International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:5, No:4, 2011

Protocol and Method for Preventing Attacks from the Web

Ryuya Uda

Abstract**—**Nowadays, computer worms, viruses and Trojan horse become popular, and they are collectively called malware. Those malware just spoiled computers by deleting or rewriting important files a decade ago. However, recent malware seems to be born to earn money. Some of malware work for collecting personal information so that malicious people can find secret information such as password for online banking, evidence for a scandal or contact address which relates with the target. Moreover, relation between money and malware becomes more complex. Many kinds of malware bear bots to get springboards. Meanwhile, for ordinary internet users, countermeasures against malware come up against a blank wall. Pattern matching becomes too much waste of computer resources, since matching tools have to deal with a lot of patterns derived from subspecies. Virus making tools can automatically bear subspecies of malware. Moreover, metamorphic and polymorphic malware are no longer special. Recently there appears malware checking sites that check contents in place of users' PC. However, there appears a new type of malicious sites that avoids check by malware checking sites. In this paper, existing protocols and methods related with the web are reconsidered in terms of protection from current attacks, and new protocol and method are indicated for the purpose of security of the web.

Keywords**—**Information Security, Malware, Network Security, World Wide Web

I. Introduction

NOWADAYS, computer worms, viruses and Trojan horse become popular, and they are collectively called malware. Malware has been popular in the world of computer networks. However, recent malware steals more money than that of a decade ago. Services on the web have so much increased that malicious people take aim at the money related with the services. Of course, software which protects PC against malware, that is called anti-virus software, becomes popular. However, for ordinary internet users, countermeasures against malware come up against a blank wall. Methods for personal anti-virus software are roughly classified into three as follows.

A. Pattern matching method

This method is standard and simple to find malware. Characteristic pattern in a binary of malware is called signature. Anti-virus software has stored signatures of malware and compares a target with each signature. There are some problems in this method. When many patterns of malware are made at the same time, signatures that have to be compared with a target increase. Moreover, if vulnerabilities of software are not fixed for a long time, many patterns of malware can be made. Nowadays, malware making tool also becomes popular among malicious people. Making malware becomes easy for malicious people, even if they are not specialists of attacking vulnerability of target software. Metamorphic and polymorphic malware are worse than normal malware. In metamorphic malware, arrangement of operation codes is changed from that of original codes, or dead codes that are independent from operation of the malware are inserted. Usual pattern matching method is not appropriate to metamorphic malware, since its signature has been changed. Some types of metamorphic malware change themselves every time when they infect another computer with themselves. In polymorphic malware, operation codes are encrypted into other codes. Usual pattern matching method is neither appropriate to dealing with polymorphic malware, since its signature has been changed as well as signature of metamorphic one has.

B. Generic method

This method is applied for processes that are operated in a computer. Operation rules are written in a definition file, and anti-virus software compares the next operation of a target with the rules. If the next operation is out of the rules, the operation is canceled. For example, if there is an e-mail that is to be sent via different SMTP server from a server defined in the rules, anti-virus software judges the process of sending the e-mail may be malicious. This method is superior to the pattern matching one, since it can find subspecies of a malware. Subspecies of a malware behave as same as original one, even if they are metamorphic or polymorphic malware.

C. Heuristic method

This method is applied for programs in a computer before the programs are operated. With this method, programs are analyzed by anti-virus software. If a suspicious operation is found in the analysis, anti-virus software judges the programs may be malicious. The important thing in this method is that target programs are once operated somewhere. When the target program is operated in a virtual machine, the method is called dynamic heuristic method. Besides, there is also a kind of heuristic methods with which the target program is operated in a stand alone computer.

Among those above methods, pattern matching method seems to consume less computer resources than other methods, if all patches for vulnerability of installed software are appropriately applied. Anti-virus software has to compare a target with only signatures that relate with malware which

Ryuya Uda is with Tokyo University of Technology, 1404-1 Katakuramachi, Hachioji City, Tokyo 192-0982, JAPAN (corresponding author to provide phone: +81-42-637-2111; fax: +81-42-637-2112; e-mail: uda@cs.teu.ac.jp).

targets unsolved vulnerabilities on the installed software or on the OS of the computer. Release of patch of vulnerability sometimes delays although the vulnerability has been reported. In that case, signatures with which anti-virus software have to compare increase. Moreover, prompt attack called zero-day becomes major. In most of the cases, patch for vulnerability is released at the same time as the vulnerability is reported, since a researcher who discovers a vulnerability of software reports the fact first to the vender of the software and the researcher stands mute till the patch of the vulnerability is released. However, all the people all of the world cannot apply patch to their software within zero second after the patch is released. Some hackers make malware as promptly as possible when they know a new vulnerability and release the malware within the same day as the patch for the vulnerability is released. Some victims catch the malware before they apply the released patch. Still worse, patch is sometimes huge, although not all computers have hi-performance CPU and hi-speed communication network. For example, an attendee of an academic international workshop with a laptop computer might not have hi-performance CPU and hi-speed communication network.

Generic method is usually heavier than pattern matching method. With the method, almost all processes are watched by anti-virus software, even if the processes are generated by non-malicious famous software, since virus can be attached with existing software installed in a computer.

Heuristic method is heaviest among the methods above. Almost all the people do not have such environment as virtual machine or stand alone computer for special use.

Generic method and heuristic method are required for casual internet users, since a kind of malware, e.g. spyware or adware, is independent from vulnerability of software. Such malware is not computer virus in a narrow sense but application software that works stand alone. For example, a snake game which sends personal information to the net was found as application software on iPhone in 2010.

In this paper, problems of existing services that prevent various attacks from malware are described, and the way how to make the services secure especially in terms of HTTP protocols is indicated. Related works are described in section 2. Problems and solution is explained in section 3. The best way for security on the web is considered in section 4. The research is concluded in section 5.

II. Related Works

Scanning malware on casual user's PC consumes many resources as is mentioned in section 1. Nowadays, malware scanning service that scans malware in place of users becomes spread. Some web sites provide a service in which the service site scans a target web site before a user goes to the target site. LinkScanner Online [1], Dr. Web Online [2], Unmask Parasites (beta) [3] and vURL Online [4] represent such service, and for Japanese users, aguse [5] and gred [6] are provide such service in Japanese. Users input URL that they want to go in the scanning service site then they can get a report whether the target site is safe or not. In that service, addresses of malicious sites are on the blacklist.

Among those sites, aguse provides an advanced service that users can access any target site via aguse's site. On that service, the target site is changed into a picture file on aguse's site and users see the picture on aguse site, so that users need not execute malicious codes on their computer.

Moreover, in the paper of malware analysis, Yoshioka et al. [7] and Kasama et al. [8] report malware sandbox analysis as an online service. There are some researches analyzing malware dynamically in sandbox of a computer [9][10][11][12][13]. In those researches, malware is moved into special environment for analysis called sandbox then it is executed to see what it does. The analysis is effective for metamorphic or polymorphic malware, as is described in section 1. Besides, there is a service that analyzes malware online in place of users, such as Norman Sandbox [14] and Anubis [15], since the analysis is difficult for ordinary internet users. However, Yoshioka and Kasama give a caution to internet users in their papers, since it is reported that there is a kind of malware that avoids the analysis.

Therefore, in this paper, the way how to make the services secure is shown especially in terms of HTTP protocols.

III. Problems and Solutions

In this section, problems of HTTP protocol are argued in section 3.A. As for solving to the problems, effective malware checking method is suggested in section 3.B.

A. Problems of HTTP Protocol

Communication protocols are deployed on a layer according to OSI (Open Systems Interconnection) Reference Model as shown in Fig. 1.

Layer	Function
Application	Network process to application
Presentation	Data representation,encryption and decryption,convert machine dependent data to machine independent data
Session	Interhost communication
Transport	End-to-end connections and reliability,flow control
Network	Path determination and logical addressing
Data Link	Physical addressing
Physical	Media, signal and binary transmission

Fig. 1 OSI reference model

OSI reference model defines seven layers in communication protocols [16]. HTTP is a protocol on the application layer. OSI defines that the function of each layer is independent from other layers. Therefore, HTTP has been working well on TCP/IP whether other layers change their implementation. For example, HTTP works whether SSL or TLS is applied on the presentation layer or not. In fact, IP (Internet Protocol) address is not required in HTTP request header and HTTP response header in HTTP protocol, since IP is on the network layer.

However, HTTPD (HTTPD daemon) can change its behavior by IP address of web clients. For example, a web server can respond to anti-social messages while the server responds to social messages when a client is in the region of IP addresses that the government organizations have. The most of people believe that the same URL shows the same page, but it is dangerous to believe it. Malicious sites can respond to a page with malware when usual people access the site, while the sites response the same page (which looks like the same for usual people, but functionally not the same) without any malware. If the response is implemented in malicious sites, pre-scanning service for web site explained in section 2 does not work well, since it is easy for malicious users to know the IP addresses of the service site.

HTTP proxy may be one of the solutions of above problem. With using HTTP proxy, HTTP request from clients goes through HTTP proxy to servers. HTTP proxy can cache the response to the next client who is to access the same page of the server. If scanning services described in section 2 are implemented in HTTP proxy servers, clients under the same proxy server will be safe.

However, there are problems with using HTTP proxy. HTTP proxy decides whether it sends client's IP address to web servers or not in HTTP request header on HTTP protocol. One that sends client's IP address is called anonymous HTTP proxy, and the other is called non-anonymous HTTP proxy. Anonymous HTTP proxy is efficient to prevent malicious web sites from selecting clients. However, with anonymous HTTP proxy, good web sites cannot distinguish good clients from malicious clients, then the number of anonymous HTTP proxy, especially the one which any users can freely access, has decreased.

HTTPS is a secure protocol for the web. It guarantees confidentiality of communication and authentication between a server and a client. HTTPS seems the best solution, if the malicious third party tries to listen or change someone's data in communication. However, HTTPS provides no forensics when a client or a server is malicious. To make matters worse, HTTPS prevents security software from scanning virus via HTTP communication, since any software between a server and a client cannot see the communication data.

B. Effective Malware Checking Method

On the basis of arguments in section 3.A, an effective malware checking method is suggested as shown in Fig. 2.

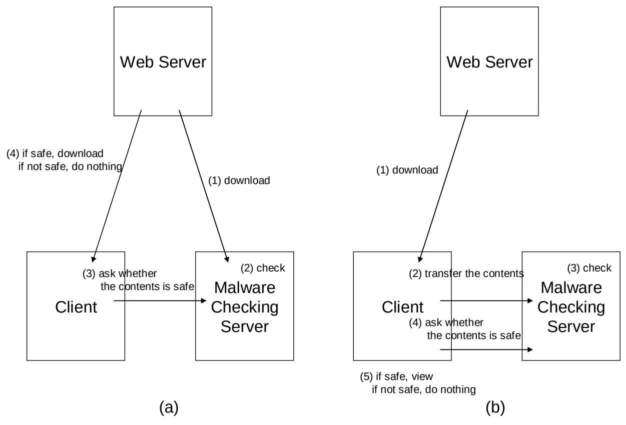


Fig. 2 Malware checking method

Fig. 2 (a) shows the method of existing online scanning services. In this method, if a malicious web server sends different contents to client from malware checking server, the answer of malware checking server does not make sense. Therefore, a new method is suggested as shown in Fig. 2 (b). The important thing is that clients never execute and show any content form web servers before checking. As soon as web browser downloads contents from the web server, it sends the contents directly to malware checking server without executing or viewing. Malware checking server checks contents and answer the client whether the contents are safe or not. After receiving the answer, the client shows the contents.

The method still has a problem that data size in communication becomes double. If the client downloads big size contents from the web server, the client have to upload the same size to the malware checking server, although uplink is narrower than downlink in popular broadband communication services.

Another suggestion is shown in Fig. 3.

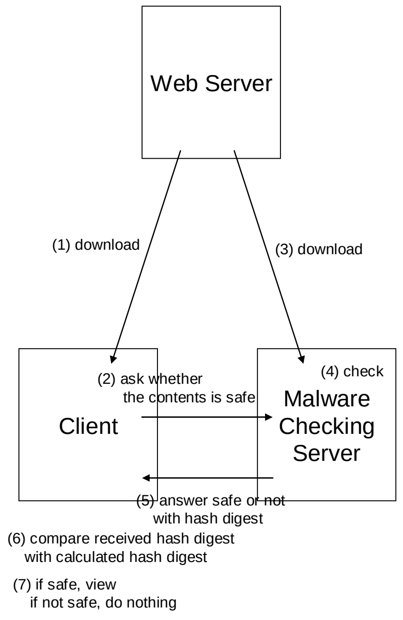


Fig. 3 Fast malware checking method

In this method, both a client and a malware checking server download the same page from the same web server. The client never sends contents to the malware checking server, so that client can avoid uploading big size contents. The malware checking server answers whether the contents are safe or not with a hash digest of the contents. The client compares the hash digest received from malware checking server with a hash digest which is calculated from the contents that the client download from the web server, so that the client would be aware of the difference, if the web server send different contents to the client and the malware checking server.

IV. Problems in Cloaking

The method that is shown in Fig. 3 in section 3.B works effectively, even if HTTPS is used, since the same contents are downloaded by the client and the malware checking server. However, if contents are dynamically created on the web server, contents that the client has and those that the malware checking server has are different from each other. The method above is called cloaking. For example, an article of usual news sites is shown in Fig. 4.

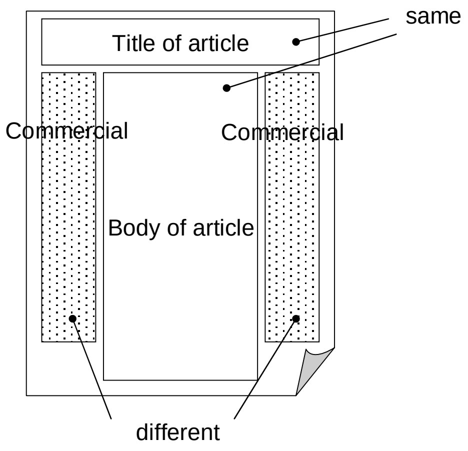


Fig. 4 News article on the web

The article is the same for all the people. However, commercials attached with the article is sometimes randomly changed or changed according to users' taste. In this case, title, body and commercial parts should be partially calculated to make hash digests, and then only parts that can not get the assurance of the malware checking server are replaced with safe picture etc.

Another solution for the news article is to combine two methods in Fig. 2 and Fig. 3 in section 3.B. First, the client tries the method in Fig. 3. Then if the two hash digests are different from each other, the client tries the method in Fig. 2 (b) or abandons viewing of the contents.

Another example is shown in Fig. 5.

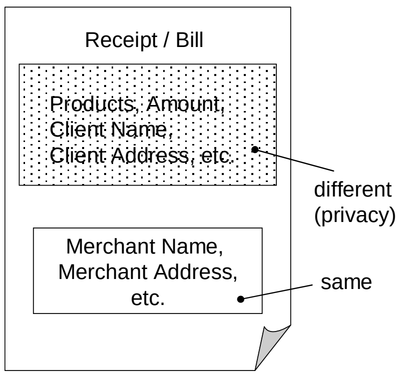


Fig. 5 Receipt or bill on the web

Fig. 5 shows receipt or bill displayed on the web. As the same way as in the news article, the different part and the same part are included. Moreover, in that case, personal information and privacy is also included.

The method in Fig. 3 is not appropriate to this case, since personal information and privacy is sent to the malware checking server. Moreover, to make the receipt or bill, the malware checking server has to copy the behavior of the client, i.e. which button the client push and what the client input in the textbox etc. Web browser on the client has to send that information to the server.

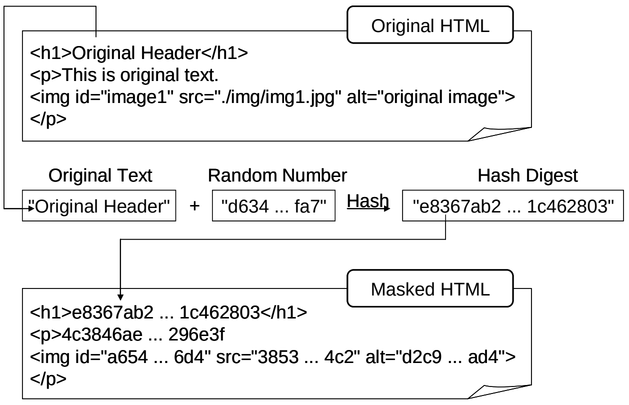
I think one of the best solutions is applying mask for contents. Under consideration of the problem of malware sandbox analysis described in section 2, the best solution seems to partially apply the method in Fig. 2 (b). The method in Fig. 2 (b) can solve both the problem described in section 2 and the problem of privacy. The method of checking malware without revealing any personal information to the malware checking server is explained in section 5.

V. Masking Contents on Web

In this section, the method of masking contents on the web is explained. As is mentioned in section 4, the method in Fig. 2 (b) has to be applied to prevent malware from recognizing that it is in malware sandbox analysis described in section 2, while keeping privacy on clients of users. There are some types of contents delivered on HTTP such as HTML, script, XML, XSLT, image, sound, etc. Through the method in this paper, each type of them is masked on clients as follows.

HTML can be masked more easily than other types of contents as shown in Fig. 6.

Each value of tags such as "Original Header" is concatenated with a random number, and then is hashed into a hash digest. Original value is replaced with the hash digest. Each value of attributes of tags such as "image1" is replaced with hash digests in the same way. The random number must be the same in one communication session between a client and the malware checking server in order to replace the same values with the same hash digests. For example, the value of "id" attribute must be referred by other HTML tags, style sheets or scripts on the same HTTP session. If one of the values of "id" attribute has changed, others which have had the same value must be changed into the same value as that one. The random number is required to prevent the malware checking server from guessing original text by comparing with hash digests in other contents by other clients.

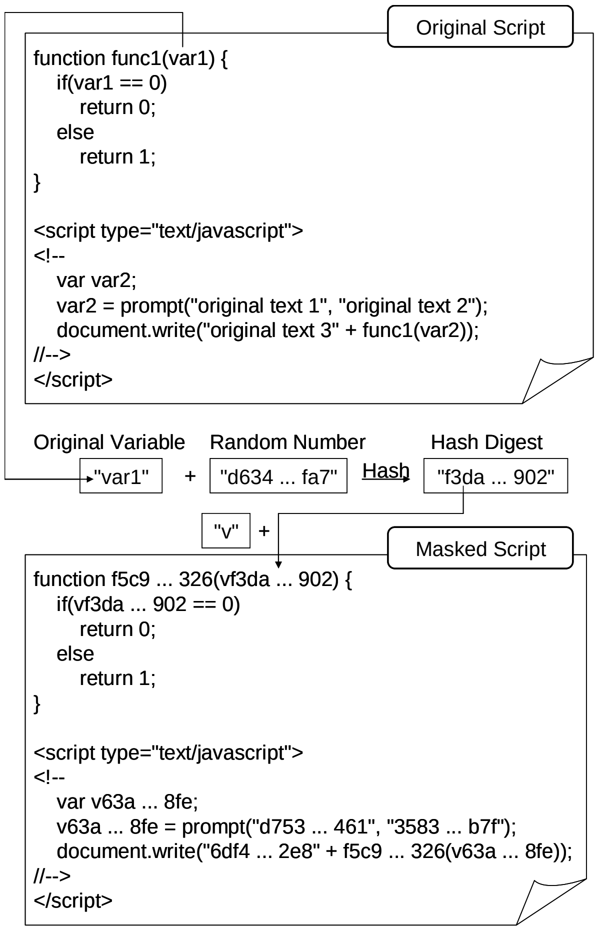


```
<h1>Original Header</h1>
<p>This is original text.
<img id="image1" src="./img/img1.jpg" alt="original image">
</p>
```

```
<h1>e8367ab2 ... 1c462803</h1>
<p>4c3846ae ... 296e3f
<img id="a654 ... 6d4" src="3853 ... 4c2" alt="d2c9 ... ad4">
</p>
```

Fig. 6 Masking HTML

Script as represented by JavaScript is masked as shown in Fig. 7.



```
function func1(var1) {
   if(var1 == 0)
      return 0;
   else
      return 1;
}

<script type="text/javascript">
<!--
   var var2;
   var2 = prompt("original text 1", "original text 2");
   document.write("original text 3" + func1(var2));
//-->
</script>
```

```
function f5c9 ... 326(vf3da ... 902) {
   if(vf3da ... 902 == 0)
      return 0;
   else
      return 1;
}

<script type="text/javascript">
<!--
   var v63a ... 8fe;
   v63a ... 8fe = prompt("d753 ... 461", "3583 ... b7f");
   document.write("6df4 ... 2e8" + f5c9 ... 326(v63a ... 8fe));
//-->
</script>
```

Fig. 7 Masking scripts

Names of variables and functions are replaced with the masked value. For example, one of the variables name "var1" is concatenated with a random number, and then is hashed into a hash digest. A character "v" is added in front of the hash digest to make a masked value. After that, "var1" is replaced with the masked value. In order to keep variables and functions referred by the name, the value of the random number must be the same in one communication session between a client and the malware checking server. The character "v" is required because the first character of the name of both variables and functions must not be numeric one. In masking the name of functions, character "f" is added in place of "v", because of the same restriction as in the name of variables. On the other hand, string in scripts requires no additional character in front of the value of the hash digest. Additionally, the value of "type" attribute in "script" tag must not be masked in order to tell the malware checking server the type of the script.

XML may be able to be masked in the same way as the masking way of HTML. In masking of XML, all of the names of tags must be masked. Moreover, character "/" for XPath in values of attributes must be eliminated from masking. XSLT may also be masked in the same way as the masking way of XML. By way of exceptions in masking XSLT, operators and some specific characters such as wildcard, "@", "[", etc. in values of attributes must be eliminated from masking.

Image and sound cannot be masked in the way as described above. However, ordinary image players and sound players usually execute no code in contents. Therefore, it seems to be enough to find malware on the malware checking server, if there are information in header of a file and information of the size of the file.

Application software cannot be masked either. To avoid the problem of malware sandbox analysis described in section 2, most of properties of clients should be transferred to the malware checking server, when the masked contents are transferred.

There remain problems that personal information might be leaked to the malware checking server, if numeric values in scripts include personal information such as credit card number, and if application software contains personal information in it when the server inserts the information at downloading. Solutions of the problems will be appeared in future works.

VI. Conclusion

In this paper, the problems of existing services that prevent various attacks from malware are described. Existing countermeasure against malware is not enough, since cipher algorithm in communication protocol is used only for confidentiality and authentication. HTTPS provides authentication between servers and clients, but nothing is assured about information exchanged between customers and traders. The method suggested in this paper especially pays attention to digital forensics on the web. With this method, electronic commerce or web surfing will be safer than with existing one. Some browsers provide environments for implementation of plug-in software. It seems the time that we have to consider the web of security with keeping existing protocols.

ACKNOWLEDGMENT

This research was partially supported by the Grant-in-Aid for Young Scientists (B), 21700088, 2010, the Ministry of Education, Culture, Sports, Science and Technology.

REFERENCES

[1] LinkScannerOnline, http://linkscanner.explabs.com/linkscanner/default.aspx
[2] Dr. Web Online, http://online.us.drweb.com/?url=1
[3] Unmask Parasites (beta), http://www.unmaskparasites.com/
[4] vURL Online, http://vurldissect.co.uk/
[5] aguse, http://www.aguse.jp/ (Japanese)
[6] gred, http://www.gred.jp/ (Japanese)
[7] K. Yoshioka, Y. Hosobuchi, T. Orii, T. Matsumoto, “Vulnerability in Public Malware Sandbox Analysis Systems”, in Proc. 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, 2010, pp.265–268.
[8] T. Kasama, T. Orii, K. Yoshioka, T. Matsumoto, “Vulnerability of Malware Sandbox Analysis as an Online Service (Part 2)”, IPSJ Anti Malware Engineering Workshop 2010, 2E1-1 (Japanese).
[9] U. Bayer, C. Kruegel, E. Kirda, “TTAnalyze: A Tool for Analyzing Malware”, in Proc. 15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR), 2006.
[10] D. Inoue, K. Yoshioka, M. Eto, Y. Hoshizawa, K. Nalao, “Automated Malware Analysis System and its Sandbox for Revealing Malware's Internal and External Activities”, IEICE Trans. Vol.E92D, No.5, pp.945–954, 2009.
[11] S. Miwa, T. Miyachi, M. Eto, M. Yoshizumi, Y. Shinoda, “Design and Implementation of an Isolated Sandbox with Mimetic Internet Used to Analyze Malwares”, in Proc. DETER Community Workshop on Cyber Security Experimentation and Test 2007, pp.6, 2007.
[12] C. Willems, T. Holz, F. Freiling, “Toward Automated Dynamic Malware Analysis Using CWSandbox”, Security & Privacy Magazine, IEEE, Vol.5, Issue 2, pp.32–39, 2007.
[13] K. Yoshioka, T. Matsumoto, “Multi-pass Malware Sandbox Analysis with Controlled Internet Connection”, IEICE Trans. E93A No.1, pp.210–218, 2010.
[14] NormanSandbox, http://www.norman.com/technology/norman_sandbox/
[15] Anubis, http://analysis.seclab.tuwien.ac.at/
[16] ITU-T Recommendation X.200, 1994.