J Inf Process Syst
http://dx.doi.org/10.3745/JIPS.03.0026

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

WebSHArk 1.0: A Benchmark Collection for Malicious Web Shell Detection

Jinsuk Kim*, Dong-Hoon Yoo*,**, Heejin Jang*, and Kimoon Jeong*

Abstract

Web shells are programs that are written for a specific purpose in Web scripting languages, such as PHP, ASP, ASP.NET, JSP, PERL-CGI, etc. Web shells provide a means to communicate with the server's operating system via the interpreter of the web scripting languages. Hence, web shells can execute OS specific commands over HTTP. Usually, web attacks by malicious users are made by uploading one of these web shells to compromise the target web servers. Though there have been several approaches to detect such malicious web shells, no standard dataset has been built to compare various web shell detection techniques. In this paper, we present a collection of web shell files, WebSHArk 1.0, as a standard dataset for current and future studies in malicious web shell detection. To provide baseline results for future studies and for the improvement of current tools, we also present some benchmark results by scanning the WebSHArk dataset directory with three web shell scanning tools that are publicly available on the Internet. The WebSHArk 1.0 dataset is only available upon request via email to one of the authors, due to security and legal issues.

Keywords

Benchmark Test Collection, Malicious Web Application, Webshell, Web Shell Collection, Web Shell Detection

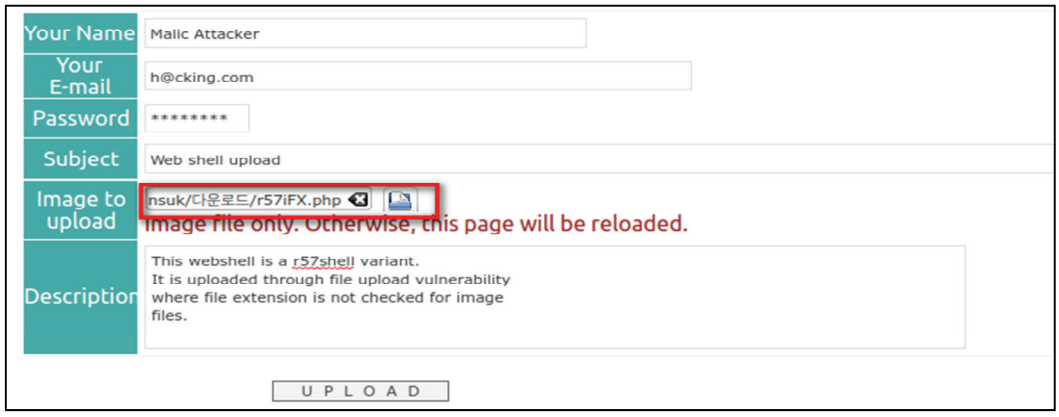
1. Introduction

As of January 2014, there was more than 861 million websites [1] operated every day for both personal and professional use. Being so common, these websites have high chances of becoming targets of web attacks by malicious hackers. Out of web attacks, uploading web shells to the target websites is the most widespread means of compromising and exploiting the victim server. There has been a report that one or more malicious web shells are found in 91% of hacked websites in South Korea [2].

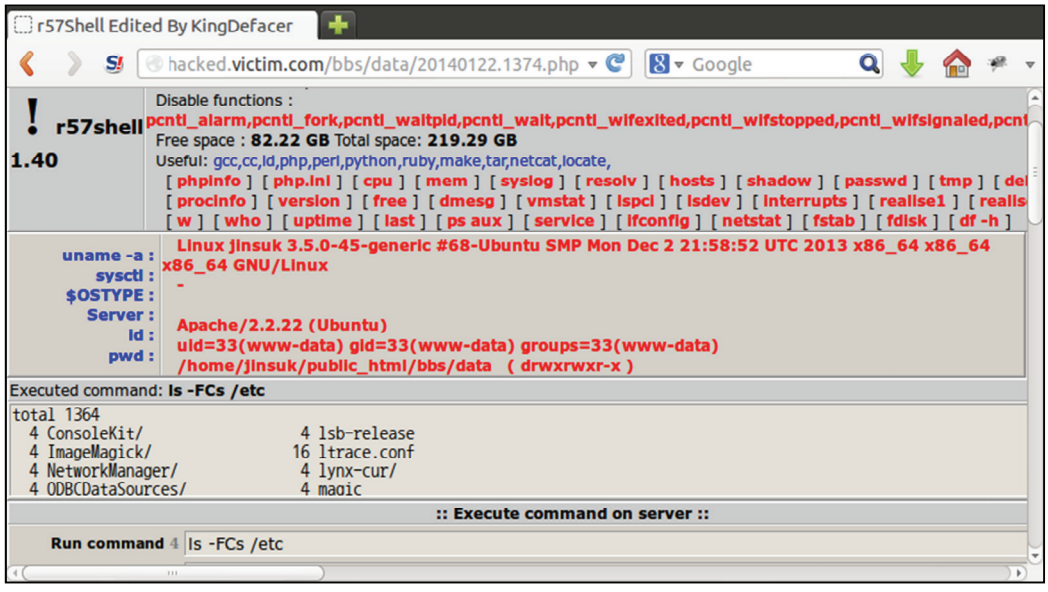
Malicious web shells are small programs or scripts that can be opened from a web browser to provide a web-based interface to run OS specific system commands. These run purely over the WWW via HTTP protocol. Usually, web shells provide a quick GUI interface to do one or more of the following common tasks: 1) executing OS shell commands, 2) traversing across directories, 3) viewing files, 4) editing files, 5) downloading files, 6) deleting files, 7) uploading files, 8) executing DB SQL queries and commands, 9) bypassing mod_security, 10) sending spam mails, 11) running IRC bots, and 12)

※ This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Manuscript received February 25, 2014; accepted April 16, 2014.
Corresponding Author: Kimoon Jeong (kmjeong@kisti.re.kr)
* Department of Science & Technology Security, National Institute of Supercomputing & Networking (NISN), Korea Institute of Science & Technology Information (KISTI), Daejeon 305-806, Korea ({jinsuk, jhj, x82, kmjeong}@kisti.re.kr)
** Department of Computer Engineering, Chonnam National University, Gwangju 500-757, Korea (x82@kisti.re.kr)

Copyright© 2015 KIPS



(a)



(b)

Fig. 1. An example of (a) uploading a web shell file by exploiting image file upload vulnerability and (b) remote execution of the uploaded web shell, r57shell version 1.40, in a web browser.

opening reverse connections to the attacker [3-5]. A malicious attacker may use one of these web shells in targeting website's vulnerabilities, such as arbitrary file upload and remote/local file inclusion. If a web server suffers from any type of these vulnerabilities, the attacker would then upload or include a web shell, open it from the web browser with the correct path, and get the interface to run an arbitrary command on the target system (see Fig. 1).

There are plenty of web shell examples in multiple languages, such as PHP, ASP, ASP.NET, and Java (JSP). Basically, there is no essential difference between malicious web shells and normal web application scripts. Therefore, it is difficult to detect the malicious activities of uploaded web shells using tools, such as antivirus software and Intrusion Detection Systems (IDS). To mitigate this situation there have been many attempts to build web shell detection tools, such as PHP WebShell Detector [6], NeoPI [3], and Web Security Framework (WSF) [7]. Linux Malware Detect (LMD), which is a well-known malware

scanner for Linux operating systems, also contains web shell detection and cleaning functions [8].

Most of the web shell detectors are signature- or pattern-based tools. Since traditional web shells are simple and easy to detect, signature-based web shell detectors have performed well in drawing out conventional web shell files. However, as various encryption and obfuscation techniques are applied to bypass antivirus (AV) software and IDSs, it is getting more difficult for these signature-based tools to detect web shells correctly. Some web shells have implemented mechanisms, such as compression and encryption techniques, which are specifically aimed at avoiding detection. This is especially effective at thwarting signature-based detection systems or keyword-search tools. In addition, finding such malicious files on enterprise web servers that contain tens of thousands of pages can be extremely difficult due to the sheer volume of data.

Since web shells have no essential difference with normal web applications and thus are difficult to detect with AVs and IDSs, several tools that are specialized to detect malicious web shells have been developed [3,6-8]. Surprisingly, no web shell dataset for evaluating web shell detection approaches has been built yet, as far as we know. Standard datasets, which are sometimes called test collections, will enable researchers and developers to test their ideas. Also, standard datasets also can allow researchers to objectively compare their results with published studies. For example, to compare their ideas with other approaches in the IDSs, most IDS researchers have used the KDD datasets that were compiled in 1998 and 1999 [9-11]. In this paper we introduce the WebSHArk test collection, which is abbreviated from “Web SHell Arkive”. This test collection is a set of web shells written in various languages, such as PHP, ASP, ASPX, JAVA, and CFM. We hope that this dataset can be used as a standard dataset to improve their own web shell detection approaches, and to be able to compare their own with published data and other tools’ performance.

The remainder of this paper is organized as follows: in the next section, related work about the detection and collection of web shells is briefly presented. We describe the WebSHArk 1.0 collection in Section 3. In Sections 4 and 5, we present our benchmark method and the results of publicly available web shell detection tools applied to our WebSHArk 1.0 collection. Finally, we conclude with the discussion and propose some future works in Section 6.

2. Related Work

As mentioned briefly in the introduction, there has been several attempts to develop web shell scanning tools that are mainly based on signature-based detection [3,6-8]. Emposha [6] reports that the PHP Shell Detector (PSD) used 141 web shells to build its signature database. Behrens and Hagen [3] note 90 web shells in their document describing the design policy of the NeoPI webshell scanner. R-fx Networks [8] reports that LMD 1.4.0 has a total of 7,241 signatures without notice of the portion of web shell patterns. Most of the AV software can also detect various kinds of web shell files. However, they do not publicly share their web shell collection or detection features due to the collection’s and features’ confidential or proprietary nature and also because of some security reasons.

We surmise that most of the existing web shell detection tools also use their own web shell collections but they do not share them publicly, which means it is not possible to compare their performance with that of other tools. There is a web shell collection that can be accessed via HTTP [12], but it is arbitrarily collected and hence, it is not organized as a standard web shell dataset.

In spite of the seriousness of web shells in Internet security, it is surprising that there is no systematic approach for building a standard web shell dataset. We hope that our work presented in this paper can be used as a standard dataset and help to compare the performance of web shell detection against

Table 1. Statistics of files in WebSHArk v1.0 dataset current and future approaches.

Partition	Script language	Directory	# of files	Description
Web shell	PHP	PHP/	433	PHP web shell files
		PHP/SPAM/	3	PHP Spam mailer
	ASP	ASP/	145	ASP web shell files
	ASP.NET	ASPX/	39	ASP.NET web shell files
	JAVA	JSP/	132	JSP web shell files
	PERL	PL-CGI/	30	Web shell files written in Perl script
	CFM	CFM/	12	ColdFusion web shell files
	Etc.	C/ PY/ SH/ ETC/	15	Web shell files written in C, Python, Shell script, and TCL.
	Total number of web shells		809	
	Miscellaneous		128	Supplementary files with .dsc extension and other files
True negative	ASP	true-negative/AspNuke	800	ASP Nuke version 0.80 An open source ASP CMS (GPL).
	ASP.NET	true-negative/Umbraco	2,684	Umbraco CMS version 7.0.2 An open source ASP.NET CMS (MIT License).
	ASP	true-negative/asp-vbscript-cms	692	ASP VBScript CMS version 0.4 An open source ASP CMS (MIT License).
	ASP	true-negative/instant-content-management	85	Instant Content Management version 1.1 An open source ASP CMS (GPL).
	PHP	true-negative/joomla	5,379	Joomla CMS version 3.2.1 An open source PHP CMS (GPL).
	JAVA	true-negative/ magnolia-5.2.1	298	Magnolia CMS version 5.2.1 An open source JAR CMS (GPL)
	ASP	true-negative/mysql-aspCMS	106	MySQL ASP Web Content Management 1.1 An open source ASP CMS (GPL).
	PHP	true-negative/openengine191	1,667	openEngine version 1.9.1 An open source PHP CMS (GPL)
	PHP	true-negative/wordpress	1,162	WordPress version 3.8 An open source Web publishing software (GPL).
	Sub total		12,873	
Total			13,810	

3. WebSHArk Collection Version 1.0

WebSHArk 1.0 is a collection of web shells written in server-side programming languages, such as PHP, ASP, ASP.NET, Java/JSP, CFM, PERL, etc. These were collected from various sources, including an unsorted web shell archive [12], a collection of web shells and RFIs [13], and other various web resources. Also, we included some web shells that we found during our own penetration tests for finding vulnerabilities in web applications. If a newly arrived web shell file has an identical MD5 hash value with an existing file in the collection, it was considered to be a duplicated file and discarded. We also added about 13,000 non-web shell files to the dataset. As such, the WebSHArk 1.0 dataset consists of a total 809 web shells (*true positives*) and 13,001 normal web files (*true negatives*), resulting in totally 13,810 files in 2,507 directories (shown in Table 1). For each web shell file, we tried, whenever it was possible, to provide a description file with a .dsc extension, which describes the date and/or sources of the web shell.

With true web shell files (true positives) only, we can measure recall but not precision. Precision is defined as (true positives retrieved)/(true positives retrieved + false positives retrieved). But in the real field, precision can also be a matter of efficiency. Thus, we added nine open-source content management systems (CMSs) written in script languages, such as PHP, ASP, ASP.NET and JAVA. We selected these languages to reflect the real usage of server-side programming languages for websites in the year 2014 (PHP 81.7%, ASP.NET/ASP 18.0%, Java 2.7% [14]). The sources of CMS added as true negatives are 1) ASP Nuke 0.80 [15], 2) Umbraco CMS 7.0.2 [16], 3) ASP VBScript CMS 0.4 [17], 4) Instant Content Management: ICM 1.1 [18], 5) Joomla! CMS 3.2.1 [19], 6) Magnolia CMS 5.2.1 [20], 7) MySQL ASP Web Content Management 1.1 [21], 8) openEngine CMS 1.9.1 [22], and 9) WordPress 3.8 [23]. These true negative files were added to mimic the real world web environment and to evaluate the scanning speed of web shell detection tools.

The WebSHArk 1.0 dataset is distributed as a gzipped tar archive file, WebSHArk-1.0.tar.gz, of which size is 172 MB. It is only available upon request to the authors due to security reasons and legal issues in Republic of Korea. Please send an e-mail to one of the authors to obtain the dataset.

4. Benchmark Method

An important part of the value of a test collection or standard dataset is the availability of published benchmark results. Good benchmark results serve to ensure that apparently superior new approaches are not being compared to artificially low baselines. To provide such baselines, and to evaluate the current status of existing web shell detection techniques, we ran three web shell detection tools against the WebSHArk 1.0 dataset. We used LMD 1.4.2 [8], PSD 1.65 [6], and Web Shell Scanner (WSS) v0.4 [7] to provide benchmark results. These tools categorize each file as malicious, suspicious, or normal. ‘Malicious’ means that the file is a well-known web shell. ‘Suspicious’ means that the file is unknown, but it is potent to be a web shell. We present the sum of malicious and suspicious files as the web shell detection result.

Web shell detection can be regarded as a binary classification since it is a task of identifying whether or not a file is a web shell. In the statistical analysis of the binary classification, the F_1 score is one of the most widely used measures of a test’s accuracy. To measure the effectiveness of the three web shell

detectors, we used the F_1 score [24], which corresponds to the harmonic mean of precision and recall, as shown below:

$$P = \frac{\text{number of true webshell files identified correctly}}{\text{number of all files identified as webshells}}$$
$$R = \frac{\text{number of true webshell files identified correctly}}{\text{number of all true webshell files}} \quad (1)$$
$$F_1 = \frac{2RP}{R+P} = \frac{2P_t}{2P_t + P_f + N_f}$$

where P_t is the number of files that a tool *correctly* identifies as web shells (true positives), P_f is the number of files that the tool *incorrectly* identifies as web shells, N_f is the number of true web shell files that the tool cannot identify as web shells (false negatives), P is the precision, i.e., $P_t/(P_t+P_f)$, and R is the recall rate, i.e., $P_t/(P_t+N_f)$.

The severity of any web shell is that it is very hazardous to the victim sites. The existence of even a very simple web shell can lead to a catastrophic result on the web server and even on the local area network that hosts the victim server. Thus, recall is more important than precision. This means that the more true positive web shells are identified, the better the web shell detection tool performed. However, in the real web service environment, tens of thousands of web service files exist and a shell scanner with a signature-based detection algorithm may sometimes detect too many false positives during incident handling process for identifying the web shell attacks. As such, precision also plays an important part in measuring the effectiveness of a web shell detection tool with regard to practical aspects, since false negative detection results can lead to misunderstanding of the web attacks and failing to remove true web shells. To weigh precision against recall, we also used the more general formula of F_1 in the results (Table 2). The general form of Eq. (1) can be denoted as an F_β measure where β is a positive real number. F_β is shown in Eq. (2):

$$F_\beta = (1+\beta^2)\frac{RP}{R+\beta^2 P} \quad (2)$$

In addition to the F_1 score, two other commonly used F_β measures are the F_2 measure, which weights recall higher than precision, and the $F_{0.5}$ measure, which puts more emphasis on precision than recall. During our analysis of the results, we found that plotting F_β values against β is helpful in understanding the behavior of the scanners in some cases. In light of this, we will present the plotted results of effectiveness measures in the form of $F_{0.5}$, $F_{1.0}$, $F_{1.5}$ and $F_{2.0}$ as baselines for future studies.

5. Benchmark Results

We scanned the WebSHArk directory with LMD 1.4.2, PSD 1.65, and WSS 0.4 to detect web shell files. The benchmark results for the three web shell detectors are shown in Table 2 and Fig. 2. From true

positive (TP), false positive (FP), and false negative (FN), precision and recall can be calculated. Four levels of F_β values that were computed from precision and recall are also presented in Table 2. They are denoted as $F_{0.5}$, $F_{1.0}$, $F_{1.5}$, and $F_{2.0}$. PSD and WSS present the scanned results in the two detection levels of malicious and suspicious. The files detected as being malicious are usually well-known web shells. Files containing functions that can do system calls, such as eval(), system(), exec(), passthru(), and shell_exec(), are detected as being suspicious (i.e., potent web shells). In analyzing the results of PSD and WSS, we regarded both malicious and suspicious results as being positives. As shown in Table 2, LMD only detects known web shells.

LMD outperforms the other two scanners in precision. If a web shell is detected by LMD, it is guaranteed that the file is really a web shell, as shown in the result of precision=1.0 for LMD. WSS outperforms in the recall. As mentioned above, recall is more important than precision since a potent web shell can lead to a catastrophic result for the victim web server. However, during scanning with WSS and PSD, painstaking macrography may be required in some cases because these two tools detect too many false positives.

In Fig. 2, the results are plotted in regard to F_β measures for the three web shell scanners applied to the WebSHArk 1.0 dataset. While the F_β values of PSD and WSS have a steady tendency for varying β = 0.5, 1.0, 1.5, and 2.0, LMD shows a reversely proportional pattern in β-vs-F_β plot. LMD has 100% of precision, but it also has a poor recall of 34.9%, which results in huge biases in β–vs–F_β plotting. This

Table 2. Benchmark results

Partition	LMD	PSD	WSS
TP	282	381	549
FP	0	364	299
FN	527	428	260
P	**1.000**	**0.511**	**0.647**
R	**0.349**	**0.471**	**0.679**
$F_{0.5}$	0.728	0.503	0.653
$F_{1.0}$	**0.517**	**0.490**	**0.663**
$F_{1.5}$	0.436	0.483	0.669
$F_{2.0}$	0.401	0.479	0.672

LMD=Linux Malware Detect, PSD=PHP Shell Detector, WSS=Web Shell Scanner, TP=true positive, FP=false positive, FN=false negative, *P*=precision, *R*=recall.

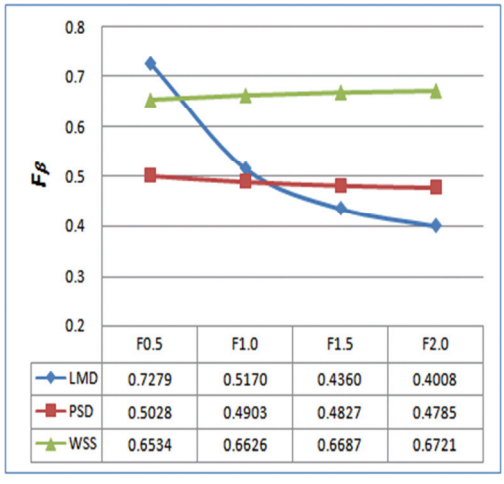


Fig. 2. Plotting of the benchmark results, F_β.

means that LMD needs to be improved in the respect of recall measure. Compared with LMD, PSD and WSS showed relatively high recall but poor precision rates of 51.1% and 64.7%, respectively. Poor precision means a labor-intensive examination of suspicious (i.e., potential) web shells. PSD and WSS should further improve their precision measure.

As seen in Fig. 2, if there is big bias in precision and recall and/or if one wants to weigh precision against recall, we recommend that β–vs–F_β plotting is a superior measure to the F_1 measure. Otherwise, the F_1 measure may be acceptable as a general purpose measure in web shell detection.

We also considered the speed of web shell scanning. All of the tests were carried out on a Linux PC with Intel i7-3770 3.4 GHz CPU, 16 GB RAM, and a 256-GB SSD hard drive while the machine was under light load. To scan the whole WebSHArk 1.0 directory, LMD took 11 minutes 40.5 seconds, which corresponds to the scanning speed=19.7 files/sec. PSD took 1 minute 4.3 seconds, scanning 214.8 files/sec; and WSS took 6.4 seconds, scanning 2,147.8 files/sec. Regarding enterprise level websites with more than hundreds of thousands of files, LMD may suffer from low scanning speeds, while WSS can scan the website very quickly.

6. Conclusions and Discussion

In many research and engineering fields, standard benchmarking datasets exist and they are used to objectively compare various approaches on the same research subject. To accomplish this goal in the web shell detection field, we built the WebSHArk 1.0 dataset, which contains 809 malicious web shell files and an additional 13,001 normal files. We also tested three publicly available web shell detection tools against the dataset and presented the results as baselines for future studies. From the results in Table 2, we suggest that LMD should collect more web shells and signatures not to miss real web shells. On the other hand, since PSD and WSS have too many false positive results, they should improve the detection patterns more accurately not to misidentify normal files as web shells.

The results with the $F_{1.0}$ value between 0.490 and 0.663 was far less than as we expected. This is mainly due to obfuscated web shells being determined to be false negatives. Let's take a look at the following PHP file without any obfuscation techniques applied (this file can be found at `PHP/simple-`webshell-request-cmd.php in the WebSHArk dataset):

```
<pre><body bgcolor=white><? @system($_REQUEST["cmd"]); ?></body></pre>
```

This shell is straightforward and web shell scanners can catch the system() function from the web shell file. If it is uploaded to a victim website as simple-cmd.php, an attacker can execute an OS command by accessing a fabricated URL like http://www.victom.com/upload/simple-cmd.php?cmd=cat+/etc/passwd. In our results, both PSD and WSS judged the file as a suspicious web shell.

Let's take a look at another PHP code with simple obfuscation applied (this string is found at PHP/simple-request-cmd-obf.php):

```
<pre><? $xx = 's'.'y'.'s'.'t'.'e'.'m'; @$xx($_REQUEST["mycm"]); ?></pre>
```

Though this PHP shell is almost identical to `simple-webshell-request-cmd.php`, all three of the scanners did not draw out this file as a malicious web shell. The detection of system calls in a runtime environment may help overcome this situation. Using a kind of pre-compiler to generate a normal PHP code can also mitigate this case. The detection of obfuscated web shells is expected to be a major obstacle in web shell detection. To overcome the weakness of signature-based scanners in detecting encrypted and/or obfuscated web shells, we need algorithms other than pattern-matching ones to detect these web shells. NeoPI showed that features other than a web shell signature could partially help in detecting pesky shells with encryption and obfuscation. Examples of such features include the longest string, entropy, and index of coincidence [3]. However, since NeoPI is still in the

seed phase it has a weakness in its being practically applied in the real world. Future studies are needed to adopt one or more of these algorithms, in order to improve the web shell detection effectiveness.

Web shells can be defined as an *undocumented way* to gain console access to a computer system through a dynamic server side webpage [3]. Most of the undocumented ways are deliberately created by malicious attackers. However, sometimes they can come from the unwanted results of a goodwill programmer's mistakes in server-side programming in which they did not follow the secure coding guidelines. Undocumented ways can also be called "web vulnerabilities". Webshell scanners with delicate design may be able to detect such *unintentional* vulnerabilities since they scan the *intentional* vulnerabilities of web shell files. We expect that further studies can take web shell detection tools to the level of vulnerability scanner/analyzer software for server-side scripting languages.

So far, we have described the WebSHArk 1.0 dataset, which is the first standard web shell collection. We have also explained how the benchmark results of three publicly available web shell detection tools can be used as the baseline performance for future studies. We hope that our web shell dataset can contribute to the improvement of web shell scanning techniques and thereby, more secure Internet life

Acknowledgement

The authors gratefully acknowledge the Ministry of Science, ICT, and Future Planning (MSIP), Republic of Korea for their partial support of this research.

References

[1] Netcraft, "January 2014 Web server survey," 2014; http://news.netcraft.com/archives/2014/01/03/january-2014-web-server-survey.html.

[2] KrCERT/CC, *Monthly trend and analysis of Internet attacks (May 2008)*. Seoul: Korea Internet & Security Agency, 2008.

[3] S. Behrens and B. Hagen, "What's up with these pesky shells?" 2011; http://resources.infosecinstitute.com/web-shell-detection/.

[4] X. Mingkun, C. Xi, and H. Yan, "Design of software to search ASP web shell," *Procedia Engi-neering*, vol. 29, pp. 123-127, 2012.

[5] D. Canali, D. Balzarotti, and A. Francillon, "The role of web hosting providers in detecting com-promised websites," in *Proceedings of the 22nd international conference on World Wide Web*, Rio de Janiro, Brazil, 2013, pp. 177-188.

[6] Emposha, "PHP shell detector: web shell detection tool," 2011; http://www.emposha.com/securi ty/php-shell-detector-web-shell-detection-tool.html.

[7] H. Park, J. Kim, K. Jeong, and Y. Lee, *User Guide to WSF (Web Security Framework)*. Daejeon, Korea: Korea Institute of Science & Technology Information, 2014.

[8] R-fx Networks Linux Malware Detect, https://www.rfxn.com/projects/linux-malware-detect/.

[9] 1998 DARPA Intrusion Detection Evaluation Data Set, http://www.ll.mit.edu/ideval/data/1998data.html.

[10] 1999 DARPA Intrusion Detection Evaluation Data Set, http://www.ll.mit.edu/ideval/data/1999data.html.

[11] G. Creech and J. Hu, "Generation of a new IDS test dataset: time to retire the KDD collection," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, 2013, pp. 4487-4492.

[12] Webshell, https://github.com/tennc/webshell.

[13] Web shells and RFIs collection, http://www.irongeek.com/i.php?page=webshells-and-rfis.

[14] W3Techs, "Usage of server-side programming languages in websites," http://w3techs.com/technologies/overview/ programming_language/all.
[15] ASP Nuke CMS ASP Nuke 0.80, http://asp-nuke-cms.soft112.com/.
[16] Umbraco, http://umbraco.com/.
[17] ASP VBScript CMS: open source content management for window IIS, http://code.google.com/p/asp-vbscript-cms/.
[18] Instant content management systems, http://www.instant-cms.com/.
[19] Joomla!, http://www.joomla.org/.
[20] Magnolia CMS, http://www.magnolia-cms.com/.
[21] MySQL ASP Web Content Management, http://www.mysql.aspcontentmanagement.com/.
[22] Web Content Management System openEngine, http://www.openengine.de/html/pages/de/index.htm.
[23] WordPress.com, http://wordpress.com.
[24] C. J. Van Rijsbergen, *Information Retrieval*, 2nd ed. London: Buttersworths, 1979.

Jinsuk Kim

He received the B.S. and M.S. degrees in Life Science from KAIST in 1993 and 1995, respectively. He also received the M.S. degree in Computer Science from KAIST in 2002. During 1995-2012 he worked for KISTI and Search Solutions Inc. to develop information retrieval systems. Since 2013 he works as a vulnerability analyst in KISTI. His research interests include Web vulnerability analysis and information retrieval.

Dong-Hoon Yoo

He received the M.S. degree in Computer Science from Chonnam National University in 2010. And now he is undertaking a doctorate course as a member of Information Security Lab at Chonnam University. During 2001-2012 he worked for InetCop Inc. in the field of smartphone security. Since 2013 he works as a vulnerability analyst in KISTI. His research interests include Android vulnerability analysis and malware detection in mobile devices.

Heejin Jang

She received the B.S. and M.S. degrees in Computer Science from POSTECH in 2001 and 2003, respectively. During 2003-2011 she worked for Samsung Electronics Inc. and National Security Research Institute as a network software developer and network security researcher. Since 2012 she works as a network security and vulnerability analyst in KISTI. Her research interests include Web vulnerability analysis and network security.

Kimoon Jeong

He received the B.S., M.S., and Ph.D. degrees in Computer Science from Chonnam National University in 1999, 2001, and 2009, respectively. During 2001-2005 he worked for KISA and NIS as a security researcher. Since 2005 he works as network security and vulnerability analyst in KISTI. His research interests include system vulnerability analysis, computer emergency response and malware detection in mobile devices.