Hindawi
Security and Communication Networks
Volume 2020, Article ID 7672941, 16 pages
https://doi.org/10.1155/2020/7672941

Review Article

Classification and Analysis of Security Techniques for the User Terminal Area in the Internet Banking Service

Kyungroul Lee,[1] Sun-Young Lee ,[2] and Kangbin Yim [2]

[1]*R&BD Center for Security and Safety Industries (SSI), Soonchunhyang University, Asan-si 31538, Republic of Korea*
[2]*Department of Information Security Engineering, Soonchunhyang University, Asan-si 31538, Republic of Korea*

Correspondence should be addressed to Kangbin Yim; yim@sch.ac.kr

Received 2 January 2020; Revised 18 March 2020; Accepted 21 May 2020; Published 5 June 2020

Academic Editor: Cristina Alcaraz

As the Internet banking service has become popular, many users have been able to exchange goods online. Even though this offers advantage, there are cases of incidents in the Internet banking service arising from security threats. In order to counteract this problem, various security techniques have been applied over the whole area of the Internet banking service. In this paper, we analyze the results of security techniques applied in the user terminal area. In detail, we classify the existing security technologies into network domain, financial institution domain, and user terminal domain, to ensure the security of the Internet banking service. This paper focuses on security techniques in the user terminal domain. Specifically, we classify the security technologies in the user terminal domain into secure keyboard program, PKI applications, E2E encryption, antihacking program, personal firewall, removable media security, and antireverse engineering technique and describe detailed and key techniques of each security technology. For easy understanding and security analysis of security technologies, we describe each security technology in detail and explain each technology by way of example. We consider that this paper will provide criteria for evaluating the security of Internet banking services.

1. Introduction

Security crimes occur in Internet banking service. To regulate these incidents, laws related to the Internet banking service have been enacted, such as electronic signature law; the laws are a means of taking action after an incident but, by themselves, do not prevent such incidents. For this reason, various security technologies have been studied to provide security for the online identification methods utilized in the Internet banking service. Therefore, in this paper, we classify the existing security technologies into network domain, financial institution domain, and user terminal domain, to ensure the security of the Internet banking service, and describe in detail the surveyed results of the security techniques applied to the user terminal domain.

2. Classification of Security Technologies

The first Internet banking incident in South Korea occurred in May 2005. To counteract such threats, security companies adopted security technologies, such as secure keyboard software and PKI application software. Figure 1 shows that these technologies evolve every time major hacking crimes occur [1]. Figure 2 shows that security technologies, such as electronic devices, including user PCs, wired/wireless based networks, Internet banking servers, authentication servers, and bank hosts, are currently studied, in order to cope with various threats.

Specifically, security technologies for electronic devices, such as user PCs, are classified into secure keyboard program, PKI applications, E2E encryption, antihacking program, personal firewall, removable media security, and antireverse engineering techniques. The secure keyboard program is a program to protect the information input from a keyboard device, in order to prevent sensitive information exposure related to the account and transaction input by the user to the attacker. PKI applications are applications that deploy cryptography technology to securely transfer account and transaction related information across the network between the user terminal domain and the financial

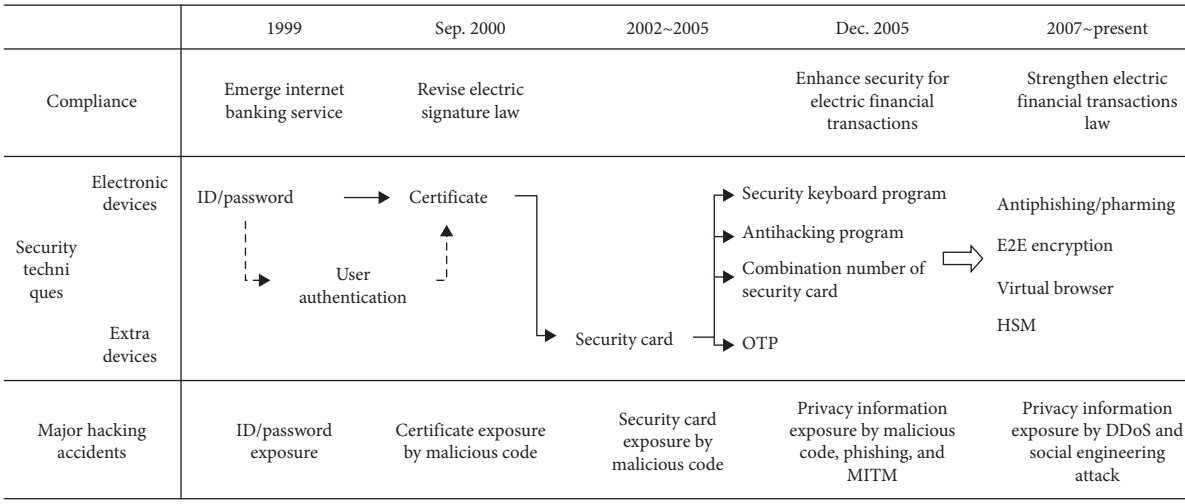

Figure 1: Development process of security technologies for the Internet banking service.

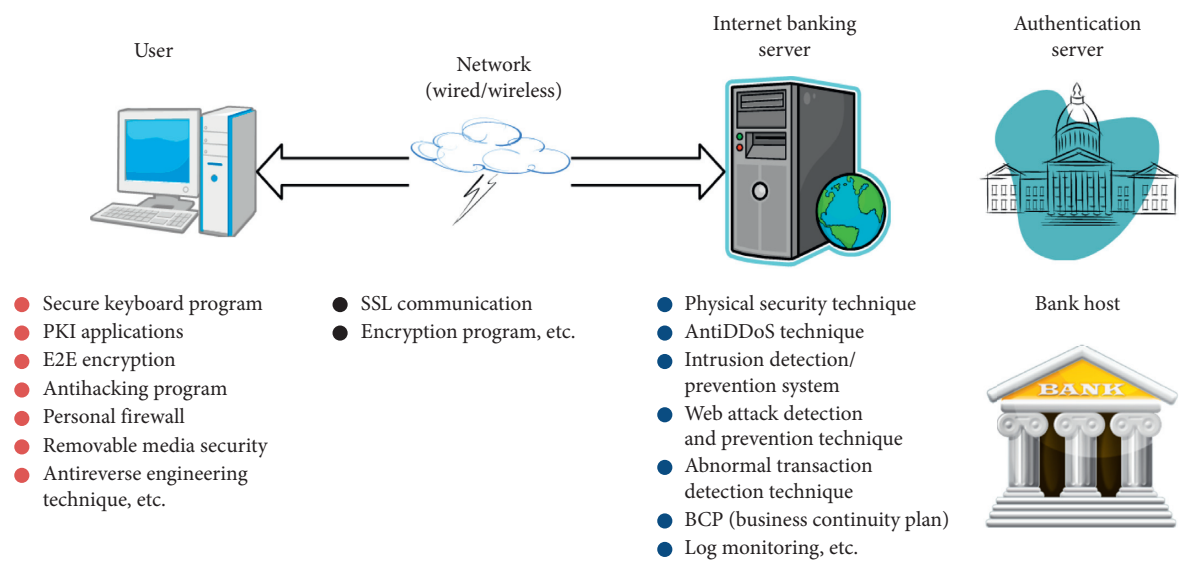

Figure 2: Classification of security techniques for the Internet banking service.

institution domain. E2E encryption is a method of applying end-to-end encryption technology to securely transfer information related to accounts and transactions input by the user to the server, in order to solve the problems caused by the interworking between the secure keyboard program and PKI applications. The antihacking program is a program that detects and prevents attacks, in order to protect the user's electronic devices from external hacking; if an attacker has already penetrated the user terminal, this program neutralizes that attack. The personal firewall is a firewall program that detects and blocks malicious packets passing through the network, to detect and prevent intrusion from the outside. Removable media security is a security technology for protecting sensitive data, such as certificates stored on removable media. Antireverse engineering techniques are prevention techniques to protect software, in order to prevent attacks from finding program vulnerabilities using reverse engineering techniques. Security technologies for wired/wireless networks are classified into SSL communication and encryption programs; security technologies for financial institutions, including Internet banking servers, authentication servers, and bank hosts, are classified into physical security technique, anti-DDoS technique, Intrusion detection/prevention system, web attack detection and prevention technique, Business Continuity Plan (BCP), and log monitoring. By applying all these security technologies, the Internet banking service ensures security and safety from threats for the user terminal domain, the network domain, and the financial institution domain.

3. Security Techniques in the User Terminal Domain

Security techniques in the user terminal domain are classified into secure keyboard program, PKI applications, end-to-end encryption, antihacking program, personal firewall, removable media security, and antireverse engineering

technique. Attackers generally concentrate their attack target on the user terminal domain, and for this reason, many techniques have been studied to counteract attacks. Because the network and financial institution domains are cryptographically secure and have an environment that is not open to the public, attackers have to spend a lot of time and effort trying to attack. Hence, they lack enthusiasm, or do not need to attack, because of the difficulty in penetrating these domains. Table 1 shows the classification of security techniques for the user terminal domain.

In detail, security techniques for the secure keyboard program are classified into message hooking, filter driver insertion, interrupt object replacement, IDT replacement, generation of random scan code using 0xD2 command, generation of random scan code using keyboard internal memory, utilization of debug exception handler, and inline hooking. PKI applications include security techniques such as self-design and library utilization. E2E encryption includes security techniques such as interlocking secure keyboard program and PKI application using encryption and decryption modules, and double encryption using only encryption module. Antihacking programs include security techniques such as anti-virus products. Personal firewall includes security techniques such as program access management, IP address access management, and network connection management. Removable media security includes security techniques such as software approach, hardware-based partitioning approach, and hardware approach. Antireverse engineering techniques include security techniques such as layout obfuscation, data obfuscation, control obfuscation, and preventive obfuscation.

3.1. Secure Keyboard Program. In the Internet banking service, an input device is indispensable for inputting the transaction information or the authentication information; generally, a keyboard is used for such inputting. However, the information input from the keyboard can be easily exposed by an attacker, so a method to protect the information is studied. Table 2 shows that, in South Korea, various vendors provide secure keyboard programs based on their techniques [2]. The surveyed secure keyboard programs mainly focus on products certified by the Telecommunications Technology Association (TTA) in South Korea, and the vendor, certification number, and features are briefly described.

Company A has security features that include keyboard hooking prevention using self-implemented keyboard driver, keyboard hacking prevention using tunnelling, PIC/AIOAPIC security, port scanning prevention, hacking prevention of keyboard input information using BHO, and kernel area hacking prevention. Company B has security features that include encrypt keyboard input data, hacking detection tools that include automatic key recorder and USB monitoring tool, hacking detection tools such as the keyboard driver control feature, and control keyboard driver. Company C has security features that include PS/2 keyboard security feature, USB keyboard security feature, and wireless USB keyboard security feature. Company D has security features that include driver security, encrypting keyboard input data, and self-protection. Company E has security features that include EasyKeyTec notification feature, secure edit control feature, virtual keyboard feature, using 128 bits AES algorithm, and support USB and wireless keyboards.

The keyboard is connected to a PS/2 interface or USB interface, and the structure and transferal process from the keyboard to the application program are analyzed. Figure 3 shows the process in which a user inputs a key from a PS/2 keyboard and then passes it to the application program [3].

The PS/2 keyboard architecture consists of a key matrix and a keyboard processor inside the keyboard, a keyboard controller and host processor inside the host, a programmable interrupt controller (PIC) and advanced programmable interrupt controller (APIC) inside the host processor, and device drivers and applications within the internal PS/2 keyboard device stack in the operating system.

When a user inputs a key from the PS/2 keyboard, the keyboard processor inside the keyboard extracts the scan code from the key matrix, and then the scan code is transmitted to the keyboard controller inside the host. The keyboard controller receives the transferred scan code and then sends it to PIC and APIC to request the keyboard interrupt service. PIC and APIC are the controllers to route interrupts that occur from various devices. Interrupt signals input from the keyboard is processed through PIC, input/output (I/O) APIC, local APIC, etc., and then the keyboard interrupt is generated to the CPU. The CPU prepares specific tables and handlers for input and output processing, which are termed the interrupt descriptor table (IDT) and interrupt service routine (ISR). The keyboard interrupt calls the keyboard interrupt service routine through an entry assigned in association with the keyboard interrupt in the tables, and then the input keyboard data is passed to the application program through the PS/2 keyboard device stack in the operating system.

Figure 4 shows the transmission process of keyboard data using the USB keyboard after a user inputs a key. The USB keyboard structure consists of a USB host controller, a host processor, a USB device stack, and an application program. The USB keyboard is also associated with interrupts, but it does not generate the keyboard interrupt differently from the PS/2 keyboard. Actually, the USB host controller generates the USB keyboard interrupt by periodically reading the transfer descriptor (TD), by which data is transferred from the USB device. Finally, the USB host controller reads the keyboard data transmitted from the USB keyboard and sends it to the host processor, and then the operating system transmits the keyboard data to the application program through the USB device stack. In detail, the host controller provides control registers, status registers, and pointers to manage the transfer buffers for each transfer method defined in the USB specification, such as pHCCA, pControl, and pBulk. The host controller also provides frame counter registers (FrmCounters) for updating frame numbers and RHregs for controlling the root hub. Control registers contain bits to allow or prohibit transmission at the next frame time for control, bulk, interrupt, and isochronous transfers, set the service rate for

Table 1: Security techniques for the user terminal domain.

Major category	Subcategory	Security techniques
Secure keyboard program	PS/2 keyboard	(i) Message hooking (ii) Inserting filter driver (iii) Interrupt object replacement (iv) IDT replacement (v) Generating random scan code using 0xD2 command (vi) Generating random scan code using keyboard internal memory (vii) Using the debug exception handler
	USB keyboard	(i) Inserting filter driver (ii) Inline hooking
PKI applications	Secure channel	(i) Self-design (ii) Using the crypto library
E2E encryption	Initial E2E	(i) Interlocking secure keyboard program and PKI applications (encryption/decryption module)
	Extended E2E	(i) Double encryption (only encryption module)
Antihacking program	Pattern-based	(i) Anti-virus products
Personal firewall	Behavior-based	(i) Program access management (ii) IP address access management (iii) Network connection management
Removable media security	Secure USB	(i) Software approach (ii) Hardware-based partitioning approach (iii) Hardware approach
Antireverse engineering technique	Layout obfuscation Data obfuscation Control obfuscation Preventive obfuscation	(i) Changing format, removing comment, scrambling identifiers (ii) Data storage, data encoding, data > aggregation, data ordering (iii) Computation transformation, aggregation transformation (iv) Unique transformation method, targeted transformation method

Table 2: Certified secure keyboard software.

Company	Software	Certification number	Features
Company A	ClientKeeper KeyPro v2.2	07-0073	(i) Keyboard hooking prevention using self-implemented keyboard driver (ii) Keyboard hacking prevention using tunnelling (ii) PIC/AIOAPIC security (iv) Port scanning prevention (v) Hacking prevention of keyboard input information using BHO (vi) Kernel area hacking prevention (vii) Encrypt keyboard input data (viii) Hacking detection tools that include automatic key recorder and USB monitoring tool
Company B	nProtectKeyCrypt v6.0	09-0108	(i) Hacking detection tools that include the keyboard driver control feature (ii) Control keyboard driver
Company C	Secure Keystroke v5.0	11-0028	(i) PS/2 keyboard security feature (ii) USB keyboard security feature (iii) Wireless USB keyboard security feature
Company D	K-Defence R6	11-0032	(i) Driver security (ii) Encrypt keyboard input data (iii) Self-protection, etc.
Company E	EasyKeyTec v1.0	—	(i) EasyKeyTec notification feature (ii) Secure edit control feature (iii) Virtual keyboard feature (iv) Using 128 bits AES algorithm (v) Support USB and wireless keyboards

control transfers and bulk transfers, manage the status of the state machine of the host controller for the host controller drive, and provide support for power management. Therefore, when a keyboard interrupt occurs, the USB host controller stores information related to the host controller communication area (HCCA) in the control registers, and the HCCA contains the head pointers to the interrupt Endpoint Descriptor lists. Finally, these interrupts are scheduled to generate a TD in which keyboard interrupt information is stored.

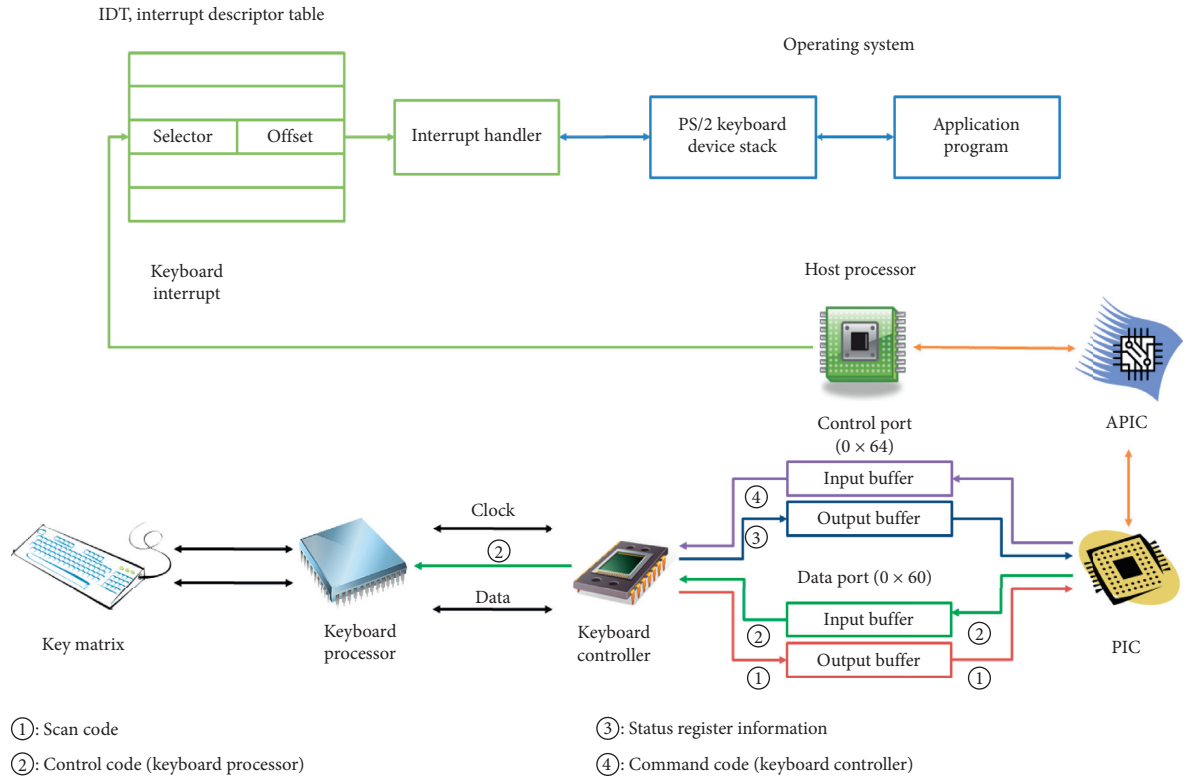

Figure 3: Structure and flow of the PS/2 keyboard.

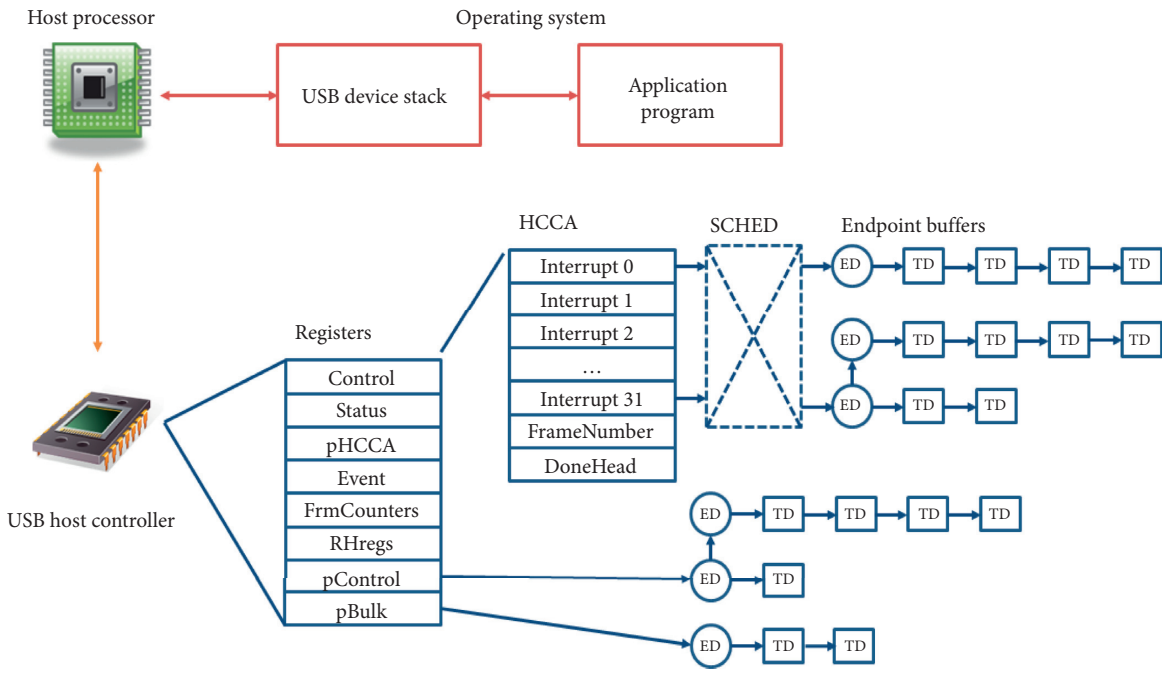

Figure 4: Structure and flow of the USB keyboard.

PS/2 keyboard security techniques are message hooking, inserting the filter driver, interrupt object replacement, IDT replacement, generating random scan code using the 0xD2 command, generating random scan code using the keyboard internal memory, and using the debug exception handler. USB keyboard security techniques are inserting the filter

driver and inline hooking, and Table 3 shows the details of PS/2 keyboard and USB keyboard security techniques [2, 4, 5].

3.2. PKI Application. The PKI application is a program implemented in the user terminal domain to apply the cryptography techniques for the network domain; for the details of the security techniques in the network domain, refer to the literature [6].

3.3. E2E Encryption. Input information from the keyboard by a user is difficult to safely transmit to the Internet banking server by using the secure keyboard program and PKI application described above. This is a problem that occurs when each program is not interworked with the others. For this reason, the end-to-end encryption technique was proposed for interworking between those security techniques. The initial end-to-end encryption protects the keyboard information input by the user using the substitution method or encryption method, and then the PKI application encrypts the protected information and sends it to the network to securely transmit the protected information to the Internet banking server. This technique is classified into substitution and encryption, depending on the method of the secure keyboard program. The substitution method replaces the keyboard information with the substitution table generated by the random number generator, and the encryption method encrypts the keyboard information based on the generated key [7].

In the internal operation process of the E2E encryption, the information as the transaction information or the authentication information input from the keyboard counteracts the security threats that can occur between the device driver and the application program, by transforming the information into a substitution or encryption method in the secure keyboard program. The transformed information is extracted by resubstitution or decryption to be transmitted to the PKI application, and the decrypted plaintext is encrypted, based on a shared key between the PKI application and the secure keyboard program in the secure keyboard program, and is sent to the PKI application. In the PKI application, the encrypted information received from the secure keyboard program is decrypted, and then the decrypted plaintext is sent to the Internet banking server. Finally, the PKI application encrypts and sends the decrypted plaintext to the Internet banking server based on a shared key between the Internet banking server and PKI application, and then the server receives the information and decrypts it; consequently, the server extracts the information input by the user [8].

However, there are several vulnerabilities of the initial end-to-end encryption. The first vulnerability is that there is exposure of the plaintext in the secure keyboard program and PKI application, while the second vulnerability is that the shared session key between the secure keyboard program and PKI application is used as a fixed key. Lastly, there is the vulnerability that the key sharing method is insecure. For these reasons, an extended end-to-end encryption technique has been proposed [4]. The extended end-to-end encryption technique includes only an encryption module in the secure keyboard program and PKI application to prevent decryption of the encrypted keyboard information in the user terminal domain, and the Internet banking server includes only a decryption module to prevent the exposure of plaintext in the user terminal domain. This technique deals with a dual encryption method in which the keyboard information is transformed into a substitution or encryption method in the secure keyboard program, and then the transformed information is transformed again in the PKI application. More specifically, the secure keyboard program and the Internet banking server share a session key or a seed value to generate a substitution table, and then the keyboard information input by the user is transformed based on a shared substitution table or session key between the Internet banking server and the secure keyboard program, and then the transformed information is transformed again, based on a session key between the Internet banking server and the PKI application [5]. Figure 5 shows a flowchart of the initial end-to-end encryption and extended end-to-end encryption.

3.4. Antihacking Program. The antihacking program is a program that protects the user terminal domain from hacking, such as a commonly known anti-virus product, and detects and neutralizes hacking programs when the user terminal is already infected. In general, known attack codes are registered in the database, and if an unknown code is the same as the registered attack code, this is determined to be an attack, and the attack is blocked. This kind of technique is called a pattern-based technique [9, 10]. These techniques have already been studied and have a wide range, so for the detailed description, refer to the literature.

3.5. Personal Firewall. The personal firewall is a technique to prevent security threats by inspecting packets and blocking them, firstly, when they are malicious packets coming from the outside to the inside of the network, to prevent intrusion from the outside, and secondly, when they are packets from the inside to the outside, to prevent the leakage of important internal information. The personal firewall provided by the financial institution is mainly based on the behavior-based method, in which trusted programs are allowed access to the network, whereas untrusted programs are blocked or must request permission from the user. Table 4 shows the basic features of this personal intrusion prevention program (IPS) [5].

The basic features of intrusion prevention program (IPS) are classified into personal firewall, shared folder control/monitoring, network, real-time hacking tool detection/prevention, and logging. The personal firewall is classified into program access management, IP address access management, and port access management. The program access management feature is to notify the user by detecting the execution of the program communicating with the outside

Table 3: Security techniques for the PS/2 and USB keyboards.

Interface	Security techniques	Description
PS/2	Message hooking	Preempts keyboard data by hooking keyboard message
	Inserting the filter driver	Preempts keyboard data by inserting filter driver inside the PS/2 keyboard driver stack
	Interrupt object replacement	Preempts keyboard data by replacing the object processing interrupt related PS/2 keyboard
	IDT replacement	Preempts keyboard data by replacing the interrupt descriptor table related PS/2 keyboard
	Generating random scan code using the 0xD2 command	Disrupts keyboard data by generating random scan code using the 0xD2 command
	Generating random scan code using the keyboard internal memory	Disrupts keyboard data by generating random scan code stored in the keyboard internal memory
	Using the debug exception handler	Preempts keyboard data using the debug exception handler when accessing input and output memories or ports related to the PS/2 keyboard
USB	Inserting filter driver	Preempts keyboard data by inserting filter driver inside the USB keyboard driver stack
	Inline hooking	Preempts keyboard data by hooking function processing the USB keyboard data

of the user's PC and allow/block settings by IP address and port. IP address access management is to allow or block designated IP addresses. Finally, port access management is to allow or block designated ports. The shared folder control/monitoring feature is classified into shared access management and shared folder management. Shared access management is to allow or block access to shared folder on the user's computer from other computers. Shared folder management is to disable the shared folder. The network feature is classified into network connection management and network monitoring. Network connection management is to display the allowed TCP/UDP connection information for external communication, alert or block the port when it is a known hacking tool, and counteract self-disguised and hidden hacking tools when detection is difficult. Network monitoring is to graphically display the amount of network inbound and outbound traffic. Real-time hacking tool detection/prevention is classified into known hacking tool detection and prevention. Known hacking tool detection and prevention is to inspect hacking tool process, detect and block key loggers, detect and block console-based hacking programs, and detect and block the Back Orifice hidden process. Logging is classified into usage history. Usage history is to record program access management information and display logs history by date.

3.6. Removable Media Security. The majority of computer peripherals today use the USB interface as a protocol for communication between a host and peripherals. There are peripherals using the USB interface, such as keyboard, mouse, printer, and removable media, and as usage increases, interest in data protection is increasing. In particular, the keyboard and the removable media contain sensitive privacy data, so more attention is paid to the protection of the information.

Because of portability and compatibility, these USB memory devices are now the most common auxiliary storage medium. However, the USB interface for connecting USB memory does not support the security feature at all; for this reason, various application devices connected via the USB interface have weak security. USB memory is the most popular device connected through the USB interface that is vulnerable; it is impossible to protect the information stored in the memory in an environment in which the countermeasures do not provide against security threats, such as general USB devices. Security issues have been raised to counteract these problems, which are especially caused by the fact that extremely personal and confidential information is stored in the USB memory, and security and theft are frequently caused by portability when lost.

In response to the demand for security enhancement of the USB memory as described above, various types of secure USB memory have been developed in recent years and popularly implemented by the technology that has been developed to protect data transmitted via USB interface and to protect data stored in the USB memory. As a technique for protecting the information stored in the USB memory, there are several protection methods; one method is authenticating a legitimate user using the ID and password in the host, and then the use of the USB memory is allowed; while another method is encrypting data for protection by the application program, and then the encrypted data is stored in the USB memory through the standard USB protocol. Moreover, various and compositive security techniques are researched; one method is to protect the storing data by isolating the data storage area of USB memory as a secure area, while another method is to allow access to the secure area of USB memory by recognizing the fingerprint of the user in the USB memory itself. Therefore in this paper, we focus on secure USB among removable media.

Since April 2008, secure USB, which is mandatory in public institutions, has launched products based on their security techniques, such as nProtect, Jiransecurity, Chaewool, and overseas dotCure. According to the "Guidelines for Security Management of Auxiliary Storage Media such as USB memory" [11] by the National Intelligence Service (NIS) in South Korea, the following four features must be

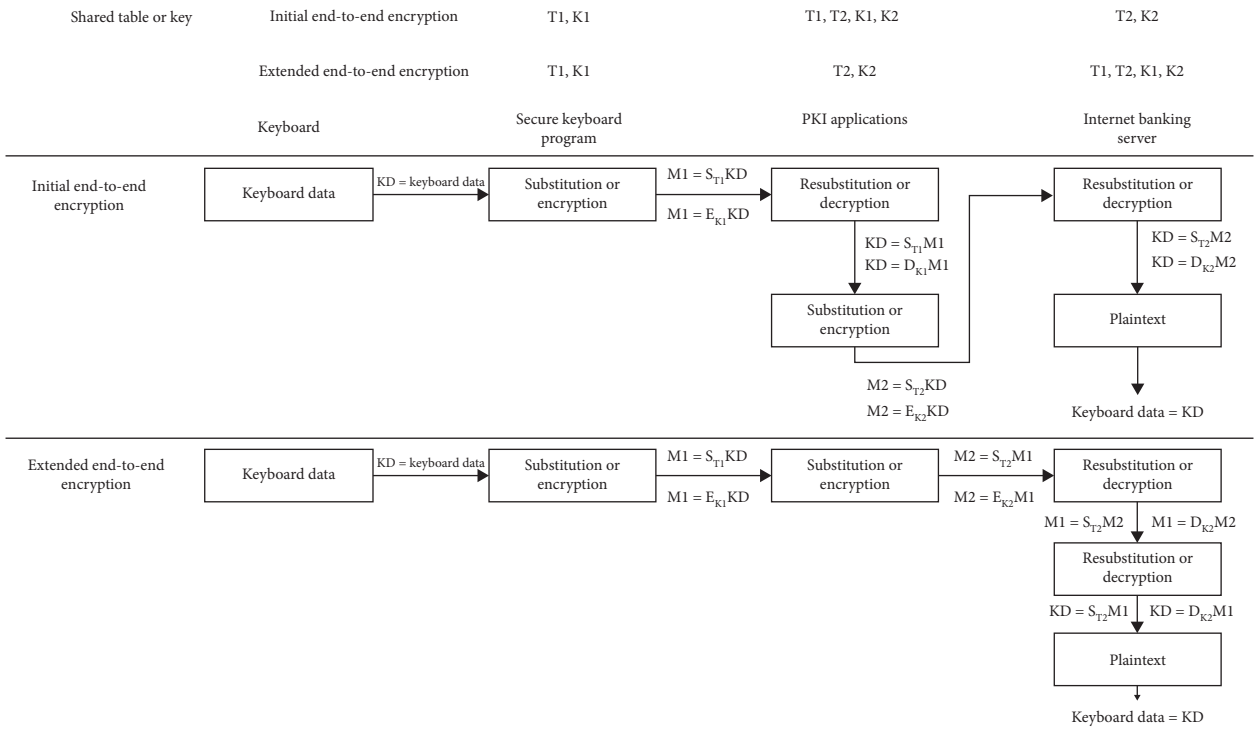

Figure 5: A flowchart of the initial end-to-end encryption and extended end-to-end encryption.

Table 4: Basic features of the intrusion prevention system (IPS).

Category	Features	Description
Personal firewall	Program access management	(i) Notifies user by detecting the execution of the program communicating with the outside of the user's PC (ii) Allows/blocks settings by IP address and port
	IP address access management	(i) Allows or blocks designated IP addresses
	Port access management	(i) Allows or blocks designated ports
Shared folder control/ monitoring	Shared access management	(i) Allows or blocks access to shared folder on the user's computer from other computers
	Shared folder management	(i) Disables shared folder
Network	Network connection management	(i) Displays allowed TCP/UDP connection information for external communication (ii) Alerts or blocks the port when it is a known hacking tool (iii) Counteracts self-disguised and hidden hacking tools when detection is difficult
	Network monitoring	(i) Displays graph of the amount of network inbound and outbound traffic
Real-time hacking tool detection/prevention	Known hacking tool detection and prevention	(i) Inspects hacking tool process (ii) Detects and blocks key loggers (iii) Detects and blocks console-based hacking programs (iv) Detects and blocks Back Orifice hidden process
Logging	Usage history	(i) Records program access management information (ii) Displays logs history by date

mandatory when verifying the security suitability of secure USB. The first feature is user identification and authentication, the second feature is designated data encryption and decryption, the third feature is the copy protection of stored data, and the last feature is deletion to protect the stored data when lost. Moreover, the auxiliary storage media are classified into general use, confidential use (including confidentiality), and authorized certificate storage, and the management was guided to suit the usage. In this paper, commercial secure USB products are selected and surveyed, and Table 5 shows the results [12].

Generally, exposure of data stored in the secure USB is prevented by software method, hardware partitioning method, and hardware method [13–17]. The software

Table 5: Commercial secure USB products.

Company	Product name	Certification authority
Company F	Defenstick	—
Company G	USBSafe	KICA, Korea Information Certificate Authority Inc.
Company H	uToken	NIS, National Intelligence Service
Company I	SePros	NIS, National Intelligence Service
Company J	SafeUSB+	—
Company K	DefCon secure USB	—
Company L	nTracker	NIS, National Intelligence Service
Company M	SecuYouSB	—
Company N	Secure-i	—

method uses only software, and an example of the domestic product is USBSafe in South Korea, while an example of overseas product is SecureAccess. This method creates a temporary file for use as a disk and then protects data from a third party by encryption and decryption of the generated temporary file based on the password input by the user. This approach is not access control for the hardware itself, but access control for the image file used as a disk based on the password. The hardware partitioning method supports physical partitions using a hardware chip. After receiving the password from the user, this method performs access control by encryption and decryption of the partition information inside the secure USB memory based on the password. Namely, access control is performed on the hardware resource indirectly, because the access control is performed according to the hardware partition information, based on the password. The hardware method also receives the password from the user and then encrypts and decrypts files based on the encryption and decryption keys generated inside the secure USB memory, based on the input password. This approach also requests the password and handles the plaintext from the host, but directly controls access to the hardware resource as a provision against physical theft.

A variety of secure USB techniques have also been studied against the problem of the exposure of vulnerability, in which most secure USB products authenticate the user by comparison routine on the host side. To counteract this vulnerability, robust authentication protocol against reverse engineering by processing comparison routine on the device side was researched [13, 14]. Moreover, the analysis process of security vulnerability has been studied, and the process includes (i) analysis step of feature, operation, and characteristic of secure USB and its management software, (ii) analysis step of operation process by each security method, (iii) vulnerability analysis step using eavesdropping, command, and reverse engineering, and (iv) deduction step of vulnerabilities, countermeasures, and security requirements.

The analysis step of feature, operation, and characteristic of secure USB and its management software is a step to run secure USB and management software and then analyze the results. When a user registers a password and all functions for protecting the file are ready, this step is an analysis of whether the created file or the file stored in the secure USB memory is protected by encrypting data, or the data is protected by hiding the file.

The analysis step of operation process by each security method is to determine whether the secure USB is a software method, a hardware partitioning method, or a hardware method and to analyze the operation process by each method. In the case of the software method, this step collects the information for analysis focused on the generated image file for the secure area or configuration file to generate the encryption and decryption key, and traces the correlation between such generated files. In the cases of hardware partition method and hardware method, it is possible to construct the environment and the information for analyzing the vulnerability, because these methods can construct a secret copy by physical copy or reconstruct the software by obtaining the data sheet that includes the information about the chip using physical decapsulation. Namely, the reasons are that the general-purpose chip is mostly used, and the circuit diagram can be easily referenced. Moreover, when the hidden function is applied for the protection of the file, further analysis is needed, because the host system can add or manipulate functions that utilize the file filter driver or system function hooking.

The vulnerability analysis step using eavesdropping, command, and reverse engineering is an actual step of analyzing vulnerabilities using eavesdropping, command, and reverse engineering based on the information collected from the previous two steps: the analysis step of feature, operation, and characteristic of secure USB and its management software; the analysis step of operation process by each security method. The vulnerability analysis step using eavesdropping arises from the fact that the USB interface does not consider security at the time of design, so data is exposed in the communication between the USB device and the host. For this reason, an attacker can steal information about the key and user password by analyzing the information transferred between the host and the secure USB drives. The vulnerability analysis step using command is classified into a method using an unlock command and a method using an unlock command, including authentication information [18]. When a password is requested from the host to the secure USB, the secure USB sends the password to the host, and then the host verifies it, and sends the unlock command to the secure USB. Therefore, it is possible to access the files in the USB simply by collecting information about the unlock command of a specific controller and sending the unlock command. The vulnerability analysis step using reverse engineering does not exploit a vulnerability of secure USB, but rather a vulnerability in the management software. Reverse engineering can be used to analyze the

operation of management software and to derive vulnerabilities based on the analyzed results. By doing this, the attacker can steal information related to the user password and encryption/decryption key, so the analysis results can be used in various ways.

The deduction step of vulnerabilities, countermeasures, and security requirements draws the results of the vulnerabilities analyzed from the previous three steps, and this step is to derive countermeasures and analyze the security requirements against security threats. This step finds the actual security threats by attempting attacks based on the vulnerabilities derived in the previous vulnerability analysis step and then verifies it by implementing a proof-of-concept tool to prove it [19–21]. Consequently, this step analyzes the reasons of the occurrence of the vulnerabilities, and the countermeasures and security requirements to solve them are derived in terms of eavesdropping, command, and reverse engineering.

3.7. Antireverse Engineering Technique. Reverse engineering is intended to analyze the file to find the characteristic and useful information about the software. Malicious attackers can find vulnerabilities of the program by reverse engineering, and then they can implement malicious code, such as viruses or worms, based on the analyzed vulnerabilities. A number of prevention techniques and tools have been developed to protect against software attacks by reverse engineering, and code obfuscation is one of the countermeasures.

The lexical meaning of obfuscation is "to embarrass" or "to confuse." In other words, code obfuscation means translating the original code into a form that is difficult to understand, to embarrass and confuse the reverse engineer. However, there are conditions for obfuscation transformation, because the behavior of the code must not be changed to make analysis difficult. Collberg defines these conditions, as follows [22]. Let $T(P)$ be the transformation for program P. If $T(P)$ is equal to P, then T is an obfuscation transformation. In addition, the obfuscation transformation T must satisfy the following conditions:

(i) $T(P)$ should have the same result as P. If a program P is terminated or fails to terminate due to an error condition, $T(P)$ should also terminate or fail to terminate as P in the same situation.

(ii) If a program P terminates normally, $T(P)$ should terminate normally and should have the same result as P.

Obfuscation techniques are classified into the source level obfuscation and the binary level obfuscation, depending on the location of the applied technique, and the techniques are also classified into layout obfuscation, data obfuscation, control obfuscation, and preventive obfuscation, depending on the applied method [22]. In many cases, obfuscation has evolved into binary level obfuscation due to software copyright issues, and source level obfuscation has been actively pursued as a protection method for software in the form of recent source distribution. However, generally speaking, obfuscation means binary obfuscation. Moreover, obfuscation distinguishes more complex code obfuscation, which consists of layout obfuscation, data obfuscation, and control obfuscation, from prevention obfuscation, which consists of techniques, such as anti-debugging and anti-disassembly. Usually, obfuscation means layout obfuscation, data obfuscation, and control obfuscation. Table 6 shows the detailed classification of obfuscation.

The layout obfuscation means to transform the physical structure of the program into another structure. That is, obfuscation is performed by changing the contents of the string contained in the executable file. For example, when reverse engineering is attempted by attackers, this obfuscation is difficult to understand by changing names, annotations, source code format, and variables that are not related to the actual program execution, to words with no more intricate words or meaning. The layout obfuscation is classified into changing format, removing comment, and scrambling identifiers, according to type. This obfuscation makes retransformation difficult by reducing readability, but there is the advantage that this obfuscation does not increase the execution time or storage space. Figure 6 shows an example of changing format, while Figure 7 shows an example of scrambling identifiers. The example of scrambling identifiers in Figure 7 transforms the name of a class or a method to a or w, respectively. The importance of this obfuscation is to make the code difficult to analyze by changing the structure of the program and making it less readable.

The data obfuscation performs transformation based on the data structure used in the program, rather than the structure of the program. Data obfuscation is classified into data storage, data encoding, data aggregation, and data ordering; these techniques make it difficult for analysis to read the data structure when an attacker knows an important data structure. The data storage obfuscation is a method of modifying the way data is stored in memory. There are several techniques of how to combine two arrays into one array, or split one array into two arrays keeping the features intact, and how to combine array data. Moreover, there is a way to convert a static variable into a function call. Figure 8 shows an example of dividing a limited BOOL variable and a limited range variable into two or more other variables. The importance of this obfuscation is to make the code harder to interpret, by making the data structures difficult to analyze.

Data encoding obfuscation prevents intuitive understanding of the program by analyzing variables or simple expressions. Figure 9 shows that i in the array is converted to the same value as the previous calculated value, after changing the variable i with initial value (1 to 11). Finally, 8 is added to make sure the result is the same before the modification. Like this, the multiplication or the shift operation makes it difficult to interpret the code, but the result can be changed to make the interpretation even the same data. Data aggregation obfuscation means to change the group of data by integrating or separating arrays, or the inheritance relation of the variable or object. For example, it is possible to convert a two-dimensional array into a one-dimensional array, or *vice versa*; this is a method of such

obfuscation. Data ordering obfuscation means to change the order of data, such as objects, variables, methods, and arrays [23]. For example, in programming, a programmer often uses the variable i for an integer to point to an index in the array. For this reason, it is possible to cause confusion to code analysis by specifying an index by separating the i variable to a method, such as $f(i)$. The importance of this obfuscation is to make the program harder to understand intuitively, by configuring it to have the same results, even if the operations are complex.

Control obfuscation transforms the control flow while keeping the function intact, changing the order and flow of the program, to make analysis difficult. Control obfuscation is classified into computation transformation, aggregation transformation, and order transformation [24]. Computation transformation is a way of transforming the control flow structure into a high-level programming language, while maintaining the semantics of the original control flow structure. For example, this transformation might change the control flow of the program by removing control flow information from the program or by adding a new control flow instruction. Methods for computation transformation include a method of transforming flow from collapsible flow graph to noncollapsible graph, a method of extending a loop condition, and a method of using a program interpretation table.

For example, the method of transforming flow from a collapsible flow graph to a noncollapsible graph is to point the *goto* statement used in translated code for the *for* statement to the middle of the loop, to transform the flow into a noncollapsible graph. For this reason, it is difficult to decompile the code into the original *for* statement without analyzing the code, because the codes are tangled into each other. The method of extending a loop condition is that the conditions of the loop are added within the range of not changing the function of the program, so that the calculation becomes complicated in the analysis, as shown in Figure 10. The method of using a program interpretation table is for the user to define commands, store the meaning in a table, and transform the program using the commands defined. As a result, the transformed program can be executed by the table. This method can confuse both analysts and reverse obfuscation programs, and in general, the method of using a program interpretation table is difficult to interpret by static analysis. The importance of this obfuscation is that it makes analysis difficult by changing control flows, such as the order and flow of the program.

Aggregation transformation means a method of combining or separating pieces of code, and this transformation is classified into an inline method, an outline method, a duplication method, and a loop release method. The inline method is to copy the contents of a function to a location of the calling function. In other words, the compiler replaces the function call part by the actual function code, instead of calling the function code from multiple places. To do this, internal concepts created by the developer are removed; hence, the reverse engineer does not know which part of the function is the actual inline function that is called multiple times throughout the program. The outline method is to extract some of the specific code inside a function by creating another function. This method does exactly the opposite of the inline method, and this is useful when creating functions with arbitrary pieces of code. The duplication method is a way to prevent analysis of the actual code by cloning a function, and the loop release method is a way to prevent analysis, so that it is difficult to determine the entire set of loop statements by releasing the loop.

Order transformation is an obfuscation that changes the order of statements, loops, expressions, etc. This transformation can use the jump instruction to change the order of blocks, or change the order of the inner and outer loops in a nested loop, or change the order of computation, without compromising dependencies. For this reason, if the order of

Table 6: Classification of obfuscation.

Major category	Subcategory
Layout obfuscation	Changing format Removing comment Scrambling identifiers
Data obfuscation	Data storage Data encoding Data aggregation Data ordering
Control obfuscation	Computation transformation Aggregation transformation Order transformation
Preventive obfuscation	Unique transformation method Targeted transformation method
Other obfuscation	Packing Virtualization

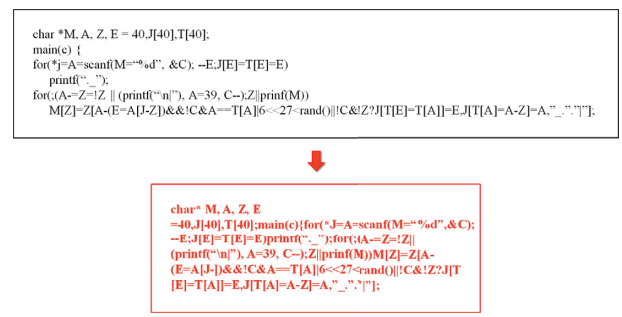

Figure 6: An example of layout obfuscation (changing format).

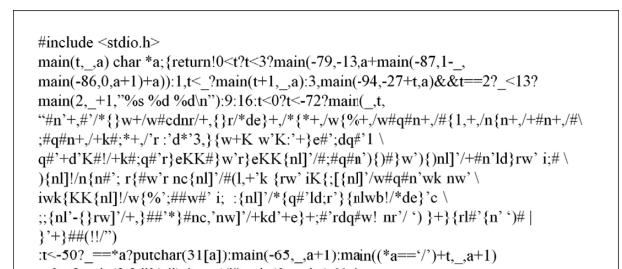

Figure 7: An example of layout obfuscation (scrambling identifiers).

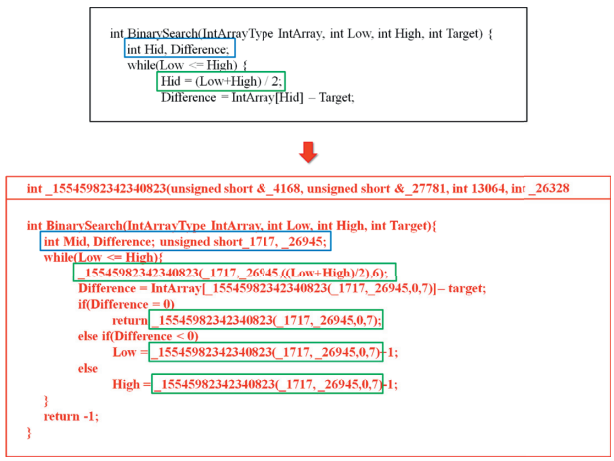

Figure 8: An example of data obfuscation (variable division).

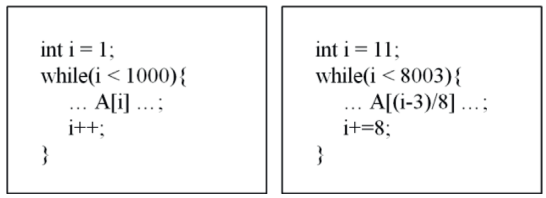

Figure 9: An example of data obfuscation (data encoding).

instructions is changed, it is possible to make the interpretation difficult, because reverse engineers must analyze the code, in order to analyze it.

An important concept in the control obfuscation concerns unclear expressions. Unclear expression refers to an expression that, even though the result of executing the code is always true or false, intuitively, does not have clear results. This is a basic component in control obfuscation. An example of an unclear expression is the case of complicating the control flow by inserting a dummy conditional statement [24].

Preventive obfuscation, that is, preventive transformation, refers to a method of neutralizing an attack by a specific decompiler or an automated reserve transformation tool. The prevention obfuscation is classified into the unique transformation method and the targeted transformation method. The unique transformation method investigates and counteracts the problems of the reserve transform techniques that automatically analyze codes of the program, while the targeted transformation method makes it difficult to apply the technique of the decompiler or reverse engineering tools. Table 7 lists the well-known prevention obfuscation techniques by feature [25].

3.8. Other Technologies. In addition, various security technologies have been studied to protect the user terminal domain. Among these, the prevention technique of reverse engineering is a technique that can not only be used by defenders, but also be used by an attacker. For this reason, analysis of malicious code using antireverse engineering

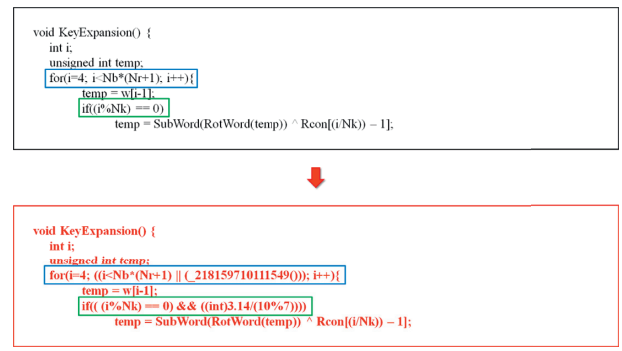

Figure 10: An example of control obfuscation (expansion of loop condition).

technique was researched by using reverse obfuscation. Reverse obfuscation refers to a method of analyzing obfuscated programs and finding the necessary information. This technique is similar to the program optimization technique in the process of deriving the results, although there are differences in purpose. In order to improve performance, program optimization is generally simplified, so the program is transformed for this purpose. The optimization process simplifies the program through the following process, and this process can be applied to the reverse obfuscation process. The first step is to detect and remove parts that are not executed through the control flow analysis, while the second step is to check whether the value stored in the specific variable is necessary to obtain the result by the data flow analysis and, if not, remove the variable. The last step is to integrate two pointers pointing to the same position into a single pointer, by pointer overlay analysis. The reverse obfuscation technique is classified into malicious code detection using static analysis and reverse obfuscation to prevent disassembly. Malicious code detection using static analysis is classified into malicious code detection based on API call characteristics and malicious code detection using semantic structure, while reverse obfuscation to prevent disassembly is classified into memory analysis and stack analysis.

Malicious code detection using static analysis has been studied to solve the problem that the malicious code uses a method of avoiding pattern detection. Generally, in order to avoid detection from anti-virus software, the malicious code is stored in the ciphertext; for this reason, in order to detect the malicious code, it needs to be executed. A typical virus using encryption is the metamorphic virus. To avoid detection, the metamorphic virus changes its appearance each time it is replicated, through methods such as instruction substitution and register reallocation. Signature-based antivirus software can easily be neutralized by malicious code that changes its appearance, and Christodorescu showed these results from experiments [26]. Therefore, in order to detect malicious code that transforms itself, a static analysis method is needed.

Bergeron proposed a prototype for malicious code detection based on API call characteristics by using static analysis [27]. This method consists of three steps. First, the

Table 7: Classification of preventive obfuscation.

Classification by feature	Detailed classification
Unique transformation method	Hardware breakpoint detection
	Detecting breakpoints by CRC
	Ring3 debugger detection via LDR_MODULE
	Context modification
	Kernel32!CloseHandle and NtClose
	popf and trap flag
	User-mode timers
	INT 2Dh debugger detection
	LordPE anti-dumping
	RDG OEP signature spoofing
	Stack segment register
	Using the CMPXCHG8B with the lock prefix
	CheckRemoteDebuggerPresent windows API
	Debug register manipulation
Targeted transformation method	OllyDbg INT3 exception detection
	OllyDbg IsDebuggerPresent detection
	OllyDbg instruction prefix detection
	OllyDbg OpenProcess string detection
	PEID GenOEP spoofing
	PEID OEP signature spoofing
	ProcDump PE header corruption

binary code is analyzed and converted into an intermediate representation; second, it is converted into a control flow graph (CFG), including API system calls, by control flow analysis and data flow analysis. Finally, it is described using security automata, and model check technique is used to check the CFG. However, this method has disadvantages in that when the call obfuscation is applied, the code hidden API call does not reverse obfuscation, and the speed is considerably slow, because it is an analysis method that depends on the model checker. Christodorescu proposed the use of semantic structure to detect malicious code [28]. According to this study, a malicious code detection program using semantic structure can examine most of the code applied by a simple *nop* command or stack obfuscation. This malware detector obtains the signature of the malicious code by analyzing the memory value used by the malicious code. Accordingly, malicious code is detected by comparing malicious code signatures with memory types of malicious code suspicious programs. Although this method can solve the obfuscation that does not change the memory, such as instruction reordering, register renaming, and dummy instruction insertion, the method does not solve the obfuscation that changes memory values or changes the order of memory accesses, using a mathematical operation, such as shift operation. Moreover, it takes a lot of time to reverse obfuscate, and because it is hard to analyze the memory access pattern, it is difficult to define signatures.

Reverse obfuscation to prevent disassembly analyzes the destination address of jump operation stored in memory, because this technique relies on the method of preventing the jump destination address from being found, using the indirect jump instruction. To solve this problem, Balakrishnan and Thomas proposed a value set analysis (VSA) to analyze the memory value [29]. The VSA uses the characteristic that when a high-level language is compiled, the global variable is fixed at a specific position in the heap, and the local variable is fixed at a specific position in the stack frame. Accordingly, the location of the memory is defined as an *a-loc* (abstract location), and the relationship between the program objects and *a-loc* is analyzed. The final step is an estimation of all values that each entity can have at each point in the program. Through VSA, aggregation information of used, killed, and possibly killed *a-locs* can be obtained, and this information can be used for data flow analysis. This technique compensates for the disadvantages of data flow analysis, which has difficulty in estimating the value of memory, and thus enhances the accuracy. Stack analysis is performed primarily to monitor API calls against obfuscated malicious code. This technique monitors the API calls by simulation of the stack; for this reason, a summary execution method is used to increase efficiencies. This method models the stack as a summary stack and the stack instruction as a summary instruction; and then the stack graph expresses the shape having the stack at each point in the program [30]. For this reason, it is possible to perform reverse obfuscation using stack instruction, by using the summary stack graph.

In addition, in the case of malicious code using packing [31], it is difficult to analyze code using reverse engineering, so research has been conducted to find a starting point for analysis to solve this problem [32, 33]. It is difficult for packed files to reverse disassembly by reverse engineering in a simple way, because the dynamic analysis prevention code in the protector interferes with reverse engineering. Packed files are protected against the reverse engineering described above by compression or encryption, but there is a vulnerability in that jump operation has to return to the original entry point (OEP) after restoration, because the protected code must be restored at runtime by unpacking code. Here, unpacking means restoring the encrypted or compressed file

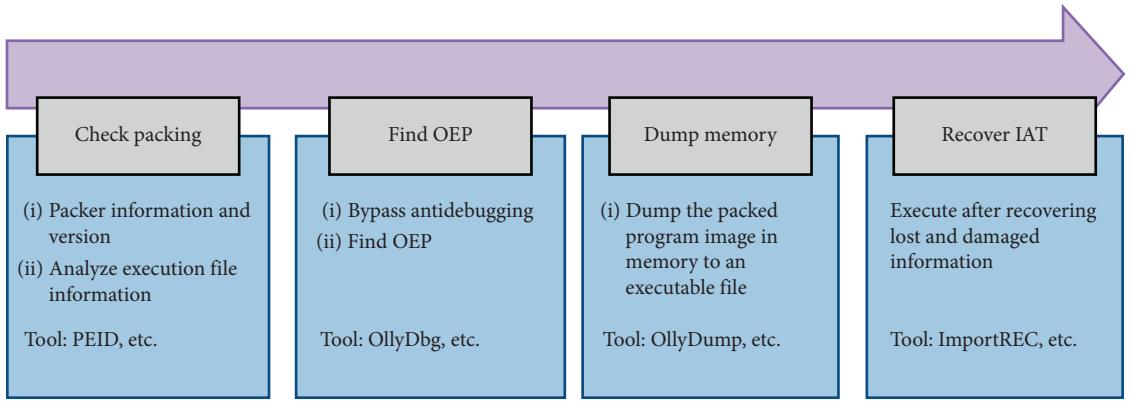

Figure 11: The unpacking process.

Table 8: A summary of security techniques for the user terminal area in the Internet banking service.

Security techniques	Key strength	Weakness
Secure keyboard program	Prevents exposure of important information input from users	It is difficult to implement and has race condition between attacker and defender
PKI applications	Prevents the exposure of important information transmitted to the network domain	Requires PKI module implementation and deployment costs
E2E encryption	Prevents the exposure of important information transmitted from the user terminal domain to the financial institution domain	Requires E2E module implementation and deployment costs
Antihacking program	Protects the system by preventing and detecting attacks from the outside in the user terminal domain	It is impossible to completely protect the system, and it is difficult to detect unknown attacks, due to various system attacks
Personal firewall	Protects the system by detecting and blocking malicious packets in the network domain	It is impossible to completely protect the system, and it is difficult to detect unknown attacks, due to the variety of network attacks
Removable media security	Enhances the security of user authentication by protecting personal and confidential information such as certificates stored on removable media	It is difficult to completely protect stored important information by bypassing authentication and exposing the cryptography key, due to various vulnerabilities
Antireverse engineering technique	Protects program code by making it difficult for attackers to analyze the program and taking a long time for analysis	It is difficult to completely protect the analysis of the program code; finally, there is a drawback and weakness in that it can be analyzed, even if it takes a long time

to execute the original code. The reason for unpacking is that the packed files have to restore the original file in its original state; unpacking can also be used for patching or cracking normal files using vulnerabilities. The unpacking method is classified into a method of using a packed dedicated program and a method of executing the OEP by bypassing the protection code of the packed file at runtime. Figure 11 shows the process of unpacking and the following operations that are performed in each process.

In the step of checking the packing, a tool such as PEID or PEInfo is used to check whether the file is packed. At the same time, information of the packer, version, and execution file information can be obtained. In the OEP finding step, the OEP is searched by bypassing the protection code existing in the packed file. This is performed by tracking the API used by the original program or by using memory write access points. OllyDbg or IDA Pro is used to find the OEP. In the memory dump step, the packed program image in memory is dumped to an executable file using OllyDump and ProcDump. In the IAT recovery step, the image is lost or corrupted by the protection code of the packer at the time of the dump, so errors occur when the dump file is executed. Thus, it is possible to execute unpacked files by restoring lost and damaged external reference information. The tools used for IAT recovery are ImportREC, Universal Import Fixer, etc. Moreover, security technology has emerged by using steganography to hide important information [34, 35]. Table 8 provides a summary of the security techniques for the user terminal area in the Internet banking service in this paper.

4. Conclusion

In this paper, we describe the security technology applied to the user terminal domain among the security technologies applied to the entire Internet banking, to cope with security threats arising from the Internet banking service. The user terminal domain is weaker than the financial institution domain and the network domain, so it is a major target for the attacker. We classified the security technologies in the

user terminal domain into secure keyboard program, PKI applications, E2E encryption, antihacking program, personal firewall, removable media security, and antireverse engineering technique and described the detailed and key techniques of each security technology. In detail, the secure keyboard program included message hooking, filter driver insertion, interrupt object replacement, IDT replacement, generation of random scan code using 0xD2 command, generation of random scan code using keyboard internal memory, utilization of debug exception handler, and inline hooking. PKI applications included security techniques, such as self-design and library utilization. E2E encryption included interlocking secure keyboard program, PKI application using encryption and decryption modules, and double encryption using only encryption module. The antihacking program included security techniques, such as anti-virus products. Personal firewall included security techniques, such as program access management, IP address access management, and network connection management. Removable media security included security techniques such as software approach, hardware-based partitioning approach, and hardware approach. Antireverse engineering technique included security techniques such as layout obfuscation, data obfuscation, control obfuscation, and preventive obfuscation. Therefore, there are various security threats; hence, it is expected that this paper will be useful as reference material to counteract such threats. Future studies will need further research to discover additional security threats that can occur and to cope with them, even though classified security technologies are applied to the Internet banking service described in this paper.

Disclosure

A part of this paper was presented at the Conference on Information Security and Cryptography (CISC-S 17), June 22-23, 2017, Asan, South Korea.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was partially supported by the National Research Foundation of Korea (NRF) grant, funded by the Korean government (MSIT) (No. 2018R1A4A1025632), and the Soonchunhyang University Research Fund.

References

[1] J. Seung, "Effective electronic financial security response system by analyzing domestic and international electronic financial security policies" Ph.D. thesis, Interdisciplinary Program of Information Security Graduate School of Chonnam National University, Gwangju, South Korea, 2011.

[2] K. Lee and K. Yim, "Analysis on the keyboard and display hacking techniques," Technical report, Korea Internet & Security Agency (KISA), Seoul, South Korea, 2012.

[3] K. Lee and K. Yim, "Cybersecurity threats based on machine learning-based offensive technique for password authentication," *Applied Sciences*, vol. 10, no. 4, p. 1286, 2020.

[4] Financial Security Agency (FSA), *End-to-End Encryption Application Guide*, FSA, Tokyo, Japan, 2007.

[5] Telecommunications Technology Association (TTA), "Security threats analysis and management methods in electronic financial services," Technical report, TTAR-12.0008, TTA, Seongnam, South Korea, 2011.

[6] K. Lee, K. Yim, and J. Seo, "Status and future of security techniques in the Internet banking service," *Journal of Internet Computing and Services*, vol. 18, no. 2, pp. 31–42, 2017.

[7] Y. Keun Jang, "A study on the alternative methods of improving security of internet banking user's input information," Master thesis, Department of Information Security, Graduate School of Dongguk University, Seoul, South Korea, 2009.

[8] S. Jeong, "A proposal of two-channel authentication technique based on QR CODE on the electronic financial transaction," Master thesis, Graduate School of Information Management Engineering of Korea University, Sejong City, South Korea, 2010.

[9] J. I. Lee, "End-to-End(E2E) encryption and considerations for the powerful securities of electronic financial transactions," Master thesis, Department of Computer Engineering, Graduate School of Information & Technology of Sungkyunkwan University, Seoul, South Korea, 2008.

[10] M.-J. Jeon and S.-Y. Hwang, "Design and implementation of high-speed pattern matcher using multi-entry simultaneous comparator in network intrusion detection system," *The Journal of Korean Institute of Communications and Information Sciences*, vol. 40, no. 11, pp. 2169–2177, 2015.

[11] National Intelligence Service (NIS), *Guideline for Security Management of Auxiliary Storage Media Such as USB Memory*, NIS, Seoul, South Korea, 2007.

[12] Security 21c, Secure USB All Guide, 2008.

[13] K. Lee, K. Yim, and E. H. Spafford, "Reverse-safe authentication protocol for secure USB memories," *Security and Communication Networks*, vol. 5, no. 8, pp. 834–845, 2012.

[14] K. Lee, H. Yeuk, Y. CHoi, S. Pho, I. You, and K. Yim, "Safe authentication protocol for secure USB memories," *Journal of the Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications (JoWUA)*, vol. 1, no. 1, pp. 46–55, 2010.

[15] K. Lee, H. Yeuk, and K. Yim, "Design of a process for vulnerability analysis of the secure USB flash drive," in *Proceedings of the Korea Institute of Communications and Information Sciences,* Seoul, South Korea, June 2012.

[16] J. Lee, J. Park, S. W. Jung, and S. Jung, "The authentication and key management method based on PUF for secure USB," *The Journal of Korea Information and Communications Society*, vol. 38, no. 12, pp. 944–953, 2013.

[17] K. Lee, I. Oh, Y. Lee, H. Lee, K. Yim, and J. Seo, "A study on a secure USB mechanism that prevents the exposure of authentication information for smart human care services," *Journal of Sensors*, vol. 2018, Article ID 2089626, 17 pages, 2018.

[18] J. Kim, Y. Lee, K. Lee, T. Jung, D. Volokhov, and K. Yim, "Vulnerability to flash controller for secure USB drives," *Journal of Internet Services and Information Security*, vol. 3, no. 3-4, pp. 136–145, 2013.

[19] K. Lee, B.-G. Son, S.-Y. Lee, and K. Yim, "Vulnerability analysis of secure USB: based on the fingerprint authentication of product B," in *Proceedings of the 2018 Conference on*

Research in Adaptive and Convergent Systems, pp. 167–169, Honolulu, HI, USA, October 2018.

[20] K. Lee, I. Oh, C. Chang, and K. Yim, "Vulnerability analysis on a secure USB memory: based on a commercial product A," in *Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 498–502, Taichung, Taiwan, October 2018.

[21] K. Lee, S.-Y. Lee, H. Ahn, J. Choi, and K. Yim, "Vulnerability analysis of secure USB: based on the replay attack of the authentication result of product B," in *Proceedings of the International Conference on Smart Media & Applications*, pp. 171–173, Toulon, France, December 2019.

[22] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Technical report #148, The University of Auckland, Auckland, New Zealand, 1997.

[23] S.-W. Choi, H.-W. Park, and T. Han, "A obfuscating technique for Java program by distributing methods," in *Proceedings of the Korean Institute of Information Scientists and Engineers*, pp. 238–240, January 2003.

[24] C. Collberg, C. Thomborson, and D. Low, "Manufacturing cheap, resilient, and stealthy opaque constructs," in *Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pp. 184–186, San Francisco, CA, USA, January 1998.

[25] M. V. Yason, *The Art of Unpacking*, IBM Internet Security Systems, Atlanta, GA, USA, 2001.

[26] M. Christodorescu and S. Jha, "Testing malware detectors," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 4, pp. 34–44, 2004.

[27] J. Bergeron, M. Debbabi, J. Desharnais, M. Erhioui, Y. Lavoie, and N. Tawbi, "Static detection of malicious code in executable programs," in *Proceedings of the Symposium on Requirements Engineering for Information Security (SREIS)*, 2001.

[28] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant, "Semantic-aware malware detection," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 32–46, Oakland, CA, USA, May 2005.

[29] G. Balakrishnan and R. Thomas, "Analyzing memory accesses in x86 executables," *LNCS*, vol. 2984, pp. 5–23, Springer, Berlin, Germany, 2004.

[30] A. Lakhotia and E. U. Kumar, "Abstracting stack to detect obfuscated calls in binaries," in *Proceedings of the 4th IEEE International Workshop on Source Code Analysis and Manipulation*, pp. 17–26, Chicago, IL, USA, September 2004.

[31] Financial Security Agency (FSA), "Packing technique to enhance software security," Issue report, FSA, Tokyo, Japan, 2011.

[32] K. Lee and K. Yim, "A new analysis method for packed malicious codes," *Journal of the Korea Navigation Institute*, vol. 16, no. 3, pp. 489–494, 2012.

[33] K. Lee, S. Pho, W. Kim et al., "A hint to the analysis of the packed malicious codes," in *Proceedings of the 11th International Workshop on Information Security Applications (WISA)*, pp. 37-38, Jeju Island, South Korea, August 2010.

[34] X. Liao, G. Chen, and J. Yin, "Content-adaptive steganalysis for color images," *Security and Communication Networks*, vol. 9, no. 18, pp. 5756–5763, 2016.

[35] X. Liao, Z. Qin, and L. Ding, "Data embedding in digital images using critical functions," *Signal Processing: Image Communication*, vol. 58, pp. 146–156, 2017.