Hindawi
Security and Communication Networks
Volume 2024, Article ID 9827176, 1 page
https://doi.org/10.1155/2024/9827176

Retraction

Retracted: Software Security Testing through Coverage in Deep Neural Networks

Security and Communication Networks

Received 8 January 2024; Accepted 8 January 2024; Published 9 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] W. Fu and L. Wang, "Software Security Testing through Coverage in Deep Neural Networks," *Security and Communication Networks*, vol. 2022, Article ID 2834982, 7 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 2834982, 7 pages
https://doi.org/10.1155/2022/2834982

Research Article

Software Security Testing through Coverage in Deep Neural Networks

Weiyu Fu [1,2] and Lixia Wang [3,4]

[1] *School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu 221116, China*
[2] *Jiangsu Vocational College of Finance and Economics, Huai'an, Jiangsu 223003, China*
[3] *School of Management, China University of Mining and Technology, Xuzhou, Jiangsu 221116, China*
[4] *School of Business Administration, Henan Polytechnic University, Jiaozuo, Henan 454003, China*

Correspondence should be addressed to Weiyu Fu; 19800341@jscj.edu.cn and Lixia Wang; wanglx@hpu.edu.cn

Received 6 July 2022; Revised 7 August 2022; Accepted 13 August 2022; Published 31 August 2022

Academic Editor: Zhiping Cai

With the continuous progress of society, computer technology and information technology are also experiencing rapid development. Especially in recent years, the application of computer technology has rapidly entered into people's daily life. As people's lives become richer, these applications have become particularly complex. For some large software, tens of thousands of function points or millions of lines of source code may be triggered to support it when performing related tasks. As a result, the security of such a complicated and excellent software becomes quite essential. The most effective way to ensure software security is to test the security of software products during the development process. A precise and effective security testing process is the basis for ensuring that software is tested for security. Without a detailed scientific software security testing model to guide software development for security testing, software security testing will become very difficult. This not only wastes more time and money but also does not guarantee the security of the software. A great security testing methodology should be able to find security problems that may be hidden deep within the software. In addition, a scientific process management can greatly facilitate the implementation of software security testing. As a result, it is relatively meaningful to establish a complete software security testing process model, generate excellent security test cases, and develop security process management tools for software security testing. At the same time, in recent years, deep learning has gradually entered more and more people's lives. However, the widespread application of deep learning systems can bring convenience to human life but also bring some hidden dangers. Hence, deep neural networks must be adequately tested to eliminate as many security risks as possible in some safety-critical software that involves personal and property safety. As the foundation of deep learning systems, deep neural networks should be adequately tested for security. However, deep learning systems are fundamentally different from traditional software testing, so traditional software testing techniques cannot be directly applied to deep neural network testing. In recent years, many scholars in related fields have proposed coverage guidelines based on deep learning testing, but the usefulness of these guidelines is still debatable. Based on the complexity of the large software development process and the fact that the interrelationship between nodes often constitutes a complex network of collaborative relationships, this study applies coverage-based testing in deep neural networks to test the security of software. To be specific, this research applies metrics such as peak coverage, speed to peak, and computational speed to evaluate coverage criteria and to investigate the feasibility of using coverage to guide test case selection to select solutions for security testing.

1. Introduction

With the rapid development of information technology, the third technological revolution has led mankind into the information age. In this context, the pervasive information technology has changed the traditional production, life, and communication methods of human beings [1]. However, the rapid development and unprecedented prosperity of the information industry have also brought negative effects [2]. Software security incidents have occurred more frequently in recent years, bringing not only a lot of inconvenience to the country and people's lives but also a lot of damage. What is worse, the rate of such incidents is also increasing [3]. Nowadays, the security of software is more important than ever, so it is necessary to ensure and use secure software. How to ensure the security of software products has become an important issue that people must address [4]. Software testing is a key technology to ensure the software security and an important method to ensure software quality and reliability. To be specific, software testing is the process of manually or automatically analyzing software to check whether it meets design requirements [5]. As the complexity of software increases and the scale of software expands, the importance of software testing grows. Software security testing is a crucial way to ensure the quality and safety of software and to reduce its own instability [6]. It can prevent security incidents from occurring or minimize damage in the event of an incident. For technical and cost reasons, the focus has been solely on the usability of computers and networks, with no consideration given to security from design to implementation [7]. This has proven to create a variety of security risks for information infrastructures. After all, software is the soul of the information infrastructure [8]. Due to the historical lack of security and the reality of increasingly complex functional requirements, software designs often fail and malfunction, causing huge losses to society and users. Software testing is the process of manually or automatically analyzing software to check whether it meets design requirements [9]. Static source code analysis and dynamic target code runs are used to determine whether the software has the expected functions and properties. Security is a nonfunctional attribute of software, and software testing can identify, locate, and then eliminate software security risks [10]. As a result, software testing is a key technology to ensure the software security and is a necessary security tool in software development and maintenance.

Currently, software security testing is treated the same as regular software testing [11]. Most of the software security testing is done in the context of general software testing. As a result, there is a lack of a specific software security testing process model to guide the security testing process [12]. In addition, there is a lack of tools to manage the security testing process. This is obviously unfair and unscientific for both software security testing and software security quality assurance [13]. As security testing cannot be standardized scientific guidance, then the software security testing cannot be fully carried out, and the quality of software security cannot be strongly guaranteed. A precise security testing process is the basis for ensuring that software is tested for security [14]. Specifically, it not only ensures that software security testing is performed efficiently, saving time and costs but also further improves the security quality of the software. A great security test case can find the security problems that may be hidden in the software, it is the most basic guarantee and basis for security testing [15]. The analysis and design of the security test cases are based on the hazard analysis of the entire software requirements, which is fundamental to the quality of the security test cases. Excellent use cases can be used several times, which can also improve the efficiency and quality of testing [16]. A scientific process management can bring great convenience to the software security testing work and can also improve the efficiency of software security testing, saving manpower and material resources.

In terms of computer system framework, information security includes four levels: physical security [17], operational security [18], data security [19], and content security [20], as shown in Figure 1. Physical security refers to hardware security, which is mainly ensured by the security of the power supply system, hardware reliability, electromagnetic shielding, and personnel management [21]. Operational security refers to software security such as operating systems and application software. To be specific, this includes intrusion detection, vulnerability scanning, virus prevention and control, and emergency response [22]. Data security refers to the security of information in processing, storage, transmission, and use. This process can take authentication, cryptographic encryption, integrity verification, digital signature, and other algorithms and protocols for security reinforcement. Content security refers to the concealment and discovery of the true content of information to ensure its security [23]. This can be achieved through information identification, data mining, information filtering, and information hiding.

With the rapid development of computer technology, more and more software and applications are taking a significant place in human life [24]. These applications not only bring a lot of convenience and efficiency in doing various things but also bring a lot of possibilities to people. In recent years, deep learning and machine learning systems have gained great popularity in various applications, such as speech processing [25], building construction [26, 27, 28], image processing [29, 30], and human traffic detection [31]. Deep neural networks as a deep learning system are the key driver behind the recent success. However, while software systems based on deep neural networks bring convenience to humans, they also bring many serious problems. For example, a few years ago, there was a car accident involving a Google self-driving car in which people lost their lives, and several cases in which self-driving vehicles were unable to handle accidents and corner situations with varying degrees of consequences. All were misbehaviors exhibited by deep neural network software that led to serious consequences. In security-critical and other critical domains, deep neural networks exhibit misbehavior that can lead to irreversible and serious consequences. Therefore, it is necessary to adequately test deep neural networks to avoid misbehavior as much as possible.

At this stage, traditional software testing techniques have gradually matured. However, because of the fundamental difference between traditional software and neural networks, traditional software testing techniques cannot be directly applied to deep neural network testing [32]. In traditional software, each statement in a program performs certain operations that either transform the output from the previous statement to the next statement or change the state of the program. For traditional software, scholars have defined

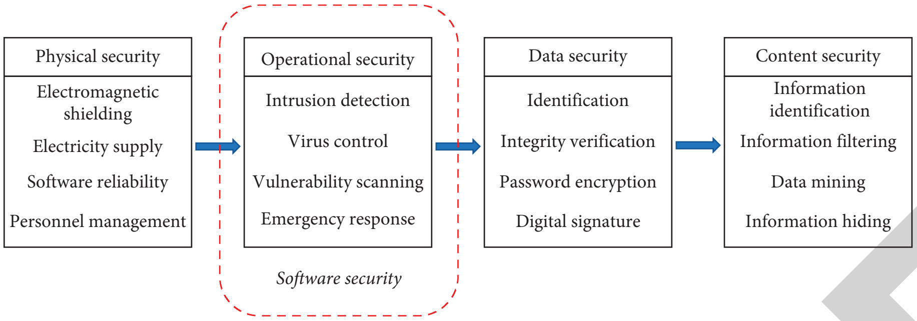

Figure 1: Computer system framework.

many coverage criteria at different levels in order to analyze the runtime behavior of the software from different perspectives. At the code level, there are now statement coverage, branch coverage, data flow coverage, and mutation testing [33]. Among them, statement coverage measures whether the target has executed every instruction. Branch coverage puts the target to test whether each branch of the control structure is covered. Both of these tests are based on control flow statements. Data flow coverage counts whether the definition of each variable is covered and thus detects data flow anomalies. At the model level, there are state coverage and transformation-based coverage. The model-based coverage criterion aims to cover more program behavior through abstract models. The many and varied coverage criteria above can be adapted to different testing methods and granularity.

The objective of testing deep neural networks is to identify potential errors, i.e., erroneous behaviors exhibited at runtime, and correct them in time [34]. In the test experiments, some counter samples are usually artificially added. In other words, samples that cause the model to give incorrect outputs with high confidence should be added to the data set by deliberately adding subtle disturbances. If these adversarial samples are well picked out by the experiment, it can be of great help to the deep neural network testing technique. In order to improve the efficiency of software testing as soon as possible, it is necessary to execute important test cases as early as possible so that defects in the system to be tested can be detected as soon as possible [35]. There are many criteria to measure the importance of a test case, including coverage, runtime, and how well the test case meets the requirements. If the coverage rate of a coverage criterion is used as the measure for test case selection, and more adversarial samples can be found in the selected test cases than in the random selection, then the coverage test can well guide the selection of adversarial samples to better find faults and defects.

As a result, it is of great theoretical and practical significance to establish a scientific and detailed software security testing process model to guide the software security testing in the development process. In the development-oriented process, it can improve the quality of software security testing and ensure that software security testing is carried out smoothly and efficiently, thus further improving the security quality of software. At the same time, based on the established software security testing process model, we design and implement software security testing process management tools, which have important engineering significance. In these contexts, this study investigates coverage testing in deep neural networks. First, some recently proposed coverage criteria will be evaluated, followed by a series of experiments to assess the efficiency of using coverage to guide the selection of test cases for testing the security of software development.

2. Deep Neural Network

Deep learning is a new field in machine learning research, which is motivated by the creation of neural networks that simulate the human brain for analytical learning. The "depth" of deep learning refers to the properties of the flow graph. In a deep neural network, data are input from the input layer and output from the output layer after passing through the hidden layer of the neural network. As a result, the computation involved in the data can be represented by the flow direction, and the property of this flow graph is the depth. Neural networks are the basis of deep learning, which is a technology that simulates the neural network of the human brain in order to achieve machine learning for artificial intelligence.

2.1. Framework of Deep Neural Network. A deep neural network consists of many neurons in several layers. That is, a deep neural network can connect multiple neurons to form a network. Figure 2 shows the most common model of a fully connected neural network. In Figure 2, there is one neuron in the input layer, one neuron in the output layer, and six neurons in the hidden layer. The hidden layer has two layers, each containing three neurons. In general, the number of neurons in the input and output layers is fixed and is determined by the input data and the format of the output. In contrast, the number of hidden layers and the number of neurons in each hidden layer are variable and can be defined by the training programmer. The neuronal interconnections represent the interactions of the neurons, and each connection corresponds to a weight. Therefore, a layer in which all neurons in the graph are connected to all neurons in the neighboring layers is called a fully connected layer.

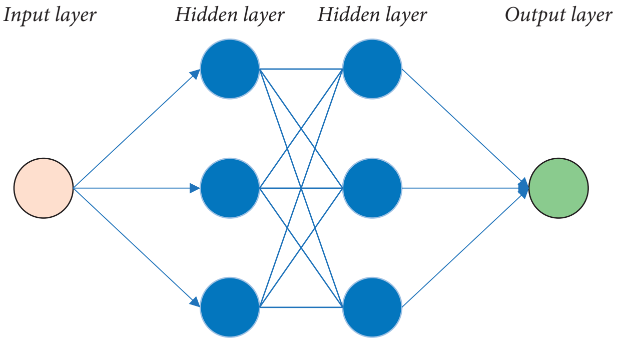

Figure 2: Fully connected neural network.

2.2. Calculation Process of Deep Neural Network. Neural network technology is a simulation of the human brain's nervous system, which consists mainly of neurons and a large number of synapses. As a result, a neuron can be understood as a function containing weights. The neuron in the upper layer passes the data to the neuron, which then performs the computation and passes the result to the neuron in the next layer. After multiple layers of computation, the result is abstracted to a higher level and finally, a computational result is an output based on the processing of multiple layers of neurons. Therefore, the calculation process of a deep neural network can be seen in Figure 3.

2.3. Adversarial Sample. Adversarial samples are mainly used for deep neural network testing. In fact, an adversarial sample is an input sample composed by intentionally adding a very small amount of interference to the data set. Although this sample is only slightly different from the original input sample, the model gives a completely wrong output with a high confidence level. In practice, the error rate of the test set can be reduced by adversarial training. That is, neural network training can be performed on the training set samples of the adversarial perturbation.

For example, in Figure 4, both the left and right images appear to be cats to the human eye, and humans cannot tell the difference between them. However, the right image is actually a sample image obtained by artificially changing a few pixels in the left image. For the model, the left image gives a 43.6% confidence level that it is a cat, while the right image gives an 89.6% confidence level that it is a dog. This is a completely different confidence result. The main reason for the formation of the adversarial sample is the excessive linearity of the model. Neural networks are mainly constructed from linear blocks. In some experiments, the overall function they implement is highly linear. If a linear function has many inputs, then its value can change very quickly. In deep neural networks, if a particular input is changed, even though the change is very small, it can have a huge impact on the result after a multidimensional computation.

2.4. Coverage in Deep Neural Network. In recent years, many deep neural network-based coverage criteria have been proposed by scholars in related fields. These criteria are either for the activation state of a single neuron or for the state of a combination of neurons in the same layer.

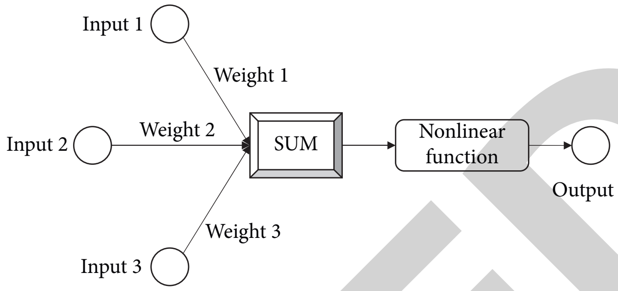

Figure 3: Calculation process of deep neural network.

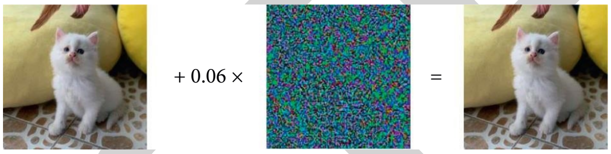

Figure 4: Example of adversarial sample.

This research assumes that the set of neurons in the neural network is $M = \{m_1, m_2, , \ldots\}$, defines the set of test inputs as $K = \{k_1, k_2, \ldots\}$. The value of neuron m at the input k is recorded as $w(m, k)$. The threshold of activation is set to 0.

A neuron m is an active neuron if it is active under one of the test case inputs. The neuron coverage is the percentage of activated neurons to the total neurons. Therefore, the neuron coverage can be defined as follows:

$$MCov(S, k) = \frac{|\{m|\exists k \in S: w(m,k) > 0\}|}{|M|}. \quad (1)$$

When training the model, each neuron m can calculate its upper bound value based on the analysis of the training set data. As a result, the strong activation neuron coverage criterion can be defined as follows:

$$SMACov(S, k) = \frac{|\{m|\exists k \in S: w(m,k) \in (high, \infty\}|}{|M|}. \quad (2)$$

With the widespread use of deep learning, adversarial samples are an increasing threat to deep learning systems. By continuously feeding new types of adversarial samples and performing adversarial training, the robustness of the network can be continuously improved. This method is called brute force adversarial training because of the large amount of training data required. In addition, input gradient regularization can improve the robustness of adversarial attacks. The combination of this method and brute force adversarial training has good results, but there is still the problem of high computational complexity.

3. Software Security Testing

Software security is an engineering approach that enables software to continue to function correctly in the face of malicious attacks. In other words, software security is a systematic, quantitative, and disciplined approach to guiding the construction of secure software. Software security is a key issue in information security. Due to the complexity of software functions and the inherent characteristics of programming languages, software flaws and vulnerabilities are inevitable. With the worldwide availability of the Internet, there have been numerous incidents of hackers exploiting software vulnerabilities to compromise systems via the network. As a result, there are high-risk security vulnerabilities in application and system software. By exploiting these vulnerabilities, attackers are often able to perform a range of unauthorized actions such as system access without local access to the computer. The knowledge architecture of software security is shown in Figure 5.

3.1. Principle of Software Security Testing. Software testing is a complicated system engineering. During software testing, the tester must be well aware of the basic principles of software testing. Inadequate testing is a foolish act, while excessive testing of software is a sin. In general, the basic principles of software security testing are as follows.

Firstly, the most serious mistake made during software development is the failure of a software system to meet user requirements. Problems identified during system development can occur at some point in the early stages of development, so correcting errors must be done retroactively. In addition, software testing should be implemented early, preferably in the requirements phase. After all, the most unacceptable error is the failure of a system to meet user requirements. Figure 6 represents the general pattern of software vulnerabilities.

Therefore, if problems are identified in a timely manner, the smaller the cost of solving the problem, which is the golden rule of software development.

3.2. Requirement of Software Security Testing. Software security testing is a specific process for solving problems that has been developed and tested over a long period of time in software development practice. It can guide the software development process from a macro perspective, control the risks in the software development process, shorten the software development cycle, and improve the quality of software. It is with the software security testing model that the cost of software can be predicted scientifically and the quality of software can be controlled reasonably. As a result, the software security testing should have the requirements as shown in Figure 7.

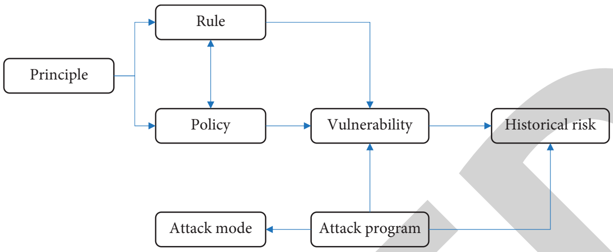

Figure 5: Knowledge architecture of software security.

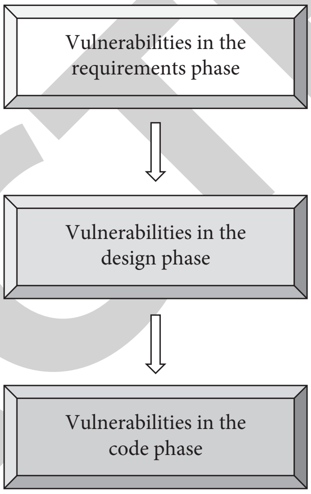
Vulnerabilities in the requirements phase
Vulnerabilities in the design phase
Vulnerabilities in the code phase

Figure 6: General pattern of software vulnerabilities.

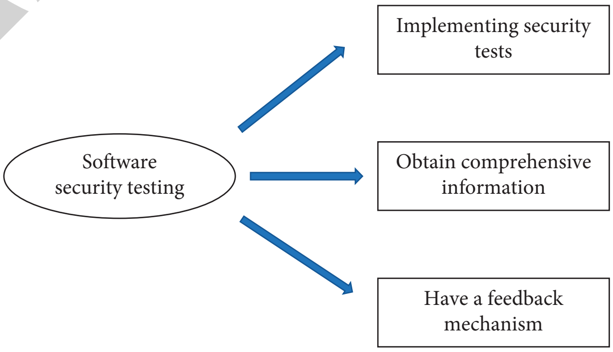
Software security testing
Implementing security tests
Obtain comprehensive information
Have a feedback mechanism

Figure 7: Requirements of software security testing.

3.3. Trajectory Model Design. The design of the software security testing process model aims to improve the confusion and confusion of software security testing in the software development process, and thus improve the efficiency of software security testing in the software development process. Based on the idea of the software development process and common testing model, as well as the analysis of software security and security testing content, a scientific security testing process model needs to be designed. The model starts from the perspective of software engineering and system security and links the software development process to the system security testing process. Then, the development process is analyzed and prepared accordingly, and security testing is performed for each of its process hazards.

In each subprocess of the software development process, its security is first analyzed. Once the security analysis is complete and ready, security testing is performed on it. This is a good way to deal with the security testing lag that can occur in software security testing. Each subprocess of the security testing process consists of several small cycles of testing activities, which can well solve the problem of frequent alternation and iteration of security testing.

Software security testing activities exist throughout the life cycle of software product development and are conducted simultaneously with the development process. The model is characterized by its ability to show not only the software development process but also to support the independence of the software development process from the software security testing process. Thus, the software development process and the software security testing process can be performed independently of each other. The development activity is an iterative process, and the security testing activity is an iterative security testing process that follows the development activity.

4. Conclusion

With the rapid development of computer technology, deep learning and deep neural networks are becoming more and more important in people's daily life. As deep learning systems are used in critical areas where security is important, it is even more necessary to test their security. Obviously, if the system behaves completely wrongly due to some errors in the neural network in critical areas such as autonomous driving, automatic medical diagnosis, and face recognition systems, the results would be unthinkable. Although the testing of traditional software has become increasingly sophisticated, there are fundamental differences between deep neural networks and traditional software. As a result, traditional software testing techniques cannot be directly applied to deep neural network testing. In this context, how to test deep neural networks and how to judge the adequacy of a test has become a recent challenge. In this context, this paper introduces the basic concepts of neural networks from the structure, computational process, and adversarial samples. Then, this research introduces the testing of artificial neural networks, focusing on the testing framework of deep learning systems.

Although some theoretical and technical progress has been made in this study, there are still many problems that need to be further investigated. The following work needs to be further developed and studied in depth. For the dynamic knowledge base, the number of samples in the database should be expanded to improve the accuracy of behavior determination. The attack tree model is further refined to be closer to the realistic attack situation in terms of weight setting. Also, in addition to finding adversarial samples, efforts can also be focused on generating adversarial samples. Knowing which samples make the system make wrong judgments and thus improve the system is also a possible direction to improve the robustness of deep neural networks.

Data Availability

The labeled dataset used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] M. Salahshour Rad, M. Nilashi, and H. Mohamed Dahlan, "Information technology adoption: a review of the literature and classification," *Universal Access in the Information Society*, vol. 17, no. 2, pp. 361–390, 2018.

[2] T. Alam, "Cloud computing and its role in the information technology," *IAIC Transactions on Sustainable Digital Innovation*, vol. 1, no. 2, pp. 108–115, 2020.

[3] H. Mumtaz, M. Alshayeb, S. Mahmood, and M. Niazi, "An empirical study to improve software security through the application of code refactoring," *Information and Software Technology*, vol. 96, pp. 112–125, 2018.

[4] L. Ben Othmane, G. Chehrazi, E. Bodden, P. Tsalovski, and A. D. Brucker, "Time for addressing software security issues: prediction models and impacting factors," *Data Science and Engineering*, vol. 2, no. 2, pp. 107–124, 2017.

[5] O. Bin Tauqeer, S. Jan, A. Omar Khadidos, A. Omar Khadidos, F. Qudus Khan, and S. Khattak, "Analysis of security testing techniques," *Intelligent Automation & Soft Computing*, vol. 29, no. 1, pp. 291–306, 2021.

[6] B. Aloraini, M. Nagappan, D. M. German, S. Hayashi, and Y. Higo, "An empirical study of security warnings from static application security testing tools," *Journal of Systems and Software*, vol. 158, Article ID 110427, 2019.

[7] P. Zech, M. Felderer, and R. Breu, "Knowledge-based security testing of web applications by logic programming," *International Journal on Software Tools for Technology Transfer*, vol. 21, no. 2, pp. 221–246, 2019.

[8] D. He, Q. Qiao, J. Gao, S. Chan, K. Zheng, and N. Guizani, "Simulation design for security testing of integrated electronic systems," *IEEE Network*, vol. 34, no. 1, pp. 159–165, 2020.

[9] P. Ping, Z. Xuan, and M. Xinyue, "Research on security test for application software based on spn," *Procedia Engineering*, vol. 174, pp. 1140–1147, 2017.

[10] L. Ma, H. Yang, J. Xu, Z. Yang, Q. Lao, and D. Yuan, "Code analysis with static application security testing for Python program," *Journal of Signal Processing Systems*, pp. 1–14, 2022.

[11] L. M. Alrawais, M. Alenezi, and M. Akour, "Security testing framework for web applications," *International Journal of Software Innovation*, vol. 6, no. 3, pp. 93–117, 2018.

[12] M. Peroli, F. De Meo, L. Viganò, and D. Guardini, "MobSTer: a model-based security testing framework for web applications," *Software Testing, Verification and Reliability*, vol. 28, no. 8, p. e1685, 2018.

[13] O. M. Pisareva, V. A. Alexeev, D. N. Mednikov, A. V. Starikovsky, and V. B. Kurguzov, "Creating a national certification system for unmanned vehicles: tasks of information security testing," *St. Petersburg State Polytechnical University Journal. Economics*, vol. 14, no. 2, pp. 63–80, 2021.

[14] N. M. Mohammed, M. Niazi, M. Alshayeb, and S. Mahmood, "Exploring software security approaches in software development lifecycle: a systematic mapping study," *Computer Standards & Interfaces*, vol. 50, pp. 107–115, 2017.

[15] M. Alenezi and S. Almuairfi, "Security risks in the software development lifecycle," *International Journal of Recent Technology and Engineering*, vol. 8, no. 3, pp. 7048–7055, 2019.

[16] M. Cheah, S. A. Shaikh, J. Bryans, and P. Wooderson, "Building an automotive security assurance case using systematic security evaluations," *Computers & Security*, vol. 77, pp. 360–379, 2018.

[17] B. Ali and A. I. Awad, "Cyber and physical security vulnerability assessment for IoT-based smart homes," *Sensors*, vol. 18, no. 3, p. 817, 2018.

[18] R. Fu, X. Huang, Y. Xue, Y. Wu, Y. Tang, and D. Yue, "Security assessment for cyber physical distribution power system under intrusion attacks," *IEEE Access*, vol. 7, pp. 75615–75628, 2019.

[19] K. Abouelmehdi, A. Beni-Hssane, H. Khaloufi, and M. Saadi, "Big data security and privacy in healthcare: a Review," *Procedia Computer Science*, vol. 113, pp. 73–80, 2017.

[20] N. Kaaniche and M. Laurent, "Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms," *Computer Communications*, vol. 111, pp. 120–141, 2017.

[21] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K. K. R. Choo, "Blockchain: a panacea for healthcare cloud-based data security and privacy?" *IEEE Cloud Computing*, vol. 5, no. 1, pp. 31–37, 2018.

[22] J. Ma, W. Kong, and Y. Lei, "Data security and privacy information challenges in cloud computing," *International Journal of Computational Science and Engineering*, vol. 16, no. 3, pp. 215–218, 2018.

[23] S. Giri and S. Shakya, "Cloud computing and data security challenges: a Nepal case," *International Journal of Computer Trends & Technology*, vol. 67, no. 3, pp. 146–150, 2019.

[24] R. Sahay, W. Meng, and C. D. Jensen, "The application of software defined networking on securing computer networks: a survey," *Journal of Network and Computer Applications*, vol. 131, pp. 89–108, 2019.

[25] R. Haeb-Umbach, S. Watanabe, T. Nakatani et al., "Speech processing for digital home assistants: combining signal processing with deep-learning techniques," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 111–124, 2019.

[26] B. Cheng, K. Lu, J. Li, H. Chen, X. Luo, and M. Shafique, "Comprehensive assessment of embodied environmental impacts of buildings using normalized environmental impact factors," *Journal of Cleaner Production*, vol. 334, Article ID 130083, 2022.

[27] B. Cheng, C. Fan, H. Fu, J. Huang, H. Chen, and X. Luo, "Measuring and computing cognitive statuses of construction workers based on electroencephalogram: a critical review," *IEEE Transactions on Computational Social Systems*, pp. 1–16, 2022.

[28] B. Cheng, J. Huang, J. Li, S. Chen, and H. Chen, "Improving contractors' participation of resource utilization in construction and demolition waste through government incentives and punishments," *Environmental Management*, pp. 1–15, 2022.

[29] L. Jiao and J. Zhao, "A survey on the new generation of deep learning in image processing," *IEEE Access*, vol. 7, pp. 172231–172263, 2019.

[30] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: a survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.

[31] Y. Qian, S. Chen, J. Li et al., "A Decision-Making Model Using Machine Learning for Improving Dispatching Efficiency in Chengdu Shuangliu Airport," *Complexity*, vol. 9, 2020.

[32] S. Naseer, Y. Saleem, S. Khalid et al., "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[33] R. Geirhos, J. H. Jacobsen, C. Michaelis et al., "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, no. 11, pp. 665–673, 2020.

[34] S. Feng, H. Zhou, and H. Dong, "Using deep neural network with small dataset to predict material defects," *Materials & Design*, vol. 162, pp. 300–310, 2019.

[35] Y. Jia, M. Wang, and Y. Wang, "Network intrusion detection algorithm based on deep neural network," *IET Information Security*, vol. 13, no. 1, pp. 48–53, 2019.