JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 28, 911-924 (2012)

DOI:10.1688/JISE.2012.28.5.6

Efficient Detection of Malicious Web Pages Using High-Interaction Client Honeypots*

HONG-GEUN KIM[1], DONGJIN KIM[2], SEONG-JE CHO[2,+], MOONJU PARK[3] AND MINKYU PARK[4]

[1]*Korea Internet Security Agency*
Seoul, 138-950 Korea
[2]*Department of Computer Science*
Dankook University
Gyeonggi, 448-701 Korea
[3]*Department of Computer Science and Engineering*
University of Incheon
Incheon, 406-772 Korea
[4]*Department of Computer Engineering*
Konkuk University
Chungbuk, 380-701 Korea

Drive-by-download attacks are client-side attacks that originate from web servers clients visit. High-interaction client honeypots identify malicious web pages by directly visiting the web pages and are very useful. However, they still have shortcomings that must be addressed: long inspection time and possibility of not detecting certain attacks such as time bombs. To address these problems, we propose a new detection method to identify web pages with time bombs. The proposed method introduces a pattern-based static analysis for detecting time bombs efficiently. A high-interaction client honeypot performs the static analysis before carrying out execution-based dynamic analysis. The static analysis classifies sample web pages into two groups, the first one assumed to be time-bombs and the second one assumed to be no time-bombs. We then perform dynamic analysis for the first using sequential visitation algorithm with long classification delay and for the second using divide-and-conquer visitation algorithm with short classification delay. Experimental results demonstrate that our method is more accurate and costs less than conventional methods.

Keywords: high-interaction client honeypot, malicious web page, visitation algorithm, logarithmic divide-and-conquer (LDAC) algorithm, detection method, time bombs, static analysis

1. INTRODUCTION

Drive-by-download attacks have emerged as a new threat to the integrity of PC systems [1-9, 11-13]. It is estimated that approximately 150 million malicious web pages that launch drive-by-download attacks exist in 2010 [13]. Drive-by-download attacks are client-side attacks that originate from web servers that web browsers visit. If a vulnerable web browser retrieves a web page, a malicious web server can push malware to a client

Received May 31, 2011; accepted March 31, 2012.
Communicated by Jiman Hong, Junyoung Heo and Tei-Wei Kuo.
* This work was supported partly by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0026301), and by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development.
+ Corresponding author.

system. This malware can be executed without the user's notice or consent. The detection of malicious web pages is one of main issues among many security researchers.

Client honeypots are active security devices that are able to detect a malicious server's web pages [1, 3, 5, 6-13]. Client honeypots directly visit a web page, and then inspect state changes of the system on which they are running. A detection of unauthorized changes to client honeypots may indicate the occurrence of an attack. Such changes include a new file or process, an altered file, and the modification of registry file, etc. As in the case of traditional server honeypots, client honeypots are mainly classified by their interaction level: high or low. An interaction level denotes the level of functional interactions a server can utilize on a client honeypot. We focus on high-interaction client honeypots in this paper.

High-interaction client honeypots (HICHPs) are effective at detecting unknown attacks and obfuscated patterns of malware because they use dynamic analysis. However, they have challenges to overcome: they are inherently slow and miss attacks sometimes [1, 5, 7, 10, 13]. One challenge of HICHPs is how they should visit web pages to reduce total detection time. It takes very long time to crawl millions of web servers. Detection of malicious web pages in those quantities would take very long time and cost millions of US dollars [7, 13]. Therefore, detection speed is an important factor to HICHP's operating costs. Detection accuracy is another factor that largely impacts on operating costs. While an HICHP's false positive rate is negligible, the false negative (FN) rate drives the cost of identifying malicious web pages [7, 13]. The FN rate means a ratio of the failure to detect a malicious web page when it is inspecting one. A malicious web page can employ several evasion techniques that cause an HICHP to fail at detecting it. One of the major evasion techniques is time bombs. Time bombs are exploits contained on a malicious web page. These exploits trigger only after a predefined time has elapsed.

We propose a method for identifying a maliciou web page with time bombs effectively. The method performs a static analysis before dynamic one by HICHPs. The static analysis filters the suspicious web pages with the pattern assumed to be a time bomb attack. The dynamic analysis is able to detect time bombs effectively by extending the visitation time of the filtered pages. Moreover, we use a sequential visitation algorithm to determine whether each filtered page is malicious or not. The sequential algorithm is known to be excellent in an environment a ratio of malicious web pages is high. With a cost-based evaluation method, we show that our method is better than the conventional one in terms of detection accuracy and cost.

The remainder of this paper is organized as follows. In section 2, we describe related works including an evaluation method of client honeypots, time-bomb attacks, and the existing visitation algorithms. In section 3, we propose a new method to detect effectively the malicious web pages with time bombs and evaluate the performance of our method in section 4. We conclude and provide an outlook on our future research in section 5.

2. RELATED WORKS

We describe an evaluation method of HICHPs, a time-bomb attack, and the existing visitation algorithms.

2.1 Performance Evaluation of an HICHP

We need a method to evaluate and compare HICHPs in an environment in which they operate. Seifert presented a cost-based evaluation method: the *true positive cost curve* (*TPCC*) [7, 13]. It allows us to evaluate HICHPs against their primary purpose of identification of malicious web pages. It takes into account the unique characteristics of HICHPs, speed, detection accuracy, and resource cost. It provides a simple, cost-based mechanism to evaluate and compare HICHPs.

The TPCC represents the *cost* per malicious web page identified (c_{URL}) over the *base rate p*, the percentage of malicious web pages in a set of web pages. Cost per malicious web page identified indicates how efficiently HICHPs identify malicious web pages and can easily be compared. As more pages that are malicious exist in the sample, the cost is reduced, because an HICHP will naturally identify more pages that are malicious.

The t_{Algo} is the time in seconds required for an HICHP to inspect a sample of n web pages. If ability of two HICHPs is identical, the faster one will detect a greater number of malicious web pages while consuming fewer resources at a lower cost.

The HICHP consumes resources while inspecting potentially malicious web pages. These resources include hardware costs and costs associated with network and power consumption. c_r (*resource cost*) represents this resource consumption.

Detection accuracy also affects operating costs. The false negative (*FN*) expresses the failure of an HICHP to detect a malicious web page when it is inspecting one. A malicious web page can employ techniques that cause the HICHP to fail at detecting it. These techniques include time bombs, IP tracking, etc. These are suspected to exist or have been observed by various studies [7, 10-15]. The HICHP with the lower false negative rate will be able to identify more web pages that are malicious using the same resources, resulting in an overall lower cost per malicious web page identified.

$$c_{URL} = \frac{t_{Algo} c_r}{np(1 - FN)} \tag{1}$$

As a result, the cost associated with identifying a malicious web page c_{URL} is calculated as Eq. (1) [7, 13]. The time to inspect the sample n, t_{Algo}, is multiplied by the resource costs per time unit c_r. It is divided by the number of malicious web pages identified in the sample, which is the number of web pages in the sample multiplied by the base rate and the true positive rate of the client honeypot: $np(1 - FN)$.

Various factors influence the speed of HICHP to inspect a sample of n web pages [1, 7, 13]. How to detect state changes influences the speed. The network bandwidth and average size of the request/response influence the time to retrieve a web page t_i. Other factors are as follows: the time to render and display a web page t_d, the overhead of starting the client application t_s, and the overhead of resetting the client honeypot into a clean state after visiting a malicious web page t_r, which overall is impacted by the base rate p, and the classification delay t_w.

The classification delay t_w is a period needed to classify a web page more accurately. This is necessary because some time passes before many exploits trigger. This might be due to the nature of the exploit or intentionally introduced by the attacker to avoid detec-

Table 1. Factors for performance evaluation of client honeypots.

Symbol	Description
c_{URL}	The cost in US dollars to identify one malicious web page
p	The base rate of malicious web page in a sample of n web pages
t_{Algo}	The time in seconds required for a client honeypot to inspect a sample of n web pages
c_r	The resource costs, such as hardware costs, per period t in US dollars
T_q	The average time in seconds required to generate a list of n URLs to be inspected by a client honeypot
t_i	The average time in seconds required to retrieve a web page over the network
t_w	The classification delay in seconds introduced to give an exploit the opportunity to trigger
t_r	The average time in seconds required to reset a virtual machine
t_d	The average time in seconds required to render a web page
t_s	The average time in seconds required to start the client application
FN	The false negative rate of a client honeypot
K	The number of web pages in a buffer

tion. When HICHPs inspect web pages, the classification delay consumes most of the time. In addition, the time of creating a queue of URLs T_q is added to the duration to inspect web pages. It is usually constant [1, 7, 13].

The TPCC can be used to tune an HICHP in a specific operating environment. We use the TPCC in this way to tune an HICHP to identify malicious web pages that employ time bombs or IP tracking functionality.

2.2 Time Bombs

A malicious web page can employ evasion techniques that cause HICHP to fail at detecting it. Time bombs are one of the major evasion techniques. Time bombs are exploits embedded in malicious web pages that trigger only after a few seconds have passed. Time bombs are the primary reason why an HICHP waits t_w seconds after having retrieved a web page. The value was generally set to 25 seconds, because the majority of web pages seem to launch an attack during that period [7, 13]. If attackers change the trigger time of their time bombs, the ability of an HICHP to identify malicious web pages, and therefore the cost is affected.

According to Seifert's study, cost is the greatest if the classification delay t_w is increased to counter the evasion technique employed [7, 13]. Therefore, if 10% of malicious web pages trigger only after 35 seconds, it is best for the operator to ignore these 10% and continue to operate the HICHP unchanged with a classification delay t_w of 25 seconds. In this case, there is a serious problem that the client honeypot cannot detect the malicious web pages with time bombs.

To solve the above problem, we develop a better detection method with low cost and high detection accuracy for time bombs.

2.3 Web Page Visitation Algorithms

HICHP use a visitation algorithm to find malicious servers on a network. We consider four visitation algorithms: sequential, bulk, binary divide-and-conquer (*BDAC*), and logarithmic divide-and-conquer (*LDAC*) [1, 7, 13, 16]. Assume that there is a sample of n web pages to be inspected (Fig. 1). The sequential algorithm visits web pages sequentially. After each visitation, the HICHP waits for t_w seconds before checking for state changes on the system to classify the web pages as malicious or benign. If the web page was identified as malicious, the page is classified as malicious and the state of the system is reset before the next web page is visited. If the page is benign, the page is classified as benign and the next web page is visited.

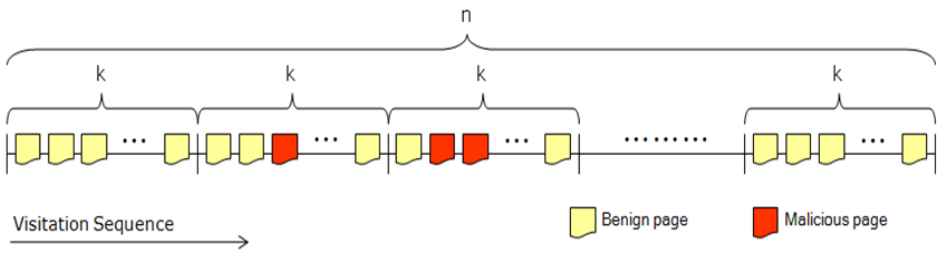

Fig. 1. Web pages visitation sequences.

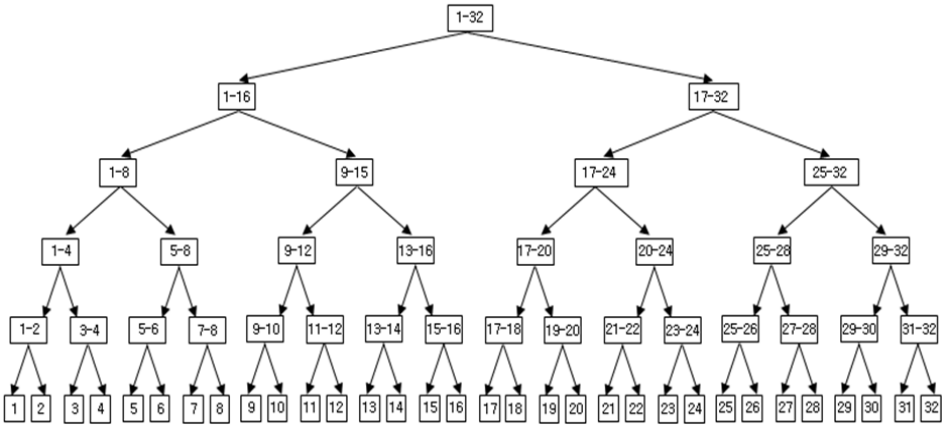

Fig. 2. Binary divide-and-conquer (BDAC) visitation.

The *BDAC* algorithm uses the divide-and-conquer design paradigm. It divides the sample of n web pages into buffers of size k and state changes are only checked after one classification delay per buffer. That is, the *BDAC* algorithm visits a buffer of k web pages at the same time, waits for t_w seconds, and makes a classification after the buffer has been inspected. At this phase, this algorithm is not capable of pinpointing which web page committed malicious activity. To determine which server attacked, the buffer of k web pages is divided into two portions and recursively visited until the malicious web page or pages are identified (see Fig. 2). Similar to the sequential algorithm, the state of the sys-

tem is reset if malicious web pages are encountered before continuing to visit the next set of web pages.

Similar to the *BDAC* algorithm, the bulk algorithm visits a buffer of k web pages at the same time, but it utilizes an enhanced state monitoring mechanism that allows it to pinpoint the specific web page that caused the unauthorized state change. As a result, the bulk algorithm does not need to visit web pages repeatedly. However, the load on the system increases by causing an overall slowdown compared to the *BDAC* algorithm's shared process. The reason is why the enhanced state monitoring mechanism requires each web page to be visited in its own process of the system using the bulk algorithm. Like the sequential and *BDAC* algorithms, the state of the system is reset before the next set of web pages is visited if malicious web pages were encountered.

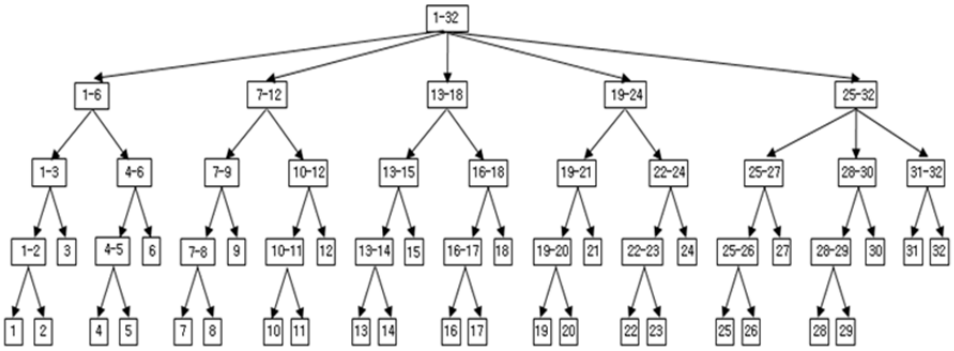

Fig. 3. Logarithmic divide-and-conquer (*LDAC*) visitation.

The LDAC algorithm is an enhanced version of the BDAC algorithm. If any malicious server attack is detected after interacting with a buffer of k web pages, we divide the buffer into several portions similarly to the BDAC algorithm to identify malicious servers. However, the algorithm divides the buffer in a logarithmic way to enhance the performance. Fig. 3 shows the search tree constructed by the LDAC algorithm when the bulk size is 32. Traversing down the tree, the algorithm divides each bulk of size x into $\lfloor \log_2 x \rfloor$ portions, and is recursively applied to each portion. In the example of Fig. 3, since the bulk size is 32 at the root, the bulk is divided into $\lfloor \log_2 32 \rfloor = 5$ portions with size 6. At the level 1 of the tree, the size of the divided portion is 6, so it is again divided into $\lfloor \log_2 6 \rfloor = 2$ portions with size 3. In some cases, like the portions at the level 2, a portion can be divided into different sizes of pieces. When the size of a piece is 3 or 2, we have $\lfloor \log_2 size \rfloor = 1$.

According to [16], the LDAC algorithm shows better performance than BDAC algorithm with respect to t_{Algo}.

3. EFFECTIVE DETECTION OF TIME BOMBS

In this section, we propose a new test method to detect malicious web pages with time bombs effectively. The method employs static analysis before dynamic analysis

with HICHP. By the static analysis, we can classify suspicious web pages into one with or without time bombs.

3.1 Detection Model for Time Bombs

As shown in Fig. 4, the proposed detection method consists of three stages. At the first stage, a dedicated system collects suspicious web pages for a certain period. The static analyzer classifies the collected web pages into the web pages suspected to be time-bomb attacks or ones suspected not to be time-bomb attacks at the second stage. Finally, the third stage performs in parallel the dynamic analysis using appropriate strategies to each classified group of pages and determines which page is malicious.

3.2 Classification of Web Pages with Time Bombs through Static Analysis

The second stage, static analysis, is intended to minimize the classification delay t_w for detecting malicious web pages that are time bomb suspects or not. The classification delay is introduced to detect malicious web pages including time-bombs in previous work too. The static analyzer scans the suspicious web pages and filters web pages with scripts or time operations, which delay the loading time or an attack, such as the JavaScript codes shown in Fig. 5.

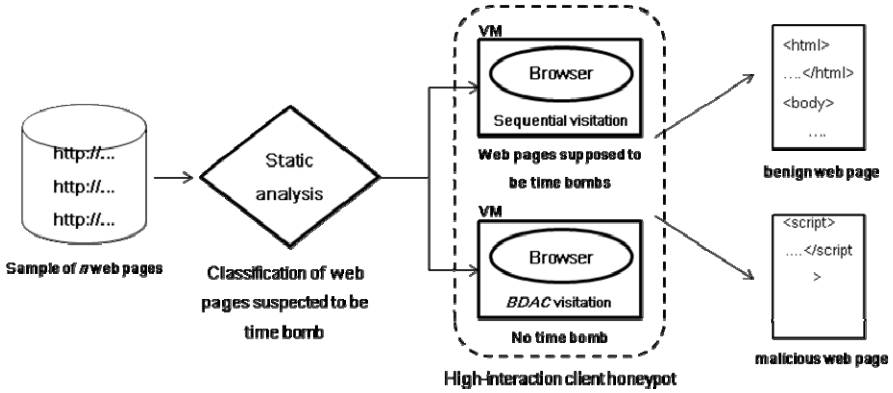

Fig. 4. Detection model for time bombs.

```
<script type="text/javascript">

function time_proc() {

setTimeout('attack()', 40000);

}
</script>
```

(A) Time parameter with a large value

```
<script type="text/javascript">

function time_proc() {

 while(i < 20) {
  setTimeout(2000);
  i++;
 }
 attack()
}
</script>
```

(B) Iteration of time operations with a small value

Fig. 5. Example of time-bomb script.

The function setTimeout() in this example is seldom used in ordinary web pages. Even when the function is used in some cases, if it is used without any hostile intent, the delay would not longer than 25-30 seconds. Using this information, the static analyzer classifies web pages calling setTimeout() with longer than 30 seconds (Fig. 5 (a)) or using repetition of short-period delays (Fig. 5 (b)) into time-bombs suspects.

3.3 A Strategy for Analyzing Web Pages with Time Bombs

To analyze web pages classified as time-bomb suspect, we perform dynamic analysis using a dedicated virtual machine of an HICHP. In case of web page suspicious to have time bombs, the web page may be malicious with high probability. Thus, we increase t_w for interacting with the web pages suspected to be time bomb attacks and use the sequential visitation algorithm. The sequential visitation algorithm shows comparable performance to the BDAC when the base rate $p = 0.1$ and outperforms the BDAC when $p > 0.1$ [1].

3.4 A Strategy for Analyzing Web Pages without Time Bombs

We use the *BDAC* or *LDAC* visitation algorithm for analyzing the web pages assumed not to have time bombs because two algorithms show better performance than the sequential algorithm when p is low [1]. The performance of the *BDAC* and *LDAC* algorithms is affected by the size buffer k. In [17], they measured the optimal k for various values of p. We use the optimal value shown in Table 2 in our experiments.

Table 2. Optimum values for buffer size k depending p (from [17]).

p	k	
	LDAC	*BDAC*
0.005	61	62
0.015	29	32
0.025	22	17
0.035	22	16
0.045	16	8
0.055	6	8
0.065	6	8
0.075	6	4
0.085	6	4
0.095	6	4

4. PERFORMANCE EVALUATION OF THE PROPOSED METHOD

To evaluate the performance of the proposed method, we simulated and analyzed the following four models from the previous studies [1, 7, 13].

(1) n = 5000, t_w = 25 seconds, 'percentage of time bombs = 0.
(2) n = 5000, t_w = 25 seconds, 'percentage of time bombs = 20% of p, trigger time of time bomb = 40 seconds.
(3) n = 5000, t_w = 40 seconds, 'percentage of time bombs = 20% of p, trigger time = 40 seconds.
(4) n = 5000, 'percentage of time bombs = 20% of p, trigger time = 40 seconds.
  (a) t_w = 25 seconds for one without time bombs.
  (b) t_w = 40 seconds for one with time bombs.

We set t_q = 0, t_s = 0.5, t_i = 4.3, t_d = 1.3, t_r = 5 seconds for all models as in [1]. We assume True-Positive (1 – *FN*) is 1 for malicious activities occurring within t_w and set p and k according to Table 2. For model 1~3, we use the *BDAC* and *LDAC* visitation algorithms to visit the sample of n web pages. For model 4, we use the *BDAC* and *LDAC* algorithms to visit web pages classified as without time bombs and the sequential algorithm to visit ones classified as with time bombs. For model 1, 2, and 4 (a), we set t_w = 25 seconds because malicious web servers without time bombs is likely to do some harmful acts within 25 seconds. With this t_w, an HICHP cannot detect malicious web pages with time bombs in model 2. We set t_w = 40 seconds for model 3 and 4.b in order to detect time bombs triggering after 40 seconds.

We simulated each models 1000 times and calculate average t_{Algo} and TPCC. Distribution of malicious web pages affects performance of HICHP and eventually t_{Algo}. Therefore, locations of malicious web pages are randomly determined. TPCC incorporates t_{Algo} and other factors such as detection accuracy, resource cost, and a proportion of malicious web servers. When calculating TPCC, we assumed resource cost c_r 0.125 US dollars per hour [7, 13, 18].

4.1 Total Inspection Times

The performance of four models in terms of t_{Algo} is shown in Figs. 6 and 7. Total inspection time of model 3 is longest among the models. The reason is that model 3 lengthens t_w from 25 seconds to 40 seconds for all n pages, despite that only 20% of malicious pages may have time bombs.

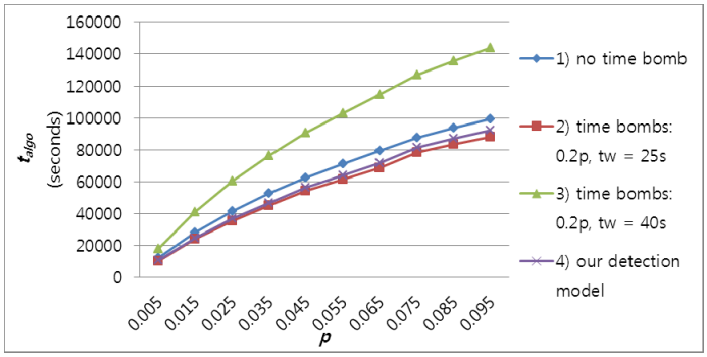

Fig. 6. Simulation results in terms of t_{Algo} (*BDAC*).

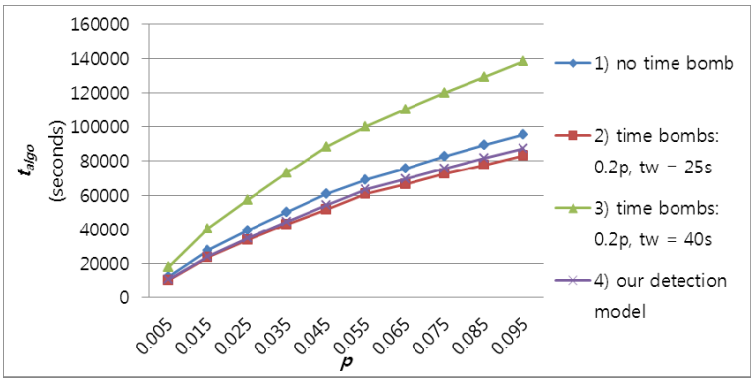

Fig. 7. Simulation results in terms of t_{Algo} (*LDAC*).

Note that model 2 has the shortest t_{Algo} because 20% of malicious web pages are not detected. In other words, HICHP in model 2 did not detect pages with time bombs because its t_w is 25 seconds and time bombs need 40 seconds to trigger. As a result, the number of reverts is minimum among the four models and therefore the time associated with t_r decreases. However, true positive rate of model 2 decreases to 0.8. Unfortunately, the t_{Algo} does not reflect this effect of *FN*.

In model 4, an HICHP uses different visitation algorithms for each two groups of suspicious web pages (see Fig. 4). For the web pages suspected to be time bombs, it uses the sequential visitation algorithm and consumes '$t_s + t_i + t_d + t_w + t_r$'. For the web pages assumed no time bombs, it uses the *BDAC* or *LDAC* visitation algorithm. The sequential and BDAC or LDAC algorithms are concurrently executed. Our model shows better total inspection time than model 1 and 3 because of appropriate visitation algorithms and classification delays. The t_{Algo} is reduced by 37.8% (BDAC), 38.1% (LDAC) compared to model 3. The detection accuracy (1 – *FN*) is 1, which is the same as model 3. Furthermore, an HICHP using LDAC algorithm performs better by 3.7% on average with respect to t_{Algo} than one using BDAC.

Performance comparison in terms of t_{Algo} does not consider detection accuracy and consumption of other additional system resources. Thus, we must use more effective performance evaluation criterion, c_{URL}.

4.2 True Positive Cost Curve (TPCC)

Figs. 8 and 9 compares the performance of four models in terms of TPCC cost model. Note the difference between Figs. 6-9. In Figs. 6 and 7, model 2 shows better performance than model 1, but in Figs. 8 and 9, vice versa. This is because TPCC incorporates detection accuracy and resource cost as performance factor, but t_{Algo} does not. In model 2, detection accuracy (1 – *FN*) is 0.8 because model 2 cannot identify malicious web pages with time-bombs (20% of total malicious web pages). Model 1 assumes no time bomb and so (1 – *FN*) is 1. Model 1 performs better than model 2 in terms of detection accuracy and overall costs.

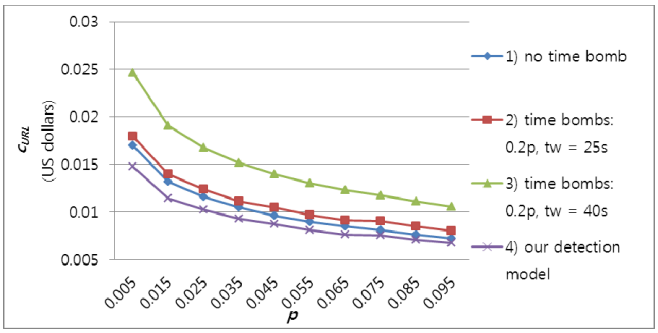

Fig. 8. Simulation results with TPCC cost model (*BDAC*).

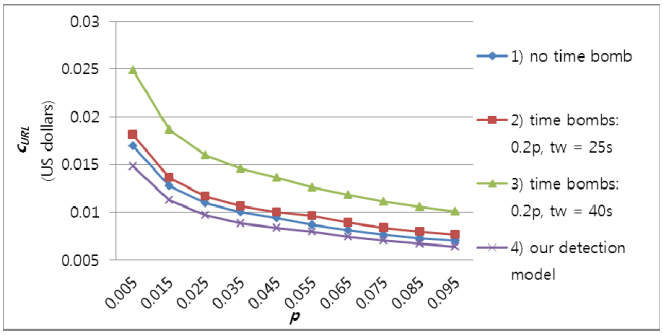

Fig. 9. Simulation results with TPCC cost model (*LDAC*).

Our model shows the best performance among the simulation models. The model has introduced static analysis before dynamic analysis. Moreover, the model uses a separate visitation algorithm and classification delay to detect web pages with time bombs effectively. This approach raises the value $(1 - FN)$ to 1 and lower overhead of t_{Algo}. This model improves TPCC of model 3 by 37.8% (*BDAC*), 38.1% (*LDAC*) on average, and that of model 2 by 16.9% and 16.7%. Furthermore, an HICHP using LDAC algorithm performs better by 3.7% on average with respect to c_{URL} than one using BDAC.

5. CONCLUSIONS

A high-interaction client honeypots (HICHP) is a useful device to detect drive-by-download attacks. An HICHP, however, has limitations such that they need long time to identify the malicious web pages and tend to miss some attacks such as time bombs. To address these limitations, we presented a new method to detect malicious web pages with time bombs effectively. Our detection method carries out static analysis before applying dynamic analysis. The static analysis filters suspicious web pages with patterns assumed time-bomb attacks. By extending the waiting time in visitation on only the filtered pages in the dynamic analysis, we have enhanced the detection accuracy of time bombs. Moreover, we used the sequential visitation algorithm for the filtered pages, known to be superior in case the ratio of malicious web pages is high. With a cost-based performance

evaluation method, the proposed method reduced costs of identifying the malicious web pages using HICHPs.

We plan to investigate additional means to detect malicious web pages with IP tracking and client honeypot detection effectively. IP tracking is a technique to launch an attack just once. Repeated interaction with the same page would make the examiner consider the web server hiding the attack benign falsely, because the malicious web server serves a false benign page. Therefore, an HICHP would fail to detect the malicious nature of the web server. Client honeypot detection is another evasion technique a malicious web page can use to identify an HICHP and selectively serve a benign web page instead of its usual malicious web page.

REFERENCES

1. C. Seifert, I. Welch, and P. Komisarczuk, "Application of divide-and-conquer algorithm paradigm to improve the detection speed of high interaction client honeypots," in *Proceedings of ACM Symposium on Applied Computing*, 2008, pp. 1426-1432.
2. M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda, "Defending browsers against drive-by downloads: Mitigating heap spraying code injection attacks," http://www.iseclab.org/papers/driveby.pdf, 2008.
3. B. Endicott-popovsky, J. Narvaez, C. Seifert, D. A. Frincke, L. R. O'Neil, and C. Aval, "Use of deception to improve client honeypot detection of drive-by-download attacks," in *Proceedings of the 5th International Conference on Foundations of Augmented Cognition*, 2009, pp. 138-147.
4. N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. "The ghost in the browser: Analysis of web-based malware," in *Proceedings of USENIX Workshop on Hot Topics in Understanding Botnets*, 2007.
5. M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," in *Proceedings of the 19th international conference on World Wide Web*, 2010, pp. 281-290.
6. J. Narvaez, B. Endicott-Popovsky, C. Seifert, C. U. Aval, and D. A. Frincke "Drive-by-downloads," in *Proceedings of Hawaii International Conference on System Sciences*, 2010, pp. 1-10.
7. C. Seifert, P. Komisarczuk, and I. Welch, "True positive cost curve: A cost-based evaluation method for high-interaction client honeypots," in *Proceedings of the 3rd International Conference on Emerging Security Information*, *Systems and Technologies*, 2009, pp. 63-69.
8. R. Steenson and C. Seifert, "Capture – Honeypot client," http://www.nz-honeynet.org, 2007.
9. Y. M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King, "Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities," in *Proceedings of the 13th Annual Network and Distributed System Security Symposium*, 2006.
10. K. Wang, "HoneyClient," http://www.honeyclient.org/trac, 2007.
11. C. Seifert, "Know your enemy: Malicious web servers," The Honeynet Project, KYE paper, http://www.honeynet.org, 2007.

12. A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy, "A crawler-based study of spyware on the web," in *Proceedings of the 13th Annual Network and Distributed System Security Symposium*, 2006.
13. C. Seifert, "Cost-effective detection of drive-by-download attacks with hybrid client honeypots," Ph.D. Thesis, Computer Science Department, Victoria University of Wellington, 2010.
14. J. Zhuge, T. Holz, C. Song, J. Guo, X. Han, and W. Zou, "Studying malicious websites and the underground economy on the Chinese web," *Managing Information Risk and the Economics of Security*, 2009, pp. 225-244.
15. N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, "All your iFRAMEs point to us," in *Proceedings of the 17th Conference on Security Symposium*, 2008, pp. 1-15.
16. H. Kim, D. Kim, S. Cho, M. Park, and M. Park, "An efficient visitation algorithm to improve the detection speed of high-interaction client honeypots," in *Proceedings of ACM Research in Applied Computation Symposium*, 2011, pp. 266-271.
17. D. Kim, H. Kim, M. Park, and S. Cho, "An improvement of visitation algorithm for analyzing suspicious web-pages using a client honeypot," in *Proceedings of the 38th KIISE Fall Conference*, Vol. 38, 2011, pp. 47-50.
18. S. Shankar, "Amazon elastic compute cloud (EC2) running Microsoft Windows server and SQL server," Amazon.com, 2008.

Hong-Geun Kim received the B.E., the M.E. and the Ph.D. in Computer Engineering from Seoul National University in 1985, 1987 and 1994 respectively. He works for Korea Internet and Security Agency from 1996. His current research interests include information security, critical infrastructure protection, and system vulnerability analysis.

Dongjin Kim received the B.E. and M.E. degree in Computer Science from Dankook University in 2009 and 2011, respectively. He is a Ph.D. student in Dankook University, Korea. His current research interests include computer security and system software.

Seong-Je Cho received the B.E., the M.E. and the Ph.D. in Computer Engineering from Seoul National University in 1989, 1991 and 1996 respectively. He was a visiting scholar at Department of EECS, University of California, Irvine, USA in 2001, and at Department of Electrical and Computer Engineering, University of Cincinnati, USA in 2009 respectively. He is a Professor in Department of Computer Science and Software Science, Dankook University, Korea from 1997. His current research interests include computer security, operating systems, software protection, real-time scheduling, and embedded software.

Moonju Park received the B.E. in Naval Architecture and Ocean Engineering, and the M.E. and the Ph.D. in Computer Engineering from Seoul National University in 1995, 1996, and 2002 respectively. He was with LG Electronics as a chief research engineer from 2002 to 2006, and with IBM Ubiquitous Computing Laboratory as an advisory software engineer from 2006 to 2007. He is an Assistant Professor in Department of Computer Science and Engineering, University of Incheon from 2007. His current research interests include operating systems, real-time systems, and embedded systems.

Minkyu Park received the B.E. and M.E. degree in Computer Engineering from Seoul National University in 1991 and 1993, respectively. He received Ph.D. degree in Computer Engineering from Seoul National University in 2005. He is now an Associate Professor in Konkuk University, Korea. His research interests include operating systems, real-time scheduling, embedded software, computer system security, and HCI.