Detecting Information Leakage via a HTTP Request Based on the Edit Distance*

Kazuki Chiba, Yoshiaki Hori† and Kouichi Sakurai
Institute of Systems, Information Technologies and Nanotechnologies /
Kyushu University Fukuoka, Japan
{chiba, hori}@itslab.inf.kyushu-u.ac.jp, sakurai@csce.kyushu-u.ac.jp

Abstract

Recently, we often face the problem of information leakage. In a lot of routes of leakage, the number of leakage victims via the Internet makes up approximately the half of all leakage victims. The cause of leakage via the Internet is divided into human action and malware such as spyware. For example, it occurs when human writes on the bulletin board and spyware works. Especially a technical countermeasure against spyware is needed. In any event, we cannot trust countermeasures for information leakage via the Internet completely.

When a web browser communicates with a server, it sends a HTTP request. The server replies with the information specified in the HTTP request. Some spyware takes advantage of the HTTP request. Installed spyware collects user's information and embeds it in the HTTP request, then sends it to an attacker's server. Filtering packets by a port number of TCP or UDP is not a good way because HTTP is a main communication protocol. A signature based technique is often used as a countermeasure against these spyware. If data of some software matches with signatures stored in the database, it is regarded as spyware. This technique has an advantage that it can detect most spyware if data of spyware is stored, however, it loses effects if data of spyware is not stored.

Then, we propose a leakage detection system which is independent of a database. This system focuses on the leakage caused by human action and malware. In an existing research, researchers calculate an edit distance between the last HTTP request and the new HTTP request. The edit distance is much smaller than the number of characters because a lot of HTTP requests have common characters. We can detect leakage easily because the information which is sent repeatedly is disregarded and the new information which is sent suddenly is digitized and its value stands out. We propose and evaluate a technique that uses not only the just previous HTTP request but further previous HTTP requests to further ignore unnecessary information. Furthermore, we propose a system which raises an alert when it is in danger of information leakage. When an abnormal value is detected in a continuous numerical value, this system judges that there is some possibility of leakage. Assuming that certain quantity information is leaked, some of the detection rate is higher than 90%.

Keywords: HTTP, information leakage, edit distance, behavior based detection

1 Introduction

Recently, incidents of information leakage have increased. Routes of information leakage are various, for example, human, paper, the Internet, and USB flash memory. Especially, the number of leakage victims via the Internet makes up approximately the half of all leakage victims. However, countermeasures for information leakage via the Internet are not adequate.

Journal of Internet Services and Information Security (JISIS), volume: 2, number: 3/4, pp. 18-28

*This work is partly supported by Grants-in-Aid for Scientific Research (B)(23300027), Japan Society for the Promotion of Science (JSPS). This work was partially supported by Proactive Response Against Cyber-attacks Through International Collaborative Exchange (PRACTICE), Ministry of Internal Affairs and Communications, Japan.

†Corresponding author: Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, 819-0395, Japan, Tel: +81-92-802-3666, Email: hori@inf.kyushu-u.ac.jp, Web: http://itslab.inf.kyushu-u.ac.jp/~hori/index.html

1.1 Cause of Leakage

The cause of leakage via the Internet is divided into a human and malware such as spyware. For example, it occurs when human writes personal information on the bulletin board. It also occurs when installed spyware works. Especially a technical countermeasure against spyware is needed. Spyware is malicious software, which steals personal information. The spyware which acquires the personal information such as an address, a name, a credit card number, and a password is increasing. Although the early spyware mainly collects information such as a URL history, recently, it steals important personal information in PC[1]. Fig 1 shows an example of the infection mechanism through malware.

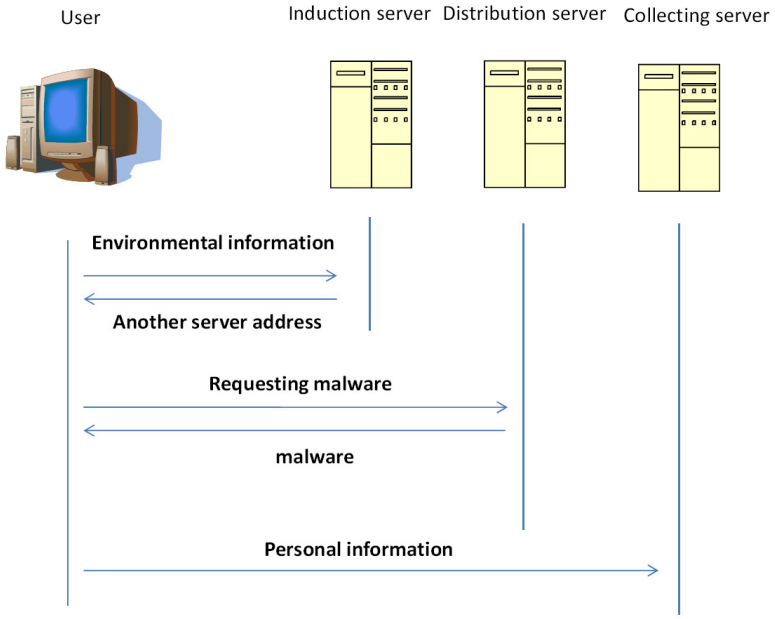

Figure 1: Example of Infection Mechanism

(1) An attacker alters Web servers in order to avoid being discovered. A user accesses the Web server.

(2) The altered Web server sends the address of a malware distribution server to the user.

(3) The user accesses the malware distribution server.

(4) The malware distribution server sends malware to the user.

(5) The infected user sends the personal information such as ID and a password to a collecting server.

Information leakage is also found in Android OS. Some application with an advertisement function collects information such as GPS and text messages, and sends it to an external server[6]. According to Pearce et al.'s study, advertising library is contained in the 49% application in Android market and 56% of them obtain the permission without necessity[9]. There is some possibility that application obtained permission to send personal information to a provider.

Spyware embeds collected information in the HTTP request, and sends it to an attacker's server. HTTP is a protocol used when a server and a client make an exchange of content, and its importance is increasing. According to Erman et al.'s study, HTTP communication makes up 74.0% among all the

```
POST /test HTTP/1.1
User-Agent: Gator/5.0
Kyushu-university
```

Figure 2: HTTP request Made by Gator

amounts of traffic in February, 2010[7]. An attacker takes advantage of the HTTP request to hide attacks. Gator, Cydoor, SaveNow, and eZula are typical spyware[10]. We describe Gator, which takes advantage of HTTP requests. Fig 2 shows the HTTP request made by Gator. Gator embeds keywords which users input into the search form, terminal ID number, URL of the Web page which users browsed, and so on. In this example, the keyword "Kyushu-university" which the user inputted into the search form is sent to an attacker's server.

1.2 Countermeasures for Spyware

The signature based technique and the behavior based technique are used as countermeasures against malware such as spyware. The system with signature based technique compares signatures of software with ones of malware stored in the database. If they are matched, the software is judged as malware. This system can detect most malware, if it is stored. However, this system loses effects if is not stored. The system with behavior based technique judges software from its action. Although it has an advantage that it can detect unknown malware, there is some possibility that it raises an alert to normal software. Therefore, the detection rate of this system has a room of improving.

1.3 Contribution

It is not realistic to filter HTTP by the port number of UDP or TCP, because HTTP communication is very often used. Furthermore, a lot of new methods of attacks are born. Therefore, it is necessary to detect attacks by investigating the action. We propose the leakage detection system based on the behavior based technique. It parses the edit distance of continuous HTTP requests. When an abnormal value is detected, this system judges that there is some possibility of leakage. Assuming that certain quantity information is leaked, some of the detection rate is more than 90%.

2 Related Work

Borders proposed a method to digitize how many new information a HTTP request has[4]. By focusing on quantity of new information, we can detect unusual behavior easily. As a method of digitizing, he calculates an edit distance between the last HTTP request and the new HTTP request. The edit distance between two character strings can be defined as the minimum cost of a sequence of editing operations which transforms one string into the other[8]. We can ignore information which is sent repeatedly by using this method. Figure 3 shows the example of the way to calculate the edit distance. In this example, the edit distance is 3 since the number of editing operation is 3.

He also proposed a method to reduce the calculating time[5]. Although the complexity is $O(n^2)$ when we normally calculate the edit distance, we can reduce the time by splitting strings and calculating the each edit distance. For example, adding 2 elements that comparing with 2 characters and 2 characters is faster than comparing with 4 characters and 4 characters.

com(replacement)co.(insert)co.j(insert)co.jp

Figure 3: the Way to Calculate the Edit Distance

He performed other operations in a Host field, a Referer field, and Cookie field, because strings of these fields often change.

- `Host` – It describes a server address. He only counts the size of the Host field if the request URL did not come from a link in another page.
- `Referer` – It describes URL a of Web page that a user browsed just before D He only counts the Referer field's size if does not contain the URL of a previous request.
- `Cookie` – It describes data sent from a website and stored in a user's web browserD If the Cookie differs from its expected value or he does not have a record from a previous response, then he counts the edit distance between the expected and actual cookie values.

Figure 4 shows relation between the method of calculating the number of characters and one of calculating the edit distance. Horizontal axis shows the order of the HTTP request, and Vertical axis shows

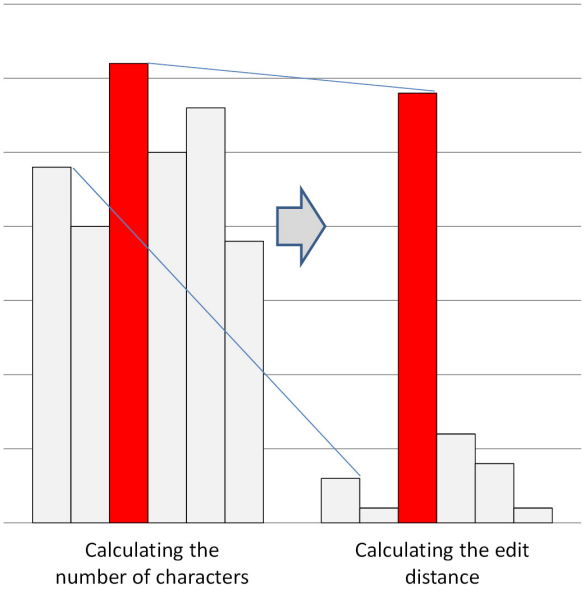

Figure 4: Merit of Calculating the Edit Distance

each value. Assuming that personal information is embedded in the 3rd HTTP request. Quantity of information is much smaller than the number of characters because the HTTP request often includes the common element. Therefore, the edit distance is much smaller than the number of characters. The decreasing width of 3rd is small because personal information is new characters. Therefore, we can find it easily because its value stands out.

The average digitized value of the HTTP request when he browsed a specific blog was 1.9 bytes. This is 0.32% of HTTP request's average number of characters. In addition, Yoshihama proposed the

Figure 5: Example of HTTP requests

technique of detecting information leakage of a Web base[11]. He analyzed HTTP, and estimates the amount of meaningful information in Web traffic. He ignores information which is sent before, the amount of meaningful information was 0.70%

Moreover, Borders mounts the spyware detection system called Web Tap Personal beta 1.2[2]. It seems that the target is abnormal communication by spyware, and it does not detect leakage caused by a user's will[3].

3 Approximate Entropy Calculation Technique

We described the digitizing technique in Related Work. As a method of digitizing, researcher calculates the edit distance between the last HTTP request and the new HTTP request. We review the technique to further ignore unnecessary information when the new HTTP request has same information as the second last one or the third last one.

3.1 Calculation Method and Merit

We use calculation method as a detection of information leakage. We define the value digitized by this method as an approximate entropy in this paper. It is difficult to find information leakage by calculating the number of characters of HTTP requests in cases where the number of leaked characters is not large. If we calculate the approximate entropy, the value is small on the whole because we can ignore a lot of repeated information. Therefore, we can find information leakage easily because the leaked information treated as new information stands out.

3.2 Referring the History Information

We describe the way to refer the history information. HTTP requests are sent from a certain PC which is selected randomly. Fig 5 shows $k-2$thC$k-1$thCkth HTTP requests. We describe the procedure of calculation.

(1) Calculating the edit distance of every field between the new HTTP request(kth) and the last one($k-1$th). The edit distance between "POST" and "GET" is 3, and one between "I'm_busy." and " " is 9. That of other fields is 0.

(2) Calculating the edit distance of every field between the new HTTP request(kth) and the second last one($k-2$th). The edit distance between "POST" and "POST" is 0, and one between "I'm_busy." and "I'm_free." is 4. That of other fields is 0.

(3) We total what has the smallest edit distance in each field. Since the edit distances of "POST" are 3 and 0, the smallest edit distance is 0. Since the edit distances of "I'm_busy." are 9 and 4, the smallest edit distance is 4. The approximate entropy of the new HTTP request(kth) is $4(=0+4)$.

```
GET /test.jpg HTTP/1.1
Host: example.com
```

Figure 6: Example of the HTTP request

```
GET / HTTP/1.1
Host: example.com
Referer: http://example.com/test.jpg
```

Figure 7: Example of the HTTP request

In this example, the number of the history reference is 2. When the number of history references is three or more, we also do the same operation. Moreover, we do another operation at the certain field.

3.3 Operation at the Certain Field

We do another operation at a Host field, a Referer field, a Cookie field, and a Request URI to ignore further innocent information.

- `Host` – Querying the DNSD We use nslookup command. If it replies normally, we can judge that leaked information is not embedded caused by unauthorized rewrite. Therefore, we assume that the approximate entropy is 0.
- `Referer` – Referring history HTTP requests and Comparing the URL made by referring. Figure 6 shows the example of the HTTP request. The Host field represents the domain name, and the request URI represents the location of file on the server. In figure 6, domain name is "example.com", and request URI is "/test.jpg". Therefore, this HTTP request is made by URL "http://example.com/test.jpg". Figure 7 shows the HTTP request made after the HTTP request of Figure 6. We can judge that the URL of the Referer field is not new information because it is made by the previous HTTP request. Therefore, the approximate entropy of the Referer field is 0.
- `Cookie` – Comparing to the content of the Cookie field stored in database. If the content of the Cookie field matches with stored one, the approximate entropy is 0. This idea is based on the character that it hardly changes.
- `Request URI` – Comparing to URL which is extracted from HTML contained in a HTTP response. URL consists of static one directly written to HTML and dynamic one generated by JavaScript. In this paper, we focus on the static one.

In each field, we calculate the edit distance normally if it does not satisfy these conditions.

4 Evaluation of the Method of Calculating the Approximate Entropy

We experiment to investigate relation between the number of history references and the approximate entropy. The subjects of the experiment are 500 HTTP requests sent from one IP address in April, 2011. The approximate entropy is calculated by applying the program to pcap files collected by tcpdump command. In this experiment, the number of the history references is from 1 to 19. Figure 8 shows relation between the number of history references and the approximate entropy. The horizontal axis shows the number of the history references, and the vertical axis shows the average approximate entropy. This result shows that the approximate entropy becomes small by increasing the number of history references. It is considered that we can further ignore unnecessary information because of increasing the number of history references.

Figure 9 shows relation between the number of history references and the execution time. This result shows that the execution time becomes large by increasing the number of history references. These

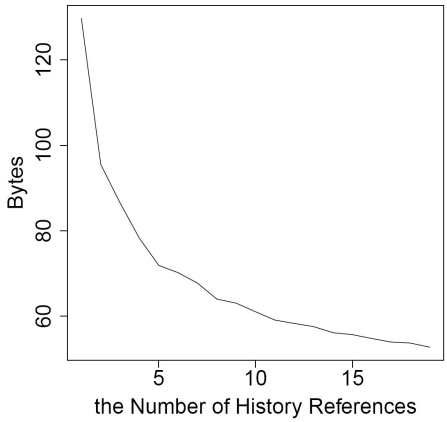

Figure 8: Relation between the Number of History References and the Approximate Entropy

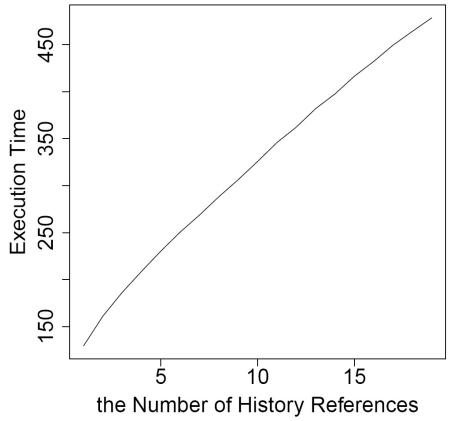

Figure 9: Relation between the Number of History References and Execution time

results show that it is difficult to fill both improvements in the approximate entropy and reduction of the execution time. Therefore, it is necessary to determine the number of history references by judging from the rate of increase or decrease.

Then, we investigate how much information we can ignore by using this technique. At first, we only calculate the number of characters of the HTTP request. The subject of the experiment is 11259 HTTP requests sent from one IP address to unspecified servers in April, 2011. Figure 10 shows that how many HTTP requests exist in each byte. This result shows that request sizes are scattered all over the range from 0 bytes to near 3000 bytes. The average size is 940 bytes.

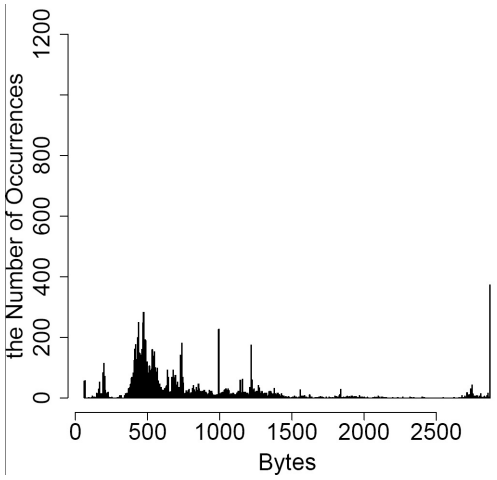

Figure 10: Distribution of the Number of Characters

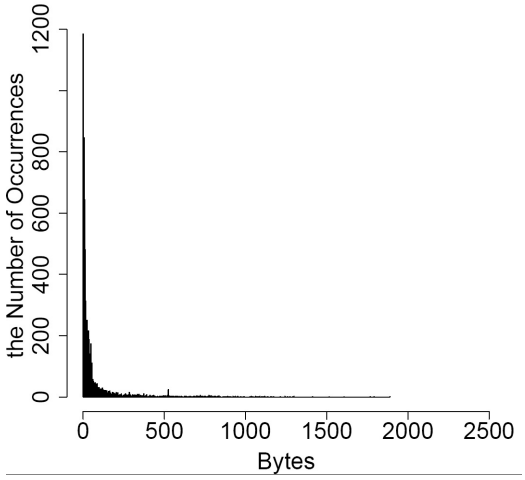

Figure 11: Distribution of the Approximate Entropy

Then, we calculate the approximate entropy. We set the number of history references 10. Figure 11 shows the distribution of the number of each approximate entropy's occurrences. This result shows that the approximate entropy concentrates on the small range from 0 byte to near 400 bytes. As a result of being able to disregard much old information with the proposal technique, it is considered that we calculate the value close to the quantity of the new information of HTTP requests.

5 Detection System Using Mahalanobis Distance

We apply the concept of the approximate entropy to the leakage detection system. This system aims to detect information leakage caused by human action and malware. This system monitors traffic and calculates the approximate entropy. For example, the approximate entropy becomes a small value when the user browses the specific Web site continually. On the other hand, it becomes a large value when a lot of new information is sent. For example, it occurs when a user writes in a bulletin board, or spyware steals information from a computer. This system raises the alert when it is a large value. We use the concept of the Mahalanobis distance to detect an abnormal value. Mahalanobis distance is a measure which means how often the certain value occurs. It becomes possible to reduce the influence of the each person's network circumstance. If it is small, we can judge that the value often occurs. On the other hand, if it is large, we can judge that the value seldom occurs. When leaked information is included in a HTTP request, the approximate entropy becomes large. Therefore, we can detect abnormal behavior easily. This system considers the Mahalanobis distance that exceeds a threshold as an abnormal value. We describe the way to calculate the Mahalanobis distance of one variable.

D means the Mahalanobis distanceCx means the approximate entropyC$\bar{x}$ means the average approximate entropyC and s_x means standard deviation of the approximate entropyD

$$D = \frac{|\, x - \bar{x} \,|}{s_x} \tag{1}$$

x_i means the approximate entropy of ith We describe the way to calculate s_x.

$$s_x = \sqrt{\frac{1}{i} \sum_{k=1}^{i} (x_k - \bar{x})^2} \tag{2}$$

We describe the way to calculate the Mahalanobis distance of ith

$$D = \frac{|\, x_i - \bar{x} \,|}{s_x} \tag{3}$$

When many values separate greatly from the average value, we should use the median value rather than the average value. Figure 10 shows the number of HTTP requests of less than 50 bytes make up the majority of the number of occurrences. In this example, the median value is 19.00 bytes, however, the average value is 65.41 bytes. The average value is affected by a few HTTP requests which have large approximate entropy(the max value: 1889 bytes). Therefore, we should use median value when we calculate the approximate entropy.

We describe the way to calculate the median value. We assume that m data of one variable is given. What arranged these in ascending order is set to $x_1 \leq ... \leq x_m$. q-quartile(m_q) is defined as a point which divides the distribution into $q : 1-q$.

$$m_q = \phi((n-1)q+1) \tag{4}$$

$$\phi(t) = \begin{cases} x_t \ (t : naturalnumber) \\ (\lceil t \rceil x_{\lfloor t \rfloor} + (t - \lfloor t \rfloor) x_{\lceil t \rceil} \ (others) \end{cases} \tag{5}$$

However, $\lfloor x \rfloor$ is the smallest integer that is not smaller than x, and $\lceil x \rceil$ is the biggest integer that is not larger than x. Also, $m_{1/2}$ means the median value.

6 Experiment of Detection Leakage

We conduct the simulation of leakage detection. On this experiment, we assume that information of certain quantity is leaked by spyware. The content of the experiment is following.

(1) We make a HTTP request dataset which contains observing HTTP requests on a campus LAN.

(2) From the dataset, we select randomly 1000 HTTP requests which is sent from one source IP address.

(3) Random characters(30 bytes, 100 bytes, 1000 bytes) as leaked information are added to 100 HTTP requests(There are 900 normal HTTP requests and 100 abnormal HTTP requests.).

(4) We Calculate the approximate entropy and the Mahalanobis distance.

(5) We investigate the number of the Mahalanobis distance which exceeds the threshold.

(6) We Calculate the detection rate and false positive rate.

We define the number of HTTP requests whose approximate entropy exceeds the threshold as x, and the number of HTTP requests with leaked information whose approximate entropy exceeds the threshold as y. Moreover, we define the detection rate and false positive rate.

$$detection\ rate = \frac{y}{100} \tag{6}$$

$$false\ positive\ rate = \frac{x-y}{x} \tag{7}$$

Table 1 shows the result of the experiment.

added information	30byte		100byte	
threshold	detection rate	false positive rate	detection rate	false positive rate
0.1	52%	90%	99%	60%
0.2	32%	87%	99%	67%
0.3	27%	83%	86%	82%

added information	1000byte	
threshold	detection rate	false positive rate
0.1	99%	68%
0.2	99%	57%
0.3	99%	57%

Table 1: The detection rate and the false positive rate

The result shows that the detection rate is high, and the false positive rate is low when the added information is large. For example, the detection rate is 99% and the false positive rate is 57% when the added information is 1000 bytes, and the threshold is 0.3. On the other hand, the detection rate is low and the false positive rate is high when the added information is small. For example, the detection rate is 52% and the false positive rate is 90% when the added information is 30 bytes, and the threshold is 0.1.

7 Conclusion and Future Work

In this paper, we described the way to calculate the approximate entropy and the detection system. As a result of calculating the approximate entropy, we found that this system further ignored unnecessary information by increasing the number of history references. Moreover, we proposed the detection system and conducted simulation. We used the Mahalanobis distance to detect an abnormal value, and to decrease dependence on the circumstance of users. In the simulation that we assumed that certain information was leaked, the detection rate was high and the false positive rate was low when the added information was large. On the other hand, the detection rate was low and the false positive rate was high when the added information was small. Therefore, we will consider another way to calculate the approximate entropy to ignore further innocent information. If the approximate entropy becomes small, we can find information leakage easily. Moreover, we will collect spywares and experiment.

References

[1] C. Alme and M. Pekrul. Systems and methods for malware detection. US Patent Application. 13/021,585, April 2011. http://www.google.com/patents/US20110197281.

[2] K. Borders. Web tap personal. Products, March 2012. `http://www.webtapsecurity.com/products_personal.html`.

[3] K. Borders and A. Prakash. Web tap: detecting covert web traffic. In *Proc. of the 11th ACM conference on Computer and communications security (ACM CCS'04), Washington, DC, USA*, pages 110–120. ACM, October 2004.

[4] K. Borders and A. Prakash. Quantifying information leaks in outbound web traffic. In *Proc. of the 30th IEEE Symposium on Security and Privacy (S&P'09), Oakland, USA*, pages 129–140. IEEE, May 2009.

[5] K. R. Borders. *Protecting confidential information from malicious software*. PhD thesis, The University of Michigan, February 2009.

[6] T. S. Danny Iland, Alexander Pucher. Detecting android malware on network level. Ben Zhao's Graduate Networking course, December 2011. `http://cs.ucsb.edu/~iland/AndroidMalwareDetection.pdf`.

[7] J. Erman, A. Gerber, and S. Sen. HTTP in the home: It is not just about PCs. *ACM SIGCOMM Computer Communication Review*, 41(1):90–95, 2011.

[8] W. Masek and M. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*, 20(1):18–31, 1980.

[9] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner. Addroid privilege separation for applications and advertisers in android. In *Proc. of the 7th ACM Symposium on Information, Computer and Communications Security (AsiaCCS'12), Seoul, Korea*. ACM, May 2012.

[10] S. Saroiu, S. Gribble, and H. Levy. Measurement and analysis of spyware in a university environment. In *Proc. of the 2004 ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI'04), California, USA*, pages 11–11, March 2004.

[11] S. Yoshihama, T. Mishina, and T. Matsumoto. Web-based data leakage prevention. In *Proc. of the International Workshop on Security (IWSEC'10), Kobe, Japan, LNCS*, pages 78–93. Springer-Verlag, November 2010.

Kazuki Chiba was born in Otuchi, Iwate prefecture, Japan on 1989. He is a master student at Kyushu University, Japan. He received his BS degree from Kyushu University. His research interests include security protocols and cyber-attacks.

Yoshiaki Hori received B.E., M.E, and D.E. degrees on Computer Engineering from Kyushu Institute of Technology, Iizuka, Japan in 1992, 1994, and 2002 respectively. From 1994 to 2003, he was a Research Associate at the Common Technical Courses, Kyushu Institute of Design. From 2003 to 2004, he was Research Associate at the Department of Art and Information Design, Kyushu University. From 2004, he was an Associate Professor at the Department of Computer Science and Communication Engineering, Kyushu University. Since 2009, he has been an Associate Professor of the Department of Informatics, Kyushu University. His research interests include network security, network architecture, and performance evaluation of network protocols on various networks. He is a member of IEEE, ACM, and IPSJ.

Kouichi Sakurai is Professor of Department of Computer Science and Communication Engineering, Kyushu University, Japan since 2002. He received B.E., M.E., and D.E. of Mathematics, Applied Mathematics, and Computer Science from Kyushu University in 1982, 1986, and 1993, respectively. He is interested in cryptography and information security. He is a member of IPSJ, IEEE and ACM.