algorithm depends on the DNS query patterns, so that the botnet, especially for the Web-based botnet, can easily avoid the filtering by directly using IP address to communicate.

Some recent research aims at detecting the decentralized peer-to-peer (P2P) botnets. Zhang et al. proposed an approach [24] aiming at detecting P2P botnets. Using decentralized architecture greatly increases the survivability because most botnet takedown actions target C&C servers. However, the decentralized architecture also has some critical disadvantages. P2P botnets often have a complex architecture. Hence, maintaining a P2P network always demands significant technical efforts. In addition, its non-client-server architecture makes it inappropriate to be integrated into existing Web services. Other early studies discussed and evaluated the scale and the takedown techniques of a botnet. Both Dagon et al. [2] and Khattak et al. [25] discussed how different kinds of botnets are organized and what activities they may have. Abu Rajab et al. focused on botnet scale evaluation [1], and Stone-Gross et al. addressed detailed issues of taking down a botnet [26]. Honeypots are often used to collect or observe malicious network traffic in early botnet research. However, honeypots usually do not provide outgoing communication. Therefore, they are not suitable for collecting botnet traffic. Nadji et al. proposed a system for the botnet takedowns [27]. Such botnet takedown solution aims at stopping those DNS servers from functioning in the botnet communication. However those C&C servers are able to reorganize using other DNS servers rapidly, since this approach targets deactivating the botnet communication, not removing botnet C&C servers.

Most approaches mentioned so far may be able but not specifically designed to detect Web-based botnet. Since the

majority of botnet owners seldom use their own hosts as the C&C servers, they often use compromised hosts instead. In other words, there must be the HTTP-supported malware on those compromised hosts performing C&C server-like operations. According to the study [28] proposed by Perdisci et al., they addressed the concrete relationship between HTTP-supported malware and Web-based botnets. They also proposed an approach to detect the HTTP-supported malware by using malicious network traces. This approach uses behavioral clustering of these HTTP-supported malware samples by finding their structural similarities among the sequences of HTTP requests. The results of behavioral clustering are used for generating signatures for an IDS system. Since they look into the sequences of HTTP requests, it means that this approach cannot be used for HTTPS-based malware. In addition, this approach still suffers from similar evasions mentioned so far, including injecting noise sequences of HTTP requests and implementing HTTP requests in a time triggering oriented approach.

Many recent botnet studies focus on problems brought by new types of botnets which utilize currently popular Internet applications. For example, some of these papers aim at detecting botnets running on social networks. Kartaltepe et al. proposed a study focusing on the social network-based botnet [11]. Wang et al. proposed an approach [29] to detect the DGA botnet by utilizing social network analysis. Venkatesh and Nadarajan proposed a survey of Stegobot [30], which is a kind of botnets using steganography to mask crucial information in digital images and then transmitting the images over social networks. Ferrara et al. addressed the rise of botnets running on social networks in a recent article [31]. Botnets utilizing IoT and mobile devices were also addressed by several prestigious conferences and projects recently. Bertino and Islam addressed the issues related to botnets and Internet of Things (IoT) security [32]. Project [33], conducted in 2017, is also motivated by Bertino and Islam to analyze the DDoS attack via IoT botnets. Mobile devices suffer from vulnerabilities as well as untrusted firmware and are also vulnerable to botnet owners. Eslahi et al. unveiled MoBots [34], which represent those botnets on mobile devices and networks. MoBots may use some existing services, such as SMS, to communicate with their bot masters. Such issue is critical to the telecommunication industry. Therefore, there is a related patent [35], which has been filed in 2016, disclosing a method for SMS-based botnet detection. Social network-based botnets, IoT botnets, or even mobile device-based botnets are not typical Web-based botnets. To be able to communicate in multiple mechanisms, they are more complicated than traditional Web-based botnets.

6. Conclusion

This study proposes a solution called WBD to detect suspicious Web-based botnets, no matter whether the botnet communication is encrypted or hidden in normal Web pages. We propose three features, two of them related to robot-like repeated contact clustering and one of them related to similar payload size, to detect the existence of botnet Web servers within the network communication. Applying our solutions to 500 GB practical network traces, we found the false positive rate of WBD is only 0.42%.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was funded by the projects of Ministry of Science and Technology of Taiwan under no. 105-2221-E-008-074-MY3 and no. 106-3114-E-002-005.

References

[1] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM on Internet Measurement Conference, IMC 2006*, pp. 41–52, Brazil, October 2006.

[2] D. Dagon, O. Gu, C. P. Lee, and W. Lee, "A taxonomy of botnet structures," in *Proceedings of the 23rd Annual Computer Security Applications Conference, ACSAC 2007*, pp. 325–338, Miami Beach, Fla, USA, December 2007.

[3] G. Gu, J. Zhang, and W. Lee, Botsniffer: Detecting botnet command and control channels in network traffic. In NDSS. The Internet Society, 2008.

[4] FBI. Botnets 101 What They Are and How to Avoid Them, 2013.

[5] A. K. Sood, R. J. Enbody, and R. Bansal, "Dissecting spyeye-understanding the design of third generation botnets," *Computer Networks*, vol. 57, no. 2, pp. 436–450, 2013.

[6] H. Binsalleeh, T. Ormerod, A. Boukhtouta et al., "On the analysis of the Zeus botnet crimeware toolkit," in *Proceedings of the 2010 8th International Conference on Privacy, Security and Trust, PST 2010*, pp. 31–38, Canada, August 2010.

[7] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, "Active botnet probing to identify obscure command and control channels," in *Proceedings of the 25th Annual Computer Conference Security Applications, ACSAC 2009*, pp. 241–253, Honolulu, Hawaii, USA, December 2009.

[8] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using machine learning techniques to identify botnet traffic," in *Proceedings of the 31st Annual IEEE Conference on Local Computer Networks (LCN '06)*, pp. 967–974, Tampa, Fla, USA, November 2006.

[9] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets, HotBots'07*, 7 pages, USENIX Association, Berkeley, Calif, USA, 2007.

[10] T. Cai and F. Zou, "Detecting HTTP botnet with clustering network traffic," in *Proceedings of the 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '12)*, pp. 1–7, September 2012.

[11] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu, "Social network-based botnet command-and-control: emerging threats and countermeasures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 6123, pp. 511–528, 2010.

[12] Cisco. NetFlow Services Solutions Guide, 2007.

[13] Behind Phishing: An Examination of Phisher Modi Operandi.

[14] Global Phishing Survey: Trends and Domain Name Use in 2H2009.

[15] Information and Communication Security Technology Center.

[16] G. K. Venkatesh and R. A. Nadarajan, "HTTP botnet detection using adaptive learning rate multilayer feed-forward neural network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7322, pp. 38–48, 2012.

[17] D. Zhao, I. Traore, B. Sayed et al., "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[18] F.-H. Hsu, C.-S. Wang, C.-H. Hsu, C.-K. Tso, L.-H. Chen, and S.-H. Lin, "Detect fast-flux domains through response time differences," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 10, pp. 1947–1956, 2014.

[19] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th Conference on Security Symposium, SS'08*, pp. 139–154, USENIX Association, Berkeley, Calif, USA, 2008.

[20] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, SS'07*, pp. 12:1–12:16, Berkeley, Calif, USA, 2007.

[21] F. Yu, Y. Xie, and Q. Ke, "SBotMiner: Large scale search bot detection," in *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining, WSDM 2010*, pp. 421–430, USA, February 2010.

[22] A. Zand, G. Vigna, X. Yan, and C. Kruegel, "Extracting probable command and control signatures for detecting botnets," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC 2014*, pp. 1657–1662, Republic of Korea, March 2014.

[23] K. Wang, C. Huang, S. Lin, and Y. Lin, "A fuzzy pattern-based filtering algorithm for botnet detection," *Computer Networks*, vol. 55, no. 15, pp. 3275–3286, 2011.

[24] J. Zhang, R. Perdisci, W. Lee, X. Luo, and U. Sarfraz, "Building a scalable system for stealthy P2P-botnet detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 1, pp. 27–38, 2014.

[25] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, "A Taxonomy of botnet behavior, detection, and defense," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 898–924, 2014.

[26] B. Stone-Gross, M. Cova, L. Cavallaro et al., "Your botnet is my botnet: Analysis of a botnet takeover," in *Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09*, pp. 635–647, New York, NY, USA, November 2009.

[27] Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee, "Beheading hydras: Performing effective botnet takedowns," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013*, pp. 121–132, Germany, November 2013.

[28] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI'10*, 26 pages, Berkeley, Calif, USA, 2010.

[29] T.-S. Wang, C.-S. Lin, and H.-T. Lin, "DGA botnet detection utilizing social network analysis," in *Proceedings of the 2016 IEEE International Symposium on Computer, Consumer and Control, IS3C 2016*, pp. 333–336, China, July 2016.

[30] N. Venkatachalam and R. Anitha, "A multi-feature approach to detect Stegobot: a covert multimedia social network botnet," *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 6079–6096, 2017.

[31] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[32] E. Bertino and N. Islam, "Botnets and internet of things security," *The Computer Journal*, vol. 50, no. 2, Article ID 7842850, pp. 76–79, 2017.

[33] R. Hallman, J. Bryan, G. Palavicini, J. Divita, and J. Romero-Mariona, Ioddos the internet of distributed denial of sevice attacks, 2017.

[34] M. Eslahi, R. Salleh, and N. B. Anuar, "MoBots: A new generation of botnets on mobile devices and networks," in *Proceedings of the 2012 IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2012*, pp. 262–266, Malaysia, December 2012.

[35] C. Adams, Sms botnet detection on mobile devices, May 24 2016. US Patent 9, 351, 167.

Hindawi
Security and Communication Networks
Volume 2017, Article ID 4184196, 10 pages
https://doi.org/10.1155/2017/4184196

Research Article

Network Intrusion Detection through Stacking Dilated Convolutional Autoencoders

Yang Yu, Jun Long, and Zhiping Cai

College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China

Correspondence should be addressed to Zhiping Cai; zpcai@nudt.edu.cn

Received 28 July 2017; Accepted 22 October 2017; Published 16 November 2017

Academic Editor: Wojciech Mazurczyk

Network intrusion detection is one of the most important parts for cyber security to protect computer systems against malicious attacks. With the emergence of numerous sophisticated and new attacks, however, network intrusion detection techniques are facing several significant challenges. The overall objective of this study is to learn useful feature representations automatically and efficiently from large amounts of unlabeled raw network traffic data by using deep learning approaches. We propose a novel network intrusion model by stacking dilated convolutional autoencoders and evaluate our method on two new intrusion detection datasets. Several experiments were carried out to check the effectiveness of our approach. The comparative experimental results demonstrate that the proposed model can achieve considerably high performance which meets the demand of high accuracy and adaptability of network intrusion detection systems (NIDSs). It is quite potential and promising to apply our model in the large-scale and real-world network environments.

1. Introduction

Network intrusion detection techniques are not trivial for cyber security to defend against malicious and suspicious activities [1, 2]. Anomaly-based network intrusion detection systems (ANIDSs) play a critical role in reacting and protecting against an increasing number of damaging threats and attacks. Furthermore, monitoring and analyzing malware traffic behaviors are especially essential tasks for network anomaly detection.

Unfortunately, network intrusion detection techniques are still facing several enormous challenges and problems to detect anomalies effectively [3]. First, with the continual increase of the number and the variety of sophisticated threats and attacks, network intrusion detection systems (NIDSs) produce high false positives or false alarms. Second, classical machine learning approaches used in network intrusion detection have several challenges, such as the paucity of labeled training data and variability of network traffics [4]. These challenges lead to the difficulties to apply conventional machine learning methods in the large-scale and real-world network environments. Additionally, traditional hand-engineered features are neither readily available nor flexible and adaptive for the emerging complex attacks.

Meanwhile, as computer hardware, such as GPUs, owns increasingly computing capabilities, deep learning techniques achieve incredibly impressive results in several research areas. Convolutional neural networks (CNNs) specially have obtained remarkable performance in the field of computer vision, such as object recognition and image classification. The most powerful part of deep learning techniques is learning feature hierarchies from large amounts of unlabeled data. Therefore, deep learning techniques are quite promising to be applied in the network intrusion detection field.

Recently, various deep learning approaches have been applied to the network intrusion detection area, such as restricted Boltzmann machines (RBMs), deep belief networks (DBNs), stacked autoencoders (SAEs), and supervised learning with convolutional neural networks (CNNs). The existing work about the application of deep learning approaches for network intrusion detection is twofold. For one thing, deep learning techniques are utilized to learn or extract valuable features automatically from raw data, which is called feature extraction. These learned features are then fed into classifiers to further complete classification tasks. For another, specific features are firstly extracted according to domain expert

knowledge. Deep learning algorithms mainly play roles of classifiers which take hand-crafted features as input data.

However, there are several problems or limitations with these studies. To begin with, obtaining large amounts of labeled network data and hand-crafted features is pretty costly, let alone other existing problems of customized features, as mentioned previously. In practice, though, getting lots of unlabeled raw network traffic data with little labeled data is relatively easy. Also, the training process of some deep learning methods, such as DBNs and SAEs, consists of unsupervised pre-training [5] and supervised fine-tuning. In this case, large amounts of unlabeled data and little labeled data are, respectively, used in the two training stages. The obvious disadvantage of these fully connected networks is having large number of training parameters because of full connection of units between adjacent layers. As a result, the number of neural network layers is limited, and training process may be very slow. Instead, CNNs reduce the number of parameters through strategies of sparse connectivity and shared weights, but CNNs for supervised learning need labeled data as input. The original motivation of this research is to propose a suitable and effective deep learning approach to bridge the gap unsupervised feature learning and the advantages of CNNs for ANIDSs.

This research aims to construct a novel network intrusion detection model which combines the strengths of unsupervised feature learning and CNNs to extract or learn critical features automatically from large volumes of raw network packets. In this paper, we propose a network intrusion detection model by stacking dilated convolutional autoencoders which actually combines the concepts of self-taught learning [6] and representation learning [7]. The model is evaluated through different classification tasks with malware traffic data which come from diverse malwares. We also observe and discuss the effects of different hyperparameters on evaluation results and find optimal parameter values for the proposed model. The experimental results demonstrate that our model can get remarkable performance and meet the demand of high accuracy and adaptability of NIDSs.

The remainder of this paper is organized as follows. Section 2 introduces recent related work on the application of deep learning approaches for network intrusion detection. Section 3 describes the proposed model and dataset construction. Section 4 presents experimental results and analysis. Section 5 further analyzes the results and discusses limitations of our method and future work. Section 6 concludes the paper.

2. Related Work

In this section, we review a little recent research that is relevant to our work. Deep learning methods used in unsupervised feature learning tasks for network intrusion detection mainly include restricted Boltzmann machines (RBMs), autoencoders, deep belief networks (DBNs), stacked autoencoders, and various variants of these methods.

In most existing studies, the unsupervised deep learning methods for intrusion detection play roles of unsupervised feature extractors to learn abstract features from hand-crafted features. The abstract features are then taken as input data of a classifier, such as the softmax classifier. For example, Fiore et al. (2013) [8] proposed discriminative RBM (DRBM) to learn abstract features from customized features that did not contain information of packet payloads. These learned features were then fed into softmax classifier for the binary classification, namely, normal and anomalous classification. Javaid et al. (2015) [9] used sparse autoencoder and softmax regression on NSL-KDD dataset [10] which is a revised version of the KDD dataset [11]. Similarly, Erfani et al. (2016) [12] combined DBNs and a linear one-class SVM for anomaly detection on various benchmark datasets. Many other studies follow this kind of pattern, namely, taking hand-engineering features which need specific domain knowledge as the input of unsupervised or supervised deep learning methods.

However, very few attempts have been made to use deep learning techniques to learn useful features or good representations from raw network traffics. Wang (2015) [13] used stacked autoencoders for traffic identification from raw network traffic data and achieved impressive high performance. In addition, Wang et al. (2017) [14] transformed 728-dimensional raw traffic data into images and used CNNs with supervised feature learning for malware or botnet traffic classification. Compared with their work, our method can learn feature representations from massive unlabeled data which contain more diverse attack types. Besides, the features our method learned include temporal information, while they only use spatial features of network traffics. Besides, we do not visualize raw traffic data because there exist huge differences on the application and the structure between network traffic data and images. Thus, it would be unsatisfactory to simply visualize the network traffic data to simulate image classification tasks using CNNs regardless of their semantic meanings.

In this paper, we propose a deep learning approach, called dilated convolutional autoencoders (DCAEs), for the network intrusion detection model, which combines the advantages of stacked autoencoders and CNNs. In essence, the model can automatically learn essential features from large-scale and more various unlabeled raw network traffic data consisting of real-world traffics from botnets, web-based malwares, exploits, APTs (Advanced Persistent Threats), scans, and normal traffics.

3. Methodology

In this section, we first introduce our model for network intrusion detection from an overall perspective. Subsequently, the deep learning method used in the model is described in detail. Finally, we briefly present construction of our datasets.

3.1. Model Description. An overview of the training process of the DCAEs-based model is illustrated in Figure 1. Raw network traffic data in the libpcap file format (.pcap) are transformed into numeric vectors through a data preprocessing module. These numeric vectors are training samples of

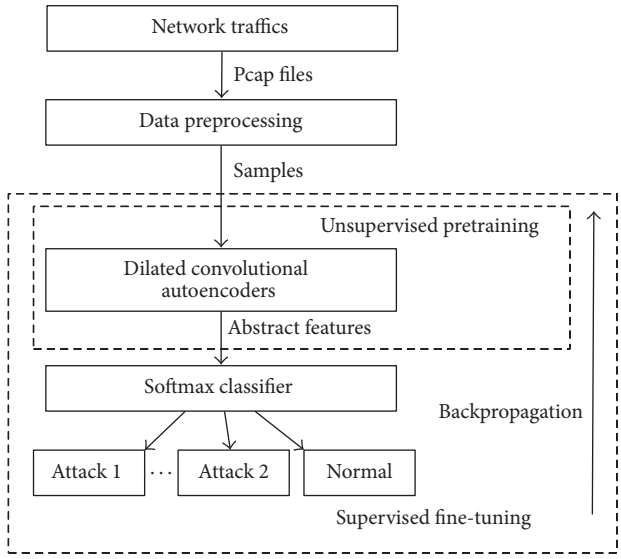


Figure 1: Overview of the training process based on the DCAEs method.

our datasets. The training process is divided into unsupervised pretraining and supervised fine-tuning. In the unsupervised pretraining process, dilated convolutional autoencoders (DCAEs) learn a hierarchy of feature representations from large volumes of unlabeled samples. Afterward, the representations learned from unlabeled data are enhanced by the supervised fine-tuning using the backpropagation algorithm and few labeled samples. Specifically, the neural network is trained as a traditional convolutional neural network without pooling layers using dilated convolutions, as shown in Figure 2. The sample is transformed into the shape of an image for applying dilated convolutions on it. There is only one convolutional layer from a convolutional autoencoder in Figure 2. The early-stopping strategy is used to prevent from over-fitting. In addition, softmax classifier is applied to perform classification task using the abstract features. The use of diverse raw network traffics and unsupervised pretraining makes our model more adaptive and flexible.

3.2. Dilated Convolutional Autoencoders. The architecture of dilated convolutional autoencoders (DCAEs) is pretty similar to classical autoencoders [15]. Figure 3 shows the structure of a dilated convolutional autoencoder. The input is mapped into feature maps through an activation function:

$$\mathbf{h}^k = f\left(\mathbf{x} * \mathbf{W}^k + \mathbf{b}^k\right), \tag{1}$$

where $\mathbf{x}$ is the two-dimensional input reshaped from a numeric vector, $\mathbf{W}^k$ and $\mathbf{b}^k$ are, respectively, a weight matrix and a bias corresponding to the kth feature map h^k. The activation function $f(\cdot)$ in our model is ReLU (Rectified Linear Unit) activation function (i.e., $f(x) = (0, x)$). The symbol $*$ denotes dilated convolution [16] operator. Subsequently, the feature maps of hidden layer are mapped into the reconstruction through a transposed convolution [17]:

$$\widetilde{\mathbf{x}} = f\left(\sum_{k \in H} \mathbf{h}^k * \widetilde{\mathbf{W}}^k + \widetilde{\mathbf{b}}\right), \tag{2}$$

where $\widetilde{\mathbf{x}}$ has the same shape of the input $\mathbf{x}$ and H is a collection of feature maps. The initial values of weight matrix $\mathbf{W}$ and $\widetilde{\mathbf{W}}$ are the same [18]. The learning objective of the dilated convolutional autoencoder is to reduce the difference between the input $\mathbf{x}$ and the reconstruction $\widetilde{\mathbf{x}}$. The cost function in our model is the mean squared error (MSE):

$$L\left(\mathbf{x}, \widetilde{\mathbf{x}}\right) = \frac{1}{n}\sum_{i}^{n}\left(\mathbf{x}_i - \widetilde{\mathbf{x}}_i\right)^2. \tag{3}$$

The DCAEs can be used to construct a deep neural network by stacking multiple DCAEs, which is similar with SAEs [15]. Specifically, the input of the next DCAE is the hidden-layer output of the previous DCAE. The process of stacking DCAEs is greedy layer-wise unsupervised training [19]. One of the advantages of dilated convolutions is that dilated convolution can have a wider range of receptive fields without losing information. This advantage makes it more suitable for text processing.

In sum, the advantages of dilated convolutional autoencoders are as follows. First, the application of dilated convolutions enlarges the layers' receptive fields to learn more global features. Compared with max-pooling, dilated convolutions can protect the input data from information loss. Second, the pretraining process of the DCAEs does not need labeled data, which is more useful in practical applications. Finally, the DCAEs have lesser parameters than fully connected neural networks, such as SAEs. Therefore, the DCAEs are more effective and time-saving than other unsupervised deep learning methods.

3.3. Dataset. In this paper, we performed three kinds of classification tasks on two types of datasets. Table 1 shows the sample distribution for the CTU-UNB dataset and the Contagio-CTU-UNB dataset. The first dataset called the CTU-UNB dataset consists of various botnet traffics from CTU-13 dataset [20] and normal traffics from the UNB ISCX IDS 2012 dataset [21, 22]. The second dataset called the Contagio-CTU-UNB dataset consists of six types of network traffic data. The normal and botnet traffics come from parts of the CTU-UNB dataset. The web-based malware traffics are from the threatglass website [23]. The traffics of exploits, APTs, and Scans come from parts of contagio or deepend research [24]. The general data preprocessing steps are shown in Figure 4. The specific elaboration about data preprocessing and CTU-UNB dataset is presented in our previous work [25].

4. Experimental Results and Performance Analysis

In this section, we first briefly introduce classification metrics for performance analysis. Experimental setup and environments are then described. Finally, we present and analyze some important experimental results.

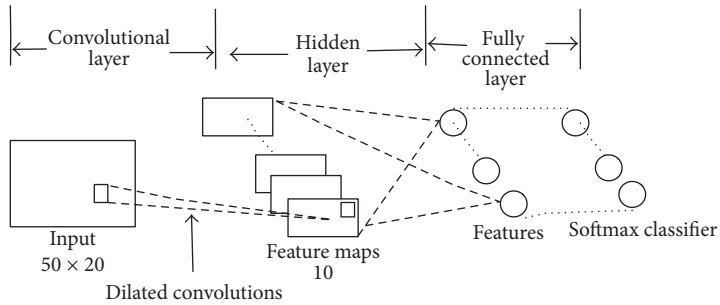


Figure 2: Neural network structure of the fine-tuning process.

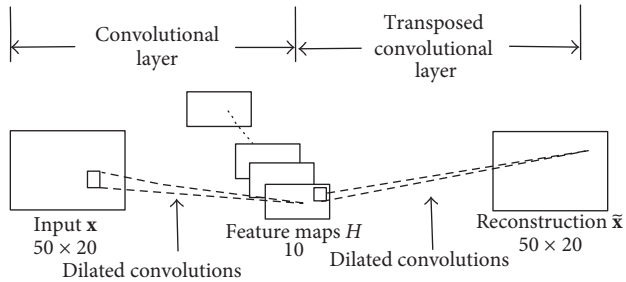


Figure 3: Structure of a dilated convolutional autoencoder.

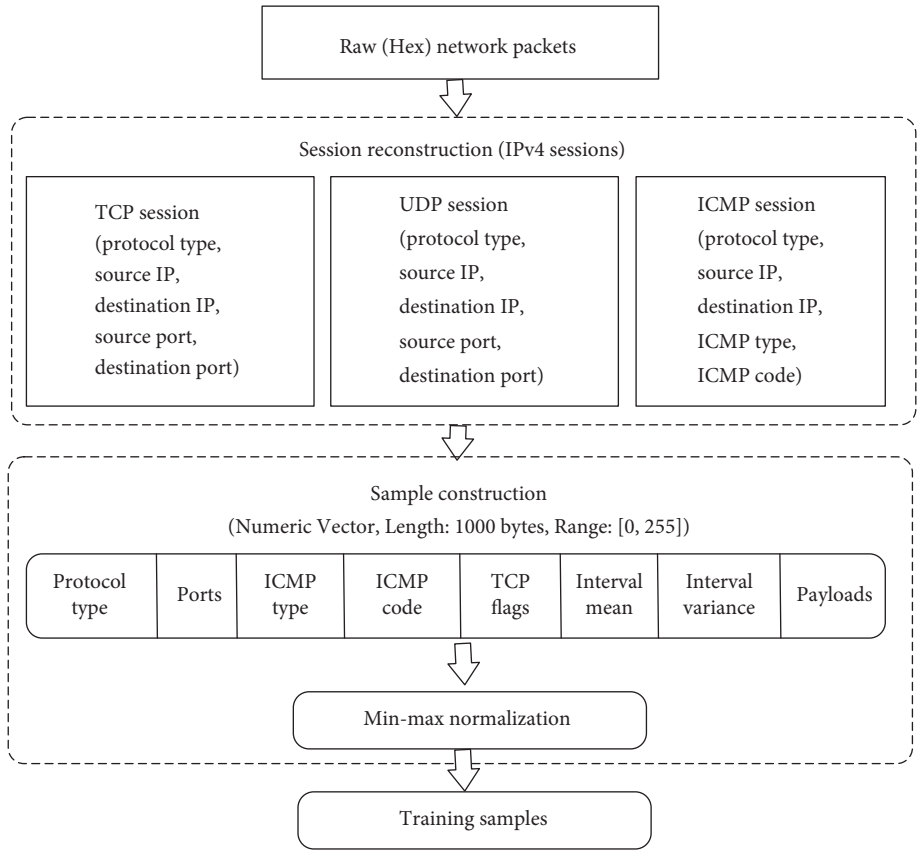


Figure 4: Data preprocessing steps.

Table 1: Sample distribution of the CTU-UNB dataset and the Contagio-CTU-UNB dataset.

CTU-UNB dataset					Contagio-CTU-UNB dataset				
Traffic type		Training set	Validation set	Test set	Total	Traffic type	Training set	Validation set	Test set	Total
Normal		41480	8123	8174	57867	Normal	10812	2053	2061	14926
Bot	Neris	8039	1567	1565	11171	Botnet	10681	2107	2047	14835
	Rbot	6073	1228	1221	8522	Web-based malware	10327	1928	1929	14184
	Virut	18914	3680	3767	26361					
	Menti	217	40	43	300	Exploit	7325	1396	1418	10139
	Sogou	34	5	5	44	APT	1439	242	276	1957
	Murlo	2013	364	358	2735					
	NSIS.ay	4395	903	867	6165	Scan	6466	1274	1269	9009
	Total	39685	7877	7826	55298					
Total		81165	16000	16000	113165	Total	47050	9000	9000	65050

4.1. Classification Metrics and Experimental Setup. Six evaluation metrics were utilized for performance analysis of our experiments. The six metrics are accuracy (AC), precision (P), recall (R), f-measure (F), the receiver operating characteristic (ROC) curve, and the confusion matrix, respectively. Specifically, these metrics are related to four classification functions, namely, true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In other words, TP and TN separately measure the number of attacks and normal classification correctly. FP and FN represent that the proportion of attacks and normal data that is incorrectly identified, respectively. These four functions can be calculated from the confusion matrix $\mathbf{C}_{i,j}$ whose elements of leading diagonal are the number correctly predicted samples. For example, the element c_{ij} $(i \neq j)$ describes the number of samples which are incorrectly identified as the class j but actually from the class i. The ROC curve illustrates the performance of the classification model through TP and FP. The larger value of the area under the ROC curve (AUC) means the higher TP and the lower FP. In addition, accuracy (i.e., AC = (TP + TN)/(TP + TN + FP + FN)) presents the percentage of correctly classified samples over all samples. Precision (i.e., P = TP/(TP + FP)) and recall (i.e., R = TP/(TP + FN)), respectively, describe the percentage of correctly identified attacks versus all predicted attacks and all actual existing attacks. F-measure (i.e., $F = 2PR/(P + R)$) is the weighted average of precision and recall.

The experimental environments are shown in Table 2. We use Theano [26] to build our neural network model. 80% of the performance of a laptop GPU was used to accelerate calculation speed. The learning rates of pretraining and fine-tuning process were, respectively, 0.001 and 0.1. The minibatch size was 100, and the pretraining epochs were 15.

4.2. Experimental Results and Analysis. We performed three types of classification tasks on the Contagio-CTU-UNB dataset and the CTU-UNB dataset to evaluate the performance of the proposed model. The classification tasks include 6-class classification using the Contagio-CTU-UNB dataset and 2-class and 8-class classification using the CTU-UNB dataset. Specifically, the 6-class classification involves normal data and five kinds of malware traffic data (i.e., botnet, web-based malware, exploit, APT, and scan). The 2-class classification contains normal data and botnet data from the CTU-UNB dataset. The 8-class classification consists of normal data and seven types of botnet data shown in Table 1.

Table 2: Experimental environments.

Name	Configuration
OS	Ubuntu 16.04.1 LTS 64-bit
CPU	Intel Core i5-4200M 2.50 GHz
RAM	8 G
GPU	GeForce GT 740M
Cuda	7.5

Table 3: Accuracy of three kinds of classification tasks.

Metric	6-class			2-class	8-class
	SAE	DBN	DCAE	DCAE	DCAE
Accuracy (%)	96.96	96.94	**98.98**	**99.59**	**98.40**

First, we evaluated the proposed model on three types of the classification tasks. In the 6-class classification task, we also compared our method with other deep learning approaches which have the similar structure and the training process. Furthermore, we evaluated the generalization ability of the proposed model through utilizing the well-trained model of the 2-class classification to detect unknown attacks which are not involved in the training set. Meantime, some important parameters of our model were analyzed.

Table 3 shows the accuracy of three types of the classification tasks. The optimal parameters from experimental tests were used in the DCAE method, as shown in Figure 6 and Table 10. The number of hidden layers of the SAE and the DBN was the same with the DCAE. In the 6-classification task, the DCAE obtained the highest accuracy in comparison with other deep learning methods. Meanwhile, our method performed best and achieved 99.59% accuracy rate in the binary classification. The result of 8-class classification was slightly worse than 6-class classification. The precision, recall,

Table 4: Precision, recall, and F-measure of various deep learning methods.

Traffic type	SAE			DBN			DCAE		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Normal	96.09	96.60	96.35	96.12	96.12	96.12	98.59	98.64	98.62
Botnet	96.38	97.41	96.90	96.38	97.61	96.99	98.69	99.66	99.17
Web-based Malware	97.87	97.56	97.72	97.72	97.56	97.64	99.12	98.96	99.04
Exploit	96.71	95.28	95.99	96.38	95.84	96.11	98.94	98.73	98.84
APT	95.56	93.48	94.51	97.33	92.39	94.80	99.24	94.93	97.04
Scan	98.50	98.50	98.50	98.58	98.50	98.54	99.84	99.61	99.72
Total	96.96	96.96	96.95	96.95	96.94	96.94	**98.98**	**98.98**	**98.98**

Table 5: Precision, recall, and F-measure of the 8-class classification task.

Traffic type	P (%)	R (%)	F (%)
Normal	99.77	99.57	99.67
Neris	91.63	97.19	94.33
Rbot	97.66	95.74	96.69
Virut	98.76	97.48	98.12
Menti	97.50	90.70	93.98
Sogou	80.00	80.00	80.00
Murlo	96.10	96.37	96.23
NSIS.ay	99.07	98.62	98.84
Average	98.44	98.40	98.41

Table 6: Confusion matrix of the 6-class classification task.

	Normal	Botnet	Web-based malware	Exploit	APT	Scan
Normal	**2033**	10	10	8	0	0
Botnet	3	**2040**	1	1	0	2
Web-based malware	16	4	**1909**	0	0	0
Exploit	8	9	0	**1400**	1	0
APT	2	3	5	4	**262**	0
Scan	0	1	1	2	1	**1264**

and f-measure of 6-class classification are presented in Table 4. The DCAE method also outperformed the compared deep learning methods and achieved the same average value with three metrics after approximation. Table 5 shows the precision, recall, and f-measure of the 8-class classification task. Combining data shown in Table 1, we found that the data size could affect classification results to some degree. Specifically, the class which has fewer data corresponds to the worse performance, which would further affect average values. However, we found that our method still performed well even when there were only few training data. This conclusion can be drawn from Tables 6 and 7.

Table 6 presents the confusion matrix of the 6-class classification task using our method. The leading diagonal shows the number of correctly classified samples of the test set. The botnets and the scans have fewer samples identified incorrectly. Similarly, Table 7 shows the confusion matrix of the 8-class classification task using our method. Though the data size of the menti and the sogou was the smallest, they still achieved relatively good performance. The ROC curves of three types of classification tasks are shown in Figure 5.

The AUC value of binary classification was equal to 1.00, which suggested that our method performed extremely well in the binary classification. Meanwhile, The AUC value of 6-class and 8-class classification was 0.99. It is almost certain that our method produces high true positives and low false alarms.

Additionally, after finishing the 2-class classification task which detected botnet data, the well-trained model was saved in order to evaluate the generalization ability of the proposed model. A new test set containing attack types of the Contagio-CTU-UNB dataset was then constructed to evaluate the generalization of features learned from the CTU-UNB dataset. As shown in Table 8, there are a total of 16000 samples in the new test set which contain two types of traffic data, namely, normal and attack. We chose normal and parts of botnet data from the test set of the CTU-UNB dataset, because the normal data and botnet data of the Contagio-CTU-UNB dataset may be included in the training set of the CTU-UNB dataset. Besides, four types of complex attacks (i.e., web-based malware, exploit, APT, and scan) come from the test set of the Contagio-CTU-UNB dataset. Specifically, we firstly saved the model well trained in the 2-class classification task using the CTU-UNB dataset. The new test set data was then

Table 7: Confusion matrix of the 8-class classification task.

	Normal	Neris	Rbot	Virut	Menti	Sogou	Murlo	NSIS.ay
Normal	**8139**	16	3	10	0	0	6	0
Neris	11	**1521**	7	23	0	0	1	2
Rbot	1	37	**1169**	8	1	1	2	2
Virut	5	74	13	**3672**	0	0	2	1
Menti	0	1	0	0	**39**	0	2	1
Sogou	0	0	0	0	0	**4**	1	0
Murlo	0	3	5	3	0	0	**345**	2
NSIS.ay	2	8	0	2	0	0	0	**855**

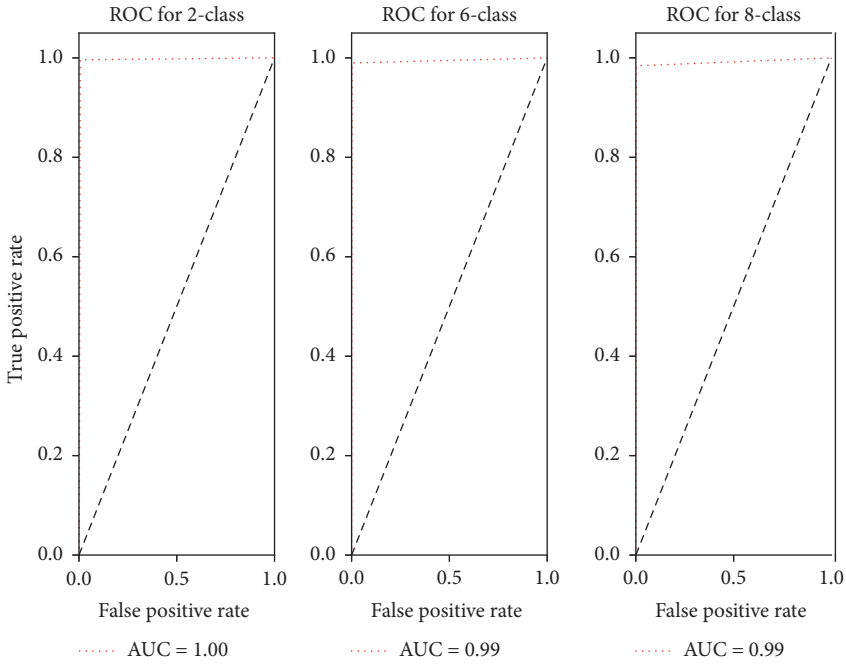


Figure 5: ROC curves for various classification tasks.

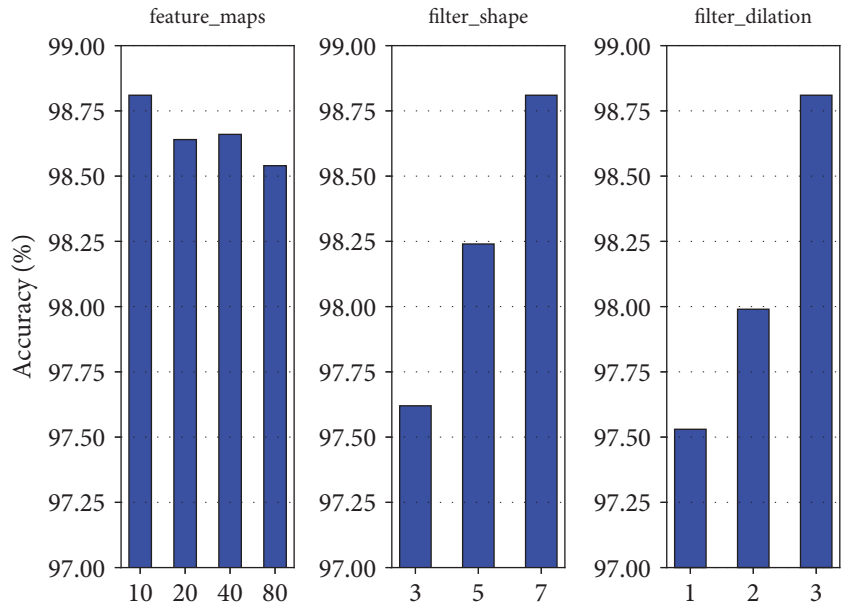


Figure 6: Accuracy comparison of various parameter settings.

Table 8: Sample distribution of new test set for evaluating generalization ability.

Traffic type	Attack					Normal
	Botnet	Web-based malware	Exploit	APT	Scan	
Data size	2934	1929	1418	276	1269	8174
	7826					

Table 9: Precision, recall, and F-measure of the generalization evaluation.

Traffic type	P (%)	R (%)	F (%)
Normal	82.25	99.56	90.08
Attack	99.41	77.56	87.14
Average	90.65	88.80	88.64

evaluated on the saved model. In other words, the proposed model was used to detect some unknown traffic data or various unknown attacks in this generalization evaluation task. The detecting accuracy is 88.8% for the new test set. The precision, recall, and F-measure of the generalization evaluation task are shown in Table 9. The evaluation results suggest that the proposed model could learn some valuable and general features through botnet data to detect unknown attacks.

In the rest of this section, the parameter comparison of experiments used controlling variable method on the 6-class classification task. The controlling variable method means the variable or parameter studied has different value while other parameters are set to optimal values. Figure 6 illustrates accuracy tendency of different parameter settings. The results show that the size of dilation (filter_dilation) and filter (filter_shape) used in the process of convolutional operation has a greater effect on the accuracy rate. Our model got the best performance when the size of dilation and the filter was, respectively, set to 7 and 3. In addition, the number of feature maps (feature_maps) of the convolutional layer has a small effect on the experimental performance. The optimal value of feature maps was set to 10.

Table 10 shows comparative experiments on different numbers of convolutional layers and two types of activation functions used in convolutional autoencoders. The unit number of fully connected layer (i.e., full_units) was set to the same with its input units because we found that the model could get better performance. We chose two types of activation functions for convolutional autoencoders, namely, the sigmoid function (i.e., $f(x) = (1 + e^{-x})^{-1}$) and the ReLU function. These two kinds of activation functions are separately the representatives of saturating nonlinearities and nonsaturating nonlinearities [27]. The cost function corresponding to the sigmoid function was the cross-entropy loss (i.e., $L(\mathbf{x}, \tilde{\mathbf{x}}) = -\sum_{j=1}^{n}[\mathbf{x}_j \log \tilde{\mathbf{x}}_j + (1 - \mathbf{x}_j)\log(1 - \tilde{\mathbf{x}}_j)]$). The experimental results show that the different number of convolutional layers and diverse activation functions do not have a significant effect on the performance of our model. However, they do have a dramatic effect on the run time. The ReLU function is more time-saving compared with the sigmoid function. The number of convolutional layers has a little effect on the run time when the activation function is the ReLU function. Therefore, the ReLU function is more effective than the sigmoid function from the perspective of whole performance.

In addition, we also presented evaluation results on adding a max-pooling layer and various activation functions of the fully connected layer, as shown in Table 11. The parameter settings were the same with the first row of the parameter setting column in Table 10. The experiment on the fully connected layer sets the activation function of convolutional autoencoders to the ReLU function. We added a max-pooling layer before the fully connected layer. We found that the max-pooling operation could not improve the accuracy of our model. But it reduced run time in the scenario of using the sigmoid function and increased run time in the scenario of using the ReLU function. Therefore, it is not suitable and wise to add a max-pooling layer when the activation function of convolutional autoencoders is the ReLU function. Moreover, the run time reached the minimum when the activation function of the fully connected layer was ReLU though the accuracy was not the highest.

We also found that increasing or reducing the length of input vector had little change for the accuracy rate, as shown in Table 12. It may be because the header information and the former parts of traffic payloads are more valuable and useful to identify various attacks. The activation function of the full connected layer was ReLU function. The numeric vector is first transformed into a two-dimensional matrix (such as [10, 20]). We achieved the best performance when the length of input vector was 1000. While the training time of model which has the shorter length of input vectors does not reduce as expected, there could be a tradeoff between training time and accuracy rate. Besides, the number of the units of the fully connected layer also has an optimal range of value. Specifically, the number of the units was better, close to the input of the fully connected lay but not more than the length of sample vectors. That maybe relates to the representation capability of neural networks.

5. Discussion

As stated previously, the purpose of this study was to learn significant features automatically and efficiently from unlabeled raw network traffic data using deep learning techniques. In general, this study shows that the proposed model can achieve high performance by learning feature representations from large volumes of unlabeled training samples. The training samples based on the session are constructed from parts of header and payload information of network packets. We found that the proposed deep learning method obtained quite good results on various classification tasks. These results provide insights into the feature representations learned from raw traffics. It is certain that these feature representations are effective to identify various malicious network traffics and generate low false alarms. The experimental results also show that more layers of convolutional autoencoders fail to

Table 10: Evaluation results on different numbers of convolutional layers and two types of activation functions.

Number of layers	Activation function	Accuracy (%)	Run time (minutes)	Parameter setting
1	Sigmoid	98.83	102.83	filter_dilation = [(3, 3)], filter_shape = [(7, 7)], feature_maps = [10], full_units = 640.
	ReLU	**98.98**	57.09	
2	Sigmoid	98.78	217.25	filter_dilation = [(3, 3), (2, 2)], filter_shape = [(5, 7), (3, 5)], feature_maps = [10, 10], full_units = 960.
	ReLU	98.61	60.64	
3	Sigmoid	98.80	178.26	filter_dilation = [(3, 3), (2, 2), (1, 1)], filter_shape = [(5, 7), (3, 5), (2, 2)], feature_maps = [10, 10, 10], full_units = 690.
	ReLU	98.51	48.65	

Table 11: Evaluation results on adding a max-pooling layer and various activation functions of the fully connected layer.

Layer	Activation function	Accuracy (%)	Run time (minutes)
Max-pooling	Sigmoid	98.42	59.91
	ReLU	97.97	90.41
Fully connected	Tanh	98.97	57.09
	Sigmoid	98.82	101.83
	ReLU	98.62	**8.77**

Table 12: Evaluation results on the different lengths of input vectors.

Length (bytes)	Accuracy (%)	Run time (minutes)	Parameter setting
200 [10, 20]	98.40	19.46	filter_dilation = [(2, 3)], filter_shape = [(3, 7)], full_units = 120.
400 [20, 20]	98.43	37.90	filter_dilation = [(3, 3)], filter_shape = [(4, 7)], full_units = 220.
500 [25, 20]	97.93	13.87	filter_dilation = [(3, 3)], filter_shape = [(5, 7)], full_units = 260.
600 [30, 20]	97.99	19.08	filter_dilation = [(3, 3)], filter_shape = [(6, 7)], full_units = 300.
800 [40, 20]	97.91	16.38	filter_dilation = [(3, 3)], filter_shape = [(7, 7)], full_units = 440.
1000 [50, 20]	**98.62**	**8.77**	filter_dilation = [(3, 3)], filter_shape = [(7, 7)], full_units = 640.
1500 [50, 30]	98.28	52.93	filter_dilation = [(4, 3)], filter_shape = [(10, 9)], full_units = 840.

significantly enhance performance as expected. It is possibly because the the number of hidden units of the first convolutional layer is enough to learn useful feature representations. In addition, diverse activation functions have a great effect on training time. The results suggest that the ReLU function is a good choice for the proposed model to reduce run time. Moreover, an additional max-pooling operation is not necessary for our proposed model compared to traditional convolutional autoencoders.

The limitation of our proposed model is that the training process takes a comparatively long time. However, it can be solved by cross-GPU parallelization technique [27] which is widely used in the deep learning field. In future work, we will implement an online network intrusion detection system in combination with high-performance computing techniques. Additionally, we would try to add missing data or noise and diverse classifiers to enhance the robustness and performance of our system.

6. Conclusion

In this paper, we proposed a novel network intrusion detection model based on dilated convolutional autoencoders. The proposed deep learning method can automatically learn significant feature representations from large volumes of unlabeled raw traffic data. The Contagio-CTU-UNB dataset and the CTU-UNB dataset are created from various malware traffic data. Three kinds of classification tasks are performed to evaluate the performance of the proposed model. We also compared our deep learning method with other similar approaches. The effects of various important hyperparameters are further analyzed. The experimental results show

the superiority of our model by effectively detecting complex attacks from lots of unlabeled data. The remarkable performance we achieved meet requirements of large-scale and real-world network environments by combining high-performance computing techniques.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants nos. 61379145 and 61105050.

References

[1] Z. Cai, Z. Wang, K. Zheng, and J. Cao, "A Distributed TCAM coprocessor architecture for integrated longest prefix matching, policy filtering, and content filtering," *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 417–427, 2013.

[2] K. Zheng, Z. Cai, X. Zhang, Z. Wang, and B. Yang, "Algorithms to speedup pattern matching for network intrusion detection systems," *Computer Communications*, vol. 62, pp. 47–58, 2015.

[3] Y. Yu, J. Long, F. Liu, and Z. Cai, "Machine learning combining with visualization for intrusion detection: a survey," in *Modeling Decisions for Artificial Intelligence*, vol. 9880 of *Lecture Notes in Comput. Sci.*, pp. 239–249, Springer, Cham, Germany, 2016.

[4] R. Sommer and V. Paxson, "Outside the closed world: on using machine learning for network intrusion detection," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 305–316, IEEE Computer Society, 2010.

[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[6] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pp. 759–766, ACM, June 2007.

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[8] U. Fiore, F. Palmieri, A. Castiglione, and A. de Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, 2013.

[9] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, New York, NY, USA, December 2015.

[10] S. Revathi and A. Malathi, *A Detailed Analysis on nsl-kdd Dataset Using Various Machine Learning Techniques for Intrusion Detection*, 2013.

[11] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2nd IEEE Symposium on Computational Intelligence for Security and Defence Applications*, pp. 1–6, IEEE, July 2009.

[12] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[13] Z. Wang, *The Applications of Deep Learning on Traffic Identification*, BlackHat, 2015.

[14] W. Wang, M. Zhu, X. Zeng et al., "Malware traffic classification using convolutional neural network for representation learning," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 712–717, Da Nang, Vietnam, January 2017.

[15] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009.

[16] Y. Fisher and V. Koltun, *Multi-scale context aggregation by dilated convolutions*, 2015, https://arxiv.org/abs/1511.07122.

[17] V. Dumoulin and V. Francesco, *A guide to convolution arithmetic for deep learning*, https://arxiv.org/abs/1603.07285.

[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[19] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the 20th Annual Conference on Neural Information Processing Systems (NIPS '06)*, pp. 153–160, Cambridge, Mass, USA, December 2006.

[20] "The ctu-13 dataset," https://stratosphereips.org/category/dataset.html.

[21] "The unb iscx 2012 intrusion detection evaluation dataset," http://www.unb.ca/cic/research/datasets/ids.html.

[22] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[23] "Threatglass by barracuda labs," http://threatglass.com/.

[24] "Contagio malware dump," http://contagiodump.blogspot.kr/2013/08/deepend-research-list-of-malware-pcaps.html.

[25] Y. Yu, J. Long, and Z. Cai, "Session-Based Network Intrusion Detection Using a Deep Learning Architecture," in *Modeling Decisions for Artificial Intelligence*, vol. 10571 of *Lecture Notes in Computer Science*, pp. 144–155, Springer International Publishing, Cham, Germany, 2017.

[26] B. James, B. Olivier, F. Bastien et al., "Theano: A cpu and gpu math compiler in python," in *Proceedings of the 9th Python in Science Conference*, pp. 1–7, 2010.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

Hindawi
Security and Communication Networks
Volume 2017, Article ID 1723658, 9 pages
https://doi.org/10.1155/2017/1723658

Research Article

Remotely Exploiting AT Command Attacks on ZigBee Networks

Ivan Vaccari, Enrico Cambiaso, and Maurizio Aiello

National Research Council (CNR), IEIIT Institute, Via De Marini 6, 16149 Genova, Italy

Correspondence should be addressed to Enrico Cambiaso; enrico.cambiaso@ieiit.cnr.it

Received 17 July 2017; Accepted 9 October 2017; Published 31 October 2017

Academic Editor: Wojciech Mazurczyk

Internet of Things networks represent an emerging phenomenon bringing connectivity to common sensors. Due to the limited capabilities and to the sensitive nature of the devices, security assumes a crucial and primary role. In this paper, we report an innovative and extremely dangerous threat targeting IoT networks. The attack is based on Remote AT Commands exploitation, providing a malicious user with the possibility of reconfiguring or disconnecting IoT sensors from the network. We present the proposed attack and evaluate its efficiency by executing tests on a real IoT network. Results demonstrate how the threat can be successfully executed and how it is able to focus on the targeted nodes, without affecting other nodes of the network.

1. Introduction

The Internet is today adopted for a wide range of different purposes and by several kinds of entities, ranging from banking and stock market sectors adoption to personal use for social networking and web surfing. The Internet is indeed populated by billions of devices of different nature. In the last years, we have seen the appearance of several categories of always connected devices: smartphones, tablets, smartwatches, and healthcare devices are today only a few kinds of components of the global network. We are today experiencing a new emerging trend related to the evolution of common "analog" sensors, making them connect to each other, creating a "parallel" network based on machine-to-machine communications.

In this context, the term Internet of Things (IoT) represents a general concept relative to the ability of common sensors to collect data from the real world and hence share the retrieved information across a network, by communicating with other connected devices. IoT networks are today deployed for different purposes. The most known and adopted ones are the home automation/domotics and the industrial (Industry 4.0) contexts: while in a domotic context IoT networks are used to provide connectivity to common and security devices (light bulbs, internal cameras, fire sensors, etc.), in an Industry 4.0 scenario, IoT is used to monitor, control, inform, and automate production processes. In order to communicate on the network, IoT devices support different communication protocols, such as Industrial Ethernet [1], Wi-Fi [2], ZigBee [3], and Z-Wave [4].

Our research, presented in this paper, investigates security aspects of IoT networks. We focus on ZigBee, a communication protocol ensuring low power consumption and characterized by low data transmission rates. During our study, we found important security issues related to a ZigBee based system and, potentially, to other IoT protocols. We identified the possibility of sending Remote AT Commands, where AT means "attention," to a connected sensor, in order to reconfigure the device, for instance, by making it join a different malicious network and hence forward captured data to the enemy. We evaluate the possibility of perpetrating a successful attack by setting up a network laboratory composed of XBee devices (XBee is one of the most adopted ZigBee radio modules in the Do-It-Yourself (DIY) scenario [5]). We describe the exposure to Remote AT Commands threats by focusing on evaluating efficiency and performance characteristics of this innovative attack.

The focus of our work is on the proposal of an innovative cyberattack. This may result in an unconventional and not needed activity. Nevertheless, especially in the research field, it is well known that offence research is as needed as defense research, in order to properly master a field and better prepare to counter cyber-criminal activities [6, 7].

The remaining of the paper focuses on the presentation of the innovative discovered threat and it is structured as follows. Section 2 reports the structure of the ZigBee protocol. Section 3 reports related work on the topic, while Section 4 reports our contribution on Remote AT Command exploitation. Then, Section 5 exposes the adopted testbed and obtained results by executing the attack on a controlled environment. Finally, Section 7 concludes the paper and reports possible extensions of the work.

2. The ZigBee Protocol

ZigBee is a wireless standard introduced by the ZigBee Alliance in 2004. It is based on the IEEE 802.15.4 standard, used in the Wireless Personal Area Networks (WPAN) context [8]. ZigBee is designed for embedded systems, often characterized by extremely low power consumption and low-rate transfers requirements [9]. The protocol is indeed able to minimize battery replacement frequency (up to 2 years) and to provide a communication rate up to 250 kbps, for a coverage radius up to 1000 meters. Figure 1 depicts the ZigBee stack protocol.

The physical layer of the IEEE 802.15.4 standard manages modulation and demodulation operations. Particularly, ZigBee supports three different frequencies:

(i) 2.4 GHz with support to 16 different channels, providing a maximum communication rate of 250 kbps (used worldwide)

(ii) 868 MHz with support to 1 channel and a maximum data rate of 20 kbps (used in Europe)

(iii) 915 MHz with support to 10 channels and 40 kbps of communication rate (used in US)

Since they work on the same frequency, in case of 2.4 GHz adoption, there may be interferences with existent Wi-Fi networks [10].

The MAC layer, also implemented in IEEE 802.15.4, takes care of ensuring a reliable and secure communication, by implementing a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to control access to the physical level [11].

The network layer of the ZigBee protocol implements instead network topologies, new devices management, and security handling. Particularly, ZigBee supports three different network topologies:

(i) A star topology, where each node communicates with a central node

(ii) A tree topology, where central nodes of different networks are connected with a bus network

(iii) A mesh topology, where all the nodes are connected to each other

Mesh networks are the most interesting ones: in this case, ZigBee implements ad hoc routing algorithms to automatically rearrange communications if a node of the network is disconnected [12].

The Application Framework layer represents the user interface and it is composed of three main components:

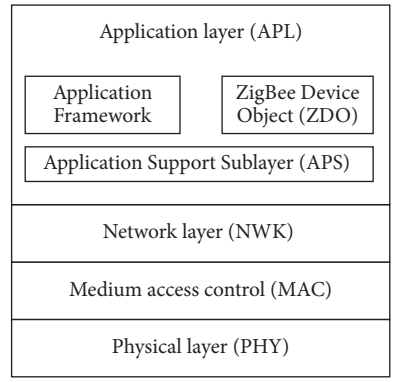


Figure 1: The ZigBee stack protocol.

(i) Application Support Sublayer (APS), providing an interface between network and application layers; moreover, it controls and manages data sent and received by other protocol layers to ensure proper packet transmission and encryption

(ii) ZigBee Device Objects (ZDO), an application object responsible for the initialization procedures of the APS and the ZigBee network layer to perform discovery of services and new nodes in the network

(iii) Application Framework (AF), an execution environment for "application objects," each ones identified by an endpoint address from 1 to 254 (0 is reserved for ZigBee Device Object (ZDO), 255 for broadcast messages): application objects are usually implemented by different manufacturers. In order to enhance products interoperability, the ZigBee Alliance has published different application profiles. The most common ones are home automation, smart energy, light link, and green power [8]

2.1. ZigBee Node Types. ZigBee supports different kind of devices with different functionalities:

(i) ZigBee end-device (ZED): it represents the sensor, usually in sleep mode most of the time and periodically waking up in order to communicate with the other nodes of the network.

(ii) ZigBee router (ZR): it is an optional node used to route packets on the network.

(iii) ZigBee coordinator (ZC): it is a ZigBee router with gateway functions used to manage the network.

While on the same network it is common to have several different ZED nodes and different routers, a single coordinator is found.

2.2. ZigBee Security. As many other wireless networks, like Wi-Fi [13] or ad hoc wireless sensor network [14, 15], security assumes a crucial role in the ZigBee protocol. The encryption algorithm used in ZigBee is Advanced Encryption Standard (AES) with a 128-bit key. Such algorithm, considered

extremely secure and reliable, guarantees confidentiality and authenticity on wireless communications [16].

ZigBee provides two different security profiles [17]: Standard Security, the basic security profile, rarely adopted because of its exposure to attacks, and High Security, mostly used since it guarantees greater security during communications. Particularly, while considering the standard security profile, the network key is shared in clear text (unencrypted), it is encrypted with the link key in case of high security profile adoption. The link key is one of the security keys adopted by ZigBee:

(i) Master key is usually hardcoded on the device or shared out-of-band. It is needed in order to retrieve the other keys but it is never directly sent on the network.

(ii) Network key is a key shared by all the devices connected to the same network. It is generated by the Trust Center and it can be sent on the network as plain text or in encrypted form, depending on the adopted security profile.

(iii) Link key is a key generated using the master key and adopted for communications between two different devices on the same network.

In a ZigBee network, if communication is unencrypted, an attacker may access all information of the network and may even sniff/capture exchanged packets. Otherwise, if communication is encrypted, a malicious user may only perform attacks that do not require access to the network, such as denial of service or jamming, since it is very difficult to retrieve the ZigBee adopted network key and hence decrypt exchanged packets.

3. Related Work

Because of the wide adoption of the ZigBee protocol, one of the most important concerns is related to ZigBee based networks protection. Many security experts have studied the protocol and identified several threats able to target such systems. In this context, an important contribution is provided by Wright, the creator of Killerbee, a framework including a set of tools able to exploit the ZigBee protocol analyzing network traffic and processing the recovered packets [18]. Although such software is extremely dangerous, its specific hardware requirements (such as Atmel AVR USB Stick or TelosB mote models) limit the execution of properly equipped attackers. Thanks to Killerbee, it is possible to execute several attacks against a ZigBee network: for instance, it is possible to retrieve the network key when sent as clear text. Such retrieval requires the attacker to be located in proximity of the network nodes, in order to sniff the key exchange. Killerbee also includes other threats such as replay [19] or manipulation/injection [20].

Other attacks focus on denial of service (DoS) activities, executed in order to disconnect a node from the network. DoS attacks are popular on the Internet [7] and they are extending to last generation fields such as mobile [21], SDN [22], and IoT [23]. Considering such kind of threats perpetrated against a ZigBee system, several attacks target battery powered sensors in order to reduce the lifetime of the device. In this context, the ZigBee end-device sabotage attack [24] is executed by keeping sensors active when a broadcast message is sent every time the device wakes up from the sleep status. In this way, a sensor under attack is forced to reply the malicious user, hence delaying the next sleep and discharging the batteries quickly. A similar threat, the ghost attack proposed by Shila, reduces the lifetime of the targeted device by sending several crafted bogus messages to the victim [25]. Vidgren et al. demonstrate instead how it is possible to discharge the batteries of a sensor if the attacker knows the adopted sensor polling rate [26]. Pacheco et al. investigate instead DDoS attacks feasibility against IoT environments [27]. Another DoS attack proposed by Vidgren et al. exploits the ZigBee frame counter. Such counter is commonly used by different network protocols to prevent threats such as replay attacks. Concerning ZigBee frame counter exploitation, a malicious user could send a parameter containing the maximum allowed frame counter value (sized 4 bytes), hence forcing the victim to set the counter to the received value. If Message Integrity Check [28] is not implemented by the victim, each packet received after the malicious one will be discarded by the victim since it will present a lower frame counter [26].

Another attack, known as same-nonce attack [29], can be carried out only if the Trust Center, a device providing reliability during the key exchange stage, provides the same-nonce encrypt with the same network key for two consecutive times. In a ZigBee network, coordinators have role of Trust Center. In this scenario, an attacker may retrieve part of the plain text simply calculating the XOR between the two sniffed packets. Although this situation rarely happens, it is possible to force this behavior by causing a power failure, for example, by discharging batteries of a Trust Center. In this case, Trust Center resets the nonce to its default value and it is possible to send a packet with the same nonce [26].

Considering other threats, ZigBee networks are also vulnerable to attacks known as Sinkhole and Wormhole, proposed by Karnain and Zakaria [30]. During a Sinkhole attack, a malicious node attracts the network packets with the aim of creating confusion in the routing phase. Instead, during a Wormhole attack, the malicious user receives packets at one point in the network and then replays these packets in other areas to interfere with all network functionality. Also, while Krivtsova et al. propose the broadcast storm attack clogging the network by sending numerous broadcast packets [31], Yang et al. introduced two attacks against ZigBee, known as Absolute Slot Number (ASN) and time synchronization tree attack. Considering that the time is split into different slots/ASNs of fixed length, during an ASN attack, since current ASN value is sent during communication, the legitimate nodes may get an incorrect ASN value from the attacker that sends on the network a broadcast message with a wrong ASN value. In this way, a node would not be able to communicate on the network since the wrong ASN packet would lead to a communication interruption. In time synchronization tree attack, the malicious user may send bogus DAG Information Objects (DIO) packets [32] to the neighbors with the aim of

desynchronizing their connections with the network [32]. A Sybil attack, proposed by Lee et al., is launched by an attacker that acquires multiple identities on the network. The aim of this attack is to convince the other devices that the malicious node is a legitimate node. In this way, a malicious node may, for example, access all services of the network or identify itself as a ZigBee router [33]. Another type of attack is performed if the enemy can physically access a ZigBee device. Indeed, the malicious user may perform a firmware dump in order to extrapolate the network key stored/hardcoded in the device [34].

Other attacks focus instead on specific version of ZigBee. In this context, a particular version of the ZigBee protocol, called ZigBee Light Link, used, for instance, by Philips Hue bulbs [35], has been exploited different times. Indeed, Gent found that the adopted ZigBee network key can be retrieved if an attacker can sniff the reinitialization process accomplished by the bulbs after a reset and if he knows the ZigBee Light Link master key [36]. Another attack on ZigBee Light Link is proposed by Ronen et al., creating a worm that automatically infects adjacent bulbs, building a custom infected firmware, and being deployed as a fake OTA update [37].

Many works focus instead on ZigBee protection. For instance, a solution to detect the Sybil attack is proposed by Marian and Mircea, presenting an interesting protection system using RSSI derived metrics to detect a Sybil attack by computing the location of a node and then classifying it as malicious or not [38]. Al Baalbaki et al. introduced instead an Anomaly Behavior Analysis System (ABAS) for the ZigBee protocol based on network traffic analysis. After detection is triggered, ABAS can classify the attack as known or unknown using information like packets origin or destination [39]. Another protection algorithm proposed by Jokar and Leung and known as HANIDPS implements machine learning based intrusion detection and prevention system. HANIDPS analyzes the network traffic and compares it with a normal in order to detect a running threat [40]. A similar approach may analyze energy consumption [41] to identify running attacks. Cui et al. proposed instead a fuzzing method based on finite state machines. A fuzzy test is implemented by injecting different testing cases into the system in order to detect vulnerabilities [42]. A defense against impulsive noise is proposed by Jia and Meng, implementing a system using a noise filtering processing activity in two steps: while during the first step an estimate of the noise is computed, in the second one, a noise cancellation is accomplished, in order to state if the estimate is suspect or not [43].

During our research work, we have studied security aspects of ZigBee based IoT networks by initially studying the protocol, thus analyzing the major threats affecting it, hence studying possible protection systems and approaches. During our study, we have discovered the proposed threat and, to the best of our knowledge, we noticed that a vulnerability analysis focused on AT Command exploitation is still missing. Nevertheless, this vulnerability should be considered extremely innovative and particularly dangerous, since it allows malicious users to retrieve/forward sensitive information or manipulate nodes functionality. Our work focuses on the proposal of the innovative Remote AT Command attack, explained in the next section, by illustrating the proposed threat and evaluating its efficiency.

4. Remote Control Exploitability

In order to properly investigate ZigBee security, we have studied the protocol and analyzed communication flows, considering the different types of packets supported by ZigBee. While, at first, we focused on packets containing data sent from the coordinator to the end-device, later, we have also analyzed other packets exchanged in the network. In this context, we found that, at the MAC layer, it is possible to send Remote AT Commands. By working at such lower layer, received packets are not processed at the application layer; hence, it may not be possible to access the packet content to avoid interpretation, except from the device manufacturer.

During our research work, we identified a particular vulnerability affecting AT Commands capabilities implemented in IoT sensor networks. Our work focuses on the exploitation of such weakness. AT Commands are specific packets, historically adopted by old generation modems to interface with the device, today used by radio modules such as XBee [44], ESP8266 (more information is available at http://esatjournals.net/ijret/2017v06/i01/IJRET20170601027.pdf), or ETRX3 [45] to configure parameters like connection type, network identifier, device name on the network, or destination address for a communication. AT Commands are today supported by many devices of different nature, providing different functionalities and hence commands. For instance, modules that provide connectivity support AT Command packets for network parameters configuration, while other modules may use these packets to alter light intensity of light bulbs.

For our research, as previously mentioned, we focused on XBee modules. Such modules, widely adopted around the world, especially in DIY contexts, implement two different AT Command packets, related to request and response operations, respectively. Concerning XBee modules, these packets can be sent remotely: we talk in this case of Remote AT Commands. Such packets belong to the (IEEE 802.15.4) MAC layer and they are interpreted by the (XBee) module automatically. Therefore, by being such interpretation demanded to the device firmware, and being such firmware provided by the manufacturer, Digi International, it is not possible to avoid implicit Remote AT Commands interpretation. In order to execute the proposed attack, the AT Command functionality of XBee has to be exploited. XBee supports several AT Command packets (more information is available at https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf). Particularly, for our aim, we have used ATID commands to target sensors (in general, other commands/approaches may be used for different purposes: e.g., to make the sensor join a different network, to forward (sensitive) data to a different malicious receiver, and to disable data encryption). ATID is used by XBee modules to set the network identifier. During the proposed Remote AT Command attack, the malicious user sends an ATID packet with a bogus identifier in order to make it join a different (inexistent, in our case) network.

In order to maliciously exploit Remote AT Command, it is assumed that the attacker is connected to the network of the target. In this case, the enemy may, for instance, disconnect an end-device from the ZigBee network and make it join a different (malicious) network and hence forward potentially sensitive data to third malicious parties. Given the nature of IoT end-devices, often associated with a critical data and operations, it may be obvious how a Remote AT Command attack represents a serious threat for the entire infrastructure.

5. Testbed

In this section, we report information about the tests we have conducted in order to validate the success and the efficiency of a Remote AT Command attack. In order to accomplish the tests, we have built a ZigBee test network, depicted in Figure 2.

The network is composed of a single ZigBee coordinator, two end-devices representing common sensors on the network, and a malicious user/node connected to the ZigBee network. As can be deduced from the figure, the attacker sends Remote AT Command packets only to one sensor and not to each device on the network. This implementation allows us to monitor the effects of the attack on the two sensors, hence evaluating the possibility of carrying out a successful attack without affecting targeted nodes. Indeed, we expect that, during a Remote AT Command attack, only the targeted sensor is affected by the threat, while other nodes keep working correctly (unless their behavior depends on the targeted sensor).

Considering the described scenario, we will now detail at first adopted hardware, hence reporting information about testbed configuration, finally exposing the obtained results.

5.1. Testbed Configuration. Different devices have been used to create ZigBee network to implement AT Command attack. For our aim, network components are composed as reported in the following:

(i) *Coordinator*, composed of a Raspberry Pi 3 equipped with an XBee USB Board and an XBee Series 2

(ii) *Targeted sensor*, composed of an Arduino UNO R3, equipped with an XBee Shield and an XBee Series 2

(iii) *Not targeted sensor*, composed of an Arduino UNO R3, equipped with an XBee Shield and an XBee Series 2

(iv) *Attacker*, composed of a Raspberry Pi 3 equipped with an XBee USB Board and an XBee Series 2

As the reader may notice, end-devices/sensors share the same hardware. Hence, our evaluation allows us to identify the efficiency of the attack on the targeted node, and simultaneously the possibility of avoiding side effects on other nodes (this is not possible, e.g., for jamming attacks).

Moreover, since XBee series 2 modules have low computational capacity, we adopted Arduino microcontrollers to generate and elaborate information, hence using XBee modules only for network communications. In order to guarantee the sleep status of end-devices, a PIN Hibernation has been implemented [46] by connecting the 7th PIN of the XBee Shield to an Arduino digital PIN. In order to implement PIN Hibernation, *D7* value, a PIN used to send and receive serial data, has been disabled (through XCTU XBee programming software). In order to test this vulnerability, the innovative attack is implemented and tested with this configuration: the attack was performed on only one sensor because by monitoring the network traffic is possible to verify the efficiency of this threat.

5.2. Network Nodes Implementation. Every 35 seconds, sensors are programmed to send a packet to the coordinator. Each packet contains a random generated number. After the message has been sent, the sensor device enters in sleep mode in order to reduce power consumption. Since the content of the message is not meaningful to us, the "random number" solution allows us to generate data to be transmitted on the network without requiring environmental sensors.

Figures 3 and 4 monitor the network traffic of the various XBee modules. Although we stated that a single packet is sent every 35 seconds, sending is relative to application layer packets, while the capture is relative to the entire ZigBee network stack. Although such capture includes additional (lower layers) packets (including, for instance, wake-up commands containing network node information and synchronization packets), it is representative of the network behavior of the sensor (e.g., we can see that, after the attack, no packets are sent by the victim node), while a capture focused on application layer packets/messages would produce single peaks missing useful information.

Data is received by the coordinator and shown to the user through an HTML based graphical interface, also reporting if sensors are correctly communicating with the coordinator. This environment is representative of a wide range of network types. For instance, sensors installed on a specific area/farm/company could be monitored through a similar approach, or industry machines and fire prevention systems may be part of a network system similar to the proposed one.

6. Results

Network traffic was analyzed from an external ZigBee device capturing data on the same channel used by the targeted network. From sniffed traffic, we are able to extrapolate communication flows of single hosts of the network.

Figures 3(a) and 3(b) report the network traffic flow of both targeted and not targeted sensors during a running attack. Traffic was monitored for 120 seconds and it is split into two phases: during the first 50 seconds, the attacker acts in a "passive" way, by scanning the ZigBee spectrum in order to identify the devices connected to the network and define the target. Instead, on the second "active" phase, the attacker sends Remote AT Command packets to the targeted sensor in order to perform the attack. Particularly, for our aim, the passive behavior is not intended as a "listen only" behavior. Instead, during this phase, the attacker does not send any malicious packet on the network. Therefore, we expect that

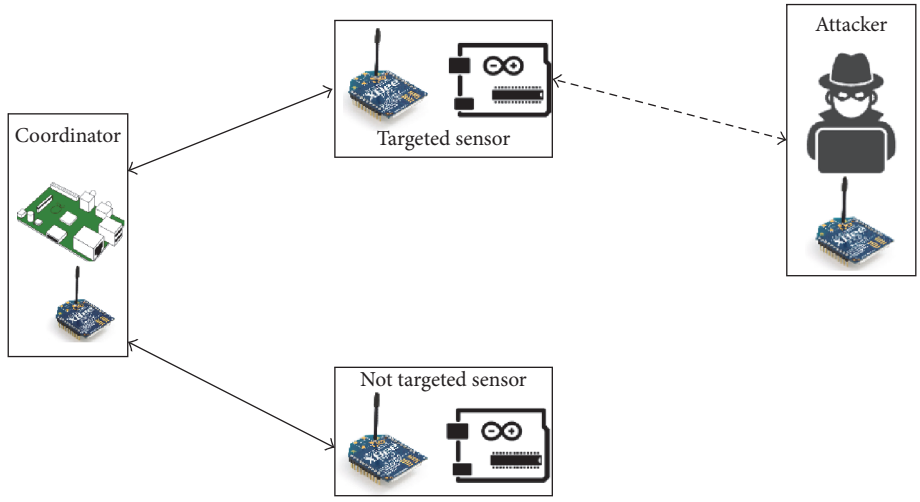
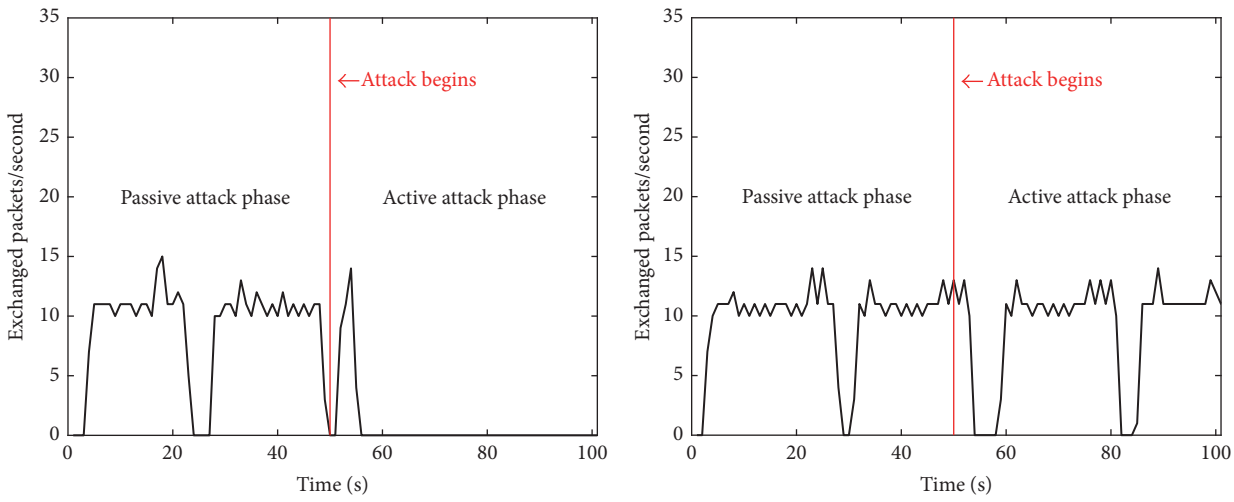


Figure 2: Test network.



(a) Traffic generated by targeted sensor

(b) Traffic generated by not targeted sensor

Figure 3: Network traffic captured during attack execution.

detecting a malicious behavior during the passive phase is particularly difficult.

Figure 3(a) reports the traffic flow of the targeted sensor. By analyzing the graph, it is possible to notice that while the sensor is correctly working during the passive phase, a few seconds after the attack is (actively) performed, the device is disconnected from the network and its communication with the coordinator is interrupted. Therefore, the attack is successful on the targeted sensor.

Instead, Figure 3(b) reports the status of the nontargeted sensor during the attack. Particularly, it is possible to notice that the connection is maintained alive for the entire considered period. Indeed, since this sensor is not directly targeted by the attacker, Remote AT Commands are not received/interpreted; hence, the network parameters of the sensor are not altered by the attacker and communication capability of the sensor is maintained and not even disturbed. This represents an important characteristic of the proposed threat, since it is able to only affect the targeted device, by making the attack not directly visible to the other sensors/devices. Such stealth behavior makes the attack more difficult to detect. Moreover, considering that device communication interruption may be related to external factors (e.g., battery drain, wireless noise, and malfunctioning device), the proposed attack should be considered a serious threat.

Figure 4 shows instead the captured attack traffic during the considered period.

By analyzing the passive phase of the attack, as previously mentioned, the enemy performs a scan of the network in order to identify each device connected to the network and choose the targeted device. Instead in the active attack phase, Remote AT Command packets are sent to the targeted sensor with the aim of disconnecting it from the network (by reconfiguring it). If we analyze the attack traffic flow, it

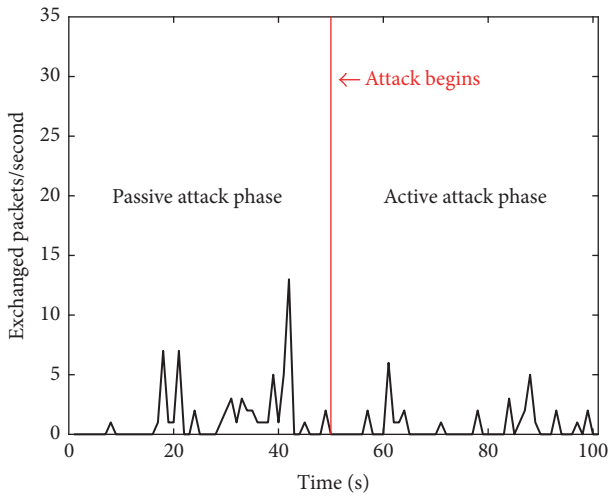


Figure 4: Traffic generated by attacker sensor.

is very difficult to distinguish a passive (hence, potentially legitimate) behavior from an active (malicious) one. Hence, detection of a running threat may require packet inspection or data flows interpretation (not easy to accomplish in case of encrypted traffic).

Although our testbed focuses on two network sensors/devices, the proposed Remote AT Commands attack is particularly scalable, due to the (minimum) requirements for the attacker (a single packet is sent to the victim to reconfigure it). Particularly, the time required to send such packet is minimal, so in case of multiple targeted sensors, the attack success is guaranteed. Of course, in case of extremely large amounts of targeted sensors, the effectiveness of the attack depends on the scan time: the attack is successful if the minimum "sleep time" of each sensor is larger than the average time required to target all the sensors.

7. Conclusion and Future Work

The proposed paper is focused on Internet of Things (IoT) environments security, by analyzing the possibility of carrying out a successful attack against a targeted node/sensor/device. During our work, we found a novel vulnerability affecting IoT devices: by exploiting a particular type of packet, Remote AT Command, it is possible to remotely reconfigure/program network nodes as the attacker wishes, hence compromising data communication security of the network.

By focusing on the ZigBee (wireless) protocol, we have described and implemented the proposed attack with the aim of interrupting the communication capabilities of a targeted device of the network. For our tests, we targeted XBee modules [44], able to communicate through the ZigBee protocol. Results show that the attack is successful and it is able to target a single node without affecting the other nodes of the network. Moreover, since the number of packets sent by the attacker is minimum, it is not easy to detect a running attack, without doing deep packet inspection. The attack is therefore particularly dangerous, since it may compromise the security of an IoT network with minimum effort for the attacker. By comparing the effects of the proposed attack to other network based threats, they can be assimilated to denial of service, man-in-the-middle (traffic sniffing), or traffic redirection activities, in function of the strategy adopted by the attacker.

Future work on the topic may concern additional tests of the attack in large scale networks composed of different nodes, in order to identify the limits of the threat, in function of the sleeping/polling times adopted by the nodes. Considering instead the design of defense systems, additional extensions of the work may be directed to the implementation of efficient protection techniques able to defend an IoT system from a Remote AT Command attack. Since detection of a running threat may not be immediate, in order to protect a remote device from a Remote AT Command attack, it may be preferred to directly work on the (potentially vulnerable) nodes. In this context, three different approaches can be adopted, working at different levels:

(i) Firmware level: creation of a modified version of the firmware, implementing Remote AT Commands filtering or allowing AT Commands elaboration at the application layer

(ii) Device configuration level: providing to the user the ability to configure a device with disabled support to Remote AT Commands

(iii) External level: demanding protection capabilities to an external application program.

Each approach provides an efficient solution to protect the device. Nevertheless, some approaches may not be adopted (e.g., device configuration, if not available). Suggested implementations provide a possible protection for this innovative threat.

The first proposed solution (firmware level protection) requires a device firmware upgrade to allow total AT Command packet management, such as the ability to process only packets received by the coordinator or secure devices. Such solution would provide the user with the possibility of configuring the device in order to avoid implicit Remote AT Commands interpretation.

Since modifying a firmware may not be easy, and the source code must be open source, it is suggested to have simpler but equally effective solutions. The second solution (device configuration level protection) implements the ability to disable Remote AT Command support of the module, by implementing a specific setting able to disable automatic Remote AT Command interpretation (e.g., packets discard). In this way, the proposed threat would be ineffective.

The last proposed solution (external level protection) is the most interesting; the main purpose is to implement protection logics on the Arduino device by implementing a function at application layer. The aim of the function is to verify if the XBee module may be communicating on the network. In this case, just before the sensor is ready to communicate on the network, an internal check is accomplished.

Although the mentioned approaches may protect IoT modules and network sensors from this innovative attack, by ensuring data transmission security, their design implementation and evaluation are on the scope of further work on the topic.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work has been supported by the following research projects: (i) My Health-My Data (MHMD) project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant agreement no. 732907; (ii) Advanced Networked Agents for Security and Trust Assessment in CPS/IoT Architectures (ANASTACIA) project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant agreement no. 731558.

References

[1] J. S. Rinaldi and P. S. Marshall, "Industrial ethernet," *ISA Press Release*, Article ID 945541, p. 04, 2004.

[2] L. Li, H. Xiaoguang, C. Ke, and H. Ketai, "The applications of WiFi-based Wireless Sensor Network in Internet of Things and Smart Grid," in *Proceedings of the 6th IEEE Conference on Industrial Electronics and Applications (ICIEA '11)*, pp. 789–793, IEEE, Beijing, China, June 2011.

[3] C. Fan, Z. Wen, F. Wang, and Y. Wu, "A middleware of internet of things(iot) based on Zigbee and RFID," in *Proceedings of the IET International Conference on Communication Technology and Application, ICCTA 2011*, pp. 732–736, October 2011.

[4] Z. W. Alliance, "The internet of things is powered by z-wave.(2016)," *Z-Wave Alliance*, vol. 28, p. 2016, 2016.

[5] R. Faludi, *Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing*, O'Reilly Media, Inc., 2010.

[6] P. Farina, E. Cambiaso, G. Papaleo, and M. Aiello, "Are mobile botnets a possible threat? the case of SlowBot Net," *Computers & Security*, vol. 58, pp. 268–283, 2016.

[7] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello, "Slow DoS attacks: definition and categorisation," *International Journal of Trust Management in Computing and Communications*, vol. 1, no. 3/4, p. 300, 2013.

[8] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, "Study on ZigBee technology," in *Proceedings of the 3rd International Conference on Electronics Computer Technology (ICECT '11)*, pp. 297–301, IEEE, Kanyakumari, India, April 2011.

[9] A. Dementyev, S. Hodges, S. Taylor, and J. Smith, "Power consumption analysis of Bluetooth Low Energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario," in *Proceedings of the IEEE International Wireless Symposium (IWS '13)*, April 2013.

[10] M. A. Sarijari, M. S. Abdullah, A. Lo, and R. A. Rashid, "Experimental studies of the ZigBee frequency agility mechanism in home area networks," in *Proceedings of the 39th Annual IEEE Conference on Local Computer Networks, LCN 2014*, pp. 711–717, September 2014.

[11] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: a survey on the state of the art and the 802.15.4 and ZigBee standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[12] J. Li, X. Zhu, N. Tang, and J. Sui, "Study on ZigBee network architecture and routing algorithm," in *Proceedings of the 2nd International Conference on Signal Processing Systems (ICSPS '10)*, vol. 2, pp. V2-389–V2-393, Dalian, China, July 2010.

[13] S. Gold, "Cracking wireless networks," *Network Security*, vol. 2011, no. 11, pp. 14–18, 2011.

[14] E. Cayirci and C. Rong, *Security in Wireless Ad Hoc and Sensor Networks*, John Wiley & Sons, 2008.

[15] L. Caviglione and F. Davoli, "Peer-to-peer middleware for bandwidth allocation in sensor networks," *IEEE Communications Letters*, vol. 9, no. 3, pp. 285–287, 2005.

[16] L. Caviglione, M. Gaggero, E. Cambiaso, and M. Aiello, "Measuring the Energy Consumption of Cyber Security," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 58–63, 2017.

[17] G. Dini and M. Tiloca, "Considerations on security in ZigBee networks," in *Proceedings of the 2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, SUTC 2010, 2010 IEEE International Workshop on Ubiquitous and Mobile Computing, UMC 2010*, pp. 58–65, June 2010.

[18] J. Wright, *Killerbee: practical zigbee exploitation framework*, 2009.

[19] B. Stelte and G. D. Rodosek, "Thwarting attacks on ZigBee - Removal of the KillerBee stinger," in *Proceedings of the 2013 9th International Conference on Network and Service Management, CNSM 2013 and its three collocated Workshops - ICQT 2013, SVM 2013 and SETM 2013*, pp. 219–226, October 2013.

[20] A. Biswas, A. Alkhalid, T. Kunz, and C.-H. Lung, "A lightweight defence against the Packet in Packet attack in ZigBee networks," in *Proceedings of the 2012 IFIP Wireless Days, WD 2012*, November 2012.

[21] R. H. Jhaveri, S. J. Patel, and D. C. Jinwala, "DoS attacks in mobile ad hoc networks: A survey," in *Proceedings of the 2012 2nd International Conference on Advanced Computing and Communication Technologies, ACCT 2012*, pp. 535–541, January 2012.

[22] R. Kandoi and M. Antikainen, "Denial-of-service attacks in OpenFlow SDN networks," in *Proceedings of the 14th IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, pp. 1322–1326, May 2015.

[23] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the internet of things: a review," in *Proceedings of the International Conference on Computer Science and Electronics Engineering (ICCSEE '12)*, pp. 648–651, Hangzhou, China, March 2012.

[24] O. Olawumi, K. Haataja, M. Asikainen, N. Vidgren, and P. Toivanen, "Three practical attacks against ZigBee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned," in *Proceedings of the 2014 14th International Conference on Hybrid Intelligent Systems, HIS 2014*, pp. 199–206, December 2014.

[25] D. M. Shila, "Ghost-in-the-wireless: Energy depletion attack on zigbee".

[26] N. Vidgren, K. Haataja, J. L. Patiño-Andres, J. J. Ramírez-Sanchis, and P. Toivanen, "Security threats in ZigBee-enabled systems: Vulnerability evaluation, practical experiments, countermeasures, and lessons learned," in *Proceedings of the 46th Annual Hawaii International Conference on System Sciences, HICSS 2013*, pp. 5132–5138, January 2013.

[27] L. A. B. Pacheco, J. J. C. Gondim, P. A. S. Barreto, and E. Alchieri, "Evaluation of distributed denial of service threat in the internet of things," in *Proceedings of the 15th IEEE International Symposium on Network Computing and Applications, NCA 2016*, pp. 89–92, November 2016.

[28] H. Li, Z. Jia, and X. Xue, "Application and analysis of ZigBee security services specification," in *Proceedings of the 2nd International Conference on Networks Security, Wireless Communications and Trusted Computing, NSWCTC 2010*, pp. 494–497, April 2010.

[29] N. Sastry and D. Wagner, "Security considerations for IEEE 802.15.4 networks," in *Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe '04)*, pp. 32–42, ACM, October 2004.

[30] M. A. B. Karnain and Z. B. Zakaria, "A review on ZigBee security enhancement in smart home environment," in *Proceedings of the 2nd IEEE International Conference on Information Science and Security, ICISS 2015*, December 2015.

[31] I. Krivtsova, I. Lebedev, M. Sukhoparov et al., "Implementing a broadcast storm attack on a mission-critical wireless sensor network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 9674, pp. 297–308, 2016.

[32] W. Yang, Q. Wang, Y. Wan, and J. He, "Security Vulnerabilities and Countermeasures for Time Synchronization in IEEE802.15.4e Networks," in *Proceedings of the 3rd IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2016 and 2nd IEEE International Conference of Scalable and Smart Cloud, SSC 2016*, pp. 102–107, June 2016.

[33] G. Lee, J. Lim, D.-K. Kim, S. Yang, and M. Yoon, "An approach to mitigating sybil attack in wireless networks using ZigBee," in *Proceedings of the 2008 10th International Conference on Advanced Communication Technology*, pp. 1005–1009, February 2008.

[34] L. Jun and Y. Qing, "Take unauthorized control over zigbee devices," 2015, https://media.defcon.org/defcon23/defcon23presentations/defcon-23-li-jun-yang-qing-i-am-a-newbie-yet-i-can-hack-zigbee.pdf.

[35] J. Wang, "Zigbee light link and its applicationss," *IEEE Wireless Communications Magazine*, vol. 20, no. 4, pp. 6-7, 2013.

[36] A. Gent, "A lightbulb worm?" in *Blackhat*, 2016, https://www.blackhat.com/docs/us-16/materials/us-16-oflynn-a-lightbulb-worm-wp.pdf.

[37] E. Ronen, A. Shamir, A. Weingarten, and C. OFlynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, pp. 195–212, San Jose, CA, USA, May 2017.

[38] S. Marian and P. Mircea, "Sybil attack type detection in Wireless Sensor networks based on received signal strength indicator detection scheme," in *Proceedings of the 10th Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics, SACI 2015*, pp. 121–124, May 2015.

[39] B. Al Baalbaki, J. Pacheco, C. Tunc, S. Hariri, and Y. Al-Nashif, "Anomaly Behavior Analysis System for ZigBee in smart buildings," in *Proceedings of the 12th IEEE/ACS International Conference of Computer Systems and Applications, AICCSA 2015*, November 2015.

[40] P. Jokar and V. Leung, "Intrusion Detection and Prevention for ZigBee-Based Home Area Networks in Smart Grids," *IEEE Transactions on Smart Grid*, pp. 1-1.

[41] L. Caviglione, M. Gaggero, J.-F. Lalande, W. Mazurczyk, and M. Urbański, "Seeing the unseen: Revealing mobile malware hidden communications via energy consumption and artificial intelligence," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 799–810, 2016.

[42] B. Cui, S. Liang, S. Chen, B. Zhao, and X. Liang, "A novel fuzzing method for Zigbee based on finite state machine," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 762891, 2014.

[43] J. Jia and J. Meng, "A novel approach for impulsive noise mitigation in ZigBee communication system," in *Proceedings of the 2014 Global Information Infrastructure and Networking Symposium, GIIS 2014*, September 2014.

[44] R. Piyare and S. r. Lee, "Performance analysis of xbee zb module based wireless sensor networks," *International Journal of Scientific & Engineering Research*, vol. 4, pp. 1615–1621.

[45] A. T. C. Dictionary, "Etrx2 and etrx3 series zigbee modules at-command dictionary".

[46] P. Manual, "Xbee/xbee-pro rf modules," http://store.express-inc.com/pdf/xa-a.pdf.

Hindawi
Security and Communication Networks
Volume 2017, Article ID 7892182, 19 pages
https://doi.org/10.1155/2017/7892182

Research Article

Predictive Abuse Detection for a PLC Smart Lighting Network Based on Automatically Created Models of Exponential Smoothing

Tomasz Andrysiak, Łukasz Saganowski, and Piotr Kiedrowski

Institute of Telecommunications and Computer Science, Faculty of Telecommunications, Computer Science and Electrical Engineering, University of Technology and Life Sciences in Bydgoszcz (UTP), Ul. Kaliskiego 7, 85-789 Bydgoszcz, Poland

Correspondence should be addressed to Tomasz Andrysiak; andrys@utp.edu.pl

Received 23 July 2017; Accepted 19 September 2017; Published 25 October 2017

Academic Editor: Steffen Wendzel

One of the basic elements of a Smart City is the urban infrastructure management system, in particular, systems of intelligent street lighting control. However, for their reliable operation, they require special care for the safety of their critical communication infrastructure. This article presents solutions for the detection of different kinds of abuses in network traffic of Smart Lighting infrastructure, realized by Power Line Communication technology. Both the structure of the examined Smart Lighting network and its elements are described. The article discusses the key security problems which have a direct impact on the correct performance of the Smart Lighting critical infrastructure. In order to detect an anomaly/attack, we proposed the usage of a statistical model to obtain forecasting intervals. Then, we calculated the value of the differences between the forecast in the estimated traffic model and its real variability so as to detect abnormal behavior (which may be symptomatic of an abuse attempt). Due to the possibility of appearance of significant fluctuations in the real network traffic, we proposed a procedure of statistical models update which is based on the criterion of interquartile spacing. The results obtained during the experiments confirmed the effectiveness of the presented misuse detection method.

1. Introduction

In the last decade, digital technologies started to cover cities, creating a skeleton of immense intelligent infrastructure based on information and communication technologies (ITC). The aim of building such a ubiquitous system is to create Smart Cities (SC), which have the ability to manage their resources in a better way to enhance the quality of life and safety of their citizens.

One of the key elements of a Smart City is a system of management, monitoring, and smart steering of street lights. This system allows for optimal use of the lighting infrastructure and facilitates reduction of lighting operating costs. It mostly involves prolonged operation of light sources and, as a result, a less often need to exchange them, which is costly. A decrease in the consumption of electric energy also causes limitation of CO_2 emission. Data presented in [1, 2] show that, in the recent decade, approximately 20 per cent of the received electricity is consumed by lighting, where the biggest share concerns roads and streets. Reduction of energy consumption thanks to the use of energy-saving light sources and introduction of Smart Lighting (SL) is performed in numerous ways, of which the most important are (i) reduction of the intensity of light in a given time and space, (ii) switching on and off the lamps precisely in time, and (iii) taking into account the variable capacity of light sources in long-term operation. Utilization of such type of activities ensures optimization of light management costs and limits the electricity consumption costs even up to 40 per cent.

Usually, the Smart Lighting system is an extension of already existing traditional lighting systems. Its implementation is based on installing controllers/drivers in every lamp. The controllers communicate with the steering server via an existing energetic network with the use of LonWorks protocol

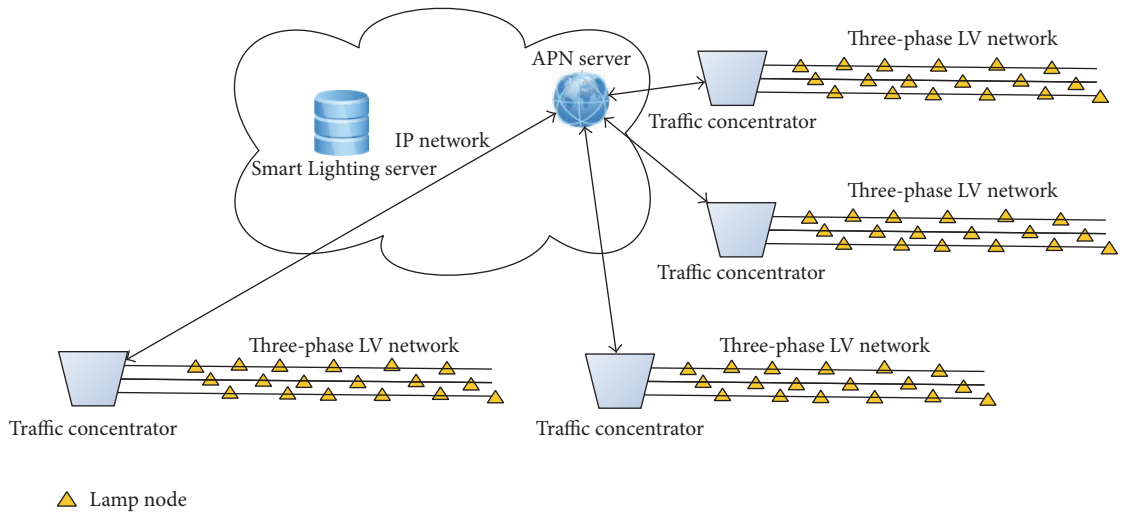


Figure 1: Smart Lighting critical infrastructure.

and Power Line Communication (PLC) technology. The steering server safely communicates through a data transmission network with a central management and control system. The central management system facilitates full control over all of the supervised lamps. This allows for configuration of parameters such as scenarios of switching on/off the lamps and time for initiation of the energy-saving function. It also supplies us with information concerning the current performance of the infrastructure, and it reports failures and provides data about lamps which work defectively [3].

In Figure 1, we can see the general block scheme of a Smart Lighting Communication Network (SLCN). The network consists of lamp nodes (yellow triangles) connected by three-phase low-voltage (LV) power mains by means of PLC modems. Traffic from the lamp nodes is received by a traffic concentrator (TC). The traffic concentrator also plays a role of a gateway between the PLC network and the Internet Protocol (IP) network. The Access Point Name Server (APN) allows us to make a connection by means of packet communication (e.g., Long-Term Evolution (LTE)) to the PLC lighting network.

Smart Lighting systems can be classified in two ways. The first way treats them as a subset of Smart City systems, which are further understood as a subset called the Internet of Things. Such classification does not include the whole area of SL application (e.g., it does not contain Road Lighting systems, which in fact are identical to the street lighting systems in terms of communication solutions). Therefore, the authors believe that a better way of classification is to define the Smart Lighting as a part of the Smart Grid (SG) system. This is a result of the fact that the Smart Lighting communication systems, next to Smart Metering (SM), are the biggest communication systems in Smart Grid when it comes to the number of nodes and the size of the geographic area where they operate. The second key similarity is the technologies used in the fields of the last-mile area of those communication systems. In SL, four technologies are applied, namely, PLC, Radio Frequency (RF), General Packet Radio Service (GPRS), and Meter-Bus (M-BUS), while in Smart Metering there are only two: PLC and RF. As far as RF technologies are concerned, in Smart Lighting and Smart Metering, the used solutions are identical. However, in case of PLC, the differences are significant, of which the most important are the following: (i) in terms of SM, the standard PLC interfaces are applied [4], while in case of SL they are not (the existing Digital Addressable Lighting Interface (DALI) [5] has only local use, e.g., steering a few lamps located on the same pole, not to control hundreds of lamps in a lighting course/string); (ii) the SM devices must communicate in Band A according to CENELEC [6], and SL devices must communicate in Band A if the system's operator is an energy supplier, or in Band B, C, or D if it is the receiver of energy; (iii) there is a requirement to encrypt the transmitted information in case of SM, while for SL there is no such obligation.

Even though, on the market, there are PLC chips equipped with encryption modules (mostly AES-128), this function is seldom used. There are numerous reasons for this fact, for instance, (i) bothersome distribution of encryption keys, (ii) extending the transmission frames and thus improving the unreliability and transmission time, and (iii) finally the cost of implementation.

Actions connected with violating the safety rules in Smart Lighting Communication Networks (especially in the last-mile area) may be deliberate or unaware. Unaware interference usually happens when the LV network powers both streets and households, in which there are connected loads that do not meet the standards of electromagnetic compatibility. On the other hand, the deliberate interference in a communication system consists in intentional switching into not only loads not being able to fulfil the norms, but also elements such as capacitors, interfering generators, or terminals emulating a hub. Switching such devices even into the LV networks dedicated only to lighting is not difficult; therefore, an intruder may use them imperceptibly for a longer period of time.

There are different reasons why the smart lights operator needs anomaly and intrusion detection for the PLC smart lights network. In case of attacks, the smart light operator is responsible for proper operation of the PLC smart lights network. The smart lights network operator is also responsible to the customer in case of improper network operation and may be exposed to penalties. Intentional and unintentional damage cause additional costs to the operator when the attacker changes smart lights to ones instantly on with maximum luminosity. Reaction on anomalies in the smart lights network is also important for public transport safety (especially in intersections) when the attacker might switch off entire segments of the PLC smart lights network. Switching lights off may also be responsible for decreasing public safety in areas where smart lights are off by an attacker. Due to similar reasons, energy suppliers use anomaly and intrusion detection systems in Smart Grid networks especially for detecting energy thefts. Energy operators detect abuses in, for example, WSN (Wireless Sensor Network) smart meter infrastructures [7].

In our system, we propose a solution concerning detection of different types of abuses in the network traffic for the SL infrastructure, which is based on automatically created models of exponential smoothing. To detect abnormal behavior that may be a symptom of possible malpractice, we counted the values of variance between the forecast in the estimated model of traffic and its real variability. In the abovementioned process, we used a two-step method of abuse detection. In the first step of the proposed solution, we identified and then eliminated outliers using the criterion based on Mahalanobis's distance. In the second step, however, we estimated proper statistical models smoothed exponentially for the analyzed network traffic parameters. As a result, the respective operations presented differences in the tested SLCN parameters, which point at possible occurrence of malpractice.

The article is organized as follows. After the Introduction, Section 2 describes the communication protocol used in the last-mile testbed network. Next, Section 3 presents related work on existing abuse detection solutions for Smart Lighting Communications Networks. Section 4 focuses on the main security risks related to the PLC network. Section 5 presents the structure and operation of the proposed solution. In Section 6, a real-life experimental setup and experimental results are presented and discussed. Finally, Section 7 concludes our work.

2. Communication Protocol Used in the Considered Solution

A communication protocol in last-mile Smart Lighting networks was proposed in 2010 and published in 2011 in [8] as EGQF protocol (Energy Greedy Quasi-Flooding) by one of the coauthors. In the same year, this protocol was implemented in Smart Metering networks, which used a low-power RF technology for communication. The EGQF protocol is independent of communication media types and may be applied in networks using RF, PLC, or even RF/PLC hybrid technologies. This protocol is dedicated to tiny communication nodes based on short distance devices connected to shared communication mediums. It uses the multihop technique for transmission range extension and also uses the multipath technique to improve reliability of data transfer. The multipath scheme is useful for delivering data in unreliable environments, such as PLC. The retransmission mechanism is used only by the destination node, without any extra RAM memory occupation, because the RESPONSE packet is already kept in the transmission buffer of a transceiver. The decision to launch the retransmission is as follows: after sending the RESPONSE, the destination node starts a retransmission timer. After the timer expires, the destination node sends RESPONSE again and stops the timer. This timer can also be stopped if a copy of RESPONSE or ACK/Cancel is received during the period of the timer's operation. The number of retransmissions is reduced by a protocol parameter, RC (Retransmission Counter). In our experiments, RC was set to 1. The architecture of the presented network is very simple because it can operate with only two types of nodes, that is, a traffic concentrator and a terminal (a lamp). All the traffic is forced and coordinated by the traffic concentrator. Due to the lack of memory, terminals do not know the network topology and even do not know the addresses of neighboring nodes.

The EGQF protocol uses a small set of packet types, that is, command packets, response packets, and ACK/Cancel packets. Command packets, in most cases, are used by the traffic concentrator for controlling or querying the lamp or the pole. The answer or acknowledgement from the lamp is transported over the response packet. The ACK/Cancel packet is a packet which acts as the low layer ACK for the destination node and as the relaying process canceler for the other nodes. The ACK/Cancel packet is sent only by the traffic concentrator to confirm the reception of the response and to cancel the flooding process of response, or even command copies. The relaying process in nodes, which are neither destination nor source nodes, depends on transmitting the copy of the packet after the sum of constant short time (60 ms) and random time in the condition of an undetected carrier. The difference between the typical flooding protocol and the EGQF protocol is that while using a typical flooding protocol, the nodes always send a copy of the packet once, whereas while using the EGQF protocol, copies are sent as often as needed, for instance, once, twice, or never. The decision of whether a copy of the packet should be sent is made when the transfer discriminator value of the packet is greater than the previously stored one. The initial (or set at the end of the process) transfer discriminator value is zero. The transfer discriminator consists of two fields organized in the following order: the packet type code and the time-to-live (TTL) counter. The TTL occupied three least significant bits of the control field of the packet, while the packet type code occupied two more significant bits in the same field. Commands are coded as 00, responses as 01, and ACK/Cancel as 11, so that the transfer process of the command packet is always canceled after receiving a response packet. This is the same as response packet propagation after receiving ACK/Cancel. The above cases show us a situation where the

relaying process was canceled, which is a difference with regard to the typical flooding protocol. The solution adopted in EGQF reduces the risk of collision. Using the same schema, it is possible to send a copy of the same packet type more than once. Such situation occurs when, after sending the copy of the packet, the same packet is received again with a greater value of TTL than the already copied packet. This situation occurs very seldom (e.g., when a packet with a greater number of hops comes earlier than a packet with a smaller number of hops), and it increases reliability [9].

3. Related Work

Every administrator of a Smart Lightning network or a safety specialist would like to be timely informed about any nontypical behaviors in the infrastructure that he is in control of (whether they are connected to attacks, abuses, or improper performance of devices or applications) [10]. The most important issue is to aim for the detection of new threats and such hazards that would break through the traditional defense mechanisms. One of the possible solutions is the use of systems based on Network Behavior Anomaly Detection (NBAD) [11]. These solutions do not utilize knowledge about the attacks'/abuses' signatures [12] but they are based on behavioral analysis [13]. Such an approach allows for the detection of numerous threats, which "manifest" their presence with nontypical behaviors in the network [14].

Generally, NBAD systems use statistical profiles or behavioral models to detect potential threats/anomalies. Most often, the model approaches are autoregressive ones, for example, AutoRegressive Moving Average (ARMA) or AutoRegressive Fractional Integrated Moving Average (AFIMA) [15], or mixed models composed of autoregressive and exponential smoothing ones [16] (combined to improve the forecasting process). There can also be found solutions applied to anomaly detection in the network traffic, which are based only on traditional, exponential smoothing models [17]. However, all those approaches do not use the processes of optimization to find defined exponential smoothing models, best matching the input data. In the subject literature, there can also be found other works (theoretical ones in particular), that is, Gardner [39, 40], Ord and Lowe [20], or Archibald [41], describing procedures of automatic prediction of future time series' values, based though on defined exponential smoothing models. In the solution proposed by us, we use a mathematical methodology presented by Hyndman [38, 43] which depends on seeking an optimal model (in the process of nonlinear optimization) and automatic procedure of prediction to find the future values of the analyzed time series. Then, detection of anomalies consists in comparing the variability of the real, time series' values with the estimated model of that traffic. Such a solution has not been yet used for anomaly/abuse detection in the Smart Lightning network traffic.

However, exhaustive description of methods and techniques of detection of anomalies and/or outlier observations can be found in review articles [14, 24]. They describe diverse approaches to anomaly detection, starting with machine learning methods, through data mining and information theory, and finishing on spectral solutions. Nevertheless, analysis of those solutions should be conveyed in close connection with their application.

Extensive research has been conducted on security in Smart Grids; most of them are done for anomaly detection in backbone networks and/or all areas of networks based on TCP/IP or UDP/IP protocol stack [25]. Not only does anomaly detection in LV network concern Smart mMtering systems, but also data transmitted over the LV network must be encrypted. In Smart Lighting systems, there are no security requirements for the transmitted data. Most works focus on data transfer reliability [26] in Smart Lighting last-mile communication networks, which is realized by using two independent technologies, for instance, PLC and wireless. In this work, the authors proposed a decentralized method of anomaly detection, similar to the one in [27], but the difference is that our method is proposed for Smart Lighting systems, not for Smart Metering.

In spite of that, we did not find anomaly/attack detection publications for Smart Lighting PLC based networks; there are different methods of anomaly detection used in Wireless Sensor Networks (WSN) or Smart Metering networks. In general, the anomaly detection methods used so far for sensor networks (especially for WSN) were divided into [18, 19] statistical methods (e.g., statistical chi-square test, kernel density estimator), signal processing methods (e.g., based on frequency analysis like Discrete Wavelet Transformation (DWT)), data mining (e.g., clustering methods like K-means, Support Vector Machines (SVM)), computational intelligence (e.g., Self-Organizing Maps (SOM)), rule-based methods, graph based methods (e.g., tree construction), and hybrid methods [18, 19, 21]. Part of the anomaly/attack detection methods work in lower protocol layers (e.g., data link layer or network layer) while others are focused on the application layer (especially for the Advanced Metering Infrastructure (AMI) used by energy operators).

4. Security Risks in PLC Smart Lighting Communication Networks

In Smart Cities, security of critical infrastructures is essential for providing confidentiality, accessibility, integrity, and stability of the transmitted data. The use of advanced digital technologies (ITC), which connect more and more complicated urban infrastructures, is risky because there may appear different types of abuses which may hamper or completely disenable proper functioning of a Smart City. Undoubtedly, one of the biggest frailties of a Smart City is the Smart Lighting system when taking into account the size of the area where it functions, potentially big number of the system's devices, and the generated operational costs. Therefore, providing a proper level of security and protection becomes a crucial element of the SLCN solutions [28].

The task of a Smart Lighting system is not only to light the streets. Depending on the kind of pavement, it must control the brightness of the lighting, its dimming, homogeneity, and reflectivity, providing drivers and pedestrians with maximum safe visibility. Therefore, lighting installations with

luminaires, which are used as light sources, must be easily controllable. Such controlling may include whole groups or even individual lamps, which may be turned on or off according to a specified schedule or dimmed up to any degree at specified times, and the state of individual devices must be easy to control. In comparison to a traditional autonomic lighting system, Smart Lighting solutions are characterized by much bigger functionality and flexibility; however, due to their intelligent nature, they may be liable to different types of abuses (attacks). Such actions may be realized by both the sole receiver of the service and intruders wanting to enforce a specific state of infrastructure [29].

The receiver most often causes destructive actions to the SLCN, which interfere with the transmission of control signals (by active or passive influence) to achieve a change in period and/or intensity of the light. Increasing the intensity of lighting in front of the receiver's property allows for switching off the light on his land, which may result in significant economic benefits. However, a much bigger problem seems to be protection against intended attacks. There are numerous reasons for performing such attacks, the main one being to disturb the controlling system in order to set a different value of lighting than the one established by the operator. Switching off the light or reduction of its intensity in some area may facilitate criminal proceedings. Another reason is malicious activity consisting in hindering the lives of neighbors or local authorities by forcing a change in the schedule of lighting (e.g., switching off the light at night or turning it on during the day). However, a much more serious challenge seems to be protection against attacks realized for criminal purposes. Then, every potential Smart Lighting lamp may become a point by means of which an attack on SLCN may be performed [10].

Such actions, in particular in the area of the last mile, may have a conscious or unconscious nature. The unconscious interference most often happens when the LV network feeds both the streets and the users' households, where the included loads do not meet electromagnetic compatibility standards. The conscious form of interference in the communication system is related to deliberate activity that consists in switching into the SLCN infrastructure such elements as capacitors, interfering generators, or terminals emulating a hub. Loading such devices, even in LV networks dedicated only to lighting, is not difficult, and using them by an intruder may remain unnoticed for a longer time.

Smart Lighting network security and protection from such attacks seems to be a harder task to solve than the prevention of possible abuses (to achieve quantifiable but limited economic benefits) from the receivers' side.

Attacks on Smart Lighting Communication Networks can be divided into two basic categories: passive and active. Passive attacks are any activities aiming to gain unauthorized access to the data or SLCN infrastructure, for which the attacker does not use emission of signals that may disturb and/or disenable correct performance of the signal. Active attacks, on the other hand, are all the attempts of illegal access to the data or the SLCN system's infrastructure by means of any signals or realization of any actions which may be detected [30].

Realizing a passive attack on the SLCN, the intruder camouflages one's presence and attempts to gain access to the transmitted data by passively listening to such a network. It is most often realized by switching into the network additional node which has similar functionalities to the original one. In such situation, we can distinguish three cases: (i) pretending to be a hub, (ii) pretending to be a particular lamp, (iii) or participating only in transferring frames in the transmission process.

To provide protection from such events, appropriate cryptographic mechanisms are most often used. Another kind of passive attack on the SLCM is activities for analyzing the traffic inside the network. In this case, the intruder's intention is not to know the content of the transmitted packets of data, but to get topological knowledge enabling the learning of the structure of the attacked network.

Contrary to the above presented passive forms of attacks on the SLCN infrastructure, in case of realization of an active attack, the intruder influences indirectly or directly the contents of the sent information and/or functionality of the system. Attacks of this kind are much easier to detect in comparison to the passive ones, because they cause visible disturbances in the SLCN performance. An effect of conducting an active attack may be degradation of a specific service or, in extreme cases, complete loss of control over the whole or some part of the SLCN infrastructure.

Due to the form, purpose, and manner of realization, active attacks can be divided into three types: (i) physical attacks aiming at destroying and/or disturbing correctness of the SLCN's node operation by means of an electromagnetic pulse (EMP), (ii) attacks on integrity and confidentiality of the transmitted data, and (iii) and attacks oriented onto particular layers of the SLCN (especially for the provided services).

Physical attacks are all kinds of destructive activities whose aim is to completely destroy or damage the SLCN infrastructure. One of their forms may be activities performed by means of an electromagnetic pulse (EPM) or injecting high pulse distortion into the power supply network [31].

Attacks directed onto the integrity or confidentiality of data, however, are especially dangerous, because they enable the attacker to gain unauthorized access to the information transmitted via the SLCN. This type of attack was presented in [32].

Another kind of attack in the SLCN consists in overloading the attacked network infrastructure, which is visible in the lack of correct data transmission or disenabling access to specific services. Such actions are usually realized by introducing to the network bigger traffic than can be served. They can also have other forms; for instance, they can occur in the physical layer performing jamming activities, and in the layer of data link they can flood the network with packets, causing as a result a collision of data and a necessity to retransmit them. The simplest way to perform such an attack is to connect an additional capacitor to the power circuit. This will cause suppression of the PLC modem carrier signal. Another method is to load into the SLCN a generator broadcasting in the transmission band of the system, which

causes reduction of the signal/noise gap more, rendering a higher level of interfering signal. Reduction of the gap causes then an increase in the number of transmission errors. Another solution is to add any PLC modem, transmitting in the same band that is used by the Smart Lighting system. This solution is a bit more advanced than the use of a generator and causes the modems remaining within the intruder's reach to stay in the "receive" state without the ability to switch to the "broadcast" mode in the period when it transmits, for instance, when it broadcasts without a break or with short breaks [9].

To ensure protection against the above presented threats, especially different kinds of active and passive attacks, it is necessary to provide a high level of security to the critical SLCN infrastructure by continuous monitoring and control of the network traffic. One of the possible solutions to the so-stated problem can be to implement a detection system of anomalies reflected in defined SLCN traffic parameters. In consequence, the detected nonstandard behaviors of specific parameters may indicate a possibility of a given abuse or any other form of attack. The present paper focuses on the above stated question.

5. The Proposed Solution: Predictive Abuse Detection System

For ensuring a high level of security to Smart Lighting Communication Network systems, it is required that they are properly protected by means of passive actions (network monitoring, storing incidents, and reporting) and active actions (constant supervision to enforce the adopted security policy). Realization of the so-stated tasks ensures connection between technologies: Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). In the hierarchy of network infrastructure protection, these systems are located just after security elements, such as a firewall.

The aim of the IPS systems is to undertake actions to prevent an attack, minimize its results, or actively respond to violation of security rules. From the technical side, IPS, in big simplification, is an IDS connected with a firewall. As far as topology is concerned, IPS systems can be divided into network solutions based on (i) a passive probe connected to the monitoring port of the switch analyzing all packets in a given network segment (ii) or a probe placed between two network segments operating in a transparent bridge mode that transmits all packets in the network. The basic aim of such solution is to compare between the real network traffic and the remembered attack signatures [12].

However, IDS systems are used to increase the security of the protected network both from the inside and from the outside. Their advantage is that they can be used for network traffic analysis and use diverse threat identification techniques. One of them consists in the detection of known attacks with the use of specified features (signatures), which describe changes in the network traffic. The second, on the other hand, is based on monitoring normal network's performance in order to find deviations from the norms (anomalies), which may indicate a break-in to the protected network infrastructure.

Anomaly detection (abuses) consists in recognition of nonstandard patterns of behaviors reflected in the network traffic parameters. All incidents deviating from those patterns (which are profiles that describe normal behavior of the network traffic) are classified as potentially dangerous and might signify an attempt of an attack or abuse. High efficiency and effectiveness of methods based on anomaly detection are closely related to the ability of recognition of unknown attacks (abuses). These methods operate on the basis of knowledge of not how a given attack runs (what is its signature), but what exceeds the defined network traffic pattern. Therefore, systems based on anomaly detection work better than those using signatures while detecting new, unknown types of attacks (abuses) [14].

In the present article, we propose a predictive abuse detection system for PLC Smart Lighting Networks based on automatically created models of exponential smoothing. Assuming that the correctness of the created statistical model directly depends on the quality of data used for designing it, at the initial stage, we identified and eliminated outlying data by means of Mahalanobis's distance (see Section 5.1). For the so-prepared data, statistical models were created (which constituted patterns) for particular network traffic parameters. This process was realized by means of exponential smoothing methods which, in turn, assume that the future forecasted value depends not only on the last observed value, but also on the whole set of the past values. Simultaneously, the influence of past values (former ones) is weaker than the influence of the newer values, that is, earlier ones (this methodology is further developed in Section 5.2). It should be noticed that the presented assumption agrees with the generally accepted rules of prediction. Bearing in mind the possibility of occurrence of essential real network traffic fluctuations (triggered by natural factors), a procedure of the pattern models' update was proposed on the basis of the interquartile spread criterion (see Section 5.3).

In Figure 2, we presented a block scheme of the proposed anomaly/attack solution for smart lights Power Line Communication networks. The presented solution is spread out across two physical localizations. On the right part of Figure 2, we can see the analyzed smart light PLC network with smart light marked as a yellow triangle connected to different phases of low-voltage power mains. The PLC traffic from different localizations of smart light PLC networks (in our case, we used 3 localizations on different streets) is gathered by the traffic concentrator and repacked into IP packets in order to send PLC network traffic by means of standard IP WAN network to distant locations where we perform anomaly/attack detection steps. We used two routers equipped with different WAN (Wide Area Network) ports or LTE (Long-Term Evolution) modems in order to connect these two localizations by means of dedicated safe connection through VPN (Virtual Private Network).

On the left of Figure 2, we can see the second part of our anomaly/attack detection solution placed on a distant location (in our case, the university building). The proposed solution is divided into two branches. The first branch is responsible for calculation of reference models for PLC anomaly/attack detection purposes. The second branch

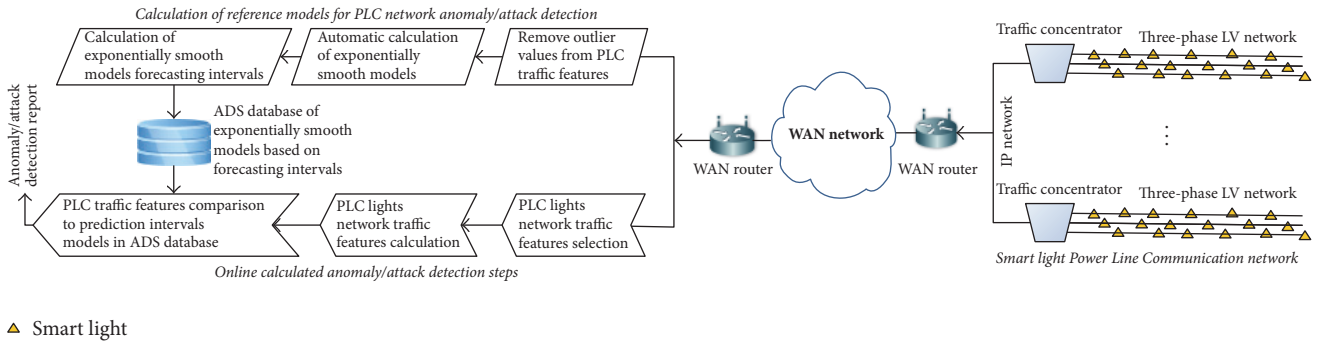


Figure 2: Block scheme of the proposed anomaly/attack solution for smart lights Power Line Communication network.

consists of steps performed online during anomaly/attack detection steps. In order to achieve reference models for PLC network traffic, we extracted traffic features from the PLC network traffic (more details are presented in Section 6.2). After removing outlier values for every traffic feature, we performed automatic calculation of exponential smoothing models and, in the end, forecasting intervals based on these models (details are presented in Section 5.2). Connection between the two branches of the proposed model is realized by means of ADS database where forecasting intervals based on exponential smoothing models are stored separately for every extracted PLC network traffic feature. Additionally, the reference models are updated when necessary to prevent the models from aging in case of changes in, for example, traffic characteristics or physical architecture (by providing additional segments of PLC smart light network). Recalculation of the model is controlled by a trigger condition presented in more detail in Section 5.3.

The second branch of the proposed model also consists of selection and calculation of the PLC network traffic features (see Section 6.2). PLC network traffic features are sampled and calculated with fixed time intervals, appropriate for smart light networks. In order to detect anomalies, we compare online calculated traffic features to prediction intervals read from the ADS database where the prediction intervals based on exponential smoothing models are stored. When the online calculated traffic features are outside the prediction intervals estimated by the model, we generate an anomaly detection report for a given traffic feature (more details are provided in Section 6).

5.1. Outliers Detection and Elimination Based on the Mahalanobis's Distance. The quality of a statistical model directly depends on the quality of data used to design it. The values of variables describing observations in actual datasets are often outlying (not typical). This is due to the specifics of the examined phenomenon or different kinds of errors. The outlier observations may have a very strong influence on the results of analysis and therefore they require special attention.

The notion of outliers is not directly defined in the literature. In the present work, a general definition, taken from Hawkins's work [33], is used. An outlier is such an observation that deviates from the remaining observations to such an extent that it generates an assumption that it was created by another mechanism; for instance, it comes from a different distribution in the dataset. It is worth noticing that, according to the above definition, such emergence indicates not fulfilling one of the most basic assumptions concerning the analyzed dataset, namely, that it is an i.i.d. set (independent and identically distributed). In that case, occurrence of an outlier means that it comes from a different distribution and should not be analyzed with other elements of the examined set of data.

Analyzing particular elements and the operational environment of Smart Lighting Communication Networks, it becomes obvious that there may appear real possibilities of considerable fluctuations of the analyzed network traffic parameters (and, as a consequence, emergence of outliers). These fluctuations may have diverse sources, for instance, (i) environmental, connected with interruptions caused by high-energy electromagnetic pulse; (ii) technical, related to changes in the infrastructure; (iii) devices' damage; (iv) as a consequence of a network attack; or (v) intentional, unfair interference in the SLCN infrastructure. Thus, an important element of the preliminary analysis of data should be the evaluation of the impact that particular observations may have on the final result, and in case of detection of outliers they should be deleted from the set of data.

In our approach, identification of outliers in the analyzed SLCN traffic parameters is performed by means of a method utilizing Mahalanobis's distance. The essence of this method lies in the estimation of the distance between the analyzed observation vector x and the average value in the examined dataset based on the calculated matrix of variance and covariance [34]:

$$MD^2(x) = (x - \hat{\mu})\,\hat{\Sigma}\,(x - \hat{\mu}), \tag{1}$$

$$\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu}), \tag{2}$$

where $\hat{\mu}$ is the average value from the analyzed dataset and $\hat{\Sigma}$ is the matrix of variance and covariance.

To underline the generality of our method, we left the original Mahalanobis's measure matrix record (the case of multiple regression); however, with time series, we have a one-dimensional case. Identification of outliers is performed by comparing Mahalanobis's square distance for each of the

observations with critical values taken from χ^2 distribution. If there are significant differences (at an accepted level of importance), the given observation is treated as an outlier. This approach has one drawback though; namely, the value of the criterion (1) itself directly depends on statistics which are very sensitive to the occurrence of distant values. To eliminate this disadvantage, modifications were proposed for calculating the meter (1) by exchanging the average $\widehat{\mu}$ with a resistant positional parameter. One of the proposals is the use of Minimum Volume Ellipsoid Estimator (MVE) [35]. In this case, $\widehat{\mu}$ takes the value of the center of gravity of the ellipsoid with a minimum volume containing at least h observations of a given set, where $h = (n/2) + 1$, and n is the complete set of elements of the analyzed dataset. The second proposal is to designate a positional parameter $\widehat{\mu}$ in formula (1) according to the following rule [35]: $\widehat{\mu}$ is an average from these h observations of the given set, for which the determinant of covariance matrix is the smallest. Such a resistant positional estimator is called Minimum Covariance Determinant (MCD) estimator. The third approach suggested in the paper [36] uses the analysis of main components and identifies the distant observations just after transformation of all observations in space of main components by determining in this space Mahalanobis's square distance. The authors of this approach propose, at the stage of preparing analytical data, to standardize the variables by means of a median as a positional parameter and MAD, that is, median absolute deviation, as a dispersion parameter. After using such standardization, calculation of Euclidean distance in space of main components is equivalent to the calculation of the resistant variant of Mahalanobis's distance.

In summary, it is necessary to state that the MD measure modifications presented above are trying to eliminate the basic drawback of the described method, that is, not always reliable inference on the basis of classical statistics, which are very sensitive to the occurrence of nontypical observations. Therefore, to make an optimal choice, numerous experiments were performed on datasets containing the subject parameters of SLCN traffic, for both the original Mahalanobis's method and its presented modifications. As a result of the analysis of the obtained results, that is, the size, location, and number of outliers, for further consideration, we chose the approach proposed by Filzmoser et al. This method uses analysis of main components for identification of outliers and it is further developed in [36].

5.2. The SLCN Traffic Features' Forecasting Using Exponential Smoothing Models. Forecasting is still one of the main tasks of the time series analysis. Construction of those predictions is usually a multistage process, including matching the adequate model on the basis of historical data and evaluation of the quality of this matching (diagnostics). Correct conduct of such analysis requires appropriate knowledge and experience. It is usually also time-consuming, which may become an obstacle when it is necessary to collect forecasts for numerous time series simultaneously. Thus, in practice, there is a natural need to automate this forecasting.

In case of some stages connected to matching the optimal model for data, complete automatization is not possible. Particularly, finding an appropriate compromise between the complexity of the model and the quality of its matching to the data often requires interpretation of the results by an analyst. Automation of the optimal model's choice usually requires adopting some assumptions simplifying the whole process (e.g., defining the statistical criterion, which will be used as a measure of matching quality of the model or the possible ranges of variation of model parameters) [37].

Algorithms allowing for automatic construction of forecasts should realize all the stages of the analysis, that is, (i) the choice of the optimal model for data, (ii) parameters' estimation, and (iii) the forecasts' construction (point and/or interval). While searching for an optimal model, it is important to use proper criteria which will protect from too good matching of the model to the learning data, which in turn may lead to bad quality of forecasts for the new periods. The algorithms should also be resistant in case of occurrence (in the analyzed time series) of outlier observations, or they should be equipped with mechanisms of their detection and elimination. Additionally, the algorithms should be easily used for a big number of diverse time series without the necessity of an analyst's interference, and they should be characterized by acceptable computational complexity [20].

One of the possible solutions to the so-stated problem of automatic forecasting is the ExponenTialSmoothing or ErrorTrendSeason (ETS) models, which constitute a family of adaptive models developed by Hyndman et al. [38], which uses generalized algorithms of exponential smoothing. Their crucial advantages are simplicity, relatively quick adaptive matching algorithm, and ease of understanding and interpretation of the results. The common denominator of these methods is assigning (exponentially) the weights, decreasing with distance in time, to the past observations during the process of designating a new forecast for a future observation. This is due to the fact that the classical assumptions of the quantitative prediction come down to the postulate of the relative invariability of the development mechanism of the studied phenomena and events. In methods based on ETS, exponential smoothing may be realized by means of different models, properly adjusted to the analyzed data.

When the time series' character and variability are analyzed, it is easy to notice that they are optionally composed of four elements: a trend, seasonal fluctuations, periodical fluctuations, and random disturbances. The seasonal fluctuations usually have an approximately constant period of time, whereas the time of the complete cycle of cyclical fluctuations is usually changeable. Optionally, the components of the analyzed time series may be connected in two ways: additively and multiplicatively [39]. In the exponential smoothing models, the trend is a combination of level c and increment g values. These two components may be connected in four different ways, including the attenuation parameter $\phi \in [0, 1]$. We then obtain diverse types of trends, such as the following [40]:

$$\text{No trend: } V_h = c, \tag{3a}$$

$$\text{Additive: } V_h = c + gh, \tag{3b}$$

$$\text{Multiplicative: } V_h = cg^h, \tag{3c}$$

$$\text{Attenuated: } V_h = cg^{(\phi+\phi^2+\cdots+\phi^h)}, \tag{3d}$$

where V_h describes the character of the trend and h parameter describes the forecast's horizon.

If we take into consideration three possible combinations of the seasonal component with a trend, that is, lack of seasonality, the additive variant, and multiplicative variant, then we obtain twelve exponential smoothing models, which can be written as

$$l_t = \alpha P_t + (1-\alpha) Q_t, \tag{4a}$$

$$b_t = \beta R_t + (\phi - \beta) b_{t-1}, \tag{4b}$$

$$s_t = \gamma T_t + (1-\gamma) s_{t-m}, \tag{4c}$$

where l_t denotes the series level at time t, b_t denotes the slope at time t, s_t denotes the seasonal component of the series at time t, and m denotes the number of seasons in a given period; the values of P_t, Q_t, R_t, and T_t vary according to which of the cells the method belongs to, and $\alpha, \beta, \gamma, \phi \in [0, 1]$ are constants denoting model parameters [38].

The method with fixed level (constant over time) is obtained by setting $\alpha = 0$, the method with fixed trend (drift) is obtained by setting $\beta = 0$, and the method with fixed seasonal pattern is obtained by setting $\gamma = 0$. Note also that the additive trend methods are obtained by letting $\varphi = 1$ in the damped trend methods [41].

The works [42] discuss specific cases of state space models with a single source of error, which may be a basis for some methods of exponential smoothing. Including the possible character of these errors, we may present the state space models for all twelve types of exponential smoothing as follows:

$$Y_t = w(z_{t-1}) + r(z_{t-1})\epsilon_t, \tag{5a}$$

$$z_t = f(z_{t-1}) + g(z_{t-1})\epsilon_t, \tag{5b}$$

where $z_t = [l_t, b_t, s_t, s_{t-1}, \ldots, s_{t-m+1}]^T$ denotes the state vector, $w(x), r(x)$, $f(x)$, and $g(x)$ are continuous functions with continuous derivatives, and $\{\epsilon_t\}$ is a Gaussian white noise process with mean zero and variance σ^2, and $\mu_t = w(z_{t-1})$ [42]. The error ϵ_t may be included in the model in an additive or multiplicative way. The model with additive errors has $r(z_{t-1}) = 1$, so that $Y_t = \mu_t + \epsilon_t$. The model with multiplicative errors has $r(z_{t-1}) = \mu_t$, so that $Y_t = \mu_t(1 + \epsilon_t)$. Thus, $\epsilon_t = (Y_t - \mu_t)/\mu_t$ is the relative error for the multiplicative model. The models are not unique. Apparently, any value of $r(z_{t-1})$ will lead to identical point forecasts for Y_t [38].

From the twelve exponential smoothing models described by dependency (4a), (4b), and (4c) after including the additive and multiplicative error ϵ_t, we obtain 24 adaptive models in the states' space. The choice of an adequate exponential smoothing model in a particular prognostic task requires the selection of the best form of the model as well as initialization of the z_0 vector's components and parameters estimation $\Theta = [\alpha, \beta, \gamma, \phi]^T$.

It is necessary to calculate the values of z_0 and Θ parameters; otherwise, the models will not be useful for the prognostic process. It is not difficult to compute the likelihood of the innovations state space model (LISSM*) (see (6)); achieving the maximum likelihood estimates (MLE) is similarly easy [38].

$$\text{LISSM}^*(\Theta; z_0) = n \log\left(\sum_{t=1}^{n} \frac{\epsilon_t^2}{z_{t-1}}\right) + 2\sum_{t=1}^{n} \log\left|r(z_{t-1})\right|, \tag{6}$$

where n is the observations' number.

Calculating the above is not difficult when recursive equations are used [43]. Minimizing LISSM* is a procedure used to calculate the parameter Θ and the initial state z_0.

The present model was selected by means of the Akaike Information Criterion (AIC):

$$\text{AIC} = \text{LISSM}^*\left(\widehat{\Theta}; \widehat{z}_0\right) + 2k, \tag{7}$$

where k is the number of parameters in Θ plus the number of free states in z_0 and $\widehat{\Theta}$ and z_0 define the estimates of Θ and z_0. From all the models applicable to the data, we selected the one which minimizes the AIC [44].

The AIC is also a method which enables us to choose between the additive and multiplicative error models. However, there is no difference between the point forecasts of the two models, to make it impossible for the standard accuracy measures, like the mean squared error (MSE) or mean absolute percentage error (MAPE), to differentiate between the error types.

The presented methodology, connected to optimal searching for proper models of exponential smoothing, requires providing some initial values. Usually, the values of parameters α, β, and γ are included in the range $(0, 1)$. However, to avoid the problem with instability, we use a narrower range of parameters, that is, $0.1 \le \alpha \le 0.9$, $0.1 \le \beta \le 0.9$, $0.1 \le \gamma \le 0.9$, and $\beta \le \phi \le 1$. We also limit the values of the initial states z_t of the vector's elements. This is done in such a way that the seasonality indexes were summed up do zero for the additive model and added to m for the multiplicative model. As the initial values in the nonlinear optimization, we use $\alpha = \beta = \gamma = 0.5$ and $\phi = 0.9$.

When we summarize the above ideas, we obtain an automatic forecasting algorithm. It operates in compliance with the following three-stage formula: (i) all proper models are applied to each of the series to optimize the parameters (smoothing the variable's initial stage), (ii) selection of the best matching model according to AIC, and (iii) creation of point forecasts on the grounds of the most effective model (with optimized parameters) for a necessary number of future stages [38].

All the above described kinds of exponential smoothing models are created in compliance with the prediction theory's assumptions, including the ongoing degradation processes (i.e., possible lack of stability in the variable correctness in time). Big flexibility of those models and their adaptive ability in case of irregular changes of the direction of speed of the trend, or deformations and shifts in seasonal fluctuations, make them a comfortable tool for short-term forecasting

and prediction. Hyndman et al. [38, 43] provide a detailed description of the proposed algorithm.

5.3. The Condition of Statistical Model's Update. The process of statistical models' designation on the basis of experimental data is usually a complex task which depends on the knowledge about the object and attributes of the measuring results (observations). The quality of the designated statistical model directly depends on the quality of data used for its estimation.

In the present work, the experimental object is network traffic of an SLCN infrastructure and data characterizing the state of the Smart Lighting system. Both datasets are represented by defined time series. While analyzing the character of the examined dependencies, in particular the SLCN traffic parameters, it is necessary to notice the possibility of occurrence of significant fluctuation of data. The reasons of this phenomenon are to be sought in possible changes in the SLCN infrastructure, that is, aging of devices, replacement with new/other models, or modifications in the topology of the network. Obviously, when the nature of the analyzed data changes, there should be made a new estimation and creation of an updated statistical model on the basis of datasets composed of the subject fluctuations. As a result, this should cause adaptation of the proposed method of anomaly detection to the changing conditions (which are not an aftermath of any attack or abuse).

For the initial data selection, that is, checking if we are dealing with significant fluctuations in the analyzed time series, we use the one-dimensional quartile criterion [45]. For every analyzed set of data, we calculate the first (Q1) and third (Q3) quartiles and the interquartile range (IRQ) IRQ = Q3 − Q1. As influential observations, we accept those whose values exceed the range (Q1 − 1,5IRQ, Q3 + 1,5IRQ). As extremely influential observations, however, we understand those exceeding the range (Q1 − 3IRQ, Q3 + 3IRQ).

In the next step, for every detected influential observation, we check fulfilling the condition of whether it fits the range of forecasts of the appropriate reference model, that is, the following condition:

$$x_i \in \left(\mu_f - \sigma_f, \mu_f + \sigma_f\right) \quad i = 1, 2, \ldots, n, \tag{8}$$

where $\{x_1, x_2, \ldots, x_n\}$ is a time series limited by n-element analysis window, μ_f is the average forecast of the given reference model in the analysis window, and σ_f is the standard deviation of appropriate prognosis.

The estimation condition of the new standard model should be an ability to detect (in the analyzed time series) significant and possibly stable statistic changeability. Therefore, updating the statistical model will be realized when in the analyzed time series over 30 per cent of analysis windows in a weekly period contain observations not fitting the acceptable prognosis range of the appropriate reference model. The above condition is a consequence of the observed dependency that the value of the false positive (FP) parameter of the presented anomaly detection system increases exponentially when in over 30 per cent analysis windows in a weekly period we note significant changeability in data.

Table 1: PLC data link and network layer traffic features extracted from the traffic concentrators.

Network feature	PLC smart lights network traffic feature description
DLN_1	RSSI: received signal strength indication for PLC lamps [dBm]
DLN_2	SNR: signal-to-noise ratio [dBu]
DLN_3	PER: packet error rate per time interval [%]
DLN_4	PPTM: number of packets per time interval
DLN_5	TTL: packet time-to-live value

Table 2: PLC application layer traffic features extracted from the traffic concentrators.

Network feature	PLC smart lights network traffic feature description
APL_1	ENE: power consumption by PLC lamp [Wh]
APL_2	TEMP: lamp temperature [°C]
APL_3	LUL: lamp luminosity level in % (value: 0–100%)
APL_4	NR: number of lamp resets per time interval
APL_5	PS: power supply value [V]

6. Experimental Installation and the Anomaly/Attack Detection Method and Results

In Figure 2, we presented a block scheme which consists of the main steps in the proposed anomaly/attack detection method. In the first step, we extracted the PLC traffic features from two experimental PLC smart lights networks (additional explanation can be found in Section 6.1). There are two main branches in the proposed method: calculation of reference models for PLC network anomaly detection and the second branch consisting of online steps for extraction of traffic features, comparison of traffic features for reference model in ADS reference models database, and generation of an anomaly/attack detection report for a given traffic feature.

Values of the PLC traffic features can be captured in an arbitrary time interval but usually a 15-minute time interval is sufficient for the PLC smart light network. The extracted PLC network traffic features (see Tables 1 and 2) are represented as a one-dimensional time series. In case of a reference model generation, we have to remove suspicious values first by removing outlier values from network traffic features (see Section 5.1). After that step, we can start to calculate exponential smoothing models (see Section 5.2) and in the end exponential smoothing models forecasting intervals. We calculate a separate model for every PLC traffic feature and store them in a database of reference models. The reference models are calculated for a one-week period with a 15-minute resolution window. An example of the calculated forecasting intervals for traffic features can be seen in Figure 3. We can see two prediction intervals for signal-to-noise ratio (SNR) PLC traffic feature. When the online calculated network traffic feature is within boundaries set by two prediction intervals (see Figure 3), we assume that there is no anomaly/attack in

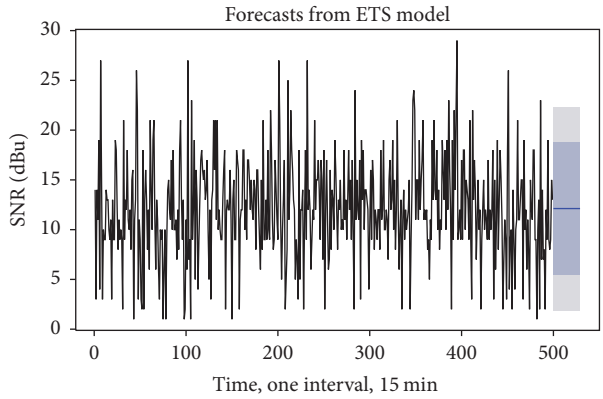


Figure 3: Two prediction forecast intervals (80% narrower, 95% wider) and 30-sample prediction interval calculated with the use of exponential smoothing model (PLC traffic feature, signal-to-noise ratio (SNR) [dBu]).

this case. We expect that 80 or 95% of the values for a given PLC traffic feature will lie inside these intervals (see Figure 3).

The second branch in our anomaly/attack detection method consists of steps calculated online during normal work of the PLC network anomaly/attack detection method. In the first two steps, we extract and calculate PLC lights network traffic features from Tables 1 and 2. Next, for every traffic feature, we check if the online calculated traffic feature values are within the intervals designated by reference models stored in ADS reference database models. When the online calculated traffic features are outside reference intervals, we generate a detection report about possible anomaly/attack triggered by the given PLC traffic feature.

The main issue of the so far proposed anomaly/attack detection conception is the problem of reference models' aging. This phenomenon comes from the fact that the PLC lights network has a dynamic structure. Connecting additional segments of PLC smart lights networks will result in changing of network traffic characteristics and, as a consequence, the necessity of changing reference models. Nonupdated reference models will cause as a result a constant increase of false positive values (FP [%]). To alleviate this drawback, we propose a trigger condition responsible for the recalculation process of the reference models (see Section 5.3 for more details). Reference models are calculated in a one-week period with the use of 15-minute windows. Based on empirical experiments, we recalculate all reference models when trigger conditions (see (8)) are not satisfied in 30% of the 15-minute analysis windows during the one-week period. We started to use new recalculated models at the beginning of the new week (the new model is valid for a minimum of one-week period).

6.1. Experimental Testbed. The analyzed data were captured in two locations: Nieszawska Street in Toruń City (Poland) and University of Technology and Life Sciences (UTP) campus in Bydgoszcz City (Poland). We also used an additional separate Smart Lighting low-voltage LV PLC network testbed constructed during studies in GEKON project [46].

The first PLC network, located in Nieszawska Street, which was dedicated to a Smart Lighting low-voltage LV network, has a length of 3 km (see Figure 4), divided by a traffic concentrator located in the middle of the street. The PLC smart lights network contains 108 lamps (only one lamp is located on every electric pole). Old gas-discharge lamps were gradually replaced by smart LED lights. We used this network for testing traffic concentrators and experiments for detecting anomalies/attacks in PLC traffic.

The second network was placed at the University of Technology and Life Sciences (UTP) campus (see Figure 4). In this case, it was not a dedicated network with a separate power supply (offices, classrooms, and labs were powered by the same power supply network). The testbed in UTP campus consisted of 36 lamps.

Tests were performed in the laboratory (located in UTP campus) with different types and numbers of lamps (gas-discharge lamps and LED lamps). The PLC traffic from both locations was captured from the WAN (from Nieszawska Street) and local network placed in the university laboratory.

6.2. Experimental Setup and Results. In this section, we present the methodology and results achieved for the proposed anomaly/attack detection with the use of exponential smoothing based models. We propose a set of different scenarios for evaluating the usability of the proposed method.

All experiments were carried out by means of two real-world PLC lights networks (see Section 6.1). A part of the testbed located in the university campus can be seen in Figure 5. The picture presents different types of smart lights used in the experiments. Connections between the 36 lamps for the testbed partially presented in Figure 5 are presented in Figure 7. We can see connection schemes between lamps assigned to three-phase power mains with signed possible high-quality and low-quality links. The entire traffic as mentioned earlier is accessible by the traffic concentrator (red rectangle in Figure 7).

Every lamp consists of a PLC modem used for communication, a lamp microprocessor controller, and a power supply. An opened LED lamp with signed internal elements is presented in Figure 6.

The first step in our method requires capturing the PLC traffic from smart lights networks presented in Section 6.1.

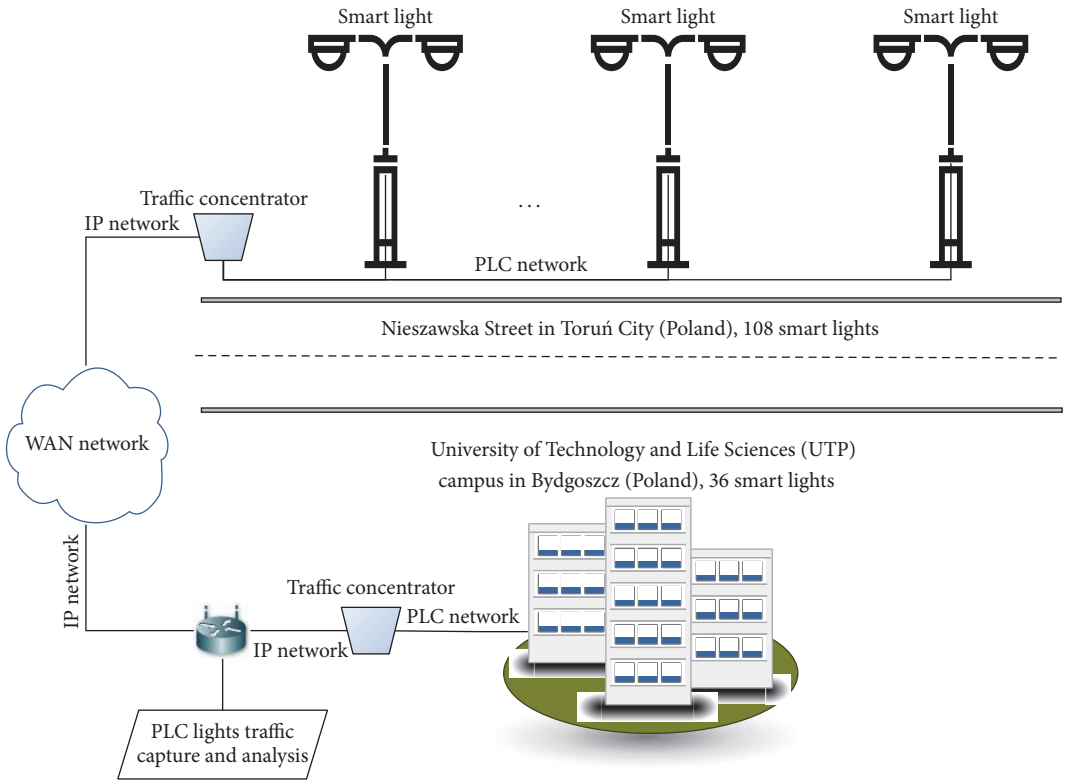


Figure 4: Experimental testbed used for evaluation of the proposed anomaly/attack detection method.

Figure 5: Part of the testbed used for achieving experimental results, located in the university campus.

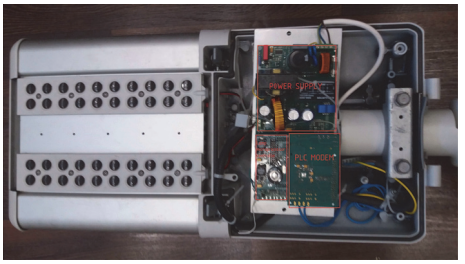


Figure 6: Opened LED smart light used in experiments.

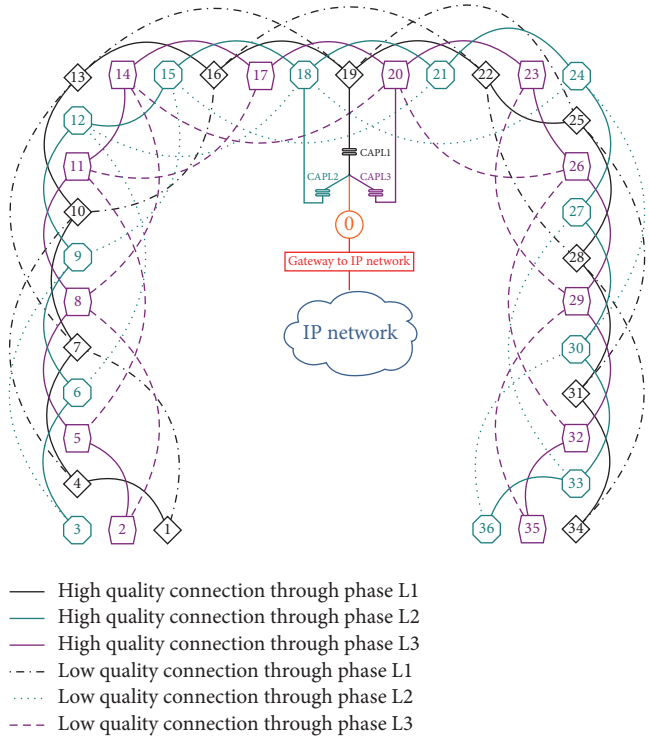


Figure 7: Schematic connection between 36 smart lamps for the testbed located in the university campus.

We collect PLC traffic from traffic concentrators which are responsible for translating the PLC network packets into IP packets. In the next step, we extract the PLC traffic features in order to analyze these features for anomaly/attack detection.

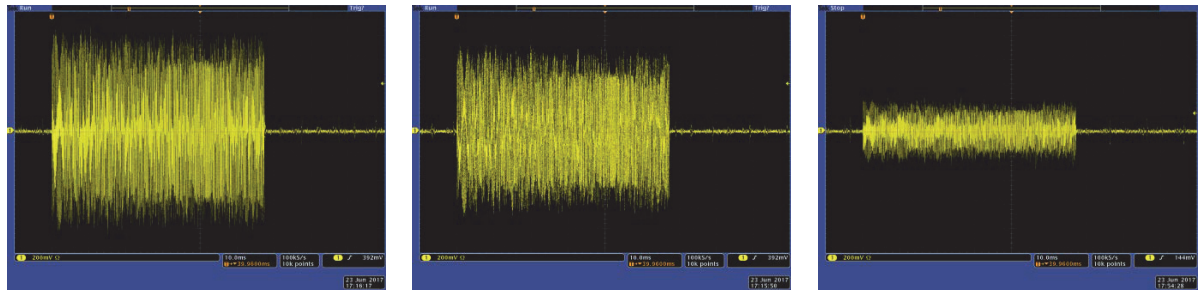
In our experiments, we extracted features that belong to every layer of a PLC protocol stack. In Tables 1 and 2, we can see the extracted PLC traffic features together with explanations.

Traffic features from Table 1 are extracted based on data link and network layers of PLC communication stack. DLN_1 and DLN_2 features give us information about the quality of the received signals transmitted through the power mains. RSSI gives us information about the received signal strength where the signal power may come from any sources (e.g., different modulations, background radiation). RSSI does not give us information about the possibility of signal decoding. SNR [dBu] measure gives us information about the relation between the desired signal and the noise level. DLN_3 traffic feature stands for Packet Error Rate (PER) per time interval. In our case, we used a 15-minute time interval. PER is calculated as a quotient between the number of destroyed packets received by the traffic concentrator and the number of all packets received by the traffic concentrator for a given period of time. DLN_4 feature PPTM stands for the number of packets per time interval. The last feature from layer 2/layer 3 DLN_5 gives us TTL information connected to packets received by the PLC concentrator. In Table 2, there are traffic features extracted from the data payload (application layer) of the PLC packets. The application layer traffic features are connected with parameters used by the energy supplier/operator management staff. APL_1 feature gives us information about power consumption for a given period of time separately for a given lamp. APL_2 carries information about the temperature read from smart lights. LUL (lamp luminosity level, in [%]) feature has values of luminosity sent by the lamp to the traffic concentrator. APL_4 carries the number of lamp resets per time interval (the value is stored in the Static Random Access Memory (SRAM) with backup power provided by a supercapacitor). The last value extracted from the data payload is PS (power supply) in volts [V] which is useful information for maintenance systems.

After PLC network features extraction, we can analyze subsequent traffic features in order to detect possible anomalies/attacks. We propose scenarios (as realistic as possible) in order to evaluate the efficiency of the proposed anomaly detection methodology.

There are different purposes of attacking smart lights PLC networks. First of all, the attacker would like to disturb the control system of a smart light operator in order to change the settings of the lamps parameters. Switching lamps off or lights' intensity reduction for a given area may cause an increase in crime or can be dangerous for car traffic (highest possibility of car accidents especially at intersections). Intentional damage or setting lamps instantly on near selected attacker possessions causes additional financial losses to the operator.

Detecting anomalies is also an important thing for the smart lights operator. The operator will be able to react faster



(a) (b) (c)

Figure 8: Impact (on signal received by the smart light) of 470 nF capacitance connected to the power line: (a) without capacitor, (b) capacitor connected close to the traffic concentrator, and (c) capacitor connected inside the lamp pole.

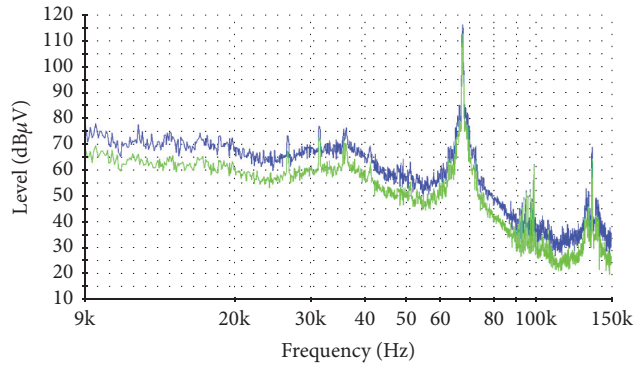


Figure 9: Characteristics of the interference signal generated by damaged notebook switching supply.

on damage, intentional damage, and network attacks, so it will be possible to limit the negative economic and social consequences.

We can divide the proposed scenarios into two main groups: (i) the first type of scenario requires physical access to the PLC network infrastructure in case of attacks on the physical infrastructure of a PLC smart lights network and (ii) the second type of attack requires knowledge about devices used in the PLC network and protocols used in the smart lights network.

Scenario 1. The first type of attack belongs to Group I of attacks. It is an attack on the physical layer and requires connection of a capacitor to the power line. The bigger the value of capacitor we connect, the higher the attenuation of PLC signal we achieve. In our case, we connected a 470 nF capacitor to the power line. In Figure 8, we can see oscillograms: Figure 8(a) without connected capacitor, Figure 8(b) with the same value capacitor connected near the traffic concentrator, and Figure 8(c) with capacitor 470 nF connected directly inside the lighting pole. In the presented oscillogram, we can see decreasing values of modulated PLC signals. When we connect a capacitor with higher values, for example, 4.7 uF, close to the PLC, the transmitter' modem would not be able to transmit any packet because of the too low current efficiency of power supply or line amplifier.

A different method of attack on the physical layer is connection of a signal generator to the power line. The connected generator has to transmit the signal with values that belong to the PLC frequency band used by the attacked network. The higher the level of the injected signal, the bigger the values of PER (DLN_3 feature) and the lower values of the SNR traffic feature. We performed such an attack by means of a damaged/prepared switching power supply which comes from a notebook computer. This is an easy and cheap way to perform such attack. We transmitted a narrow bandwidth signal with 90 dBuV power close to the disturbed device. In Figure 9, we can see the characteristics of the interference signal that comes from the damaged laptop power supply.

We also disturbed PLC power mains by a professional Electrical Fast Transient (EFT)/Burst generator [22] that is used during electromagnetic compatibility (EMC) tests and capacitive coupling clamp (in this case, there is no need for a galvanic connection to the power mains) according to the IEC 61000-4-4 [47] recommendation.

In our experiments, the capacitors and generator were connected constantly but the attacker can arbitrarily connect these elements by a microcontroller controlled device and take into consideration, for example, sunrise and sunset.

Attacks from Scenario 1 have an impact mainly on data link and the network layer from Table 1. In Table 3, we can see the results of the proposed anomaly/attack detection method.

Table 3: DR [%] and FP [%] for anomalies/attacks performed on the SLCN in Scenario 1.

Network feature	DR [%]	FP [%]	Description
DLN_1	90.80	4.80	—
DLN_2	98.00	3.60	The biggest impact on DLN_2 in Scenario 1
DLN_3	97.00	3.20	The biggest impact on DLN_3 in Scenario 1
DLN_4	81.40	5.20	—
DLN_5	75.40	7.40	—

Table 4: DR [%] and FP [%] for anomalies/attacks performed on the SLCN in Scenario 2.

Network feature	DR [%]	FP [%]	Description
DLN_1	91.40	4.30	—
DLN_2	98.80	3.80	The biggest impact on DLN_2 in Scenario 2
DLN_3	97.60	3.10	The biggest impact on DLN_3 in Scenario 2
DLN_4	82.60	6.40	—
DLN_5	78.60	7.80	—

Table 5: DR [%] and FP [%] for anomalies/attacks performed on the SLCN in Scenario 2.

Network feature	DR [%]	FP [%]	Description
APL_1	98.40	2.80	—
APL_2	91.20	5.20	—
APL_3	96.80	3.80	—
APL_4	—	—	Not important in this scenario
APL_5	—	—	Not important in this scenario

Scenario 2. In the second scenario, the attacker would like to generate random packets by means of a connected unauthorized smart lamp or a PLC modem. This is a more sophisticated attack than in case of using a generator (see Scenario 1). Constantly generated packets by the attacker's PLC modem cause modems which are within the impact of this transmission to be constantly in the receiving mode and to be unable to transmit or receive any packets. The attacker transmits packets with the use of carrier frequency/frequencies used in the attacked network one by one with the shortest delays as possible between consecutive packets. Packets transmitted by the attacker may be understandable or not from the smart lights network's point of view. Results of DR [%] and FP [%] for anomaly detection in case of Scenario 2 are presented in Tables 4 and 5.

Indirectly, this type of attack can also be seen in application layer parameters because part of the lamps will switch to maximum luminosity after three connection attempts to the traffic concentrator (we set 900 seconds between attempts). In this case, energy consumption will increase and other parameters that depend on energy consumption also will change (e.g., the lamp's temperature).

Scenario 3. The attack performed in Scenario 3 belongs to Group II of attacks. This type of attack requires knowledge about the PLC smart lights network topology, devices used in the smart lights network, communication protocols used for every layer of PLC communication stack, and so forth.

The attacker, in the presented scenario, connected an additional traffic concentrator (with the same MAC address as the valid traffic concentrator). The attacker's traffic concentrator pretends to be a valid communication device and takes part in packet exchange between lamps. The attacker is placed near lamps and wants to change the lamps' settings. In this case, the attacker is far from the concentrator and the valid concentrator does not receive the command (or a command copy) sent by the fake concentrator. In order to prevent the command from reaching the valid concentrator, it is best to send a command with TTL = 0.

We also performed a similar attack when the attacker was close to the valid concentrator. In this case, anomaly is revealed by the registration command packet with TTL = TTLmax. The valid concentrator will never hear packets' copy with TTLmax. In a proper situation, the packet should have TTL < TTLmax. In this case, the attacker does not care that packets will not arrive to the valid concentrator. Results for the presented scenario are presented in Table 6.

Scenario 4. In the presented scenario, the attacker connected an additional device with a PLC modem and tried to change and retransmit packets with destroyed bits. This action causes an increasing number of corrupted packets with wrong Cyclic Redundancy Check (CRC) bytes. In this case, we can see an increasing value of Packet Error Rate (PER) (DLN_3) network feature. For example, if we send a command to lamps with new luminosity settings, some lamps may not get this information. When a lamp does not receive any command after three connection attempts to the concentrator (number of attempts' parameter NA and time between attempts are protocol parameters in our experiments set to NA = 3

Table 6: DR [%] and FP [%] for anomalies/attacks performed on the SLCN in Scenario 3.

Network feature	DR [%]	FP [%]	Description
DLN_1	—	—	Not important in this scenario
DLN_2	—	—	Not important in this scenario
DLN_3	—	—	Not important in this scenario
DLN_4	90.60	4.60	—
DLN_5	98.60	3.40	—

Table 7: DR [%] and FP [%] for anomalies/attacks performed on the SLCN in Scenario 4.

Network feature	DR [%]	FP [%]	Description
DLN_1	—	—	Not important in this scenario
DLN_2	—	—	Not important in this scenario
DLN_3	98.40	3.40	—
DLN_4	85.40	7.24	—
DLN_5	92.60	6.60	—

Table 8: DR [%] and FP [%] for anomalies/attacks performed on the SLCN in Scenario 4.

Network feature	DR [%]	FP [%]	Description
APL_1	98.80	2.40	—
APL_2	90.30	4.80	—
APL_3	96.50	3.60	—
APL_4	—	—	Not important in this scenario
APL_5	—	—	Not important in this scenario

and time 900 seconds), then they will switch to maximum luminosity. This situation causes additional costs to the installation's operator. This type of attack can be especially seen in application layer network features, such as APL_1 (ENE, power consumption by PLC lamp [Wh]), APL_3 (LUL, lamp luminosity level received from the lamp), and, indirectly, the lamp's temperature (APL_2). Detection rate DR [%] and false positive FP [%] results for Scenario 4 are presented in Tables 7 and 8.

Scenario 5. In the next scenario, the attacker would like to prevent receiving the broadcast command (e.g., a command that wants to set a group of lamps to certain luminosity) by lamps. When the attacker's PLC modem detects a broadcast command sent by a traffic concentrator, it transmits an arbitrary command (i.e., no operation command (NOP)) in the unicast mode. Transmission in the unicast mode has a higher priority and lower delay, which is why this transmission will reach first the lamp. The lamp will respond to this packet by switching to the acknowledge ACK/awaiting state. Broadcast command receiving is only possible for lamps in IDLE state. Results for this scenario are presented in Tables 9 and 10.

Additional explanation requires application network features APL_4 (number of lamp resets per time interval (NR)) and APL_5 (power supply (PS) value). These parameters are mainly important for the smart lights network operator and were not affected by the attack simulated in our experiments. Such parameters are important for smart lights network management and may indirectly have an impact on the transmission parameter, but we did not have the chance to observe the impact of these parameters during our experiments.

Taking into account all scenarios, the detection rate (DR) values change from 75.40 to 98.80%, while the false positive ranged from 7.80 to 2.40%. We can see that, depending on the attack scenario, only part of the network traffic features selected from the PLC traffic give us meaningful information from the anomaly/attack detection's point of view. For example, in Scenario 4, we can see a direct impact on data link and network layer features and indirect influence on application layer features extracted from the data payload.

Results achieved by the proposed anomaly/attack detection proved the usefulness of the proposed method. Anomaly detection systems are characterized by higher values of false positive in comparison to classic intrusion detection systems (IDS), which are based on the database of already known attacks.

We verified the achieved results by comparing the proposed solution to methods available in the literature. Although we did not find anomaly and intrusion detection for smart lights PLC network operating in data link, network layer, and application layer, there are anomaly and intrusion detection systems applied to WSN smart meter networks in Smart Grid AMI (Advanced Metering Infrastructure). Such solutions are mainly designed for energy theft detection and for failure and maintenance purposes and operate usually in network and application layers. Anomaly and intrusion detection systems for energy theft detection use, for example,

Table 9: DR [%] and FP [%] for anomalies/attacks performed on the SLCN in Scenario 5.

Network feature	DR [%]	FP [%]	Description
DLN_1	—	—	Not important in this scenario
DLN_2	—	—	Not important in this scenario
DLN_3	—	—	Not important in this scenario
DLN_4	92.40	7.74	—
DLN_5	90.20	7.70	—

Table 10: DR [%] and FP [%] for anomalies/attacks performed on the SLCN in Scenario 4.

Network feature	DR [%]	FP [%]	Description
APL_1	94.20	2.60	—
APL_2	88.40	4.40	—
APL_3	98.40	2.40	—
APL_4	—	—	Not important in this scenario
APL_5	—	—	Not important in this scenario

the HMM (Hidden Markov Models) [48], rule-based solutions [13], and statistical methods by means of, for example, Bollinger Bands [49]. Different kinds of methods are for estimation of metering errors in AMI infrastructure with the use of, for example, DTW (Dynamic Time Warping) [23]. In general, anomaly detection systems are very diverse and so a straightforward comparison is not easy, though it can be stated, bearing in mind the available literature [7, 11, 13, 15, 16, 23, 48–50], that false positive values for anomaly detection type systems are generally less than 10% [18, 19, 21]. This level of false positive parameter is acceptable for the proposed class of systems, especially for anomaly detection systems.

We also proposed a mechanism that prevents the aging of exponential smoothing models (see Section 5.3). Such an installation like smart lights networks or Wireless Sensor Networks (WSN) changes over time, so it is important to predict such a situation and update anomaly detection reference profiles in order to prevent the increase of false positive values.

7. Conclusions

The number of potential threats in dynamically created Smart Cities, and in particular in their critical communication infrastructures, is very big and is increasing every day. Thus, protection from constantly newer vectors of attacks is becoming more complicated and requires the use of highly specific solutions. Currently, the most often used mechanisms ensuring an adequate level of security in such infrastructures are the methods of detection and classification of abuses (attacks) unknown so far, often directed onto defined sources of critical communication infrastructures. The basic aims of such solutions are the early detection and reaction to the symptoms of nontypical behavior of network traffic which may indicate various abuses originating both outside and inside the protected infrastructure.

The article presents effective solutions concerning the detection of different types of abuses in network traffic for the critical infrastructure of Smart Lighting. It proposes and describes the structure of the SLCN created for the purposes of the experiment. The structure was built with the use of Power Line Communication technology. The key security problems are also discussed, which have a direct impact on proper operation of the Smart Lighting critical infrastructure; that is, the authors described the possibilities of emergence of both external factors and active forms of attacks aiming at gaining influence on the informational contents of the transmitted data. The article proposes an efficient and effective method of abuse detection in the analyzed Smart Lighting network traffic. At the initial stage of the solution, there is identification and elimination of outliers, which is performed by means of Mahalanobis's distance. The objective of such an activity was correction of data for automatic creation of statistic models (standards) based on exponential smoothing methods. The choice of optimal values of the estimated statistical models was realized as minimization of their forecast error. The article also presents a procedure of recalculation (update) of the standard models in case there are permanent changes in the character of the SLCN traffic. The next step is the calculation of the difference value between the forecast in the estimated traffic model and its real variability in order to detect abnormal behavior, which may indicate an attempt of an abuse, for example, a network attack or unauthorized interference in the SLCN infrastructure.

The proposed anomaly/attack detection system based on predictive analysis with the use of exponential smoothing method was evaluated by five attack scenarios. The proposed scenarios have an impact on every layer of PLC communication stack. In order to detect an anomaly/attack, we extracted 10 network features from the PLC traffic network. For all scenarios, we achieved detection rate (DR) values changes from 75.40 to 98.80%, while the false positive ranged from 7.80 to 2.40%. In order to prevent ADS reference models' aging, we added a trigger condition used for reference profiles recalculation. The achieved results are promising and proved that statistical analysis of traffic features with

the use of exponential smoothing models can be useful for anomaly/attack detection and maintenance purposes for smart lights operators.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Centre for Research and Development and also by the National Fund for Environmental Protection and Water Management under the realized GEKON program (Project no. 214093), and it also was supported by the Polish Ministry of Science and High Education and Apator S.A. Company under Contract 04409/C.ZR6-6/2009.

References

[1] IEEE Standards Association, *IEEE Guide for Smart Grid Interoperability of Energy Technology and Information Technology Operation with the Electric Power System (EPS), End-Use Applications, and Loads*, The Institute of Electrical and Electronics Engineers, 2011.

[2] M. Górczewska, S. Mroczkowska, and P. Skrzypczak, "Badanie wplywu barwy swiatla w oswietleniu drogowym na rozpoznawalnosc przeszkód (light color influence on obstacle recognition," *Electrical Engineering*, vol. 73, pp. 165–172, 2013.

[3] H. Schaffers, "Landscape and Roadmap of Future Internet and Smart Cities," 2012.

[4] S. Sun, B. Rong, and Y. Qian, "Artificial frequency selective channel for covert cyclic delay diversity orthogonal frequency division multiplexing transmission," *Security and Communication Networks*, vol. 8, no. 9, pp. 1707–1716, 2015.

[5] IEC 62386-102:2014, Digital addressable lighting interface - Part 102: General requirements - Control gear, 2014.

[6] EN 50065-1:2011, Signalling on low-voltage electrical installations in the frequency range 3 kHz to 148.5 kHz, General requirements, frequency bands and electromagnetic disturbances, 2011.

[7] M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez, "Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: a feasibility study," *IEEE Systems Journal*, vol. 9, no. 1, pp. 31–44, 2015.

[8] P. Kiedrowski, B. Dubalski, T. Marciniak, T. Riaz, and J. Gutierrez, "Energy greedy protocol suite for smart grid communication systems based on short range devices," in *Image Processing and Communications Challenges 3*, vol. 102 of *Advances in Intelligent and Soft Computing*, pp. 493–502, Springer, Berlin, Germany, 2011.

[9] P. Kiedrowski, "Errors nature of the narrowband plc transmission in smart lighting LV network," *International Journal of Distributed Sensor Networks*, vol. 2016, Article ID 9592679, 9 pages, 2016.

[10] A. S. Elmaghraby and M. M. Losavio, "Cyber security challenges in smart cities: Safety, security and privacy," *Journal of Advanced Research*, vol. 5, no. 4, pp. 491–497, 2014.

[11] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[12] M. Esposito, C. Mazzariello, F. Oliviero, S. P. Romano, and C. Sansone, "Evaluating pattern recognition techniques in intrusion detection systems," in *Proceedings of the 5th International Workshop on Pattern Recognition in Information Systems (PRIS'05), in Conjunction with ICEIS 2005*, pp. 144–153, Miami, FL, USA, May 2005.

[13] R. Mitchell and I.-R. Chen, "Behavior-rule based intrusion detection systems for safety critical smart grid applications," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1254–1263, 2013.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Computing Surveys*, vol. 41, no. 3, article 15, 2009.

[15] T. Andrysiak and Ł. Saganowski, *Network Anomaly Detection Basedon ARFIMA Model, Image Processing & Communications Challenges 6, Advances in Intelligent Systems and Computing*, vol. 313, Springer, 2015.

[16] E. H. M. Pena, M. V. O. De Assis, and M. L. Proença, "Anomaly detection using forecasting methods ARIMA and HWDS," in *Proceedings of the 32nd International Conference of the Chilean Computer Science Society, SCCC 2013*, pp. 63–66, November 2013.

[17] G. Galvas, "Time series forecasting used for real-time anomaly detection on websites," 2016, https://beta.vu.nl/nl/Images/stageverslag-galvas_tcm235-801861.pdf.

[18] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: a survey," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1302–1325, 2011.

[19] P. Cheng and M. Zhu, "Lightweight anomaly detection for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 653232, 2015.

[20] K. Ord and S. Lowe, "Automatic forecasting," *The American Statistician*, vol. 50, no. 1, pp. 88–94, 1996.

[21] V. Garcia-Font, C. Garrigues, and H. Rifà-Pous, "A comparative study of anomaly detection techniques for smart city wireless sensor networks," *Sensors*, vol. 16, no. 6, article 868, 2016.

[22] EFT/Burst generator Teseq, http://www.teseq.com/products/NSG-3060.php.

[23] N. Zhou, J. Wang, and Q. Wang, "A novel estimation method of metering errors of electric energy based on membership cloud and dynamic time warping," *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1318–1329, 2017.

[24] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[25] Y. Wang, T. T. Gamage, and C. H. Hauser, "Security Implications of Transport Layer Protocols in Power Grid Synchrophasor Data Communication," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 807–816, 2016.

[26] M. Mahoor, F. R. Salmasi, and T. A. Najafabadi, "A hierarchical smart street lighting system with brute-force energy optimization," *IEEE Sensors Journal*, vol. 17, no. 9, pp. 2871–2879, 2017.

[27] C. Liao, C.-W. Ten, and S. Hu, "Strategic FRTU deployment considering cybersecurity in secondary distribution network," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1264–1274, 2013.

[28] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies," *IEEE Control Systems Magazine*, vol. 21, no. 6, pp. 11–25, 2001.

[29] Y. Wu, C. Shi, X. Zhang, and W. Yang, "Design of new intelligent street light control system," in *Proceedings of the 2010 8th IEEE International Conference on Control and Automation, ICCA 2010*, pp. 1423–1427, June 2010.

[30] T. Macaulay and B. L. Singer, *ICS vulnerabilities. In: Cybersecurity industrial control systems SCADA, DCS, PLC, HMI, SIS [Internet]*, CRC PRESS: Taylor & Francis Group, 2012, https://www.crcpress.com/Cybersecurity-for-Industrial-Control-Systems-SCADA-DCS-PLC-HMI-and/Macaulay-Singer/9781439801963 [Google Scholar].

[31] R. Smoleński, *Conducted Electromagnetic Interference (EMI) in Smart Grids*, Springer, London, UK, 2012.

[32] J. Liu, Y. Xiao, S. Li, W. Liang, and C. L. P. Chen, "Cyber security and privacy issues in smart grids," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 981–997, 2012.

[33] D. M. Hawkins, *Identification of Outliers*, Chapman and Hall, London, UK, 1980.

[34] M. J. Healy, "Multivariate Normal Plotting," *Journal of Applied Statistics*, vol. 17, no. 2, p. 157, 1968.

[35] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 871–880, 1984.

[36] P. Filzmoser, R. Maronna, and M. Werner, "Outlier identification in high dimensions," *Computational Statistics & Data Analysis*, vol. 52, no. 3, pp. 1694–1711, 2008.

[37] R. L. Goodrich, "The Forecast Pro methodology," *International Journal of Forecasting*, vol. 16, no. 4, pp. 533–535, 2000.

[38] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *International Journal of Forecasting*, vol. 18, no. 3, pp. 439–454, 2002.

[39] E. S. Gardner, "Exponential smoothing: the state of the art," *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[40] E. S. Gardner Jr., "Exponential smoothing: the state of the art-part II," *International Journal of Forecasting*, vol. 22, no. 4, pp. 637–666, 2006.

[41] B. C. Archibald, "Parameter space of the holt-winters' model," *International Journal of Forecasting*, vol. 6, no. 2, pp. 199–209, 1990.

[42] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*, vol. 24, Oxford University Press, Oxford, UK, 2001.

[43] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for R," *Journal of Statistical Software* , vol. 27, no. 3, pp. 1–22, 2008.

[44] H. Bozdogan, "Model selection and Akaike's information criterion (AIC): the general theory and its analytical extensions," *Psychometrika*, vol. 52, no. 3, pp. 345–370, 1987.

[45] J. Ramsey and D. Wiley, "Book Reviews : exploratory data analysis John W. Tukey Reading, Mass: Addison-Wesley, 1977, Pps. xvi +688. $17.95," *Applied Psychological Measurement*, vol. 2, no. 1, pp. 151–155, 1978.

[46] National Fund for Environmental Protection and Water Management under the realized GEKON program (project no. 214093).

[47] IEC 61000-4-4, http://www.iec.ch/emc/basic_emc/basic_emc_immunity.htm.

[48] S. McLaughlin, B. Holbert, A. Fawaz, R. Berthier, and S. Zonouz, "A multi-sensor energy theft detection framework for advanced metering infrastructures," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1319–1330, 2013.

[49] Y. Liu, S. Hu, and T.-Y. Ho, "Leveraging strategic detection techniques for smart home pricing cyberattacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 220–235, 2016.

[50] C.-H. Lo and N. Ansari, "CONSUMER: a novel hybrid intrusion detection system for distribution networks in smart grid," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 33–44, 2013.