Website Protection Schemes Based on Behavior Analysis of Malware Attackers

Takeshi Yagi

February 2013

List of publication

Journal papers

[1] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Design of Provider-Provisioned Website Protection Scheme against Malware Distribution," *IEICE Transactions on Communications,* vol. E93-B, no. 5, pp. 1122–1130, May 2010.

[2] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Intelligent High-Interaction Web Honeypots Based on URL Conversion Scheme," *IEICE Transactions on Communications,* vol. E94-B, no. 5, pp. 1339–1347, May 2011.

Refereed Conference papers

[1] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Enhanced Attack Collection Scheme on High-Interaction Web Honeypots," in *Proceedings of IEEE Symposium on Computers and Communications (ISCC) 2010*, June 2010.

[2] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Investigation and Analysis of Malware on Websites," in *Proceedings of IEEE Symposium on Web Systems Evolution (WSE) 2010*, September 2010.

[3] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Life-cycle Monitoring Scheme of Malware Download Sites for Websites," in *Proceedings of IEEE International Conference on Service-Oriented Computing and Applications (SOCA) 2010*, December 2010.

Non-Refereed Conference papers

[1] T. Yagi, T. Kondoh, T. Kuwahara, J. Murayama, H. Ohsaki and M. Imase, "Architecture Design for SPX: Secure networking Platform for group-oriented eXchange," in *Proceedings of Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT) 2008*, April 2008.

[2] T. Yagi, "Terabit-OBS Super-Net Experiments (invited)," in *Proceedings of OBS Workshop in COIN/NGNCON 2006*, July 2006.

[3] T. Yagi, Y. Naruse, J. Murayama and K. Matsuda, "A Distributed Traffic Monitoring Scheme for Large-Scale IP over Optical Network," in *Proceedings of World Telecommunications Congress (WTC) 2006*, May 2006.

[4] T. Yagi, Y. Naruse, J. Murayama and K. Matsuda, "Terabit-OBS Super-Net Experiments," in *Proceedings of International Workshop on the Future of Optical Networking (FON) in OFC/NFOEC 2006*, March 2006.

[5] T. Yagi, K. Matsui, Y. Naruse, J. Murayama and K. Matsuda, "A Cooperative Operation Technology for Cut-through IP and Optical Paths for the Electrical/Optical Hybrid Network," in *Proceedings of World Telecommunications Congress (WTC) 2004,* September 2004.

Preface

This thesis proposes high-accuracy attack detection schemes against diverse malware attacks on websites. To prevent websites from being attached by malware, schemes to extract characteristics from known attacks were researched. This paper discusses schemes that not only detect malware files on servers based on the characteristics of anti-virus software, but also detect malware attacks on networks based on the characteristics of intrusion detection systems, intrusion prevention systems, and web application firewalls. However, the conventional schemes cannot detect malware attacks with high probability because malware attacks have been diversifying rapidly.

Generally, attackers force victims to download malware by using vulnerabilities in web applications. Recently, detection of diverse attacks is necessary because the number of web application vulnerabilities has been increasing. Additionally, to detect many kinds of attacks, it is necessary to collect as many characteristics of known attacks as possible because the information is used to determine whether each access to and from a website is an attack. Furthermore, to achieve high-accuracy attack detection, cases in which a normal access is detected as an attack, called a "false positive," and in which an actual attack is not detected, called a "false negative," should be restricted.

Conventional schemes, deploy the characteristics of malware files and the characteristics of HTTP request messages from attackers as characteristics of known attacks. However, the variations of malware and variations of HTTP request messages for attacks are increasing rapidly, making attack detection schemes all the more difficult.

Additionally, when conventional schemes extract characteristics from known attacks,

they can only extract the characteristics from attacks that clearly led to illegal control of the website. The conventional schemes cannot extract effective information from attacks that did not cause illegal control but only attempted illegal control. As a result, conventional schemes cannot extract a lot of information from attacks.

Finally, conventional schemes monitor and filter attacks without considering the attacker's behavior. To avoid being detected, attackers try to sense the monitoring and filtering and change the characteristics of their attacks as they deem necessary. As a result, attack detection ratios by the conventional schemes are decreasing.

To achieve detection of diverse attacks, the proposed schemes collect destination URLs that attacked websites are forced to access, and they filter accesses from websites to the URLs as blacklisted URLs. To extract a lot of information from attacks that do not lead to illegal control, the proposed schemes convert the destination URLs of the attacks so as to allow the attacks to succeed under our management. To achieve high accuracy, the proposed schemes monitor and manage blacklisted URLs according to the attackers' behaviors.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisor, Professor Masayuki Murata of the Graduate School of Information Science and Technology, Osaka University, for his patient encouragement, meaningful and comprehensive advice, and valuable discussion. He directed me to the appropriate perspective in this domain and inspired me to aim at higher goals. I was able to complete my thesis owing to his kind guidance, timely encouragement and valuable advices.

I am grateful to the member of my thesis committee, Professor Koso Murakami, Professor Teruo Higashino of the Graduate School of Information Science and Technology, Osaka University, and to Professor Hirotaka Nakano of the Cyber Media Center, Osaka University, for reviewing my dissertation and providing many valuable comments.

I would especially like to express my appreciation to Vice Presidents Makoto Imase of the National Institute of Information and Communications Technology in Japan (NICT), and to Associate Professor Hiroyuki Ohsaki of the Graduate School of Information Science and Technology, Osaka University, for their critical and beneficial comments and unerring guidance that greatly inspired me. They gave me an opportunity to study for a doctorate at Osaka University. My study would not have been possible without their continuous care and support. I am deep grateful to them.

I am grateful to Assistant Professor Sho Tsugawa of Graduate School of Economics, Osaka University, and to Dr. Junichi Murayama, NTT Corporation, for their valuable discussion, advice, support and encouragement during the course of this study.

The website protection scheme was studied at NTT Corporation. Mr. Takeo Hariu,

Mr. Naoto Tanimoto, Dr. Makoto Iwamura, Mr. Yuhei Kawakoya, Mr. Kazufumi Aoki, Mr. Mitsuaki Akiyama and Mr. Eitaro Shioji supported my studies. I am grateful to them and my colleagues at NTT Corporation.

I am thankful to all the members of the NTT Secure Platform Laboratories, NTT Corporation, for their continuous support and friendship.

Finally, I deeply thank my parents, my wife, Yuka, and my son, Haruki, for their understanding and hearty support and encouragement in my daily life. This work would not have been achieved without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The expansion of network technologies means that many kinds of important information such as personal information and foundational information are now being transmitted over the Internet. Consequently, the number of cyber-attacks, in which hackers try to obtain information illegally or interfere with the provision of normal services, has been increasing rapidly. Various approaches have been investigated as countermeasures against cyber-attacks. These approaches include constructing virtual private networks and community networks [1, 2, 3, 4, 5, 6, 7], and introducing measures to prevent attacks on the Internet. This thesis examines the latter techniques since the majority of network users connect to the Internet.

There are two types of attacks on the Internet. One type targets leakages of data by humans or organizations, and the other targets vulnerabilities of networks and software. Attacks in the former category include phishing [8, 9], which is an attack that attempts to acquire personal information of Internet users by using fake websites constructed by attackers, and spam mail [10, 11, 12], which is involves sending malicious e-mails to induce the users receiving it to connect to illegal websites and advertising e-mails. Many methods to solve these problems have been proposed, but these problems can best be solved by improving the computer literacy of users. In contrast, the attacks in the latter group cannot be solved only by improving user literacy because these attacks target vulnerabilities

of networks and software that are not generated by users. This thesis examines the latter attacks because technology is necessary to prevent such attacks.

There are two kinds of attacks that target vulnerabilities of networks and software. One tries to gain illegal control and illegal usage of network routing, bandwidth, and sessions. The other tries to gain illegal control and illegal usage of computer resources. In the former attacks, for example, on layer 2 of the open systems interconnection (OSI) reference model, ARP spoofing [13, 14, 15], which is an attempt to intercept user communication by registering an attacker's MAC address to an ARP cache table instead of the user's MAC address, is known to occur. On layer 3, BGP man-in-the-middle attacks [16, 17, 18], which try to control routing information by prepending a specific autonomous system (AS) number to an IP address prefix that is more limited than a legal IP address prefix, are known to occur. On layer 4, DSN cache poisoning [19, 20, 21], which tries to register an illegal IP address to a domain name on a DNS cache server by generating and sending fake packets, which are normally sent by a DNS root server, has been confirmed. In addition, on each layer, denial of service (DOS) attacks [22, 23, 24], which are attempts to shout down services by forcing victims to receive a lot of traffic or to process a lot of data, are a serious threat. Because these attacks target network infrastructures, network carriers and Internet service providers implement countermeasures that require them to spend financial and human resources on operating and continually monitoring them, and also on developing support systems for the countermeasures. Moreover, recently, the amount of software and its vulnerabilities has been increasing rapidly, so managing these vulnerabilities is becoming more difficult. This thesis discusses countermeasures for attacks due to software vulnerabilities.

Attacks that occur due to software vulnerabilities mainly targeted operating systems (OSs) and middleware. However, recently, applications have become a primary target because the number of applications and their vulnerabilities is increasing [25]. Therefore, this thesis also focuses on attacks that occur due to application vulnerabilities. In addition, the targets of attacks due to application vulnerabilities can be classified into user terminals and servers. In many cases, servers accumulate much more important information than each user terminal. Thus, this thesis considers server protection schemes against attacks

that are caused by application vulnerabilities.

Many Internet users now regularly send and receive information via web services such as homepages, blogs, news sites, and social networks that are provided by websites. In addition, many users construct their own websites and operate them by using the environment prepared by service providers such as hosting service providers and cloud service providers [26]. Website operators can construct websites easily without needing a high level of software skill because various web applications that are necessary to construct websites are now provided not only as paid software but also open source software (OSS). However, many web applications contain vulnerabilities that are being exploited for mounting attacks on websites [27]. These attacks come in various forms, such as structured query language (SQL) injection [28] that aims to tamper with content and cross-site scripting (XSS) [29] that mixes malicious script with valid script. Especially, a malicious software (malware) attack can be called the root of attacks [30] because a website that has been attacked would be freely manipulated by the attacker and used as a source terminal of new attacks [31]. In this type of attack, a web application programmed to obtain and execute a designated file is maliciously used to have that website download and execute malware. Security for an individual website has been traditionally managed by the website's operator, but in an environment in which websites take on the role of a service platform, as in a cloud computing environment, a service provider must manage the security of a great number of websites.

Attacks on websites can be divided into two main types: targeted attacks and indiscriminate attacks. The former attack a specific website after the attacker has investigated the most suitable type of attack for that site, and the latter attack many websites using a technique having a high probability of success against multiple websites. Targeted attacks are likely to be mounted against websites of large enterprises where highly valuable information is stored. Techniques for defending against them tend to require a financial investment, such as developing secure web applications [32] and deploying website managers. Indiscriminate attacks are usually mounted to store a resource (steppingstone) that can be used as a source terminal of attacks on other websites or general users. As such, they target small and medium-sized business websites in the cloud. In this thesis, we focus on

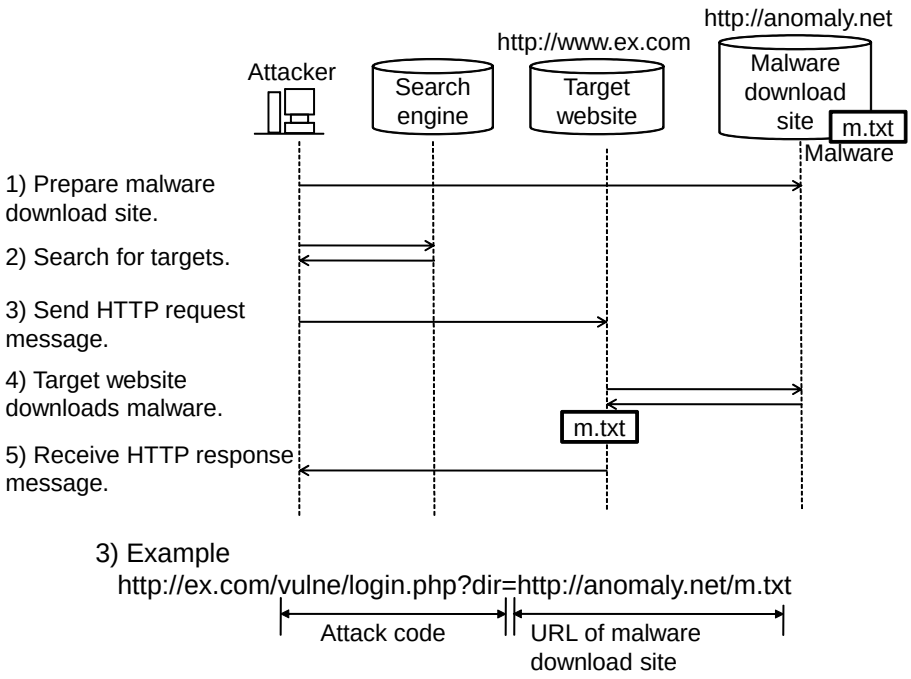

Figure 1.1: Attack procedure

indiscriminate attacks, which service providers are required to defend against.

Command injection [33] and remote file inclusion (RFI) [34] are effective techniques for infecting websites with malware. An RFI attack consists of the following steps, as shown in Fig. 1.1.

1. The attacker prepares a malware download site for holding malware. A malware download site is frequently an ordinary website being maliciously used by an attacker.In Fig. 1.1, a host name of the malware download site is anomaly.net and the filename of the malware is m.txt.

2. Using a search engine, the attacker finds a website with a vulnerable web application and treats it as the target website. In Fig. 1.1, the host name of the target website is www.ex.com.

3. The attacker sends the target website an hypertext transfer protocol (HTTP) request message that includes exploit code that can exploit the web application's vulnerability

so that the website will download malware from the malware download site. In Fig. 1.1, the exploit code is vulne/login.php?dir, which shows the vulnerable program, and http://anomaly.net/m.txt is the location of the malware.

4. The target website downloads and executes the malware from the malware download site.

5. The target website sends an HTTP response message to the attacker.

The attacker can determine whether the attack was successful from the contents of the HTTP response message and responses sent from the malware.

Attackers use botnets as attack sources. Recently, results of studies about anonymity [67, 68] are used by attackers. So it is difficult to detect not attack sources but attackers.

On the other hands, many ordinary websites are maliciously used by attackers as malware download sites. Because these sites may be listed in blacklists of security vendors for filtering accesses to and from the sites, these sites are victim, too.

Chapter 2

Related Works

There are two solutions for protecting websites from these attacks. One is the design and development of a website including web applications carried on the website. The other is the correspondence of vulnerabilities discovered after development of the website. To achieve the former solution, authentication design [35] and secure programming for websites [32] have been proposed. However, service providers should protect user websites without using these techniques because these techniques make it difficult to manage web applications for which there are many users. Therefore, we surveyed the correspondence of vulnerabilities discovered after development of a website.

To defend against attacks that infect websites with malware is to monitor access to websites and to filter out any access whose feature information matches that of an attack. The intrusion detection system (IDS), intrusion prevention system (IPS) and web application firewall (WAF) use this method. This approach requires collecting as much information about attack features as possible so that attacks can be detected with high accuracy. A method that collects attack feature information by placing decoy systems called honeypots on the Internet has been studied. As shown in Fig. 2.1, security vendors collect attacks on the Internet by honeypots and analyze the attack features to generate signature information, which is distributed to security appliances such as IPS, IDS and WAF. The security appliances detect an access from an attacker whether the access feature matches the signature

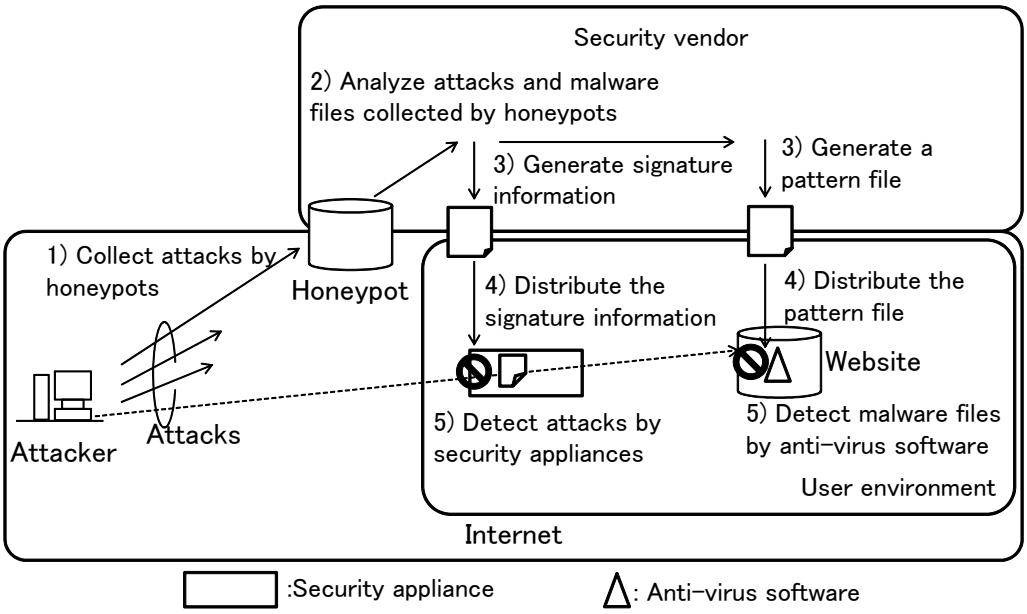

Figure 2.1: Attack prevention and collection

information. On the other hand, security vendors of anti-virus software generate pattern file, which have information of the characteristics of malware files they have collected and analyzed. In addition, anti-virus software, which is installed on websites, receives pattern files from security vendors and detects malware by comparing the characteristics of received files with the information written in the pattern files. Thus, to detect and filter attacks, not only attack prevention schemes but also attack collection schemes are necessary.

2.1 Attack Prevention

To protect websites from malware infection, which is caused by abuse of web application vulnerabilities, anti-virus software or security appliances can be used, as shown in Fig. 2.2.

Many security vendors have investigated a method for installing anti-virus software, and several types of anti-virus software have been developed to protect servers from malware infections [37]. In this solution, anti-virus software is installed on a server on which a website is hosted. The anti-virus software detects malware files from received files by

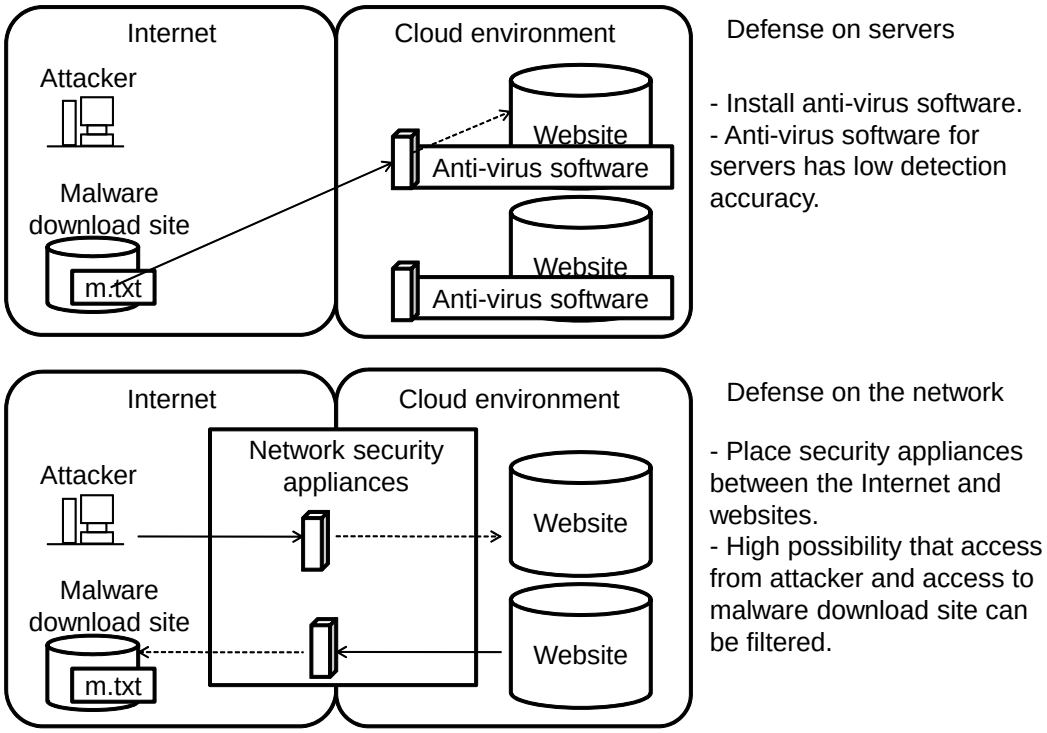

Figure 2.2: Server-based and network-based defenses

using two typical methods, a detection method using pattern files and a behavior blocking method. If the anti-virus software detects malware by using pattern files, it monitors received files and checks whether their characteristics are the same as known malware. Thus, the anti-virus software can detect malware based on the similarity between received files and known malware. Security vendors who provide anti-virus software collect malware from the Internet by using honeypots. Additionally, they analyze the malware and generate pattern files, which contain information about the characteristics of analyzed malware. Furthermore, they distribute the pattern files to anti-virus software installed on servers. When websites receive a file, the anti-virus software checks the characteristics of the file and compares them with information written in the pattern files. If the characteristics are the same with the information, the anti-virus software determines that the file is malware. To detect as much malware as possible, security vendors should collect many malware files by using many honeypots and analyze the malware files for pattern file generation. If

the anti-virus software can use behavior blocking method, the anti-virus software monitors file activities and prevents certain modifications to the operating system or related files. The anti-virus software defines malicious activities not caused by legitimate software but malware files by analyzing file activities of known malware files. For example, the anti-virus software monitors accesses to important files, execution of suspicious processes and commands, suspicious communication between external network, and huge usage of system resources such as memory. With this method, anti-virus software can detect malware files with characteristics not described in pattern files.

When anti-virus software using pattern files detects malware based on characteristics of known malware already collected and analyzed by security vendors, the anti-virus software should always update pattern files to avoid false negatives since new types of malware appear constantly. However, recently, a huge amount of malware files are generated in a very short time so it is difficult for security vendors to generate enough pattern files to prevent malware infections with high probability. Furthermore, to avoid false positives, pattern files should be generated from software used only for attacks. However, the characteristics of malware, which may be determined as legitimate software according to users and usage aims, cannot be described in the pattern files. For example, malware that sends server information to other terminals become legitimate tools when website managers use them on their websites, as shown in Fig. 2.3. In this case, it is difficult for security vendors to judge if these files are malware. Therefore, they cannot describe the characteristics of a pattern file to prevent false positives. As a result, anti-virus software may fail to detect certain types of malware. Although security vendors can use a behavior blocking method to improve the detection ratio of anti-virus software using pattern files, it is difficult to define malicious actions because of the same reason. This situation may deteriorate the detection ratio. There is therefore a need to monitor the network for attacks by using a network security appliance such as an IDS, IPS or WAF.

To protect websites in service provider environments, service providers generally place an WAF between the website and the external network. WAFs monitor HTTP accesses to a user website and detect attacks according to the access patterns. In this case, there

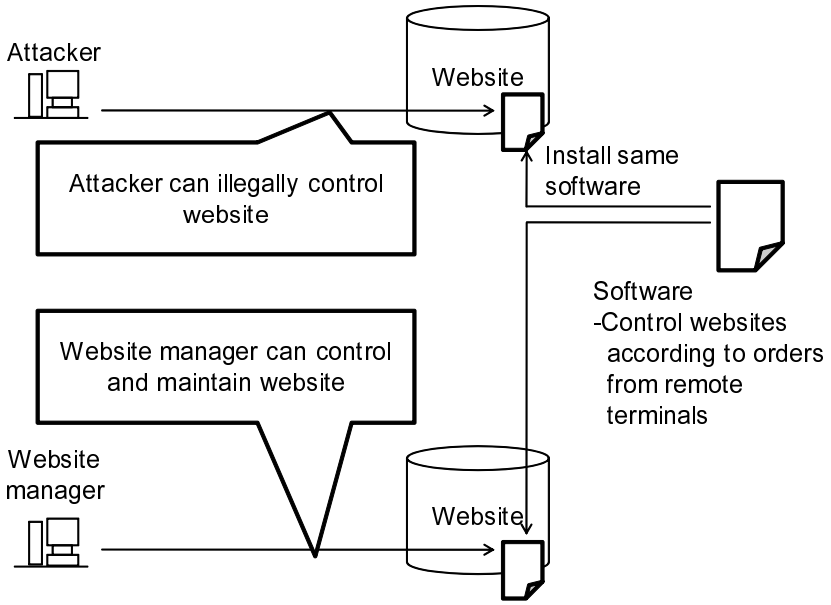

Figure 2.3: Design issue of anti-virus software

are two types of methods. One requires additional functions to be added to websites and the other does not. In the former, for example, the WAF and the website attach an identifier to forward packets between them [38]. These methods are used to detect attacks by checking the validity of the transition of the identifier. These methods are effective if each site manager protects his or her website individually. However, if service providers protect a large number of their user websites using these methods, the operation cost (time, labor, and money) is high as additional functions must be implemented for each website. Therefore, we used methods that do not require additional functions to be added to websites. These methods are classified by whether signatures are used or not. If operators want to detect unknown attacks, a WAF may not detect by signatures but instead detect the difference between a user access and regulated access patterns. In this case, an operator who manages a WAF tries to configure regulated accesses, which are likely generated by normal, or non-malicious, users, by each web application in advance as teaching data. If the WAF detects accesses that are not similar to the teaching data, it determines the accesses as those of an attacker [39, 40, 41]. These methods are effective for individually protecting

specific websites from targeted attacks. However, they must collect teaching data, which consist of information about regulated access patterns. In a service provider environment, dynamically collecting and updating teaching data is difficult because there are various web applications independently uploaded by each user in the environment. Therefore, we used methods that use signatures.

In these methods, when a WAF monitor accesses user websites, it detects an attacker's accesses by detecting the similarity between the patterns of the accesses and signatures. These signatures are generated by security vendors who collect and analyze known attack information to reveal the exploit code [42]. To generate signatures, security vendors use a technology that can discover new vulnerabilities by analyzing web applications [43] and a technology that can collect and analyze attack information by using web honeypots. The former is effective for environments that individually protect a specific website whose web applications will be provided for analysis. However, a large variety of unknown web applications will be deployed on many websites in service provider environments, so the latter technology is suitable for service providers. When web honeypots are used, normal HTTP accesses may be made to them. In addition, communication may be generated among the malware, some of which may have already infected the web honeypots. Therefore, security vendors should generate signatures by extracting exploit codes manually from communication records that consist of many kinds of information.

2.2 Honeypot

Honeypots are computing resources where their value lies in the information they capture while being probed, attacked or compromised. Many organizations research and develop honeypots and collect attacks from the Internet [66].

There are two major types of honeypots [44], as shown in Fig. 2.4. One is a client honeypot, which mimics vulnerable clients such as user terminals. This type of honeypot collects attacks on clients. For example, it is used to find malicious websites that send exploit codes to hack user terminals by abusing vulnerabilities of web browsers. On the

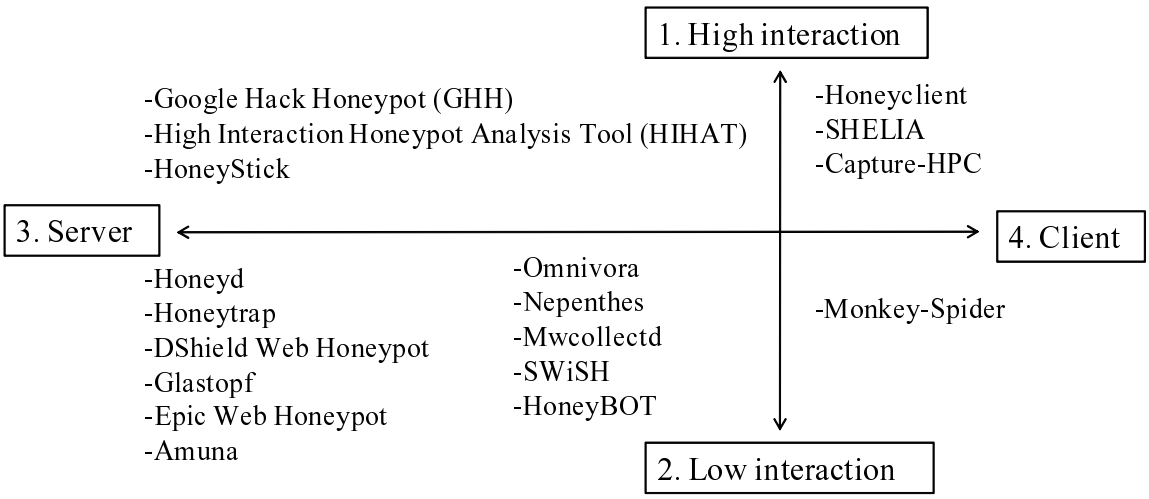

Figure 2.4: Honeypot map

other hand, a server honeypot, which mimics a vulnerable server, is used if it is necessary to collect attacks on websites.

Web honeypots [45, 46, 47, 48, 49, 50], which collect attacks on websites by exploiting the vulnerability of web applications, are classified as low interaction and high interaction. Low-interaction web honeypots are designed to emulate websites hosting vulnerable web applications. High-interaction web honeypots are composed of vulnerable systems, such as decoy websites, which accommodate such actual applications and are actually attacked through these applications, and surveillance functions. Low-interaction honeypots safely collect attack information because the systems do not execute malware. However, the attack information is limited because attackers cannot compromise these systems since the attackers interact with just a simulation. On the other hand, it is difficult to manage high-interaction honeypots because their systems temporally execute the malware, but these honeypots can collect much more attack information than what low-interaction honeypots can.

The Google hack honeypot [45], which can efficiently collect attacks only, and Glastopf [46] and the Dshield web honeypot [47], which enhance fidelity for adapting to attackers, have been studied as low-interaction web honeypots. The Google hack honeypot links from

known websites, which have already appeared in Google search results, to a website emulator, which records received traffic, so as not to be seen by humans. With this mechanism, the Google hack honeypot can collect traffic automatically sent by tools based on the results of Google searches, not traffic from normal users. There is a high probability that such traffic consists of attacks; therefore, the Google hack honeypot can efficiently collect attacks. However, if a service provider wants to continually collect attacks to find new attack information, it is necessary to enhance the fidelity of web honeypots for adapting to attackers in order to become the target of attacks. To enhance fidelity, Glastopf extracts URLs from HTTP request messages and obtains programs from those URLs. Glastopf can collect files that may have been ordered to download by the received HTTP request messages. In the Dshield web honeypot, operators can configure HTTP response messages, each of which is sent according to the information of a received HTTP request message, in advance. The Dshield web honeypot can send an HTTP response message, which is configured for a special source IP address, such as a search engine crawler's IP addresses, known attacker's IP addresses, or a destination web application program. Low-interaction web honeypots, such as Glastopf and Dshield web honeypot, can emulate actual vulnerable websites with high precision to enhance adaptability to attackers.

However, these low-interaction web honeypots cannot execute malware programs, so it is impossible to generate perfect HTTP response messages that reflect actions of web applications and malware programs. In attacks on websites, exploit codes are described in HTTP request messages, so attackers receive HTTP response messages corresponding to the HTTP request messages they sent. Therefore, attackers know whether their target is a low-interaction web honeypot by analyzing results of web application and malware program operations, which are written in HTTP response messages. Of course malware, such as a downloader, is not executed. Consequently, the web honeypot cannot collect malware programs downloaded sequentially. Thus, if service providers want to continually collect and analyze a large amount of attack information from not only simple but also complex attacks, it is better to deploy high-interaction web honeypots.

A high-interaction web honeypot contains a decoy website that has vulnerable web

applications installed. For example, in the high-interaction honeypot analysis tool [48], programs, which record access logs, are inserted into web application programs in a decoy website. The web honeypot can then record attacks, which are received by web application programs, and actions resulting from the attacks. In other high-interaction web honeypots, accesses, which result from HTTP request messages from the Internet to a decoy website, are recorded. In such high-interaction web honeypots, an HTTP response message resulting from an HTTP request message from an attacker is generated by a vulnerable web application program, which receives the attack. Therefore, these web honeypots can exhibit high fidelity.

If a vulnerable web application program on a high-interaction web honeypot received an attack from the Internet, the program downloads and executes a malware program by using functions of the vulnerable web application program, which download and execute designated files. Moreover, if the malware program downloads additional malware programs, such as a downloader, the high-interaction web honeypot downloads additional malware programs from other malware download sites and executes them. Thus, when vulnerable web application programs on the high-interaction web honeypot can receive attacks, in other words, attacks to the web honeypot are successful, the web honeypot can collect many malware programs and specify many malware download sites.

Chapter 3

Requirements and Design Issues for Website Protection Schemes

3.1 Requirements

To protect websites, on which many web applications are typically installed, from attacks, service providers need to detect various types of attacks. To detect many attacks, service providers should collect a lot of information about the characteristics of known attacks. Furthermore, to detect attacks with high accuracy, the number of false positives, in which normal information is recognized as malicious, and false negatives, in which malicious information is not recognized as malicious, should be restricted. Details are as follows.

3.1.1 Detection of Diverse Attack

Website managers select various applications for their websites. Many of these web applications have vulnerabilities that can be illegally exploited by attackers. However, the website managers not only spend money and human resources on security but also require service providers to provide security. It is difficult for service providers to confirm the construction of each website and deploy countermeasures for each website because such an effort would incur an enormous cost and require extensive human resources.

Consequently, service providers should detect various attacks occurring in accesses to and from all websites

3.1.2 Ability of Information Extraction

There are three types of attack detection schemes: signature detection, which identifies accesses whose characteristics are the same as those of known attacks; anomaly detection, which identifies unusual access frequencies and amounts of traffic; and behavior detection, which identifies accesses whose patterns are similar to those of known attacks. For all three schemes, it is necessary to collect some information to be used as a basis to determine whether accesses to and from websites are normal accesses or attacks. The basis information is generally extracted from known attacks collected from the Internet by web honeypots. However, it is difficult to collect known attacks because the attack collection depends on external requirements, for example, whether the attackers select the web honeypots as a target, and whether search engines register the web honeypots as websites.

Therefore, service providers should extract a lot of information from attacks collected by web honeypots.

3.1.3 Accuracy of Attack Detection

To carry out successful attacks, attackers try to avoid being detected so as not to be identified as malware and have their attack characteristics collected and analyzed by service providers and security vendors. A case in which an attack is not detected is called a “false negative”. To avoid false negatives, service providers and security vendors may extract accesses that contain elements that are similar to characteristics of known attacks. A case in which a normal access is detected as an attack is called a “false positive”. False negatives cause security incidents, and false positives lead to a decline in service quality.

As a result, service providers should establish high-accuracy attack detection schemes that can reduce the number of both false negatives and false positives.

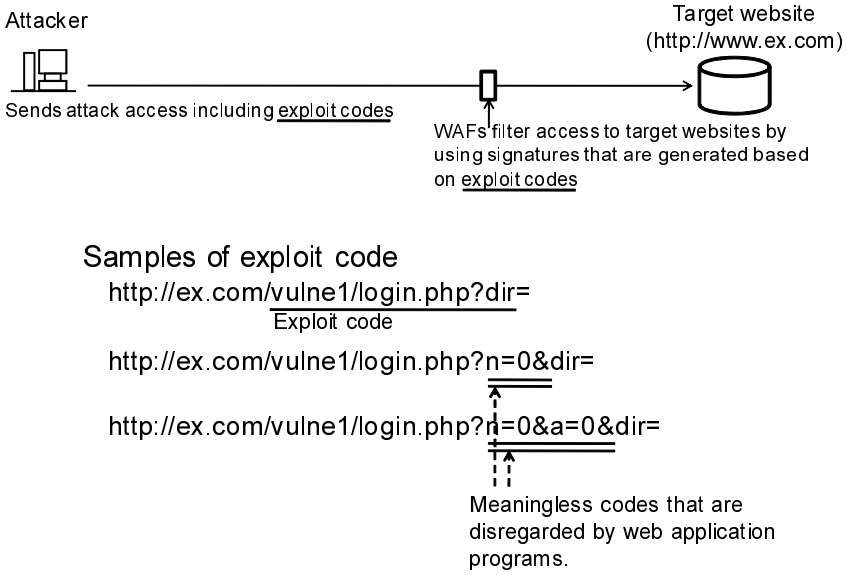

Figure 3.1: Patterns of exploit codes

3.2 Design Issue for Diversity and Accuracy

Conventional schemes, which detect attacks by using signatures based on information collected by web honeypots, cannot detect unknown attacks whose exploit codes have not yet been analyzed.

As shown in Fig. 3.1, WAFs filter access to target websites by using signatures that are generated based on exploit codes. These exploit codes are generated based on the vulnerabilities of web applications. These vulnerabilities are continuously being discovered by security vendors, attackers, developers, and others. Thus, the number of vulnerabilities, indicated as $V(t)$, increases constantly over time, t. In addition, attackers mix an exploit code with meaningless codes that are disregarded by web application programs. In Fig. 3.1, n=0 and n=0&a=0 are meaningless codes, which are located as parameters and values. The number of meaningless codes, indicated as $D(t)$, increases rapidly over time. Therefore, the function $S_o(t)$ showing the number of signatures that should be generated for a web application is as follows.

$$S_o(t) = V(t)D(t) \tag{3.1}$$

Moreover, in service provider environments, the number of web applications used by users, represented as $N_a(t)$, fluctuates over time. The function $S_a(t)$ showing the number of signatures that should be generated for service provider environments is as follows.

$$S_a(t) = D(t) \sum_{i-1}^{N_a(t)} V_i(t) \tag{3.2}$$

Currently, $N_a(t)$ is increasing rapidly because of the wide spread of OSS, Web 2.0, and cloud computing.

In contrast, the function $S_s(t)$ showing the number of signatures generated by security venders uses α, which shows the number of signatures generated by a security vender per time unit.

$$S_s(t) = \alpha t \tag{3.3}$$

It currently takes at least two weeks to generate and update a signature [51], so α would not be too large.

Consequently, the number of exploit codes that have not had signatures generated in service provider environments, $E(t)$, is as follows.

$$E(t) = S_a(t) - S_s(t) = D(t) \sum_{i-1}^{N_a(t)} V_i(t) - \alpha t \tag{3.4}$$

The $S_a(t)$ value is larger than that of $S_s(t)$, so $E(t)$ increases according to the increase in t. The value of t is over ten years as attacks exploiting the vulnerabilities of web applications on websites have been monitored for that long. Consequently, $E(t)$ increases rapidly. Thus, there is a high possibility that conventional schemes cannot detect many of the attacks.

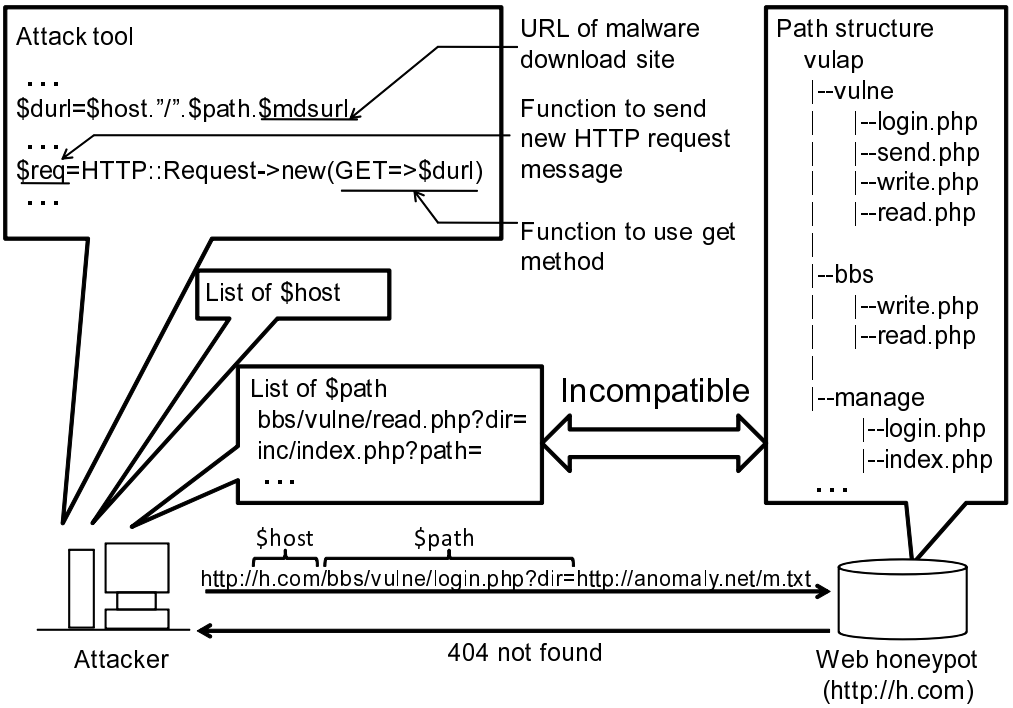

Figure 3.2: Design issue of web honeypots

3.3 Design Issue for Information Extraction

The path structure of a decoy website is determined by the installer and version of the web applications. As shown in Fig.3.2, however, automatic tools, with which many attacks on websites are sent, have path lists, which are described in the destination URL of the attack, made by tool makers or attackers in advance. Consequently, attacks may be likely to fail because a path, in which a vulnerable web application program is located, and a path, which is described in the destination URL of the attack, likely differ. When attacks fail, it is difficult for the web honeypot to specify MDSs and collect malware programs. It goes without saying that the web honeypot cannot specify additional malware programs and MDSs from malware programs that are used in failed attacks and cannot launch further network attacks. In addition, if an attack fails, an HTTP response message for the attack is an error message such as "404 not found". In this case, attackers may exclude the web honeypot from target websites because it seems to attackers that the website, the web

honeypot, does not have paths in which vulnerable web application programs are installed.

3.4 Approach toward These Issues

To meet the requirements for a website protection scheme, this thesis proposes and evaluates a malware download site blacklisting scheme to detect diverse attacks in chapter 4. To evaluate the proposed scheme and to reveal the limitations of conventional countermeasures, detection ratios of the blacklisting scheme and those of anti-virus software are compared. Additionally, in chapter 5, to maximize information that can be extracted from attacks, this thesis proposes and evaluates web honeypots, which can collect attack information such as malware download sites, attack sources, and malware. Furthermore, in chapter 6, to reduce the number of false negatives and false positives, this thesis proposes and evaluates schemes for optimizing blacklist update frequency based on the behavior analysis of malware attackers.

Chapter 4

Detection of Diverse Attacks by Provider-Provisioned Website Protection Based on Network-Based Blacklisting Scheme

4.1 Network-Based Blacklisting Scheme

In conventional schemes, increase in $S_a(t)$ causes a bottleneck. Additionally, it is problematic for conventional schemes that α of $S_s(t)$ is not large. Thus, increase in $S_s(t)$ (which will improve α) and decrease in $S_a(t)$ are requirements for website protection schemes. To reduce $S_a(t)$, it is necessary for website protection schemes to use information that is shared among various attacks using different exploit codes.

To meet the requirements for provider-provisioned website protection schemes, we propose a scheme for detecting and filtering not the exploit codes but malware download

requests from websites that receive exploit codes. In our proposed scheme, as shown in Fig. 4.1, we filter accesses that have the destination URL of a malware download request from a web honeypot to protect user websites. In Fig. 4.1, after a web honeypot received malware named m.txt from http://anomaly.net , we filter HTTP messages, whose destination URL is http://anomaly.net/m.txt, from user websites. In addition, service providers filter accesses from websites to MDSs by using URL filtering technologies. Our scheme focuses on the high possibility of attacks that do not use the same exploit code but share the same MDS. Our scheme attempts to reduce $S_a(t)$. With our scheme, accesses from web honeypots to MDSs are identified by dynamic analysis of the extracted communication data on an HTTP request message to a web honeypot. By this process, to increase $S_s(t)$, our scheme extracts communication data that results from an exploit code and specifies the URLs of MDSs automatically. A more detailed explanation of each process follows.

4.1.1 Protection using URLs of Malware Download Sites

The proposed scheme filters not HTTP request messages in which exploit codes are written but malware download request to MDSs. Our scheme identifies MDSs from the communication records of web honeypots. Therefore, many web honeypots should be located at various URL domains and blocks of IP addresses to collect a large amount of information. In this case, decrease in $S_s(t)$ is important because the number of web honeypots will increase.

4.1.2 Automated Analysis of Web Honeypots

To reduce $S_s(t)$, our scheme deploys a function for extracting communication data including an exploit code from the communication data to each web honeypot. In addition, an analyzing function, which extracts the URLs of MDSs from communication records, is implemented. It is necessary for the former function to eliminate other communication records of other exploit codes, such as malware and regulated accesses. In addition, it is necessary for the latter function to analyze downloaded programs and determine whether they are really malware. In our scheme, the extraction function is implemented in a web honeypot

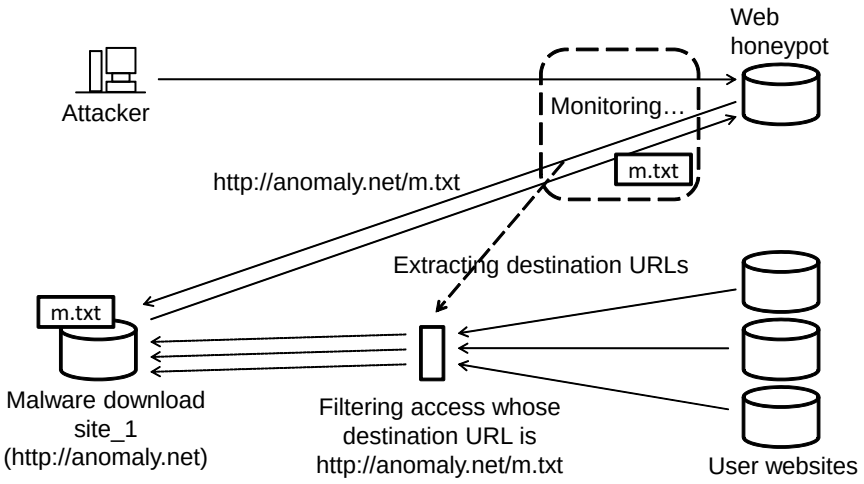

Figure 4.1: Proposed scheme

and the analyzing function is implemented as a web-attack analyzer. Our web honeypot records all communication information created by an HTTP request message. In addition, the web honeypot sends the recorded information to the web-attack analyzer if parameters and values are described in a destination URL of an HTTP request message because there is a high possibility that these messages are attacks. The web-attack analyzer confirms whether malware are included in the recorded information by recreating the communication in a closed virtual network, and specifies MDS when the recorded information includes malware. A more detailed explanation of these devices in the architecture follows.

Web Honeypot

In our scheme, a web honeypot is composed of a decoy website and a controller. A decoy website carries many types of selected vulnerable web applications. Additionally, it is constructed so as not to send download requests to an external network using standard procedures. For instance, users cannot select all the web pages of a decoy website freely using standard procedures. The controller is used as a reverse proxy server and is located between the decoy website and an external network.

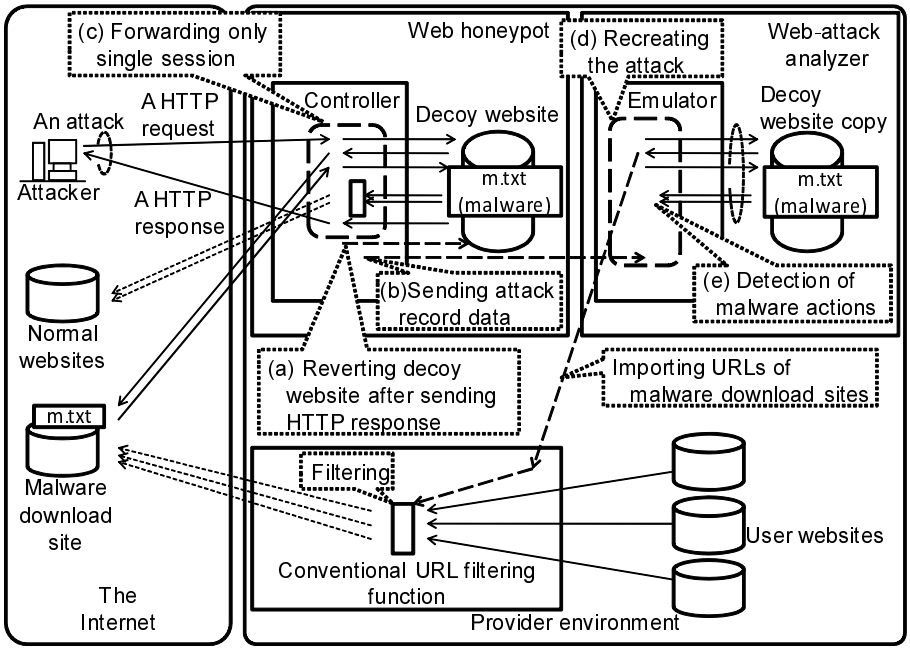

Figure 4.2: Architecture design

Exploit codes are sent to websites as HTTP request messages. Thus, websites send HTTP response messages after executing exploit codes and finishing malware downloads. Our scheme focuses on this characteristic. The controller receives all the accesses between the decoy website and an external network and records the contents before forwarding the accesses. Moreover, as shown in Fig. 4.2(a), when the controller detects an HTTP response message from a decoy website, the controller reverts the decoy website to the website's state prior to receiving the HTTP request message because the decoy website may have been infected by malware. The controller can do so because it has image files of the decoy website and can overwrite the decoy website by using these image files. Finally, the controller sends a communication record as attack record data to the web-attack analyzer (Fig. 4.2(b)). This makes it possible to prevent the communication record of an exploit code from mixing with the communication records of other malware, some of which may infect the web honeypots through other exploit codes. In addition, the controller stops forwarding other HTTP request messages to a decoy website while that decoy website is handling an

HTTP request message (Fig. 4.2(c)). By this access control, it becomes possible to prevent the communication record of one exploit code from mixing with communication records of other exploit codes. Consequently, the controller can generate attack record data composed of only the communication records of a specific exploit code.

There is a possibility that a decoy website will send attack traffic to the Internet because it may be infected by malware and thus be controlled by attackers before reverting the decoy website. From a security viewpoint, the controller should filter the attack traffic from the decoy website. However, if filtering is detected by attackers, they may avoid the decoy website. Therefore, it is necessary for attackers not to question that web honeypots are decoy websites while filtering attack traffic from the decoy website. To satisfy this requirement, when the controller receives attack traffic from a decoy website, the controller not only filters the attack traffic but also generates a response message to the attack traffic and sends it to the decoy website. For detecting attack traffic, the controller confirms any similarity between received traffic and traffic patterns that may be sent by servers infected with malware. This traffic pattern can be obtained using static or dynamic analysis [52, 53] of conventional malware. Furthermore, the controller can generate a response message to attack traffic using these static or dynamic analysis results and by using a conventional tool such as Honeyd [54].

Web-Attack Analyzer

The web-attack analyzer consists of an emulator and a decoy website copy. The emulator has an interface for controlling the web honeypot. The decoy website copy constructs a closed virtual network with the emulator; In other words, the web-attack analyzer does not have access to the Internet.

The emulator receives an attack record data from the web honeypot collector (Fig. 4.2(a)). The emulator refers to the data and recreates an attacker's accesses to a decoy website, and these accesses are sent to the decoy website copy (Fig. 4.2(d)). The decoy website copy then initiates a download request as if it were a decoy website. At this time, the emulator generates a response to the download request and sends it to the decoy website copy by

referring to the attack record data. Thus, the emulator recreates the download sequence of the program. Incidentally, malware used to exploit websites attempt to confirm the communication routes to the terminal of an attacker, such as backdoors. Therefore, after sending the response to the download request, the emulator monitors whether the decoy website sends traffic to the Internet (Fig. 4.2(e)). If the decoy website does so, the emulator determines that the program is malware and that the destination of the download request is an MDS. In addition, by using this result, the emulator generates a list that contains the URLs of MDSs. The list is composed of URLs and IP addresses, so it can be imported into conventional access control devices as a filtering list. Thus, the web-attack analyzer extracts information immediately for protecting websites through real-time reproduction of actual attacks on web honeypots. Consequently, it is possible to detect MDSs instantaneously and automatically.

4.2 Evaluation of Proposed and Conventional Website Protection Scheme

There are two requirements for protecting websites with high precision, reduction in the amount of information needed for generating filter regulations, and reduction in the calculation time for specifying filter regulations. Therefore, we first evaluate our proposed scheme for the first requirement by connecting prototype web honeypots to the Internet to collect and analyze attacks to websites. Furthermore, we evaluate our proposed scheme for the second requirement by using prototype web honeypots and web-attack analyzers.

4.2.1 Prototype Systems and Experiment Environments

The experiment environment for evaluating the first requirement, which is met by our protection scheme using the URLs of MDS, is shown in Fig. 4.3. The experiment environment for evaluating the second requirement, which is met by our automated analysis of web honeypots, is shown in Fig. 4.4. The decoy websites carried web applications of OSS, which were programmed using PHP. The web honeypots were implemented on a physical server by

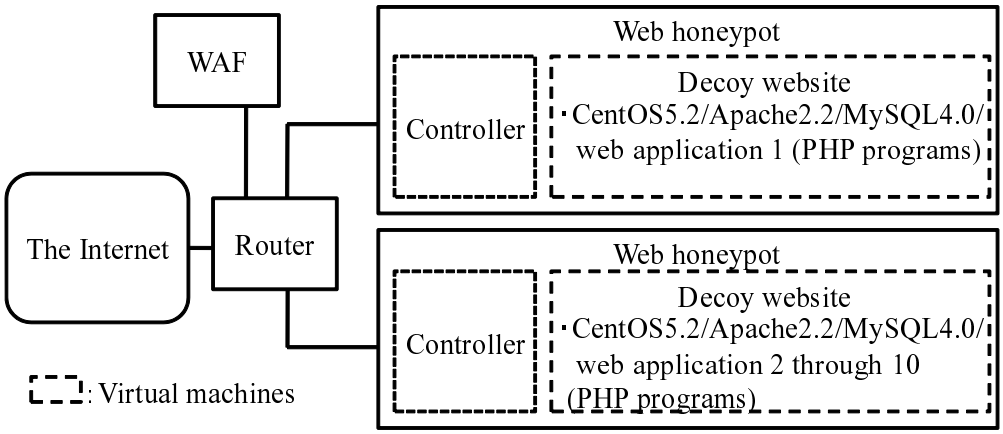

Figure 4.3: First experiment environment

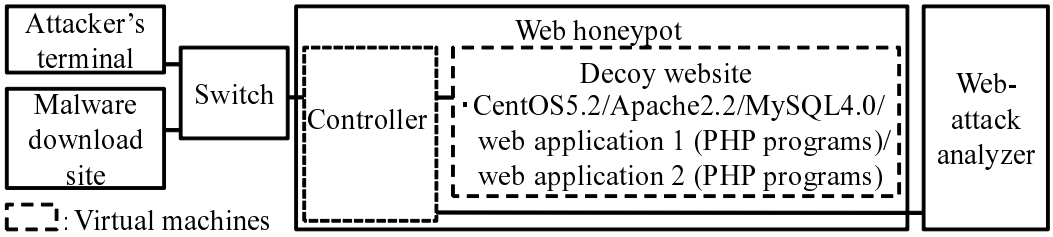

Figure 4.4: Second experiment environment

constructing a controller and a decoy website on virtual machines, which were constructed using technologies such as VMware [55]. The web-attack analyzer was also implemented on a physical server by constructing an emulator and a decoy web copy on virtual machines. The specifications of the physical server and virtual machines are listed in Table 4.1. As shown in Fig. 4.3, we connected a WAF to a router and forwarded the copied traffic, which forwards to a decoy website, to the WAF. The signature of this WAF had been newly updated.

Table 4.1: Specifications of physical servers.

CPU	3.8GigaHz x 2
VMware	VMware server 1.08
Memory	4GB (512MB for each virtual machine)
Disk space	73GB (20GB for each virtual machine)

Table 4.2: Results of WAF.

Total number of attacks	867
Attacks detected	580
Detection ratio	67%

4.2.2 Evaluation of Protection Scheme

In this experiment, we analyzed the increase in information used for protecting attacks with the conventional schemes and proposed scheme. We collected attacks on web honeypots from the Internet between January 30, 2009 and April 1, 2009 using the environment depicted in Fig. 4.3. The number of attacks detected by WAF is listed in Table 4.2. In addition, the numbers of exploit codes and MDSs are listed in Table 4.3. To reveal the relationships between MDSs and web applications whose vulnerabilities greatly affect the kinds of exploit codes generated, we investigated the probability that an MDS is used by attacks that exploit the vulnerabilities of different web applications. The results are listed in Table 4.4. Furthermore, to reveal the tendency of information used for protecting against attacks, we investigated a time series of the accumulated numbers of exploit codes and MDSs, as shown in Fig. 4.5. MDSs are generated by normal websites being abused. Therefore, managers of normal websites may find and stop the abuse of their websites. As a result, the number of websites from which malware can be downloaded may decrease. To reveal the actual status of this, we investigated a time series of the number of websites from which malware can be downloaded, as shown in Fig. 4.6.

A high probability that a WAF can detect only limited attacks is shown in Table 4.2. First, it is necessary to clarify the factors of this result. As shown in Table 4.3, the number of

Table 4.3: Number of exploit codes and MDSs.

Total number of attacks	867
Number of unique exploit codes	160
Number of unique MDSs	70

Table 4.4: Probability of sharing MDS.

	Probability
Sharing multiple web applications	60%
Used only a web application	36%
Unknown	4%

exploit codes is larger than the number of MDSs. As shown in Fig. 4.5, there are two increase patterns for the exploit codes. In the early stages of the experiment, the accumulated numbers of exploit codes increases rapidly because almost all the major exploit codes that are included in many attacks are monitored over a short term. After the early stages, minor and new exploit codes may be monitored. At this stage, the increase in the latter becomes nonlinear. To generate signatures for WAFs, it is necessary to clarify vulnerabilities of programs and consider exploit as many patterns that correspond to vulnerabilities as possible. Figure 4.5 reveals that there are many kinds of exploit codes and their number increases rapidly. It will be difficult to generate signatures by analyzing these exploit codes. In addition, in the case of WAFs, to generate signatures for all kinds of attacks, such as SQL injection, analysts should extract traffic data from each kind of attack. After that, analysts, for example, can obtain traffic data of malware distribution (Fig. 4.5). Thus, a large amount of time and labor are required to generate signatures of WAFs for defending malware distribution. On the other hand, as shown in Table 4.3, the number of MDSs is small. Additionally, the increase in the number of MDSs becomes linear (Fig. 4.5). Furthermore, the time series of the number of websites from which malware can be downloaded stabilizes (Fig. 4.6). This shows that the number of URLs that should be filtered is constant. Finally, more than half of the MDSs are used by attacks that each exploits the vulnerabilities of

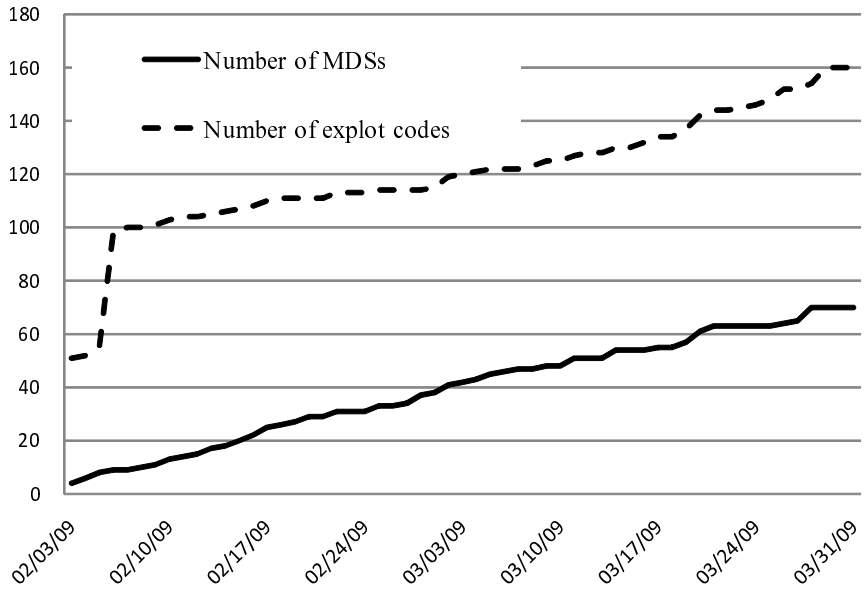

Figure 4.5: Accumulated numbers of exploit codes and MDSs

Table 4.5: Attacks data.

Kind of application	Malware
application_1	freeze1.pl, freeze2.pl, freeze3.pl
application_2	echotest.php

different web applications, as shown in Table 4.4. This shows that an MDS is shared by attacks, each of which uses different exploit codes. Moreover, the result shows that our proposed scheme can generate filter regulation without installing all the web applications of service providers.

4.2.3 Evaluation of Automatic Analysis System

In this experiment, we evaluated the calculation time for specifying filter regulation in the environment shown in Fig. 4.4 using a decoy website that had two kinds of web applications. First, we forced the decoy website to download Perl programs, which send data, using a

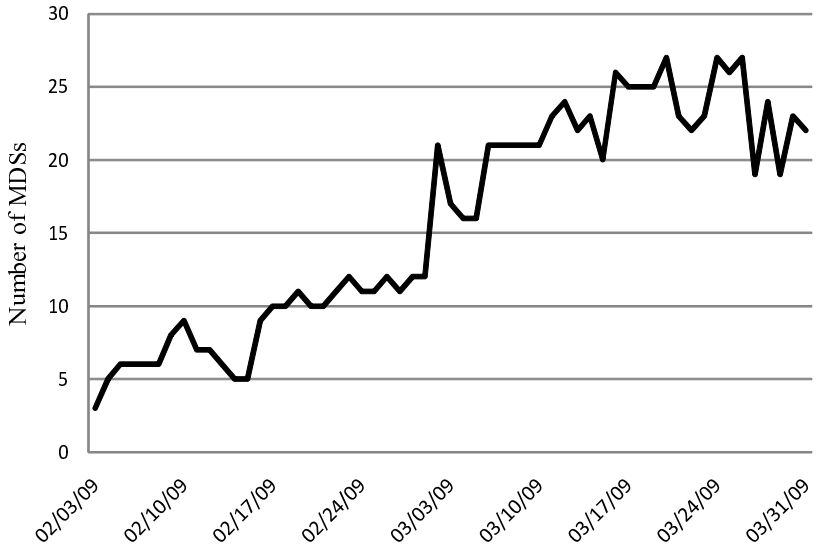

Figure 4.6: Number of sites from which malware can be downloaded

command injection into web application_1, as shown in Table 4.5. To avoid abuse of this information, the names and details of these applications are not made public. At this time, we forced the decoy website (URL http://www.ex.com) to download Perl programs such as freeze1.pl from an MDS (URL http://anomaly.net) multiple times. In this experiment, the number of downloads from web application_1 increased from one to three. Furthermore, we evaluated the calculation time for the proposed scheme to specify the URL of the MDS by inputting these attack data into a prototype of the web-attack analyzer. Next, we forced the decoy website to download a PHP program, which sends data, using remote file inclusion on web application_2. We tested each set of conditions five times. The calculation times for analyzing MDSs are listed in Table 4.6. If vulnerable programs and traffic data of an attack have been already known, the conventional signature-generation tool [56] can be deployed for generating a signature based on the exploit codes of web applications, Php-post and MyBulletinBoard. This tool requires more than 600 seconds to generate a signature. In addition, WAFs are updated every two weeks, as discussed in chapter 3.

As shown in Table 4.6, conventional tools have the advantage that vulnerable programs

Table 4.6: Results of second experiment.

Environments		Times [seconds]		
Kind of application	Number of downloads	Min.	Ave.	Max.
application_1	1	21.73	23.61	24.34
application_1	2	37.35	38.92	40.17
application_1	3	54.62	56.50	58.57
application_2	1	25.08	25.48	25.97
Conventional tool		600		
Conventional WAF		2 [weeks]		

and traffic data of an attack are already known, the proposed scheme can be used to reduce the calculation time taken by conventional schemes for analysis to 10%. It is difficult to search vulnerable programs, and it requires a large amount of time and labor to collect traffic data of attacks, as discussed in chapter 2, so the advantage of conventional tools is large. In addition, the calculation time for analysis with the proposed scheme is proportional to the number of downloaded malware. Moreover, the results of one download by each of two applications are approximately equivalent, so the calculation time for analysis does not depend on the web applications and their vulnerabilities. Thus, in the proposed scheme, the calculation time for analysis will not increase exponentially, making it possible to instantly analyze attack information. In this experiment, all the results show that http://anomaly.net/ is an MDS. Thus, the proposed scheme can be used to protect websites economically because filtering a single URL can protect multiple web applications and their vulnerabilities.

4.2.4 Discussion

As with conventional schemes, our scheme generates a filtering regulation from the attack information on the web honeypot. Because of this, there is a small possibility that websites are targeted by attackers before the web honeypots receive the attacks. To reduce this risk, web honeypots are located at various URL domains and blocks of IP addresses. In addition,

the management server has a reputation function, which collects filter lists from all of an attacker's accesses, and analyzing functions, such as web-attack analyzers. It also uses the filter lists for all the URL filtering functions. This server is located in service provider environments. By using the server, the risk of attacks on websites can be reduced. It is difficult for conventional schemes, in which filtering regulations are generated by manually analyzing web honeypots, to distribute many web honeypots because of the rapid increase in analysis costs. In contrast, the proposed scheme can distribute many web honeypots economically because filter regulations are generated automatically.

In addition, from the URLs of MDSs, service providers detect websites that already have access to MDSs by checking the access logs of the websites. By using this characteristic, service providers can detect malware infections in websites before the websites generate other cyber attacks.

The proposed scheme is not a means for constructing solid websites by spending a large amount of time and labor but a means for strengthening the security of all websites in service provider environments with high cost efficiency. Our scheme can detect and defend against malware distribution to websites efficiently and effectively, but WAFs can detect and defend against other kinds of attacks to websites, such as SQL injections and cross-site scripting. Therefore, if the service provider wants to construct solid websites for premium service, our scheme can be used with conventional technologies, such as a WAF, so that the service provider can strengthen the security of each website according to user demand. Thus, a service provider can provide various security services on the basis of cost.

4.3 Evaluation of Proposed Scheme and Anti-Virus Software

To quantitatively evaluate actual malware and communication opponents of malware infections, we investigated the detection ratio of anti-virus software to malware distributed to websites. The malware files are collected by web honeypots connected to the Internet. Moreover, we investigated communication opponents on the web honeypots and evaluated the blacklist-based filtering method using the communication information.

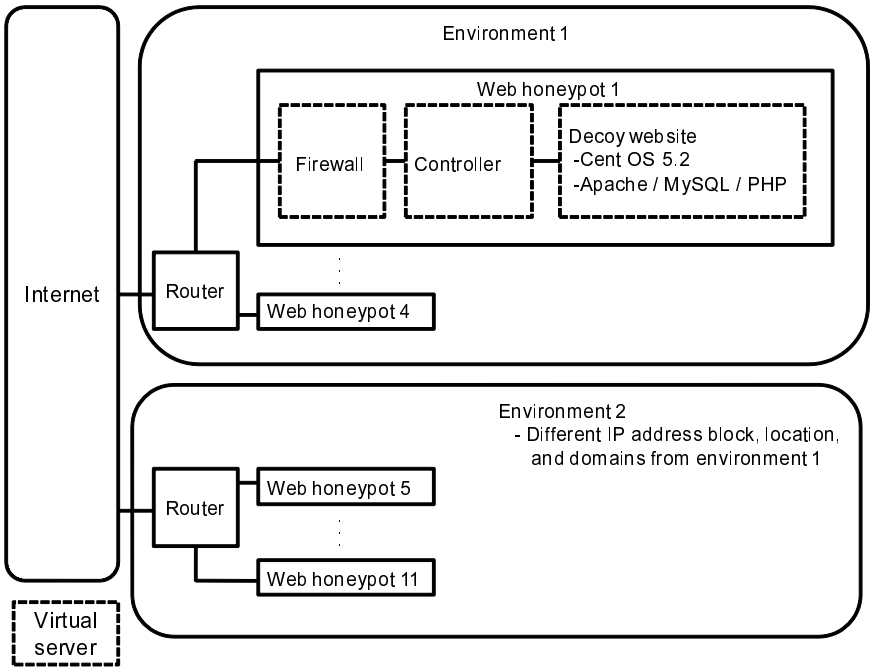

Figure 4.7: Investigation environments

Table 4.7: Specifications of physical servers.

CPU	3.8GigaHz x 2
Virtual machine	VMware server 1.08
Memory	At least 512MB for each virtual machine
Disk space	At least 20GB for each virtual machine

In these evaluations, eleven web honeypots, which included a decoy website and a controller, were deployed on a virtual machine of a physical server and located on two different address blocks, as shown in Fig. 4.7. The specifications of the physical server and virtual machines are listed in Table 4.7. Malware files were collected from the Internet between September 8, 2009 and January 30, 2010. Decoy websites in the high-interaction web honeypots were registered on search engines on the Internet during this period to collect attacks, as described in chapter 1. A total of 15 vulnerable web applications [57] were deployed. These web applications had RFI and command injection vulnerabilities. Attacks that occur due to vulnerabilities of web applications depend not on the OS and middleware but

Table 4.8: Attack information collected by web honeypots

	Total
Number of RFI attacks	4,621
Number of malware download sites	2,666
Number of malware files	366

on the applications. In addition, many service providers deploy linux OSs for many users because the OSs are free. Consequently, we used the environment shown in Fig. 4.7. We used anti-virus software programs appropriate for linux servers.

The attack information collected by web honeypots in this investigation is listed in Table 4.8. In this investigation, 4,621 attacks were collected. These attackers collectively used 2,666 malware download sites. Additionally, 366 types of malware were identified using the SHA1 value of malware files.

4.3.1 Evaluation of Detection by Anti-Virus Software

To evaluate the detection ratio of anti-virus software, we selected six types of anti-virus software for server protection, which were developed by several security vendors [58]. These anti-virus software programs can detect not only malware files but also spam emails and accesses to malicious websites. In the evaluation, detection ratios of the anti-virus software by using only malware files were investigated.

In the evaluation, as shown in Fig. 4.8, the detection ratio of malware collected by web honeypots was checked on May 20, 2010 using six types of anti-virus software whose pattern files were newly updated. This shows that security vendors were able to collect and analyze many malware files from January 30 to May 20. This evaluation reveals the limitation of the detection ratio of anti-virus software. Even though a behavior blocking method can monitor actions caused by malware when the attack is successful, many attacks on websites fail [59]. In order to generate all actions caused by malware and to check the actions by using the behavior blocking method for evaluating the detection ratio, we sent malware files to a dummy vulnerable web server, to which anti-virus software using a behavior blocking

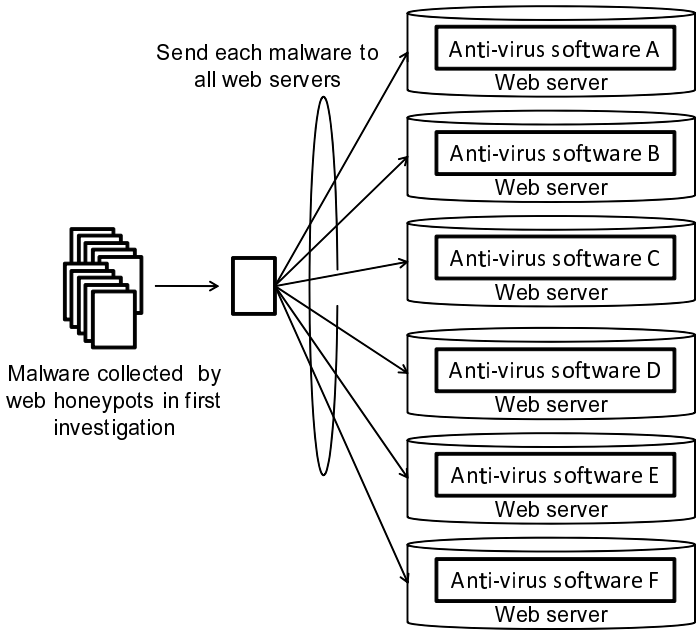

Figure 4.8: Evaluations of anti-virus software

method was installed, by using RFI attacks, as shown in Fig. 4.9. The aim was to determine the probability that websites on which anti-virus software is installed can be protected from malware.

When each anti-virus software is shown by $v(v = A, B, \ldots, F)$, the detection ratio of anti-virus software v, represented as A_v, is as follows.

$$A_v = \frac{M_v}{T} \tag{4.1}$$

Here, M_v is the number of malware files detected by anti-virus software v, and T is the number of malware files used in this evaluation.

In addition, we evaluated the number of anti-virus software programs that detected each malware file. When the number of anti-virus software programs that detect an arbitrary malware file is shown by $i(i = 0, 1, 2, \ldots, 6)$, the number of malware files, which are detected by i anti-virus software programs, can be shown by D_i. Here, the ratio of the number of anti-virus software programs that can detect each malware file, represented as P_i, is as

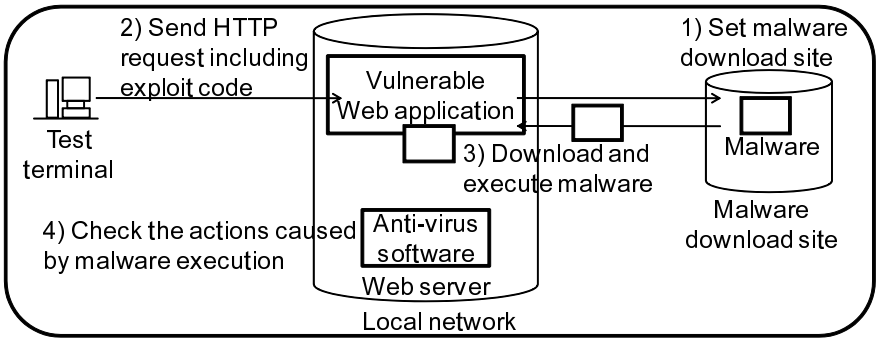

Figure 4.9: Test of anti-virus software

Table 4.9: Detection ratios of anti-virus software

Types of anti-virus software	Detection ratios [%]
Software_A (A_A)	41
Software_B (A_B)	57
Software_C (A_C)	34
Software_D (A_D)	74
Software_E (A_E)	38
Software_F (A_F)	35

follows.

$$P_i = \frac{D_i}{T} \tag{4.2}$$

4.3.2 Results of Evaluation of Anti-Virus Software

The results of A_v are listed in Table 4.9. All detection ratios were insufficient to protect websites from malware infection. In particular, four types of anti-virus software were only able to detect half of the malware files, even though security vendors collected and analyzed many malware files from January 30 to May 20. As shown in A_D, software_D detected approximately 74% of malware files. However, the other anti-virus software programs could not detect many malware files, so A_D is a particular value that depends on this evaluation environment.

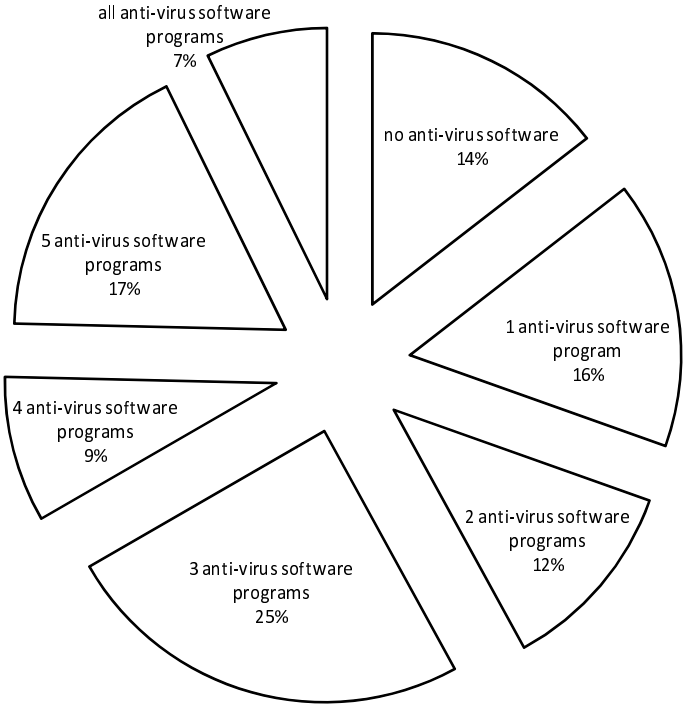

Figure 4.10: Number of anti-virus software programs that can detect each malware file

In addition, the results of P_i are shown in Fig. 4.10. For example, two types of anti-virus software detected approximately 12% of the malware, shown as P_2. As shown in the figure, 14% of the malware was undetectable by all the anti-virus software, whereas only 7% of the malware was detectable by all of the software. In addition, when only one anti-virus software program was used, it detected 16% of the malware. Generally, different anti-virus software cannot be installed at the same time. Therefore, many malware files will not be detected depending on the selected anti-virus software.

Thus, anti-virus software was able to detect only about half of the older malware, which was collected several months before. These results show that anti-virus software cannot detect a large amount of malware used to attack websites. To solve this problem, it is necessary to monitor not only files but also traffic patterns, including malware infection

routes, to detect malware.

4.3.3 Evaluation of Detection using Blacklist

Generally, it is difficult to evaluate detection ratios of blacklisting methods in actual situations because it is impossible to understand all attacks and all malicious information, such as IP addresses of attack sources and malware download sites, in the actual situation. To evaluate blacklisting methods, it is necessary to build up a hypothesis. For example, cross validation checks were deployed [69, 70]. In the methods, collected data in the actual situation was divided into two groups, one was treated as known data and the other was treated as unknown data. Blacklisting methods were evaluated whether the unknown data could be detected by using the known data.

Except for accesses generated by crawlers from search engines, it is highly probable that a large number of accesses received by web honeypots are generated by attackers because there is no reason for a normal user to access web honeypots. If there is a high reappearance of access characteristics on web honeypots, such as IP addresses of attack sources and malware download sites, the access characteristics may be useful for detecting malware infection. Specifically, if the IP addresses of attack sources and malware download sites, which are collected by web honeypots, are registered in our blacklist, we may be able to detect many attacks by using the blacklist. To reveal an actual attack situation, we evaluated the blacklist by using access data that were collected by web honeypots in our evaluation environments.

We assumed that, with the blacklist, we cannot detect IP addresses that are used for the first time, but we can detect IP addresses that are used repeatedly because the IP addresses will have already been monitored by web honeypots. Thus, the detection ratio of attack sources by the blacklist, represented as B, and the detection ratio of malware download sites, represented as W, may be evaluated by the reappearance of the attack sources and malware download sites.

Table 4.10: Reappearance ratios

	Total	IP addresses used by attack only	Detection ratio
Number of attacks	4,621	92	B =97.9%
Number of malware download sites	2,666	45	W =98.3%

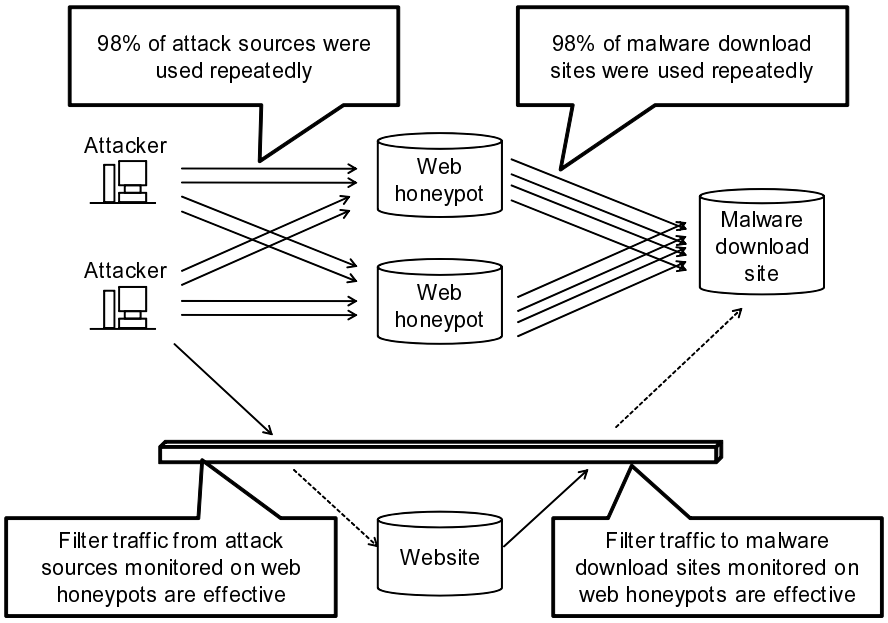

Figure 4.11: Detection method by using traffic patterns

4.3.4 Results of the Evaluation of Blacklist

The results of the analysis are listed in Table 4.10. A total of 4,621 attacks were monitored on the web honeypots, but there were only 92 attacks that had unique attacker source IP addresses. This suggests that about 98% of attacks had the same attacker source IP address. Furthermore, 2,666 attacks used malware download sites, but only 45 attacks used unique malware download sites. This suggests that about 98% of the attacks that used malware download sites used the same malware download site as other attacks. Thus, about 98% of attack information reappeared.

Table 4.11: Kinds of malware files

Type of malware	Number of malware files
Strings generator	110
Information investigator	191
Downloader	2
Bot generator	63

Another useful method for detecting attacks is checking traffic patterns such as source IP addresses of traffic to websites and destination IP addresses of traffic from websites, and checking if these patterns are consistent with the traffic pattern between web honeypots and an external network, as shown in Fig. 4.11. In fact, this detection method is enhanced with our web honeypots, which were described in this chapter. Because our web honeypots can execute downloaders and sequential attacks, they can collect much more information, such as attacker source IP addresses and malware download sites, than conventional web honeypots. As a result, the number of attacks detected with this method increased.

4.4 Discussion

4.4.1 Malware Characteristics and Detection Ratios

We describe here the four types of malware collected by our web honeypots, which are listed in Table 4.11. An overview of malware that sends specific messages to an attacker is shown in Fig. 4.12, and an overview of malware that collects server information and sends it to an attacker is shown in Fig. 4.13. Furthermore, an overview of malware that downloads other malware, such as a downloader, is shown in Fig. 4.14, and an overview of malware that enables an attacker to control an infected server is shown in Fig. 4.15.

The first type of malware sends simple specific strings to an attacker. The attacker knows whether the target website contains vulnerable web applications by confirming the

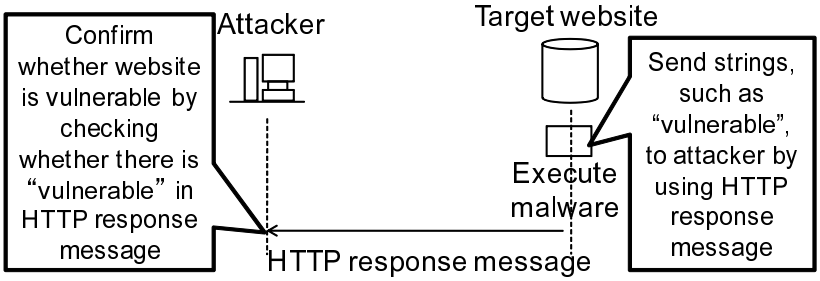

Figure 4.12: First type of malware

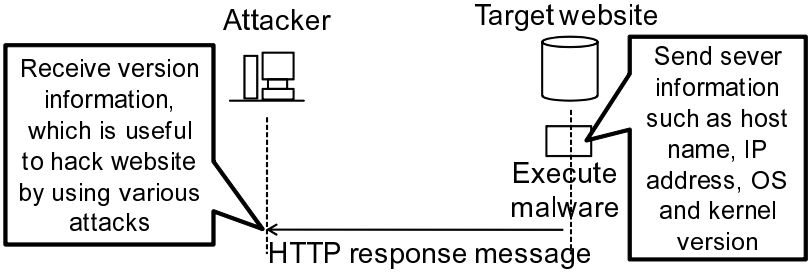

Figure 4.13: Second type of malware

existence of the strings in messages from the target website. By using this malware, attackers can generate a target list, in which vulnerable websites are described, to distribute more malicious malware with a higher success rate. It is difficult for security vendors to judge whether software for generating simple strings is malware.

The second type of malware downloads and sends website information, such as the host name, IP address, OS, and kernel version, to an attacker by referring to the environment of the website. Attackers can obtain information to hack a website by using this malware. Anti-virus software cannot detect this type of malware since security vendors judge the program to be legitimate because this kind of program may be used by website managers in managing their website. For example, website managers generally update the OS and kernel when a new version is released, and this kind of program is useful for that.

The third type of malware only downloads other malware. This malware forces a target website to download pieces of a malicious program. Attackers try to conceal their intrusion from security applications since this malware complicates the route of infection. Anti-virus

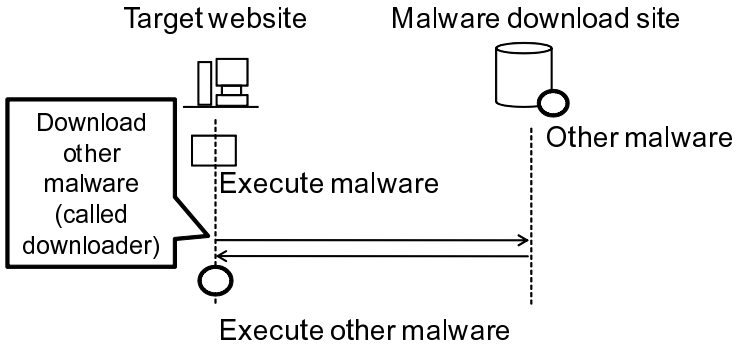

Figure 4.14: Third type of malware

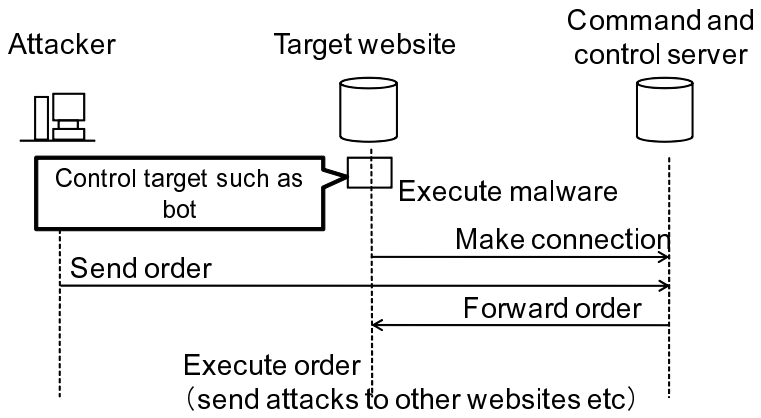

Figure 4.15: Fourth type of malware

software cannot detect this type of malware because security vendors judge software for downloading other files to be legitimate.

The last type of malware is the most malicious. This malware makes a connection or a backdoor for an attacker and allows him/her to control the server. Attackers can illegally control a server as a bot by using this malware. Orders from the attacker are sent via a command-and-control server, normally using IRC and HTTP. It may be difficult for antivirus software to detect this type of malware because there are too many malware varieties to make enough pattern files to detect all malware.

For example, attackers can set any strings on the first type of malware so it is impossible for security vendors to generate pattern files for all malware. In an attack using the first

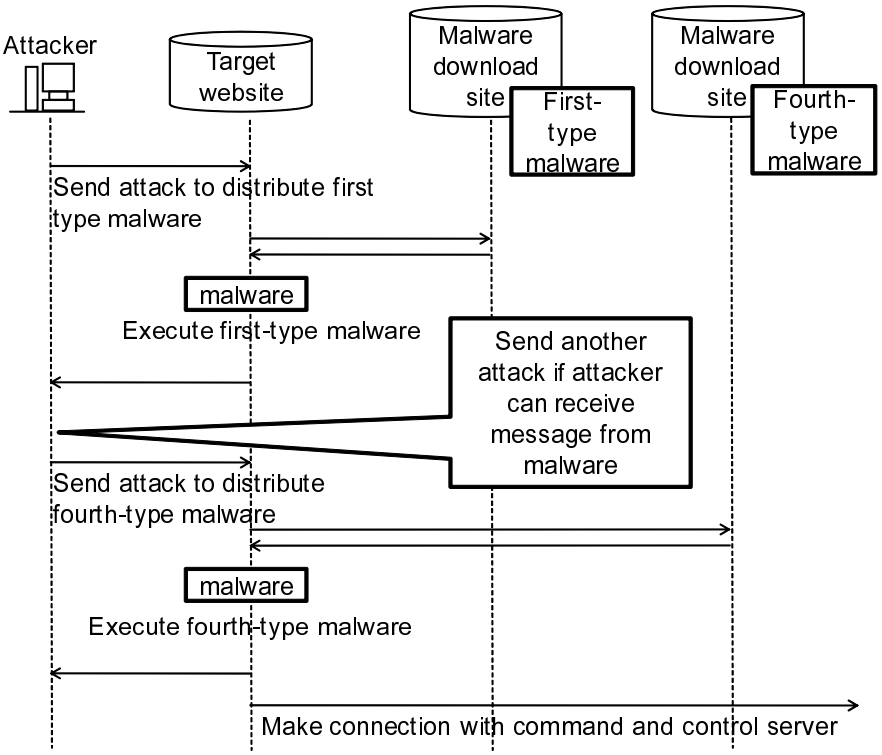

Figure 4.16: Sequential attack

type of malware, attackers can find out whether the target website has vulnerabilities. As a result, if attackers program unknown malware by remodeling conventional malware, attackers can force target websites to successfully download and execute the malware by abusing the vulnerabilities, as shown in Fig. 4.16.

4.4.2 Logs of Web Honeypots and Detection Ratios

When traffic patterns of web honeypots are used for detecting malware infections on websites, websites may receive new attacks before the web honeypots receive them. This risk can be prevented by locating many web honeypots at many IP address blocks, locations, and domains to receive attacks as quickly as possible.

Many attacks are generated by normal user terminals and websites already hacked by attackers. Therefore, normal user traffic may have the same source IP address as attack

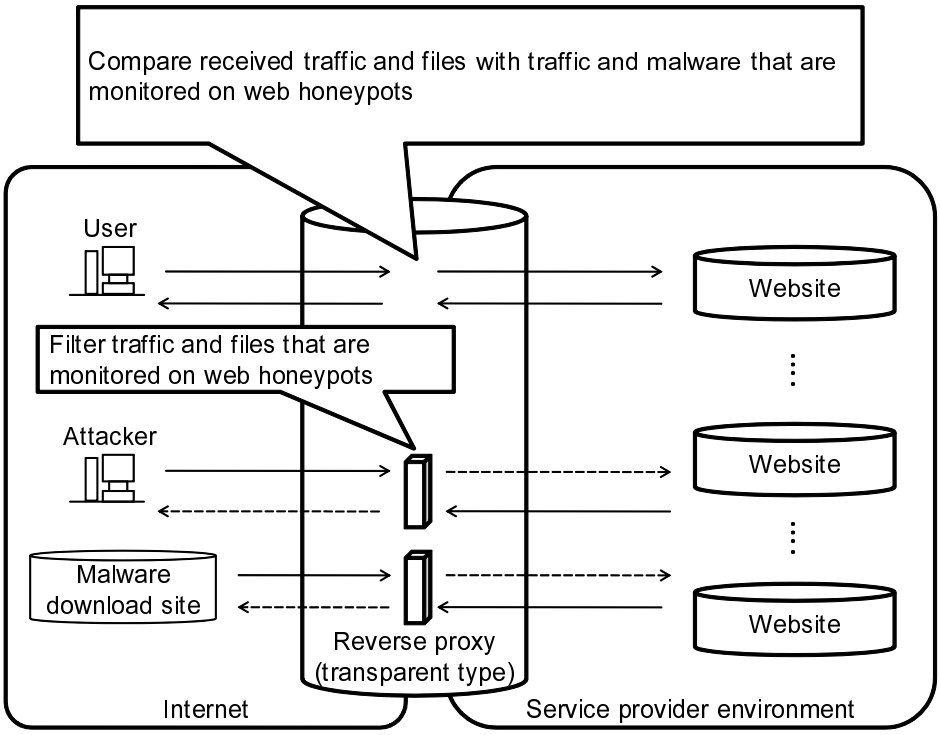

Figure 4.17: Detection method in service provider environment

traffic to websites because the user may use the same terminal as the attackers. In this case, the proposed method of using traffic patterns generates false positives. To solve this problem, a detection method for monitoring traffic patterns and the file characteristics and monitoring whether the characteristics received by websites are the same as those of files received by honeypots is effective. In service provider environments, a transparent reverse proxy is useful for checking traffic between the Internet and websites. As shown in Fig. 4.17, the reverse proxy is set on the boundary between the Internet and service provider environments. In addition, the reverse proxy contains traffic data and malware monitored by web honeypots. When the reverse proxy receives traffic between the Internet and websites, it determines whether these traffic data and files are the same as the traffic data and malware monitored by the web honeypots. Of course, the reverse proxy can contain anti-virus software at the same time. Therefore, the reverse proxy can detect attacks to websites with higher probability.

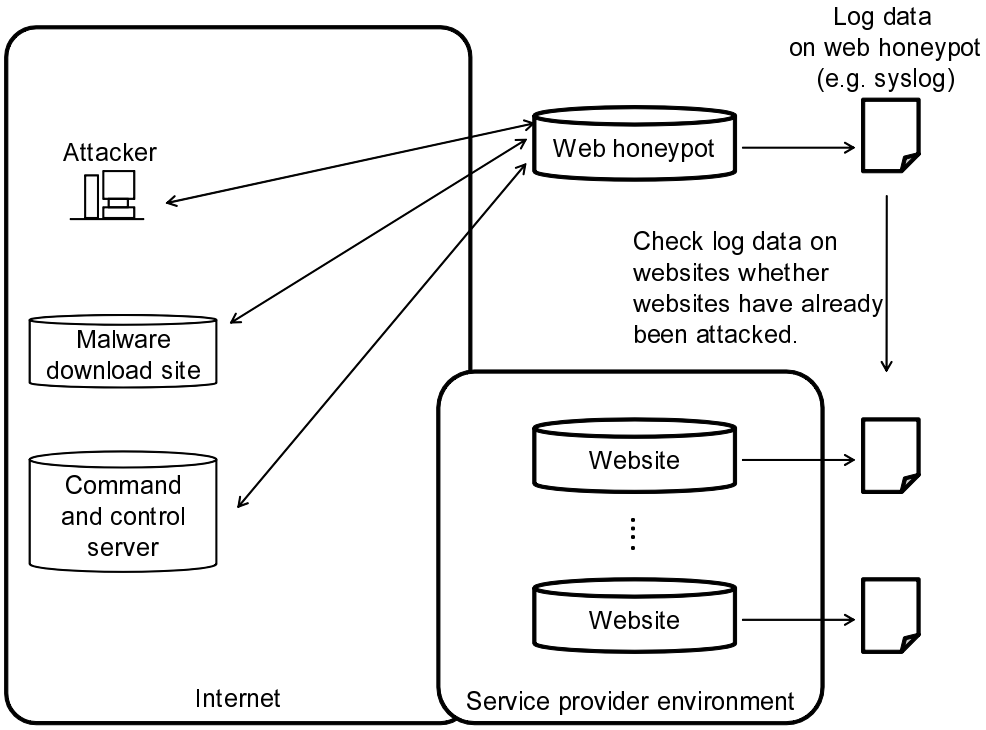

Figure 4.18: Detection method by using log data

This detection method can be useful in detecting websites that have already been hacked by attackers. Many attacks are generated by user terminals and websites infected by malware, so we can specify infected user terminals and websites by monitoring attack sources and malware download sites. Additionally, it is highly probable that websites whose traffic patterns in the log data are the same as those of web honeypots have already been attacked and infected by malware. When service providers use high-interaction web honeypots, they can obtain log data, such as syslog, from actual vulnerable web server systems in high-interaction web honeypots. Therefore, it is easy for service providers to compare log data of websites with log data of attacks received by web honeypots, as shown in Fig. 4.18. This detection method is useful for detecting malware before and after infection.

To detect malware attacks on websites, this network-based detection method, which monitors the patterns of traffic to websites, is more effective than terminal-based detection methods using anti-virus software installed on user terminals and websites. Network

security appliances are useful in detecting traffic patterns. For example, service providers can protect websites from the threat of malware according to the configurations of traffic patterns as signatures of network appliances between web honeypots and the Internet.

4.5 Conclusions

We proposed a provider-provisioned website protection scheme that specifies MDSs using web honeypots and filter accesses from websites to MDSs. The proposed scheme focuses on the characteristics of websites accessing MDSs during an attack. The proposed scheme can then deploy protection technology that filters accesses from websites to the MDSs. In addition, it focuses on the specific characteristic of attacks on websites in which exploit codes are executed as HTTP requests. By this characteristic, the proposed scheme can construct a web honeypot from which it is possible to extract the URLs of MDSs automatically. Incidentally, it is necessary to use other scheme against the attacks, such as SQL injection, that do not use MDSs. To correspond with this type of attack, the controller confirms whether the contents of decoy websites are changed before and after an access by using a tool such as a tripwire [60]. If there is a difference between the before and after content of decoy websites, the controller extracts the source IP address from the access logs and filters the access from the source IP address. By this, the proposed scheme can be used to protect websites from attacks that do not use MDSs. It also can be used to protect websites from not only attacks using websites as attack platforms but also attacks using websites as MDSs. By protecting websites from such attacks, service providers can destroy attack platforms. As a result, a service provider can protect not only websites but also user terminals from malware infections. By using our proposed scheme, a service provider can provide cost-effective and secure networking environments.

In addition, we evaluated and analyzed malware infection prevention methods for websites by using web honeypots connected to the Internet. We investigated the detection ratio of anti-virus software to malware distributed to web honeypots. The results show that it is difficult to detect a large amount of malware by using antivirus software because the

malware may be legitimate tools for users, such as websites managers, or usage aims such as management software versions on websites. Our investigation also revealed that traffic patterns of attackers appear repeatedly on web honeypots if they can automatically and safely collect a large amount of attack information, like our web honeypots. Because of this reappearance, our access filtering method for detecting malware infection by monitoring the same traffic patterns with web honeypots is effective for detecting malware infections on websites.

Service providers can use such a method to protect websites from malware infection with high probability, allowing them to construct secure platforms for websites.

Chapter 5

Enhanced Information Extraction by Intelligent Web Honeypot Based on URL Conversion Scheme

5.1 Intelligent Web Honeypot

To solve this problem, we propose a scheme in which an HTTP forwarding function, which has information of a path structure of a high-interaction web honeypot, is located between the web honeypot and the Internet. The HTTP forwarding function has three key features, a path analyzer, a cache table, and a conversion algorithm. The path analyzer checks whether a path in a destination URL of an HTTP request message from the Internet to the web honeypot corresponds to the path structure of the web honeypot. If the path does not correspond, the HTTP forwarding function converts the incorrect path to the correct path on the web honeypot using the cache table and the conversion algorithm. The following is a more detailed explanation of each key feature.

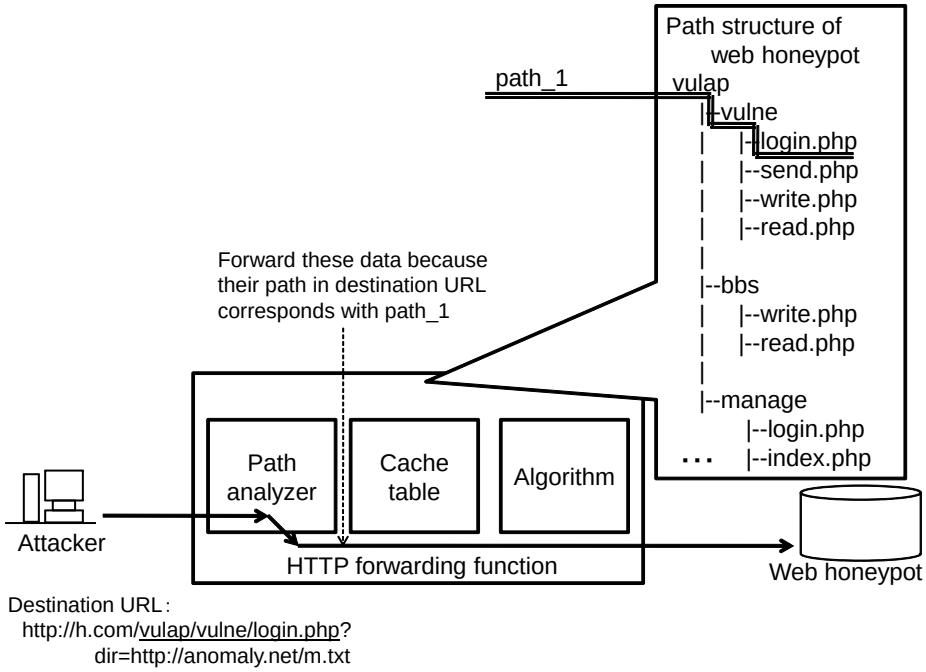

Figure 5.1: Path analyzer

5.1.1 Path Analyzer

A path analyzer monitors a path in a destination URL of an HTTP request message from the Internet to a web honeypot. The function records the path structure of the web honeypot in advance. The analyzer also checks a path in a destination URL of attacks and compares the path with those in the web honeypot using the record date of the path structure. If the path is found in the web honeypot, the HTTP forwarding function forwards the HTTP data to the web honeypot, while maintaining the destination URL. For example, the path in the destination URL corresponds to path_1, as shown in Fig. 5.1. In this case, the HTTP forwarding function forwards this HTTP data without URL conversion. On the other hand, if there is no path found in the web honeypot, the HTTP forwarding function attempts to convert the path in the destination URL to a correct path using a cache table.

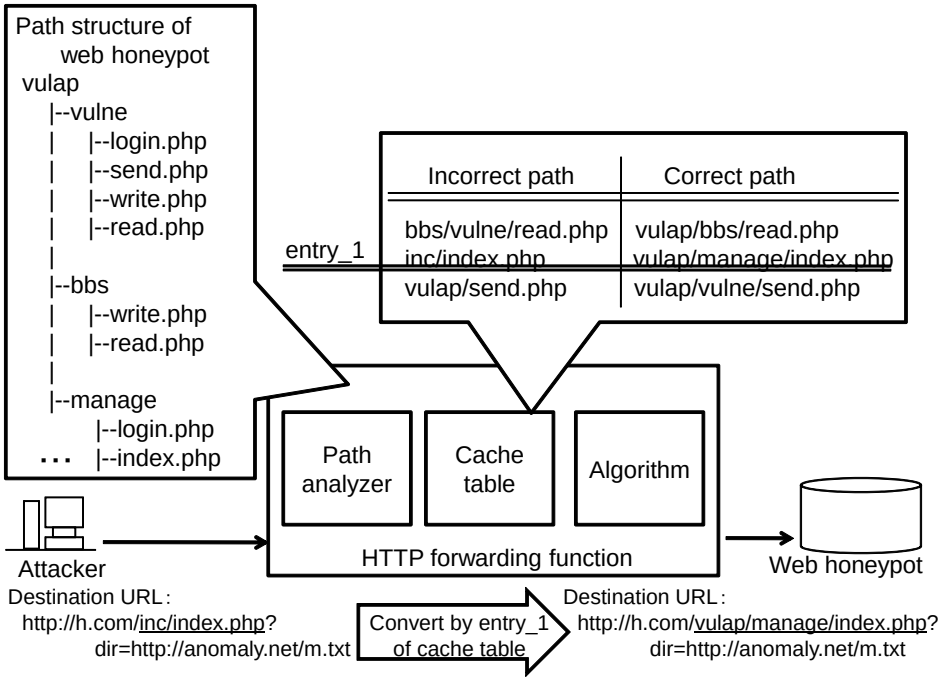

Figure 5.2: Cache table

5.1.2 Cache Table

The HTTP forwarding function has a cache table in which entries are composed of an attack path, which is likely used by attackers, and a correct path in which a vulnerable web application is installed.

The HTTP forwarding function looks up the incorrect path in the cache table. If this function finds an entry in which the incorrect path is described, it converts the incorrect path to a correct one, which is described in the entry, and forwards the HTTP data to the web honeypot. For example, the path in the destination URL corresponds to entry_1, as shown in Fig. 5.2. In this case, the HTTP forwarding function converts the path in the destination URL to the correct path described in entry_1. On the other hand, if the function cannot find an entry, it attempts to convert the path in the destination URL using the conversion algorithm described in the next section.

The operator sets the initial entries of the cache table using the analysis data of attack tools and conventional attacks. In addition, the table is updated with additional entries

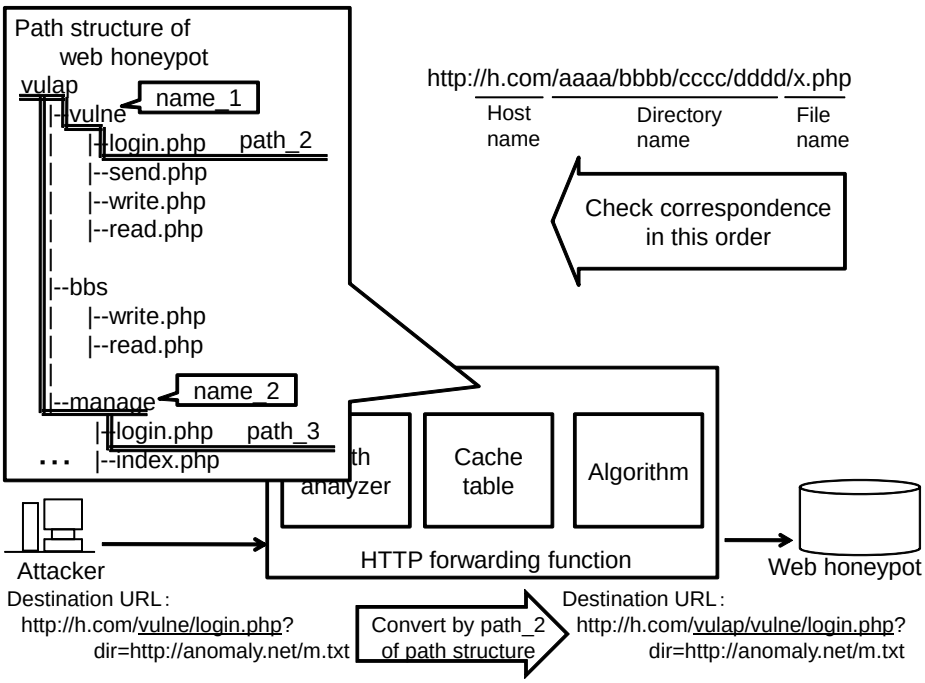

Figure 5.3: Conversion algorithm

generated using the conversion algorithm described below.

5.1.3 Conversion Algorithm

The HTTP forwarding function records an algorithm for incorrect path conversion and attempts to convert an incorrect path not described in the cache table. In our proposed scheme, the HTTP forwarding function converts the incorrect path by determining the correct path on the web honeypot from the similarity between lower paths. This algorithm is explained in detail below.

First, the HTTP forwarding function searches the same file name, which is described in the incorrect path, from names of files on the web honeypot, which are described in the path structure of the web honeypot. If the file name can be found, the HTTP forwarding function stops the conversion of the destination URL and forwards the HTTP data to the web honeypot. On the other hand, if the file name can be found and there is only one path that has the same file name, the function converts the incorrect path to the correct path.

At the same time, some paths may be found because files with the same name may exist on a web honeypot, shown as path_2 and path_3 in Fig. 5.3. In this case, the HTTP forwarding function compares the directory names that house these files, shown as name_1 and name_2 in Fig. 5.3, with those that house the file of the incorrect path, path_2 in Fig. 5.3, and selects the path that has the same directory name with the incorrect path. If we cannot select one path from all the paths, which house the same file name with an incorrect path, the function stops the conversion of the destination URL.

Thus, in our proposed scheme, the HTTP forwarding function converts an incorrect path of a destination URL to a path selected according to the similarity of the file and directory names of a lower path. With this scheme, it is possible for attacks with incorrect paths, which are likely to target web applications on a web honeypot, to succeed. Consequently, a vulnerable web application receives the attack and the result is transmitted to the attacker by an HTTP response message, leading the attacker to believe the site has been infected.

5.2 Evaluation of Proposed Honeypot and Conventional Web Honeypot

One of the main reasons to deploy high-interaction web honeypots is to allow attackers to gain full access to systems and to allow them to launch further network attacks by executing malware. By sending HTTP response messages or other types of traffic, which are generated by executed malware, to attackers, the web honeypot can trick attackers into recognizing the web honeypot as a vulnerable website. In addition, the web honeypot can collect malware downloaded with a downloader, which is distributed using an exploit code by attackers. In these cases, the attack success ratio is important because no malware is downloaded and executed if attacks fail. To confirm the effect of our proposed schemes, including the cache table and conversion algorithm, it is necessary to evaluate the attack success ratio improved by these schemes. It is also better to evaluate the ratio by using actual attack data from the Internet.

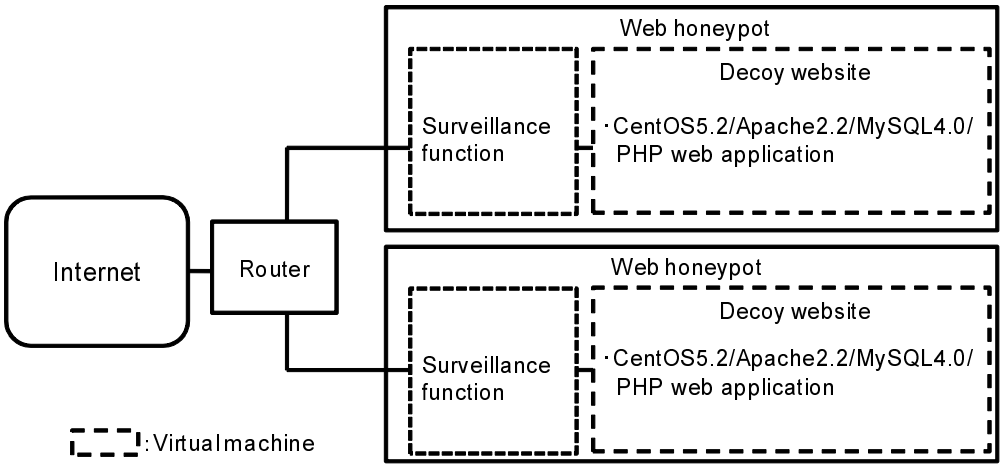

Figure 5.4: Experimental environment

To evaluate the attack success ratio, we collected attack data from the Internet using high-interaction web honeypots. We also analyzed these data by static analysis and evaluated the effectiveness of our proposed scheme.

5.2.1 Experimental Environment

The experimental environment for the evaluation is shown in Fig. 5.4. The decoy websites, deployed as vulnerable websites in high-interaction web honeypots, carried linux OS and open source web applications, which were programmed using PHP. The decoy websites and surveillance functions were constructed on virtual machines, which were constructed using software such as VMware [55]. We compared the proposed high-interaction web honeypots with conventional high-interaction web honeypots, such as the high-interaction honeypot analysis tool. We counted the number of naturally successful attacks on a decoy website and successful attacks on a decoy website with the proposed scheme because the amount of attack information collected by conventional high-interaction web honeypots depends on the number of successful attacks on a decoy website on such honeypots.

Table 5.1: Experimental results

	Total attacks	Naturally successful attacks	Successful attacks with proposed scheme
Number of attacks	1035	33 (3%)	526 (50%)
Different types of malware	63	14 (22%)	45 (71%)

5.2.2 Results of Experiment

We collected attacks on web honeypots from the Internet between January 30, 2009 and July 22, 2009 using the environment illustrated in Fig. 5.4. We analyzed RFI attacks on three web applications installed on our high-interaction web honeypots, to evaluate our proposed scheme.

The total number of attacks, naturally successful attacks and those using the proposed scheme, are listed in Table 5.1. To evaluate whether the proposed scheme can collect much more information that can be used to protect websites, the number of different types of malware, whose information, such as information of malware download sites, can be used to protect websites used in each attack is also listed. The type of malware is distinguished by the SHA1 value of a malware file. In this evaluation, the number of attacks and the different types of malware are manually searched. Furthermore, the types of malware, which would be used in failed attacks, are counted by analyzing these failed attacks and manually downloading the malware. As listed in Table 5.1, only 3% of the attacks were naturally successful, and 22% of the malware was collected from these successful attacks. On the other hand, about 50% of the attacks were successful with the proposed scheme, and 71% of the malware was collected. Thus, web honeypots can increase the probability of malware collection from 22% to 71% by path conversion of 47% of attacks with our proposed scheme.

Next, we analyzed the number of detections of each incorrect path. The number of detections of each path, which were converted using the proposed scheme, is shown in Fig. 5.5. Figure 5.6 shows the number of detections of each path that was not be converted using the proposed scheme but was likely set for using a web application program installed

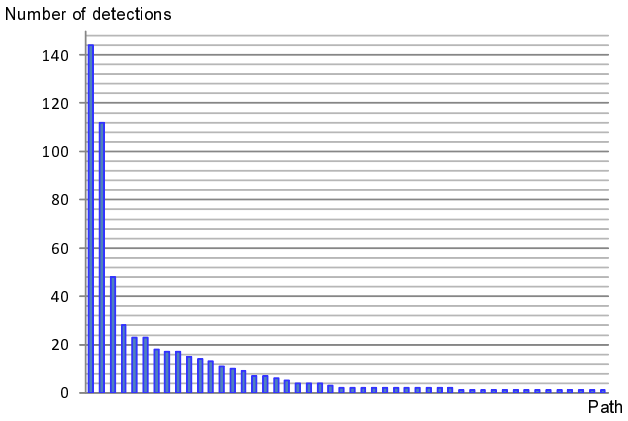

Figure 5.5: Detections of each incorrect path that was converted

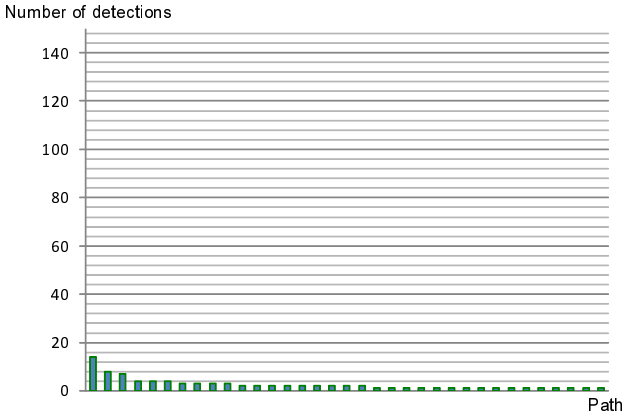

Figure 5.6: Detections of each incorrect path that was not converted but set for using applications on web honeypots

on the web honeypots. Moreover, the number of detections of each path that was likely set for using a web application program not installed on the web honeypots, or whose structure was too easy to convert, is shown in Fig. 5.6. As shown in Fig. 5.5, more than half the paths were repeatedly detected. Therefore, a cache table was referred to frequently. Additionally, the number of detections of each path in Fig. 5.6 was about 10% of the number of detections of each path in Fig. 5.5. This suggests that our proposed scheme can convert incorrect paths that are likely set for using web application programs installed on web honeypots. Furthermore, as shown in Fig. 5.7, some attacks set paths that are related to

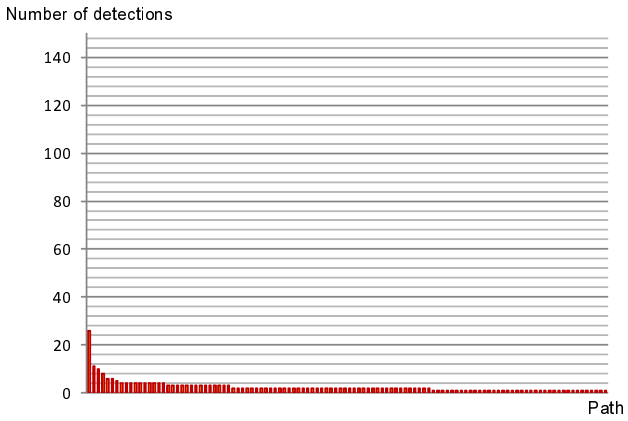

Figure 5.7: Detections of each incorrect path that did not have to be converted

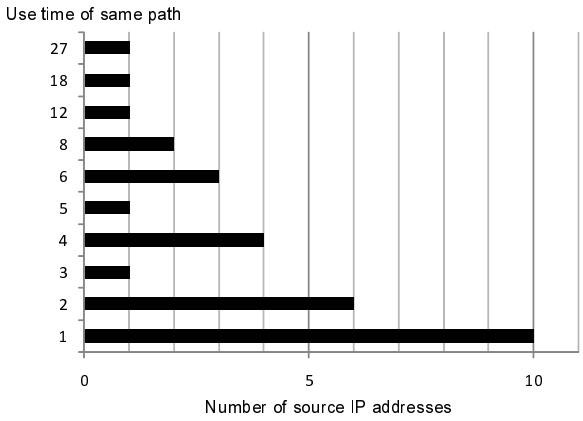

Figure 5.8: Relationship between attackers and use time of same path

web applications on web honeypots.

Finally, to investigate the usage of incorrect paths, which are described in failed attacks, we analyzed the usage trend of the most frequently monitored path shown in Fig. 5.5. The relationship between the number of usage times of the path and the number of IP addresses of attackers who use the path is shown in Fig. 5.8. Furthermore, The relationship between the use times of the path and the detection interval of attacks, in which the path is described, is shown in Fig. 5.9. For example, as shown in Fig. 5.8, an attacker sent an attack, which had the same path, 27 times. This figure shows that about 70% of attackers repeatedly used this frequently used path. In addition, about 60% of attack intervals, in which the

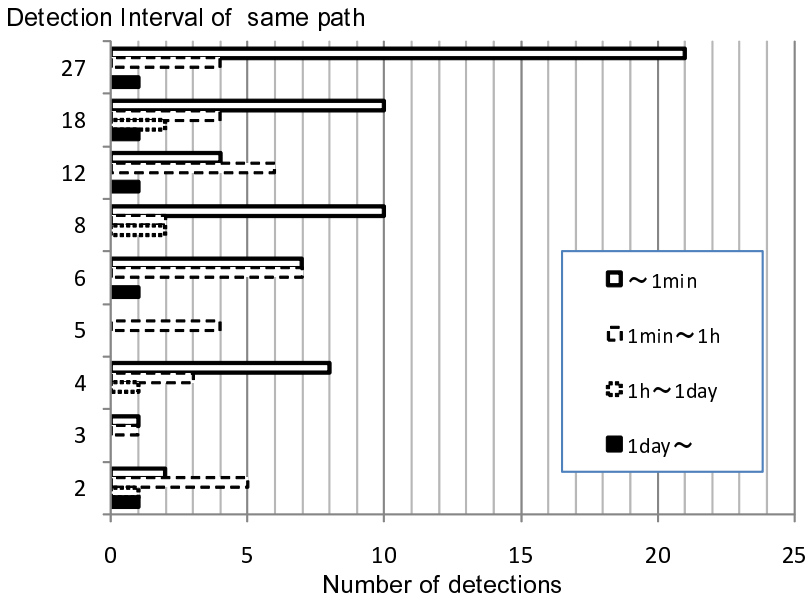

Figure 5.9: Detection interval of same path

path is described, were less than a minute. As shown in Fig. 5.9, for example, an attacker, who sent an attack 27 times using this path, sent the next attack, which had the same path as the previous attack, 21 times within a minute. This suggests that a path list, in which target paths are described, may be shared among attackers. Attacks are also repeatedly sent for a period of time. From these results, attacks are likely to be generated and sent automatically using attack tools.

5.2.3 Discussion

As shown in Table 5.1, the proposed scheme enables about half of possible failed attacks, which have an incorrect path, to succeed. In addition, the scheme can collect about 71% of the malware, which shows that it can efficiently collect malware. On the other hand, attacks whose incorrect path cannot be converted with the proposed scheme include those whose paths are set for using web application programs not installed on web honeypots, as shown in Fig. 5.7. These attacks may be automatically generated with attack tools, as shown in Figs. 5.8 and 5.9.

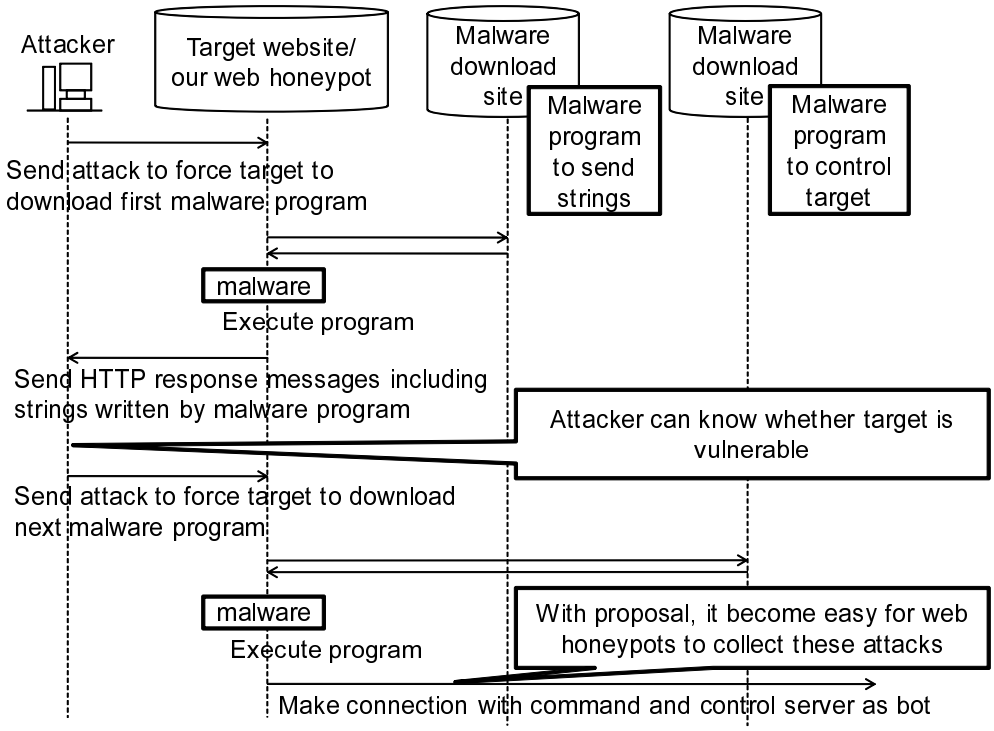

Figure 5.10: Sequential attacks

If all these attacks are successful with the proposed scheme, the behavior of web honeypots becomes forced and artificial. In this case, there is a possibility that attackers will be able to detect our web honeypots as actual web honeypots. In other words, attackers will not consider our web honeypots as target websites. From this point of view, the proposed scheme is suitable because it converts incorrect paths resembling paths on web honeypots. Moreover, as shown in Figs. 5.5 and 5.6, more than 90% of the attacks, of which each path is not correct but likely set for using a web application program installed on a web honeypot, were successful with the proposed scheme. Additionally, a cache table was used frequently. Thus, the proposed scheme can convert incorrect paths effectively and efficiently.

Attacks on high-interaction web honeypots are successful due to the following. First, if attackers used downloaders, which download other malware programs, high-interaction web honeypots can monitor additional malware programs and malware download sites. In addition, the honeypots can monitor sequential attacks, which send additional attacks

Date:
2009/7/29 5:02
Destination:
http://<host_A>/<path_B>?<paramater_C>=http://www.<siteD>.org/files/fey/<file_X>?

Date:
2009/7/29 5:03
Destination:
http://<host_A>/<path_B>?<paramater_C>=http://www.<siteD>.org/files/fey/<file_Y>?

Date:
2009/7/29 5:05
Destination:
http://<host_A>/<path_B>?<paramater_C>=http://www.<siteD>.org/files/fey/<file_Z>?

Date:
2009/7/29 8:25
Destination:
http://<host_A>/<path_B>?<paramater_C>=http://www.<siteD>.org/files/fey/<file_Z>?

Figure 5.11: Actual sequential attacks

according to the success of the first attack, as shown in Fig. 5.10. In these attacks, an attacker first uses a malware program, which writes specific strings in HTTP response messages. Such malware programs are used to confirm whether attacks are successful since the programs can be used as new malware programs by changing the strings, even though the programs are analyzed by security vendors to detect malware infection. If the attacker can confirm the strings in the HTTP response messages, the attacker sends another attack, whose exploit code is the same as the first attack, to download another type of malware program that controls the target website, high-interaction web honeypots in our proposed scheme, as a bot. In an additional investigation, the sequential attacks were monitored, as shown in Fig. 5.11. In these attacks, URLs of malware download sites, whose host names were the same, were inputted as a value of parameter_C of the program located on path_B of the high-interaction web honeypot whose host name was host_A. There are three types of malware programs, file_X, file_Y, and file_Z. File_X writes specific strings into HTTP response messages. File_Y searches and sends information of the web honeypot such as host name, IP address, OS, and kernel versions. File_Z controls the web honeypot. The attacks were successful with our proposed scheme. As a result, some types of malware programs,

such as those in Figs. 5.10 and 5.11, were used. With conventional web honeypots, it is difficult to monitor sequential attacks because attackers do not send additional attacks after a failed first attack. On the other hand, in our proposed scheme, the first attack can succeed, so we can collect sequential attacks.

5.2.4 Conclusion

We proposed a scheme for improving attack information collection on high-interaction web honeypots by converting the destination URLs of low-accuracy attacks.

In our proposed scheme, when a path described in destination URLs does not exist on a web honeypot, the path is converted to a correct path on that honeypot by determining the correct path that may be targeted by the attacker. We also showed the effectiveness of a conversion algorithm, which determines a correct path from the similarity of the file and directory names of a lower path.

From these results, about 50% of incorrect paths can be converted to correct paths with the proposed scheme. Most of the remaining 50%, which cannot be converted with the proposed scheme, are for web applications that were not installed on the web honeypots. If attacks, which have these remaining paths, are forced to be successful, attackers may conclude that the behavior of the website, which is a web honeypot in this case, is not normal. Therefore, the attack success ratio of the proposed scheme is suitable.

By deploying the proposed scheme, we can improve the attack information collected by high-interaction web honeypots. As a result, we can collect many types of attacks and analyze them in detail, improving the quality and quantity of information useful for website protection. This will enable service providers to construct secure website environments.

Chapter 6

High-Accuracy Attack Detection by Blacklist Update Scheme Based on Behavior Analysis of Attackers

6.1 Issues with Network-Based Blacklisting Scheme

The malware located on a malware download site may eventually be removed by either the attacker or the website manager and replaced with a different file [61]. Here, we define the frequency corresponding to continuous placement of the same malware as the active period of that malware download site. If malware has been removed and replaced by a normal file, that malware download site, which is no longer in its active period, must be identified and removed from the blacklist as a target of filtering prohibiting communications in order to minimize the effects of such filtering on other services. However, as shown in Fig. 6.1, there is also the possibility that a malware download site that has exceeded its active period will be loaded with different malware file and maliciously used as a new malware download site. In this case, communications with that site must continue to be prohibited.

Here, we describe four conditions that can occur when managing the URL of a malware download site (Table 6.1). Given that the URL of a non-malware-download site is not

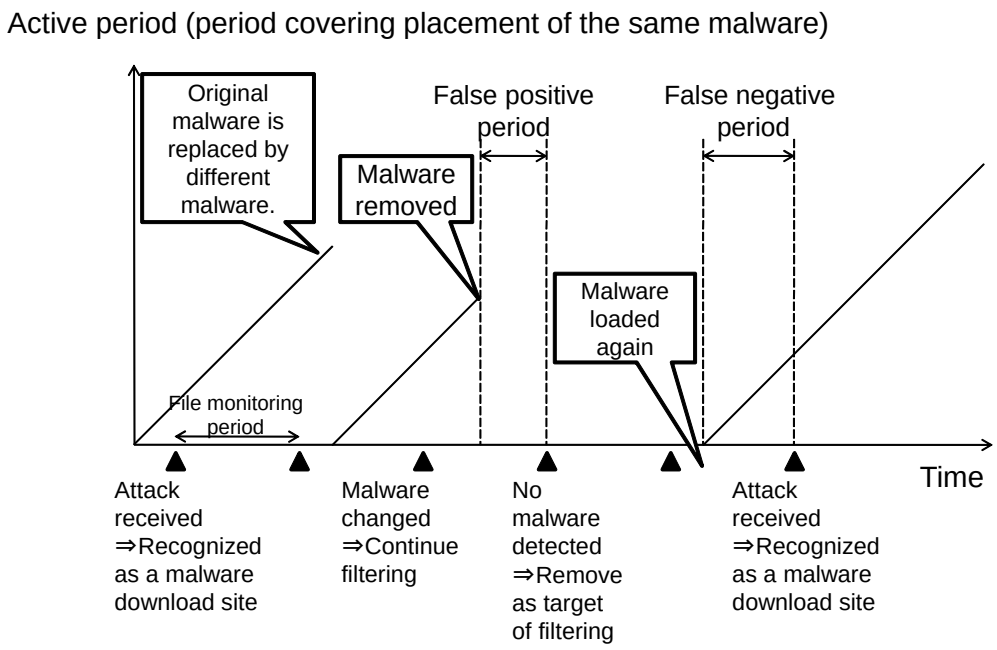

Figure 6.1: Active period of malware download site

Table 6.1: Generated condition

	Not on blacklist	On blacklist
Site not used as a malware download site	True negative	False positive
Site used as a malware download site	False negative	True positive

blacklisted (true negative (TN)), a false negative (FN) occurs if malware is placed on that site. On the other hand, a true positive (TP) can occur if that site is subsequently discovered to be a malware download site and its URL is blacklisted. Under this condition, the malware placed at the URL of that malware download site may be removed and replaced by another file. This action would generate a false positive (FP) in which access to that URL is erroneously detected as part of an attack and filtered out. Here, we point out that a malware download site may also be blacklisted in terms of its domain or IP address in addition to its URL. This means that a false positive in which access to all services provided

by that site is filtered out.

Since a service provider usually discovers that malware has been deleted from a certain malware download site by being told this by users, and unblacklists the site's URL, a false positive exists during the time period from malware deletion to URL removal. One way to solve this problem is to periodically send out a probe to obtain the malware on that malware download site to confirm its presence there. However, the attacker sense the transmission of this probe and move the malware to another site to create a new malware download site and continue the attack [61]. In this case, the service provider will not be able to detect that attack until the new malware download site is discovered by a honeypot, which will have the effect of lowering the true positive rate.

To solve this problem, the service provider needs to send out a probe to the malware download site at time intervals that prevent the attacker from sensing the probe: this will minimize false positives rate and maximize true positives rate. The positive predictive value (PPV) indicates the probability that a blacklisted URL is actually a malware download site: a larger value is preferable.

$$PPV = \frac{TPR}{TPR + FPR} \tag{6.1}$$

where TPR is the true positive rate and FPR is the false positive rate.

6.2 Blacklist Monitoring Scheme

6.2.1 Design Issue of Conventional Monitoring Scheme

Malware on malware download sites may be deleted or replaced by attackers or website managers whose websites are being used illegally by attackers. The period within which a malware program is located on a malware download site is called the life cycle of the malware download site in this work. If malware download sites whose life cycles have finished (i.e., a malware program on a malware download site was deleted) remain in the filtering list, the list becomes huge and increases the filtering load. In addition, if a malware

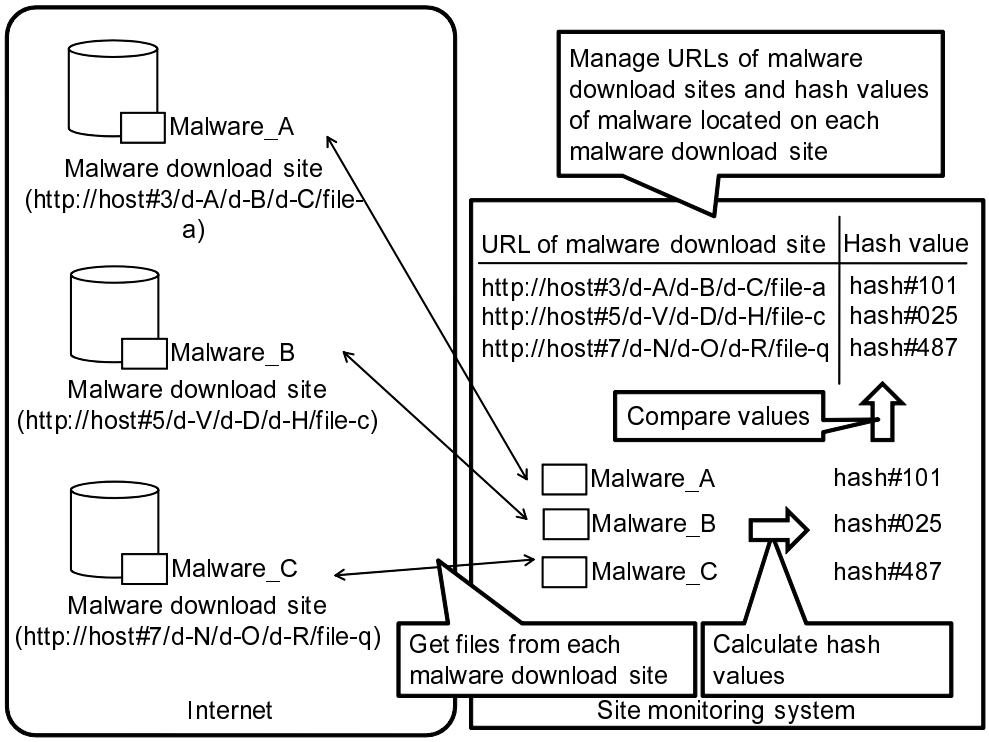

Figure 6.2: Monitoring malware download site using conventional file monitoring schemes

program is deleted and a normal file is located on the same path by website managers or legal users, accesses to the normal file are filtered, which may stop legitimate services. To avoid these situations, it is necessary to identify malware download sites whose life cycles have finished and remove the URLs of the malware download sites from the filtering list. On the other hand, malware programs may be replaced with different malware programs, and the malware download site may be used as a new malware download site. Therefore, the URL of the malware download site should remain in the filtering list.

In spite of the lack of schemes to solve this problem, conventional file monitoring schemes are applied to periodically monitor malware programs on malware download sites. In such schemes, as shown in Fig. 6.2, when a web attack analyzer detects a malware download site, the site monitoring system records the URL of the malware download site and a hash value of the malware program downloaded from the malware download site. In addition, the system downloads a file from the URL and calculates a hash value for the file to compare

the calculated hash value with the recorded hash value. If the values are the same, this indicates that a malware program has been located on the malware download site again since its first download. Therefore, the URL of the malware download site should remain in the filtering list. On the other hand, if the values are different, the service provider should confirm whether the new file is malware. If the file is not malware, the URL of the malware download site should be removed from the filtering list. Of course, if the site monitoring system cannot download any file from the URL, the URL of the malware download site can be removed since the malware may be removed. To deploy these conventional file monitoring schemes, the service provider can confirm the life cycle of each malware download site to some extent and update part of the filtering list according to the results of periodical monitoring.

However, in these schemes, when hash values are different, it is necessary to analyze the new file and confirm whether the file is malware. Generally, anti-virus software is used to confirm this; however, the detection ratio of anti-virus software to malware on websites is limited. To analyze malware programs on not websites but user terminals, there is a conventional scheme that analyzes malware programs dynamically by running them on a closed virtual Internet and monitoring their actions [53]. However, almost all malware programs on websites are scripts and run by using programs of vulnerable web applications. Therefore, it is difficult for the conventional dynamic analysis scheme to run malware programs and monitor the actions dynamically without some additional ideas. Consequently, engineers must manually analyze the new file, which increases labor costs. Thus, with the conventional schemes, it is difficult to efficiently and automatically monitor the life cycles of malware download sites.

6.2.2 Proposed Monitoring Scheme

In our proposed scheme, each HTTP request of an attack from the Internet to web honeypots is recorded, with the URL of the malware download site and the hash value of the malware program downloaded from the malware download site, as an attack management list. As

described in chapter 4, if an HTTP request of an attack is recorded, a web attack analyzer can replay the attack by using a new file to determine whether the file is malware. A site monitoring system and a web attack analyzer can replay attacks again by using the HTTP request of an attack and a copy of the decoy website, so dynamic analysis on a web attack analyzer can be conducted. As a result, the site monitoring system can automatically analyze the new file when hash values are different. The following is a more detailed explanation.

In our proposed scheme, the site monitoring function consists of an attack management list. When a web attack analyzer specifies the URL of a malware download site, the web attack analyzer notifies the URL, the hash value of the malware program located on the malware download site, and the exploit code, which is described in the HTTP request of the attack, to the site monitoring system. The site monitoring system generates a new entry in the attack management list from the notified information and sets the monitoring interval for each entry, as shown in Fig. 6.3.

The site monitoring function downloads files from each malware download site according to the interval described by each malware download site in the attack management list. If the site monitoring system cannot download any files from a malware download site, it deletes the entry of that site from the attack management list and removes the URL of that site from the filtering list, which is located in the filtering function. If the system can download a file from a malware download site, the system calculates the hash value of the new file and compares it with the hash value described in the attack management list entry in which the malware download site is described. If the values are the same, this indicates that a malware program has been located on the malware download site again since its first download, so the site monitoring system enters the next interval. On the other hand, if the values are different, the system sends the new file and an exploit code, described in the attack management list entry in which the malware download site is described, to a web attack analyzer. The web attack analyzer sends the attack to a decoy website by using the exploit code. In this case, the decoy website sends a file download request; therefore, the web attack analyzer sends the new file to the decoy website as a file download response.

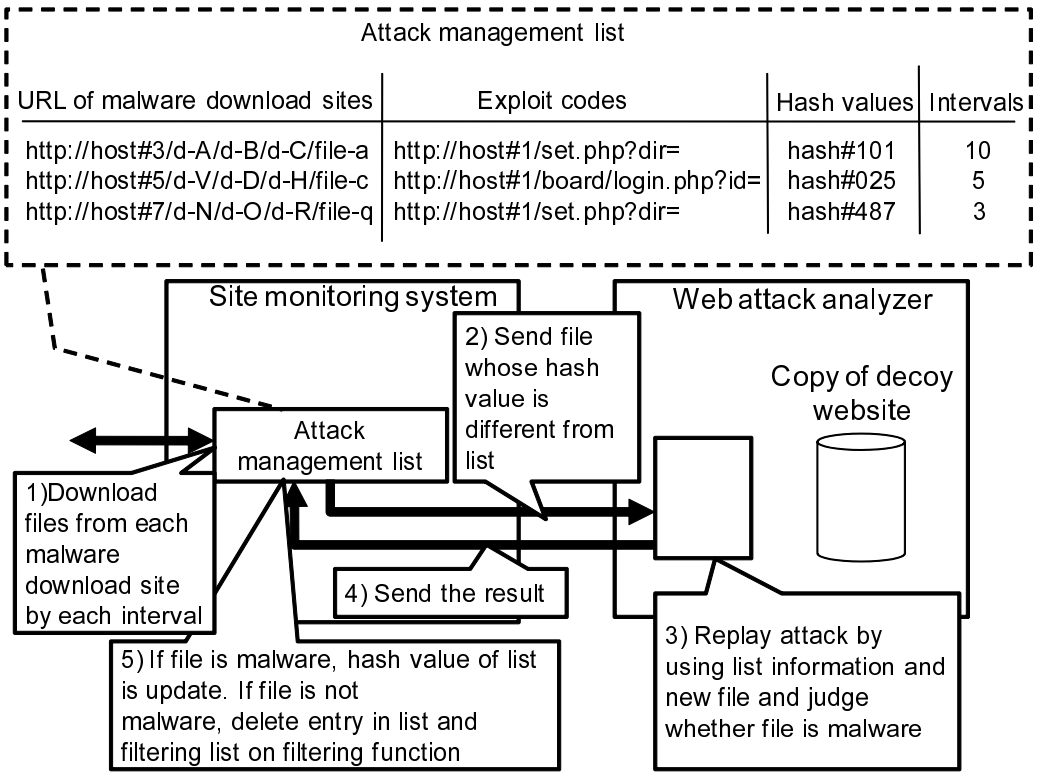

Figure 6.3: Proposed scheme

After that, the web attack analyzer can confirm whether the file is malware by monitoring the actions of the decoy website, and the web attack analyzer sends the analysis result to the site monitoring system. If the file is not malicious, the site monitoring function deletes the attack management list entry in which the malware download site is described and removes the URL of that site from the filtering list in the filter function. On the other hand, if the new file is malware, the system updates the hash value of the attack management list entry in which the malware download site is described.

Thus, in the proposed scheme, exploit codes for malware infection are recorded to replay attacks by using an exploit code and a new file. By using this scheme, a filtering service on a cloud computing environment, on which a large number of websites are run, can be provided automatically according to user demand by coupling a web honeypot, web attack analyzer, and site monitoring system.

Table 6.2: Results of first investigation

Number of sites that replaced their malware program	Life cycle[days]		
	Min.	Ave.	Max.
64	0	17.6	212

6.2.3 Investigation and Analysis

Labor costs can be cut with our proposed scheme, which monitors life cycles of malware download sites and automatically updates a filtering list. To evaluate its effectiveness, we first investigated malware download sites, for which the proposed scheme is needed, by developing a prototype system as part of the proposed scheme and connecting the system to the Internet. We also investigated the life-cycle characteristics of malware download sites.

In these investigations, 11 high-interaction web honeypots, on which a decoy website, controller, and firewall were deployed on a virtual machine of a physical server, were located on 2 different address blocks, and malware from the Internet between January 30, 2009 and March 31, 2010 were collected.

6.2.4 First Investigation

Table 6.2 lists the number of malware download sites on which malware programs were replaced and the life cycles of these sites. We confirmed life cycles every day. In addition, the life cycles of malware download sites on which malware programs were located on March 31, 2010 were cut on that day.

In Table 6.2, a life cycle equaling 0 shows that a high-interaction web honeypot could not download malware from a malware download site. This shows that attack timing of bots, which generate many attacks to websites using automatic attack tools, sometimes does not correspond to the life cycles of malware download sites. As shown in Table 6.2, there were 64 malware download sites on which malware programs were replaced. On some of these sites, malware programs were replaced every day. This could be because attackers tried to

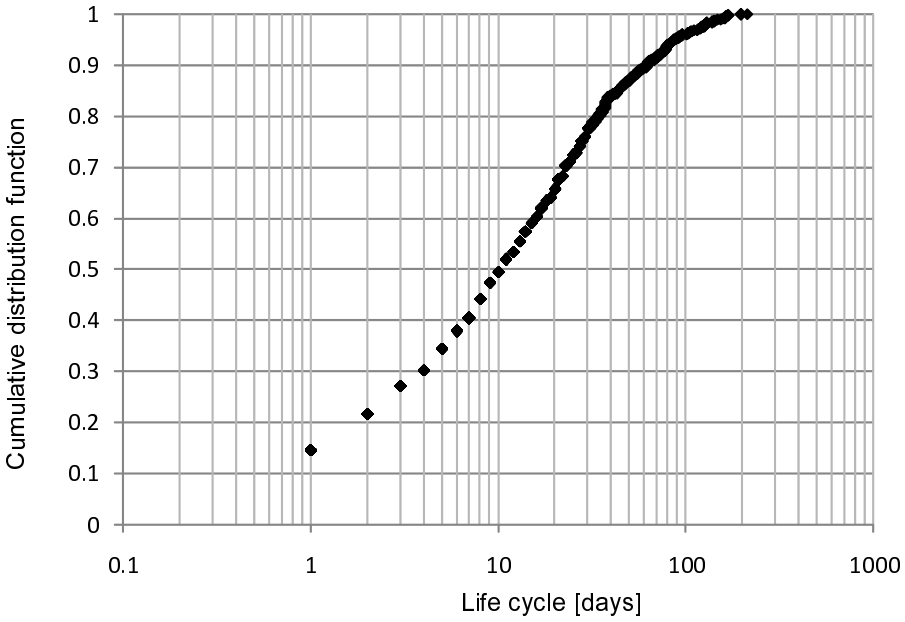

Figure 6.4: Life cycles of malware download sites

conceal their malware programs and their attack platform, such as malware download sites, from anti-virus software and security vendors. In this investigation, all malware programs were replaced not with normal files but with other malware programs. The functions of some of these malware programs were drastically changed. From these results, our proposed scheme, which can dynamically analyze files by replaying attacks, is necessary to deal with the change in malware functions.

6.2.5 Second Investigation

We investigated the life-cycle characteristics of malware download sites. To clarify and design the monitoring interval, we first investigated the cumulative distribution function, as shown in Fig. 6.4. Many malware download sites are generated by illegally using legitimate websites. In these websites, there is a probability that malware will be deleted by anti-virus software installed by website managers. Therefore, the life cycles of malware download sites may depend on the install conditions and detection ratios of anti-virus software. To clarify

Table 6.3: Relationships between life cycles of malware download sites and characteristics of malware programs

Types of malware	Detection result of anti-virus software	Number of sites that replaced their malware program	Life cycle [days]		
			Min.	Ave.	Max.
Bot generator	Not detected	64	0	17.3	212
	Detected	0	9	40.5	197
Other malware	Not detected	0	2	38.3	120
	Detected	0	2	30.7	58

this, we investigated the relationship between the life cycles of malware download sites and the characteristics of malware, especially whether anti-virus software can detect malware. The results are listed in Table 6.3. We compared the life cycles of malware download sites on which malware for generating bots is located with the life cycles of malware download sites, on which malware for other aims, e.g., sending messages to attackers, sending server information such as kernel version to attackers, and downloading other malware programs such as a downloader, is located. In addition, each malware program was checked by anti-virus software and distinguished according to the detection results.

The cumulative distribution function, as shown in Fig. 6.4, is extremely similar to the cumulative distribution function of a normal web page [62]. This result shows that the monitoring interval of malware download sites can be designed using algorithms for determining the crawling interval of web page crawlers. Crawling systems can quickly detect updated web pages by dynamically adjusting the crawling interval by each web page according to the update frequency of each web page. Therefore, it is useful to adjust the monitoring interval of malware download sites according to the change frequency of the hash values of malware located on each malware download site.

As shown in Table 6.3, at malware download sites on which malware programs except bot generators were located, only malware programs not detected by anti-virus software were replaced. In this investigation, the life cycles of malware download sites on which these malware programs are located are similar to those listed in Table 6.2. Therefore, these malware download sites are the most common. On the other hand, at malware

download sites on which bot generators are located, no malware programs were replaced. In addition, the life cycles of malware download sites were longer than those listed in Table 6.2 irrespective of the detection results from anti-virus software.

Thus, life cycles of malware download sites are not affected by the detection ratio of anti-virus software and the type of malware program. Therefore, the results of our investigations may not change according to the spread of anti-virus software to websites.

6.2.6 Discussion

In our investigations, no malware programs on malware download sites were replaced with normal files. However, if a malware program is replaced with a normal file by website managers or users, the filtering list should be quickly updated so as not to filter accesses to the normal file. On the other hand, we confirmed that many malware programs are replaced with other malware programs. This may be caused by attackers who want to avoid detection by anti-virus software. Our proposed scheme is effective because it can dynamically analyze a new file without the need for manual analysis.

The life cycles of malware download sites depend on the actions of attackers. For example, attackers replace malicious websites, which enable malware infection of a user's terminal via the user's web browser, in the short term to avoid malware detection by anti-virus software. Therefore, attackers may change the life cycles of malware download sites despite the life cycles of malware download sites recently being the same as those of normal web pages. To correspond with these trends, investigation of statistical data of life cycles is important. Additionally, interval learning functions, which can adjust the monitoring interval for malware download sites according to the shift in the distribution characteristics of life cycles, will be effective.

6.2.7 Conclusion

We proposed an automatic life-cycle monitoring scheme for malware download sites on which malware is located. In addition, we reported the results of our investigation on the

life cycles of malware download sites on the Internet.

In our proposed scheme, when a high-interaction web honeypot receive attacks from the Internet, an exploit code and URLs of malware download sites and malware programs, which are distributed by the attack, are recorded on a site monitoring system. In addition, the site monitoring system attempts to periodically download a file from each malware download site. If the hash value of the new file is different from that of the malware program recorded on the site monitoring system, the system dynamically analyzes the file and determines whether the file is malware by using the new file and the exploit code recorded on the site monitoring system.

From the results of our investigations on the Internet using our prototype system, we identified malware download sites on which malware are frequently replaced. From this investigation, we clarified the cost efficiencyof our proposed scheme. In addition, we confirmed that the life cycles of malware download sites are similar to those of normal web pages. We also found that we can design the monitoring interval of malware download sites by using an interval-determination method for web page crawlers. Furthermore, we showed that the life cycles of malware download sites will not be affected by the spread of anti-virus software.

Thus, our proposed scheme can improve the filtering of malware infection of websites by automatically updating the filtering list. By using this scheme with a website protection scheme, which filters accesses from websites to malware download sites, service providers can provide security services to many websites on cloud computing environments, in which many types of web services are provided. These security services can be provided according to user demand or service specifications economically and efficiently.

6.3 Blacklist Update Scheme Based on Behavior Analysis of Attackers

In this analysis, we consider an attacker preparing a bot (automated software tool, derived from the word robot) to be used as a malware download site, i.e., a stepping-stone site for

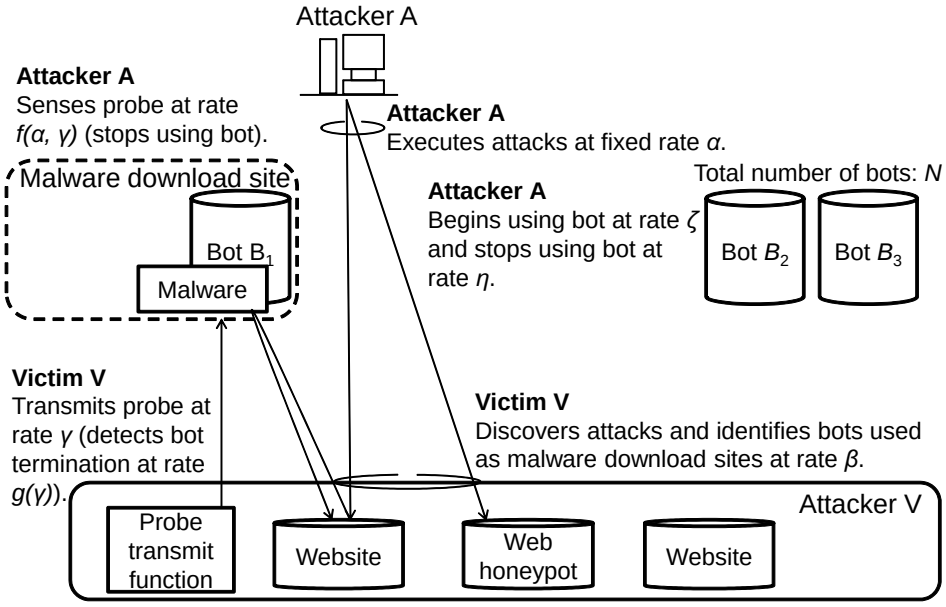

Figure 6.5: Analysis model

mounting attacks. We define states from the viewpoints of whether or not that bot is being maliciously used as a malware download site and whether or not it is currently blacklisted. We also specify transitions between such states through the behavior of the attacker and that of the victim, i.e., the service provider. In addition, we define an evaluation index in terms of the TPR, FPR, true negative rate (TNR) and false negative rate (FNR) to determine an optimal probe transmission period. Though an attacker may also use a bot to transmit attack messages in an actual attack, a "bot" in this analysis model denotes only one used as a malware download site.

6.3.1 Analysis Model

This analysis model consists of a single attacker A, a single victim V, and N bots B_1,B_2, ...,B_N as shown in Fig. 6.5.

We denote the bots used by attacker A as $\boldsymbol{b} = (B_i)$ $(1 \leq i \leq N)$ such that $B_i = 1$ when attacker A is using B_i and $B_i = 0$ otherwise, and we denote the blacklist of victim V as $\boldsymbol{l} = (l_i)(1 \leq i \leq)$ such that $l_i = 1$ when B_i is on the blacklist and $l_i = 0$ otherwise.

Next, we denote the probe transmission rate of victim V as γ. This means that victim V sends a probe to bot B_i at rate γ when $l_i = 1$, that is, when B_i is on the blacklist. Furthermore, when $B_i = 1$, attacker A monitors the packets received by bot B_i currently being used in an attack and, if a probe is sensed, terminates the use of B_i. The packets received by B_i consist of attack-related packets and probe-related packets, and if the number of packets used in an attack is large, the probability of sensing a probe is low. We therefore denote the probe sensing rate as $f(\alpha, \gamma)$, here, α is an attack rate.

Next, we denote the bot-termination detection rate of victim V as $g(\gamma)$. Here, given that victim V has been sending out probes at rate γ to bot B_i on the blacklist ($l_i = 1$), victim V detects at rate $g(\gamma)$ that an attack by B_i has terminated, i.e., that B_i is not being used by the attacker ($B_i = 0$).

The following assumptions are made in this model.

1. Attacker A executes attacks on victim V at fixed attack rate α.

2. Attacker A terminates the use of bot B_i for mounting an attack at fixed rate η [61] and begins an attack using bot B_i, which was not being used in an attack at that time, at fixed rate ζ.

3. Victim V detects an attack by non-blacklisted bot B_i at fixed detection rate β and blacklists bot B_i.

Blacklist-based defensive measures originating in the attack defense function might be detected by the attacker; however, other studies of the defense function have shown that this detection can be prevented with high probability, so this analysis assumes that this kind of detection does not occur. Likewise, the attacker might detect that a service provider is using a honeypot to detect attacks; however, other studies of web honeypots have shown that this detection can be prevented, so this analysis also assumes that honeypot detection by the attacker does not occur.

6.3.2 Derivation of State Transition Rate

A system state can be expressed in terms of B_i and l_i and the state transition rate from state $(\boldsymbol{b},\boldsymbol{l})$ to state $(\boldsymbol{b'},\boldsymbol{l'})$ as $\lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b'},\boldsymbol{l'})}$. Here, it is assumed that the occurrence of state transitions follows a Poisson process.

First, we consider the state transition rate from state $(\boldsymbol{b},\boldsymbol{l})$ to state $(\boldsymbol{b'},\boldsymbol{l})$ $(\boldsymbol{b} \neq \boldsymbol{b'})$, that is, the state transition rate for the case that attacker A adds or deletes a bot for use in an attack. Since this analysis assumes that attacker A adds a bot for use in an attack at rate ζ, we get the following equation.

$$\lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b'},\boldsymbol{l})} = \zeta \quad (|\boldsymbol{b}| = \sum B_i = \sum B'_i - 1) \tag{6.2}$$

Furthermore, since attacker A terminates a bot being used in an attack at rate η, we get

$$\lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b'},\boldsymbol{l})} = \eta \quad (|\boldsymbol{b}| = \sum B_i = \sum B'_i + 1) \tag{6.3}$$

However, if a bot being used in an attack is on the blacklist $(B_i = l_i = 1)$, attacker A terminates the use of that bot not only at rate η but also when sensing the transmission of a probe, which gives us

$$\begin{aligned} \lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b'},\boldsymbol{l})} &= \eta + f(\alpha,\gamma) \\ (|\boldsymbol{b}| &= \sum B_i = \sum B'_i + 1, B_i = l_i = 1) \end{aligned} \tag{6.4}$$

Next, we consider the state transition rate from state $(\boldsymbol{b},\boldsymbol{l})$ to $(\boldsymbol{b},\boldsymbol{l'})$ $(\boldsymbol{l} \neq \boldsymbol{l'})$, that is, the state transition rate for the case that victim V updates the blacklist. Based on the assumptions made in this analysis, a blacklist update occurs for no more than a single bot

B_i per unit time. Initially, we consider that victim V deletes bot B_i from the blacklist. Given that all bots are equivalent and that the rate at which victim V detects attack termination from bot B_i is $g(\gamma)$, we get

$$\lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b},\boldsymbol{l'})} = g(\gamma) \quad (\boldsymbol{l} \neq \boldsymbol{l'}, l_i = 1, l'_i = B_i = 0) \tag{6.5}$$

Conversely, we can consider the case that victim V places bot B_i on the blacklist. Now, given that all bots are equivalent and that the rate at which victim V detects an attack from bot B_i is γ, we get

$$\lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b},\boldsymbol{l'})} = \beta \quad (\boldsymbol{l} \neq \boldsymbol{l'}, l_i = 0, l'_i = B_i = 1) \tag{6.6}$$

If none of the above transitions occurs, the system stays in the same state, so we get

$$\begin{aligned} \lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b},\boldsymbol{l})} &= \mathrm{e}^{-(\sum \lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b'},\boldsymbol{l})} + \sum \lambda_{(\boldsymbol{b},\boldsymbol{l})(\boldsymbol{b},\boldsymbol{l'})})} \\ & \quad (\boldsymbol{b} \neq \boldsymbol{b'}, \boldsymbol{l} \neq \boldsymbol{l'}) \end{aligned} \tag{6.7}$$

Here, we point out that the rate for all other state transitions is 0.

To further examine the state transition rate, we present in Fig. 6.6 part of a state transition diagram focusing on bots B_1 and B_2 as bot B_i used for mounting attacks. First, when $B_1 = 0$, that is, when B_1 is not being used in an attack, the use of B_1 ($B_1 = 1$) begins at rate ζ, as shown by Eq. (6.2). Now, when $B_1 = 1$ and $l_1 = 0$, that is, when B_1 is being used in an attack but is not on the blacklist, victim V detects an attack from B_1 and places B_1 on the blacklist ($l_1 = 1$) at rate β in accordance with Eq. (6.6). In addition, the use of B_1 is terminated ($B_1 = 0$) at rate η, as shown by Eq. (6.3). However, when $B_1 = 1$ and $l_1 = 1$, that is, when B_1 is being used in an attack and is on the blacklist, the state transition that terminates the use of B_1 occurs at rate $\eta + f(\alpha, \gamma)$, as shown by Eq. (6.4).

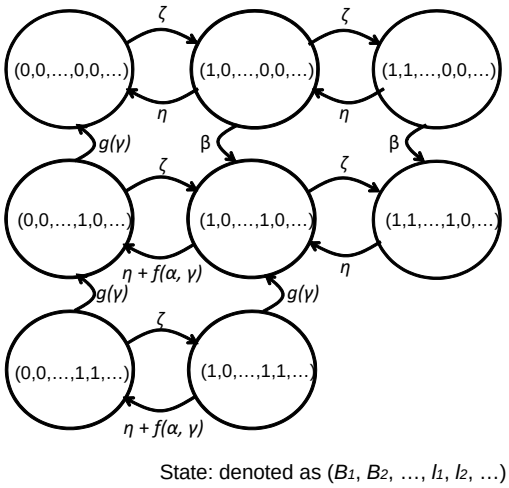

Figure 6.6: Example of state transitions

Table 6.4: States of bots in Fig. 6.6

	Bot B_1 state	Bot B_2 state
(0,0,...,0,0,...)	TN	TN
(1,0,...,0,0,...)	TN	TN
(1,1,...,0,0,...)	FN	FN
(0,0,...,1,0,...)	FP	TN
(1,0,...,1,0,...)	TP	TN
(1,1,...,1,0,...)	TP	FN
(0,0,...,1,1,...)	FP	FP
(1,0,...,1,1,...)	TP	FP

Furthermore, when $B_1 = 0$ and $l_1 = 1$, that is, when $B_1$is not being used in an attack but is on the blacklist, B_1 is removed from the blacklist ($l_1 = 0$) at rate $g(\gamma)$, as shown by Eq. (6.5). In addition the state transition expressed by Eq. (6.7) occurs at each state.

The states of bots B_1 and B_2 for each of the system states shown in Fig. 6.6 are listed in Table 6.4. The states of all bots are TP or TN whenever $\boldsymbol{b} = \boldsymbol{l}$, which is an optimal system state.

6.3.3 Numerical Examples and Derivation of Evaluation Index

Using the Markov chain for this analysis model, we compute TPR, TNR, FPR, and FNR versus γ by calculating the stationary distribution probability of each state. In this analysis, we set parameters other than α and γ on the basis of the results of surveying actual malware attacks. First, given that the number of malware download sites used in an attack at any one time is actually fixed [61], we set this number to 1. In this case, the termination of bot B_i currently being used in an attack and the commencement of an attack using newly selected bot $B_j (i \neq j)$ occur simultaneously, which gives us

$$\zeta = \eta \quad (|\boldsymbol{b}| = \sum B_i = \sum B_j = 1) \tag{6.8}$$

Moreover, since the rate of probe detection by the attacker drops as the number of attacks increases, as described earlier, we get

$$f(\alpha, \gamma) = \frac{\alpha}{\gamma} \tag{6.9}$$

Furthermore, because the rate of terminated bot detection by the victim can be determined from the probe's responses, we get

$$g(\gamma) = \gamma \tag{6.10}$$

Since, in this analysis, no differences arise among bots in terms of transition rate, we can define PPV and the negative predictive value (NPV) by the following equations, where the former is the probability that bot B_i is a malware download site when B_i is blacklisted and the latter is the probability that bot B_i is not blacklisted when B_i is not a malware download site. These can be treated as evaluation values.

$$\begin{aligned} PPV &= \frac{\sum TPR}{\sum TPR + \sum FPR} \\ &\sum TPR = \textstyle\sum_{\{(\boldsymbol{b},\boldsymbol{l}) \in S | B_i = l_i = 1\}} P(\boldsymbol{b}, \boldsymbol{l}) \end{aligned} \tag{6.11}$$

$$\sum FPR = \sum_{\{(\boldsymbol{b},\boldsymbol{l})\in S|B_i=0\cap l_i=1\}} P(\boldsymbol{b},\boldsymbol{l})$$

$$\begin{aligned} NPV &= \frac{\sum TNR}{\sum TNR + \sum FNR} \\ &\sum TNR = \sum_{\{(\boldsymbol{b},\boldsymbol{l})\in S|B_i=l_i=0\}} P(\boldsymbol{b},\boldsymbol{l}) \\ &\sum FNR = \sum_{\{(\boldsymbol{b},\boldsymbol{l})\in S|B_i=1\cap l_i=0\}} P(\boldsymbol{b},\boldsymbol{l}) \end{aligned} \tag{6.12}$$

Here, S denotes the set of all states and $P(\boldsymbol{b},\boldsymbol{l})$ denotes the stationary distribution probability of state $(\boldsymbol{b},\boldsymbol{l})$.

We can also consider as an evaluation index the sum of stationary distribution probabilities O_p such that the states of all bots are either TP or TN.

$$O_p = \sum_{(\boldsymbol{b},\boldsymbol{l})\in S|B_i=l_i} P(\boldsymbol{b},\boldsymbol{l}) \tag{6.13}$$

To calculate γ that maximizes PPV, NPV, and Op, we must monitor the total number of bots N and number of attacks α used by the same malware download site and estimate η and β. We describe a method for estimating η and β in the following subsection.

6.3.4 Parameter Estimation Based on Surveys of Actual Malware Attacks

To estimate η and $beta$ from actual attack conditions, we set up seven our web honeypots [63] on the Internet and collected attacks according to an attack-survey method [64]. We also measured the active periods of malware download sites once a day. This survey ran from October 1, 2011 to February 14, 2012, during which time we collected a total of 128,897 attacks and uncovered 97 malware download sites. The active periods of these malware download sites are summarized in Table 6.5. We also found through this survey that attacks using the same malware download site lasted for times ranging from about one second to more than a month. The unit time for this analysis was therefore taken to be on

Table 6.5: Results of surveying actual active period

	Active period[days]		
	Min.	Ave.	Max.
Active period of a malware download site	1	15.8	124

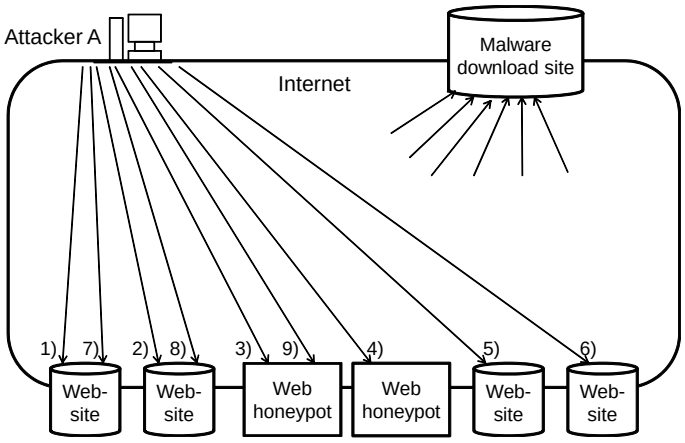

Figure 6.7: Placement of periodic attacks

the order of a minute.

The active periods of malware download sites tend to follow an exponential distribution [64]. We can use the following equation to calculate from the average active period the rate at which malware on a malware download site is moved to another site in units of minutes.

$$\eta = 1 - \mathrm{e}^{-\frac{1}{15.8 \times 24 \times 60}} = 0.44 \times 10^{-4} \tag{6.14}$$

In this survey, we uncovered two main attack patterns: intensive attacks lasting for an interval of several seconds and attacks that reappeared after several days. As an example of the latter, we observed attacks 3) and 9) from the same malware download site at the same honeypot, as shown in Fig. 6.7. This observed result was thought to occur because the

attacker was sending out attacks to attack targets on a regular basis in the order of attacks 1) through 9). In this survey, we found that attacks using the same malware download site were received by the same honeypot at an average interval of 22 days. Assuming that the attack arrival interval follows a Poisson process and denoting the number of honeypots deployed by the service provider as x and the attack detection rate of each honeypot as y, we can calculate β as follows.

$$\begin{aligned} \beta = \sum_{i=1}^{x} y_i &= x(1 - \mathrm{e}^{-\frac{1}{22\times 24\times 60}}) \\ &= 0.32 \times 10^{-4} \times x \end{aligned} \tag{6.15}$$

Here, assuming that the number of honeypots is 100 [65] as targets of observations in a survey of actual attacks on websites, we get $\beta = 0.32 \times 10^{-2}$.

Though explained in more detail in section 6.4.2, we mention here that this method for estimating η and β is just one example and that other estimation methods may be used for this analysis.

6.4 Results and Discussion

6.4.1 Results

Using η and β calculated from the results of our survey on actual malware attacks, we performed an analysis to determine optimal γ when varying the value of α. In that survey, we observed a maximum of 10 attacks per minute, so in this analysis, we calculated an optimal value for γ for values of α from 1 to 10. We also calculated evaluation values when increasing the number of bots to determine how the total number of bots may affect those values.

In this analysis model, the state in which bot B_i is being used as a malware download site and no bots have yet been blacklisted is taken to be the initial state. State transitions are repeated until each state arrives at a stationary distribution so that the stationary

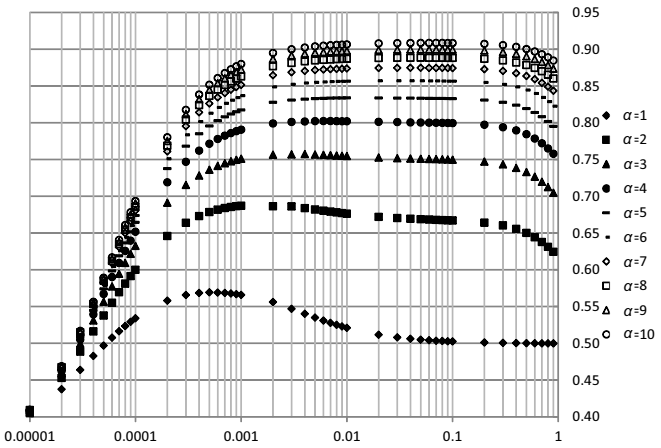

Figure 6.8: PPV versus γ for various α (for $N = 3$)

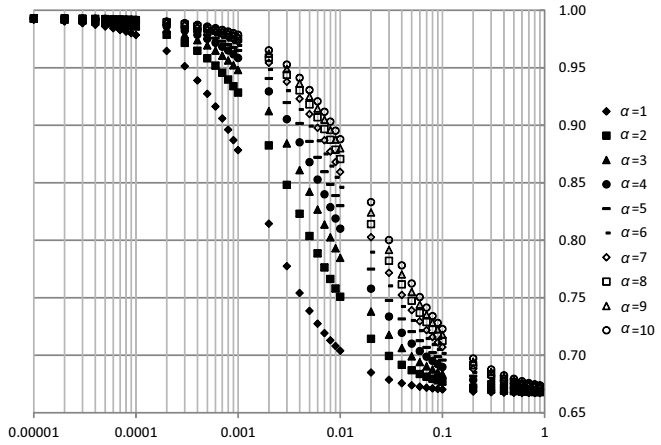

Figure 6.9: NPV versus γ for various α (for $N = 3$)

distribution probability of each state can be determined and PPV, NPV, and Op can be calculated. PPV versus γ for different values of α when $N(numberofbots) = 3$ is shown in Fig. 6.8, where the horizontal axis is γ and the vertical axis is PPV. NPV versus γ for different values of α when $N = 3$ is shown in Fig. 6.9, where the horizontal axis is γ and the vertical axis is NPV. Results for TPR, TNR, FPR, and FNR for $\alpha = 1$ and 10 when $N = 3$ are shown in Figs. 6.10 and 6.11, respectively. The horizontal axes are γ and the vertical axes are the values of TPR, TNR, FPR, and FNR. Finally, O_p versus γ for different values of α when $N = 3$ is shown in Fig. 6.12, where the horizontal axis is γ and

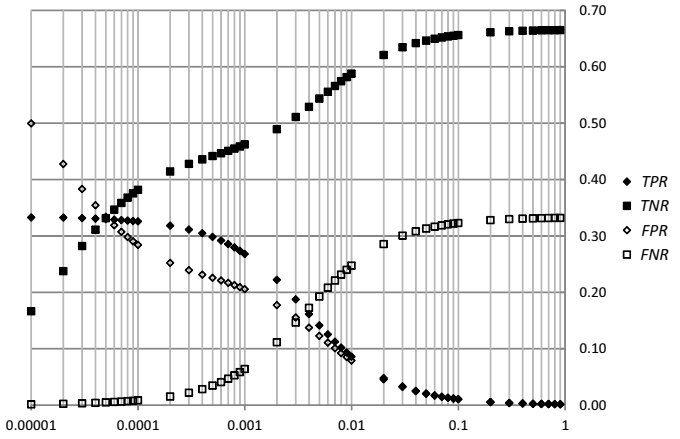

Figure 6.10: TPR, TNR, FPR and FNR versus γ for $\alpha = 1$ (for $N = 3$)

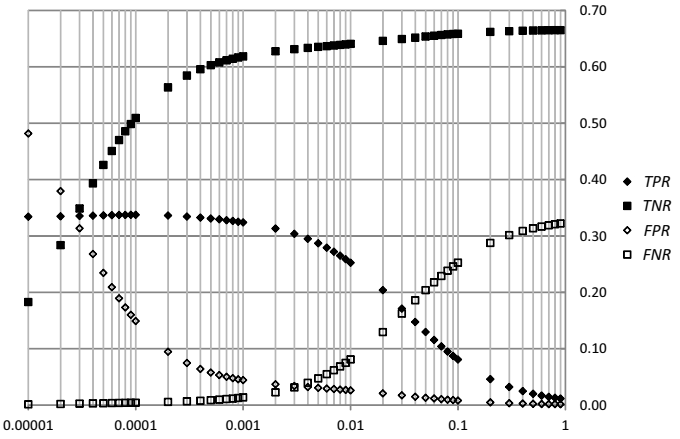

Figure 6.11: TPR, TNR, FPR and FNR versus γ for $\alpha = 10$ (for $N = 3$)

the vertical axis is O_p.

As shown in Fig. 6.8, an optimal γ exists for PPV for each value of α, or to put it another way, optimal γ changes according to α. For $\alpha = 1$, for example, this optimal value is $\gamma = 0.05 \times 10^{-2}$. Thus, in minute units, the optimal probe transmission period is 0.05×10^{-2}, which means that an optimal probe transmission period of 1.38 days. Similarly, for $\alpha = 2$, the optimal value is $\gamma = 0.01 \times 10^{-1}$, and for $\alpha = 3$, it is $\gamma = 0.05 \times 10^{-1}$. In short, an optimal γ exists for PPV. By contrast, NPV decreases monotonically versus γ in Fig. 6.9. We think that the reason for this is that, while NPV differs somewhat according

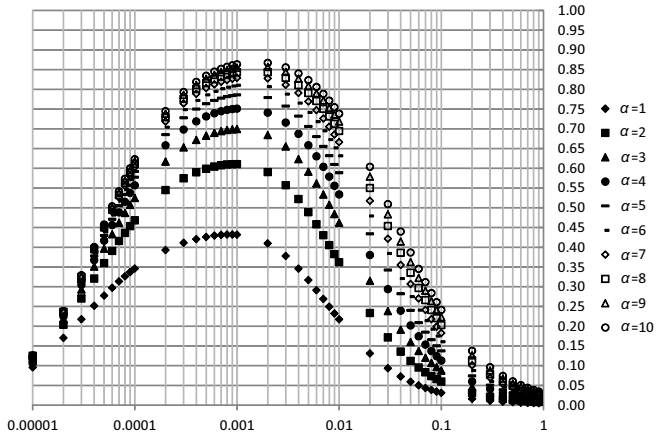

Figure 6.12: O_p versus γ for various α (for $N = 3$)

to α, the rate of increase of FNR versus γ converges sooner than the rate of increase of TNR, as shown in Figs. 6.10 and .6.11. This means that, for NPV, it is desirable for γ to be minimum. In this case, however, false positives occur frequently, as also shown by Figs. 6.10 and 6.11. Moreover, when focusing on only true positives and true negatives, the stationary distribution probability O_p of the optimal state in Fig. 6.12 features an optimal γ, though different from PPV. As described in section 6.1, a service provider must decide a probe-transmission period for a malware download site that maximizes TP while lying in a range that prevents the attacker from sensing the probe thereby minimizing FP. Based on the results of this analysis, we conclude that false positives can be minimized and true positives maximized by considering the PPV evaluation value and calculating γ in accordance with α.

Next, we calculate PPV and O_p when increasing the number of bots in this analysis model and examine how optimal γ changes with respect to this increase. PPV versus γ for different values of α when $N = 7$ is shown in Fig. 6.13, where the horizontal axis is γ and the vertical axis is PPV. O_p versus γ for different values of α when $N = 7$ is shown in Fig. 6.14, where the horizontal axis is γ and the vertical axis is O_p.

Optimal γ in Fig. 6.13 is different from that of Fig. 6.8, which tells us that the number of bots holding the same malware must be observed by web honeypots and reflected in the

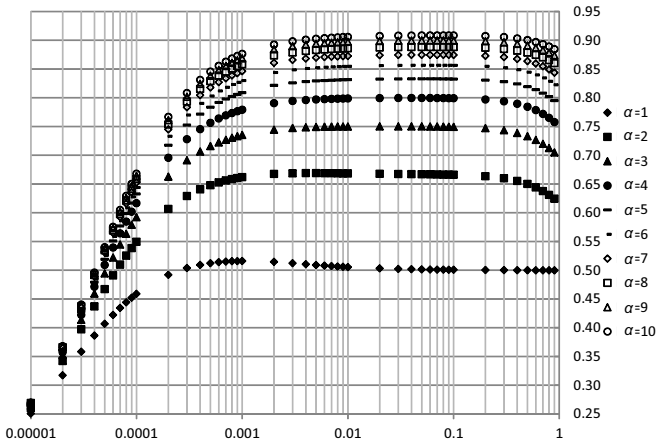

Figure 6.13: PPV versus γ for various α (for $N = 7$)

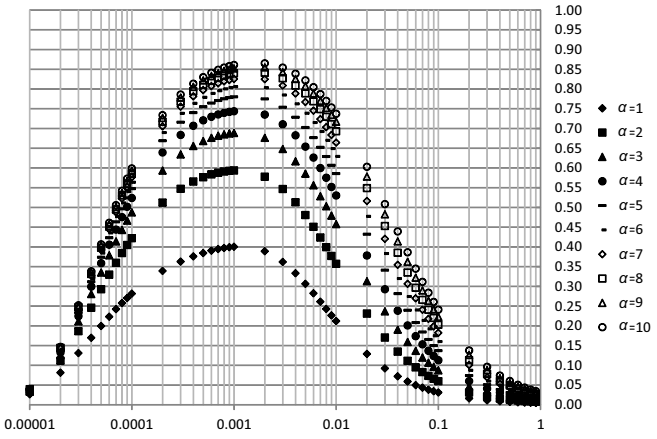

Figure 6.14: O_p versus γ for various α (for $N = 7$)

calculation of γ. In contrast to these results, optimal γ in Fig. 6.14 is the same as that in Fig. 6.12 and O_p also shows little change.

The results of our analysis show that a probe transmission period that minimizes false positives and maximizes true positives can be computed by observing the number of bots holding the same malware and the attack transmission rate used by the same malware download site and applying those observations to this analysis model. Thus, by using the analysis, we can clarify the change in attack-detection accuracy, which is difficult to assess solely by observing attacks, and to determine an optimal frequency for monitoring the

activities of a malware download site.

6.4.2 Discussion

The method for estimating η and β presented in this paper is just one example - other estimation methods may be used to perform the analysis described here. Specifically, we can consider a method for estimating β that treats the active period as the time from when the malware download site is first discovered to the time when it stops being used for mounting attacks. Likewise, we can consider calculating β from the ratio of the number of attacks received by some honeypots to the number of attacks observed at all honeypots using, for example, a cross validation technique. An optimal probe period can still be determined by applying η and *beta* estimated by these methods to the analysis method introduced in this paper.

Some caution is required in collecting attacks by web honeypots and sending out probes using a malware-download-site monitoring function [54, 66]. For example, honeypots must be arranged so as to avoid discovery by attackers. Furthermore, an attacker may be monitoring source IP addresses in packets received at the attacker's malware download site, so IP addresses should be changed after sending out probes. Constructing such a system while taking these precautions should enable a service provider to achieve a blacklist-update system that maximizes true positives while minimizing false positives. With this approach, the service provider will be able to protect user websites from attacks with a high level of accuracy.

6.5 Conclusion

In this paper, we modeled attacker and service-provider behaviors with respect to malware download sites containing website-infecting malware and modeled state transitions with respect to blacklisting malware download sites. We also analyzed these model with synthetically generated attack patterns and measured attack patterns in an operation network. This analysis enabled us to clarify the latent change in attack-detection accuracy, which is

difficult to assess solely by observing attacks, and to determine an optimal frequency for monitoring the activities of a malware download site.

In this analysis, we specified system states in terms of whether bots used by an attacker for malicious purposes are currently being used as malware download sites and whether those bots are currently blacklisted. We also specified a state transition rate taking into account attacker behavior in sensing a probe transmitted by a service provider to monitor a malware download site. Using a Markov chain for these state transitions and calculating the stationary distribution probability of each state, we clarified the process by which false positives and false negatives occur for malware detection and analyzed the relationship between the probe transmission period and attack detection rate. Through this analysis, we showed that it is possible to determine a probe transmission period that, while being dependent on the number of bots, raises the attack detection rate while minimizing false positives.

In this way, by analyzing malware-download-site activities while considering attacker behavior when communications from websites to the malware download site are being filtered out, one can create a blacklist-update system that raises the attack detection rate while minimizing false positives. In other words, the proposed method enables a service provider to construct a safe and secure website environment that can protect websites from malware infection with high accuracy.

Topics for future research include realtime estimation of parameters so that this method can be applied to actual systems and an approximation algorithm for handling an increase in computational load resulting from an increased number of bots.

Chapter 7

Conclusions

This thesis proposed and evaluates a malware download site blacklisting scheme to detect diverse attacks in chapter 4. Additionally, in chapter 5, to maximize information that can be extracted from attacks, this thesis proposed and evaluated web honeypots, which can collect attack information such as malware download sites, attack sources, and malware. Furthermore, in chapter 6, to reduce the number of false negatives and false positives, this thesis proposed and evaluated schemes for optimizing blacklist update frequency based on the behavior analysis of malware attackers.

At first, we proposed a provider-provisioned website protection scheme that specifies MDSs using web honeypots and filter accesses from websites to MDSs. The proposed scheme focuses on the characteristics of websites accessing MDSs during an attack. The proposed scheme can then deploy protection technology that filters accesses from websites to the MDSs. In addition, it focuses on the specific characteristic of attacks on websites in which exploit codes are executed as HTTP requests. By this characteristic, the proposed scheme can construct a web honeypot from which it is possible to extract the URLs of MDSs automatically. Incidentally, it is necessary to use other scheme against the attacks, such as SQL injection, that do not use MDSs. To correspond with this type of attack, the controller confirms whether the contents of decoy websites are changed before and after an access by using a tool such as a tripwire[60]. If there is a difference between the before

and after content of decoy websites, the controller extracts the source IP address from the access logs and filters the access from the source IP address. By this, the proposed scheme can be used to protect websites from attacks that do not use MDSs. It also can be used to protect websites from not only attacks using websites as attack platforms but also attacks using websites as MDSs. By protecting websites from such attacks, service providers can destroy attack platforms. As a result, a service provider can protect not only websites but also user terminals from malware infections. By using our proposed scheme, a service provider can provide cost-effective and secure networking environments. In addition, we evaluated and analyzed malware infection prevention methods for websites by using web honeypots connected to the Internet. We investigated the detection ratio of anti-virus software to malware distributed to web honeypots. The results show that it is difficult to detect a large amount of malware by using antivirus software because the malware may be legitimate tools for users, such as websites managers, or usage aims such as management software versions on websites. Our investigation also revealed that traffic patterns of attackers appear repeatedly on web honeypots if they can automatically and safely collect a large amount of attack information, like our web honeypots. Because of this reappearance, our access filtering method for detecting malware infection by monitoring the same traffic patterns with web honeypots is effective for detecting malware infections on websites.

Additionally, in our intelligent web honeypot, when a path described in destination URLs does not exist on a web honeypot, the path is converted to a correct path on that honeypot by determining the correct path that may be targeted by the attacker. We also showed the effectiveness of a conversion algorithm, which determines a correct path from the similarity of the file and directory names of a lower path. From these results, about 50% of incorrect paths can be converted to correct paths with the proposed scheme. Most of the remaining 50%, which cannot be converted with the proposed scheme, are for web applications that were not installed on the web honeypots. If attacks, which have these remaining paths, are forced to be successful, attackers may conclude that the behavior of the website, which is a web honeypot in this case, is not normal. Therefore, the attack

success ratio of the proposed scheme is suitable. By deploying the proposed scheme, we can improve the attack information collected by high-interaction web honeypots. As a result, we can collect many types of attacks and analyze them in detail, improving the quality and quantity of information useful for website protection. This will enable service providers to construct secure website environments.

Furthermore, in this thesis, we proposed an automatic life-cycle monitoring scheme for malware download sites on which malware is located. Moreover, we modeled attacker and service-provider behaviors with respect to malware download sites containing website-infecting malware and modeled state transitions with respect to blacklisting malware download sites. At first, we proposed an automatic life-cycle monitoring scheme for malware download sites. In addition, we reported the results of our investigation on the life cycles of malware download sites on the Internet. In our proposed scheme, when a high-interaction web honeypot receive attacks from the Internet, an exploit code and URLs of malware download sites and malware programs, which are distributed by the attack, are recorded on a site monitoring system. In addition, the site monitoring system attempts to periodically download a file from each malware download site. If the hash value of the new file is different from that of the malware program recorded on the site monitoring system, the system dynamically analyzes the file and determines whether the file is malware by using the new file and the exploit code recorded on the site monitoring system. From the results of our investigations on the Internet using our prototype system, we identified malware download sites on which malware are frequently replaced. From this investigation, we clarified the cost efficiencyof our proposed scheme. In addition, we confirmed that the life cycles of malware download sites are similar to those of normal web pages. We also found that we can design the monitoring interval of malware download sites by using an interval-determination method for web page crawlers. Furthermore, we showed that the life cycles of malware download sites will not be affected by the spread of anti-virus software. Thus, our proposed scheme can improve the filtering of malware infection of websites by automatically updating the filtering list. By using this scheme with a website protection scheme, which filters accesses from websites to malware download sites, service providers

can provide security services to many websites on cloud computing environments, in which many types of web services are provided. These security services can be provided according to user demand or service specifications economically and efficiently. Next, we modeled attacker and service-provider behaviors with respect to malware download sites containing website-infecting malware and modeled state transitions with respect to blacklisting malware download sites. We also analyzed these model with synthetically generated attack patterns and measured attack patterns in an operation network. This analysis enabled us to clarify the latent change in attack-detection accuracy, which is difficult to assess solely by observing attacks, and to determine an optimal frequency for monitoring the activities of a malware download site. In this analysis, we specified system states in terms of whether bots used by an attacker for malicious purposes are currently being used as malware download sites and whether those bots are currently blacklisted. We also specified a state transition rate taking into account attacker behavior in sensing a probe transmitted by a service provider to monitor a malware download site. Using a Markov chain for these state transitions and calculating the stationary distribution probability of each state, we clarified the process by which false positives and false negatives occur for malware detection and analyzed the relationship between the probe transmission period and attack detection rate. Through this analysis, we showed that it is possible to determine a probe transmission period that, while being dependent on the number of bots, raises the attack detection rate while minimizing false positives. In this way, by analyzing malware-download-site activities while considering attacker behavior when communications from websites to the malware download site are being filtered out, one can create a blacklist-update system that raises the attack detection rate while minimizing false positives. In other words, the proposed method enables a service provider to construct a safe and secure website environment that can protect websites from malware infection with high accuracy.

Service providers can use such a method to protect websites from malware infection with high probability, allowing them to construct secure platforms for websites.

In this thesis, schemes were evaluated by using attacks which were collected from the Internet because many studies evaluate their methods by using attacks collected from the

Internet[69][70]. On the other hands, it is difficult to evaluate effects in actual situation accurately because it is impossible to understand all malicious information, such as exploit codes and malware download sites. Assumption of the information and accurate evaluation in actual situation are important and future topics.

Bibliography

[1] R. Callon and M. Suzuki, "A framework for layer 3 provider provisioned virtual private networks (PPVPNs)," *IETF RFC4110*, July 2005.

[2] T. Yagi, T. Kondoh, T. Kuwahara, J. Murayama, H. Ohsaki and M. Imase, "Architecture Design for SPX: Secure networking Platform for group-oriented eXchange," in *Proceedings of the Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT) 2008*, April 2008.

[3] T. Ogasa, Y. Takahashi, H. Ohsaki, T. Yagi, J. Murayama, and Makoto Imase, "Dynamic Topology Reconfiguration Method for Service Overlay Networks Using Users' Community Information," in *Proceedings of the IEEE/IPSJ International Symposium on Applications and the Internet (SAINT) 2009*, July 2009.

[4] Y. Takahashi, K. Sugiyama, H. Ohsaki, M. Imase, T. Yagi, and Junichi Murayama, "Group-Oriented Communication: Concept and Network Architecture," in *Proceedings of the First International Workshop on Security of Computer Communications and Networks (SoCCaN 2008)*, pp. 649–655, August 2008.

[5] Y. Takahashi, K. Sugiyama, H. Ohsaki, M. Imase, T. Yagi, and Junichi Murayama, "On Network Architecture for Realizing Group-Oriented Communication," in *Proceedings of the Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT) 2008*, April 2008.

[6] K. Sugiyama, H. Ohsaki, M. Imase, T. Yagi, and Junichi Murayama, "NMF: Network Mining Framework using Topological Structure of Complex Networks," in *Proceedings of the IEEE Congress on Services (SERVICES) 2008 Part II*, pp. 210-211, September 2008.

[7] Y. Takahashi, K. Sugiyama, H. Ohsaki, T. Yagi, Junichi Murayama, and M. Imase, "On Network Architecture for Realizing Group-Oriented Communication," in *Proceedings of the Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT) 2008*, April 2008.

[8] M. Jakobsson, and S. Myers, "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft," *Wiley*, 2007.

[9] M. Aburrous, M.A.Hossain, Keshav Dahal, and Fadi Thabtah, "Experimental Case Studies for Investigating E-Banking Phishing Techniques and Attack Strategies," *Springer Science, Cong Comput 2010*, vol. 2, no. 242–253, April 2010.

[10] L. Zhuang, J. Dunagan, D. Simon, H. Wang, I. Osipkov, G. Hulten and J. D. Tygar, "Characterizing Botnets from Email Spam Records," in *Proceedings of the First USENIX Workshop on Large Scale Exploits and Emergent Threats (LEET) 2008*, April 2008.

[11] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Felegyhazi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G. M. Voelker, and Stefan Savage, "Click Trajectories: End-to-End Analysis of the Spam Value Chain," in *Proceedings of the IEEE Symposium on Security and Privacy 2011*, May 2011.

[12] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Felegyhazi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G. M. Voelker, and Stefan Savage, "Spamalytics: An Empirical Analysis of Spam Marketing Conversion," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, October 2008.

[13] J. Bae, S. Ahn, and J. Chung, "Network Access Control and Management Using ARP Spoofing in Various Windows Environment," in *Proceedings of the International Conference on Information Science and Applications (ICISA) 2011*, April 2011.

[14] X. Hou, Z. Jiang, and X. Tian, "The detection and prevention for ARP Spoofing based on Snort," in *Proceedings of the International Conference on Computer Application and System Modeling (ICCASM) 2010*, October 2010.

[15] W. Xing, Y. Zhao, and T. Li, "Research on the Defense Against ARP Spoofing Attacks Based on Winpcap," in *Proceedings of the International Workshop on Education Technology and Computer Science (ETCS) 2010*, March 2010.

[16] C. Hepner, E. Zmijewski, "Defending Against BGP Man-In-The-Middle Attacks," in *Proceedings of Black Hat DC 2009*, February 2009.

[17] O. Maennel, I. Phillips, D. Perouli, R. Bush, R. Austein, and A. Jaboldinov, "Towards a Framework for Evaluating BGP Security," in *Proceedings of the Workshop on Cyber Security Experimentation and Test 2012*, August 2012.

[18] J. Israr, M. Guennoun, H. T. Mouftah, "Analysis of impact of trust on Secure Border Gateway Protocol," in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC) 2011*, June 2011.

[19] Y. Xi, C. Xiaochen, and X Fangqin, "Recovering and Protecting against DNS Cache Poisoning Attacks," in *Proceedings of the International Conference on Information Technology, Computer Engineering and Management Sciences (ICM) 2011*, September 2011.

[20] L. Fan, Y. Wang, X. Cheng, and L. Li, "Prevent DNS Cache Poisoning Using Security Proxy," in *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT) 2011*, October 2011.

[21] Y. W. Ju, K. H. Song, E. J. Lee, and Y. T. Shin, "Cache Poisoning Detection Method for Improving Security of Recursive DNS," in *Proceedings of the International Conference on Advanced Communication Technology 2007*, February 2007.

[22] J. Nazario, "Political DDoS: Estonia and Beyond," in *Proceedings of the USENIX Security Symposium 2008*, July 2008.

[23] J. Cheng, J. Yin, Y. Liu, Z. Cai, and C. Wu, "DDoS Attack Detection Using IP Address Feature Interaction," in *Proceedings of the International Conference on Intelligent Networking and Collaborative Systems 2009 (INCOS '09)*, November 2009.

[24] Z. C. Yang, "DOS Attack Analysis and Study of New Measures to Prevent," in *Proceedings of the International Conference on Intelligence Science and Information Engineering (ISIE) 2011*, August 2011.

[25] TippingPoint,"The Top Cyber Security Risks", http://www.dunkel.de/pdf/200909_Top CyberSecurityRisks.pdf, September 2009.

[26] L. Zhang, and Q. Zhou, "CCOA: Cloud Computing Open Architecture," in *Proceedings of IEEE International Conference on Web Services (ICWS) 2009*, July 2009.

[27] The Open Web Application Security Project,"2007 OWASP Top 10 Most Critical Web Application Security Vulnerabilities", https://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf.

[28] C. Anley,"Advanced SQL Injection in SQL Server Applications," *An NGSSoftware Insight Security Research (NISR) Publication*, 2002.

[29] HTTP Service Project,"Cross Site Scripting Info", http://httpd.apache.org/info/css-security/.

[30] Z. Zhao, G. Ahn, and H. Hu, "Examining Social Dynamics for Contering Botnet Attacks," in *Proceedings of the IEEE Global Communication Conference (GLOBECOM) 2011*, December 2011.

[31] Y. Pyun, Y. Park, X. Wand, D.S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM) 2007*, May 2007.

[32] G. Hermosillo, R. Gomez, L. Seinturier, and L. Duchien, "AProSec: an Aspect for Programming Secure Web Applications," in *Proceedings of the Second International Conference on Abailability, Reliability and Security 2007 (ARES'07)*, April 2007.

[33] The Open Web Application Security Project,"Command Injection", http://www.owasp.org/index.php/Command_Injection.

[34] H. F. G. Robledo, "Type of hosts on a Remote File Inclusion (RFI) Botnet," in *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA) 2008*, September 2008.

[35] K.Sohr, T.Mustafa, and X.Bao, "Enforcing Role-Based Access Control Policies in Web Services with UML and OCL," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC) 2008*, December 2008.

[36] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Investigation and Analysis of Malware on Websites," in *Proceedings of IEEE Symposium on Web Systems Evolution (WSE) 2010*, September 2010.

[37] Kaspersky,"Component List", http://www.kaspersky.com/component_list.

[38] S. Nanda, L. Lam and T. Chiueh, "Web Application Attack Prevention for Tiered Internet Services," in *Proceedings of the International Conference on Information Assurance and Security 2008 (IAS'08)*, September 2008.

[39] C. Kruegel, and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the ACM conference on Computer and communications security*, October 2003.

[40] T. Huang, S. Huang, and T. Lin, "Web Application Security Assessment by Fault Injection and Behavior Monitoring," in *Proceedings of the International World Wide Web Conference*, March 2003.

[41] J. M. E. Tapiador, P. G. Teodoro, and J. E. D. Verdejo, "Detection of Web-based Attacks through Markovian Protocol Parsing," in *Proceedings of the ACM conference on Computer and communications security*, October 2003.

[42] M. Sharifi, M. Zoroufi, and A. Saberi, "How to Counter Control Flow Tampering Attacks," in *Proceedings of the Computer Systems and Applications 2007 (AICCSA '07)*, May 2007.

[43] D. Scott, and R. Sharp, "Specifying and Enforcing Application-Level Web Security Policies," in *Proceedings of the IEEE Transactions on Knowledge and data Engineering*, August 2003.

[44] D.Watson, and J.Riden, "The Honeynet Project: Data Collection Tools, Infrastructure, Archives and Analysis," in *Proceedings of the WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, April 2008.

[45] Chicago Honeynet Project, "The Google Hack Honeypot", http://ghh.sourceforge.net/.

[46] The Honeynet Project, "Web Applicaton Honeypot", http://www.honeynet.org/gsoc/project8.

[47] DShield Web Honeypot Project, "Web Honeypot", http://site.google.com/site/webhoneypotsite/alpha-release.

[48] The German Honeynet Project, "HIHAT", http://www.honeynet.org/project/HIHAT.

[49] M. Muter, F. Freiling, T. Holz, and J. Matthews, "A Generic Toolkit for Converting Web Applications Into High-Interaction Honeypots", http://people.clarkson.edu/ jnm/publications/honeypot-raid2007.pdf.

[50] Y. Mai, R. Upadrashta, and X. Su, "J-Honeypot:A Java-Based Network Deception Tool with Monitoring and Intrusion Detection," in *Proceedings of the International Conference on Information Technology: Coding and Computing*, April 2004.

[51] The Open Web Application Security Project, "OWAPS Best Practices: Use of Web Application Firewalls", http://www.owasp.org/index.php/Category:OWASP_Best_Practices:_Use_of_Web _Application_Firewalls.

[52] E. Eliam, "Reversing: Secrets of Reverse Engineering," *Wiley*, 2005.

[53] SecureWorks, "The Reusable Unknown Malware Analysis Net", http://www.secureworks.com/research/tools/truman.html.

[54] N. Holz, and T. Provos, "Virtual Honeypots: From Botnet Tracking to Intrusion Detection," *Wiley*, 2007.

[55] vmware, "Virtualize Your IT Infrastructure", http://www.vmware.com/virtualization/.

[56] S. Nanda, and T. Chiueh, "Execution Trace-Driven Automated Attack Signature Generation," in *Proceedings of the Computer Security Applications Coference 2008*, December 2008.

[57] Symantec, "Attacks Increasingly Target Trusted Web Sites", http://www.symantec.com/business/resources/articles/article.jsp?aid=20080513 _sym_report_attacks_increasingly.

[58] AV-Comparative, "On-demand Detection of Malicious Software", http://www.av-comparatives.org/images/stories/test/ondret/avc_report21.pdf.

[59] T. Yagi, N. Tanimoto, T. Hariu, and M. Itoh, "Enhanced Attack Collection Scheme on High-Interaction Web Honeypots," in *Proceedings of IEEE Symposium on Computers and Communications (ISCC) 2010*, June 2010.

[60] Sourceforge, "Tripwire", http://www.sourceforge.net/projects/tripwire/.

[61] T. Yagi, N. Tanimoto, T. Hariu, and M. Itoh, "Design of Provider-Provisioned Website Protection Scheme against Malware Distribution," *IEICE Transactions on Communications*, vol. E93-B, no. 5, pp. 1122–1130, May 2010.

[62] B.E. Brewington, and G.Cybenko, "How dynamic is the web?," in *Proceedings of the International Journal of Computer and Telecommunications Networking*, vol. 33, issue 1–6, pp. 257–276, June 2000.

[63] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Intelligent High-Interaction Web Honeypots Based on URL Conversion Scheme," *IEICE Transactions on Communications*, vol. E94-B, no. 5, pp. 1339–1347, May 2011.

[64] T. Yagi, N. Tanimoto, T. Hariu and M. Itoh, "Life-cycle Monitoring Scheme of Malware Download Sites for Websites," in *Proceedings of IEEE International Conference on Service-Oriented Computing and Applications (SOCA) 2010*, December 2010.

[65] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi, "Heat-seeking Honeypots: Design and Experience," in *Proceedings of the World Wide Web (WWW) 2011*, March 2011.

[66] The Honeynet Project, "About The Honeynet Project", http://www.honeynet.org/about.

[67] M. AlSabah, K. Bauer, and I. Goldberg , "Enhancing Tor's Performance using Real-time Traffic Classification," in *Proceedings of 19th ACM Conference on Computer and Communications Security (CCS) 2012*, October 2012.

[68] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, "StegoTorus: A Camouflage Proxy for the Tor Anonymity System," in *Proceedings of 19th ACM Conference on Computer and Communications Security (CCS) 2012*, October 2012.

[69] L. Blige, E. Kruegel, and M. Balduzzi, "EXPOSURE: Finding Malicious Domain Using Passive DNS Analysis," in *Proceedings of 18th Annual Network and Disributed System Security Symposium (NDSS) 2011*, February 2011.

[70] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting Malware Domains at the Upper DNS Hierarchy," in *Proceedings of 20th USENIX Security Symposium 2011*, August 2011.