

# Identification of Malicious Web Pages with Static Heuristics

Christian Seifert, Ian Welch, Peter Komisarczuk  
Victoria University of Wellington  
P. O. Box 600  
Wellington 6140, New Zealand  
Email: {cseifert,ian,peterk}@mcs.vuw.ac.nz

**Abstract**—Malicious web pages that launch client-side attacks on web browsers have become an increasing problem in recent years. High- interaction client honeypots are security devices that can detect these malicious web pages on a network. However, high-interaction client honeypots are both resource-intensive and known to miss attacks. This paper presents a novel classification method for detecting malicious web pages that involves inspecting the underlying static attributes of the initial HTTP response and HTML code. Because malicious web pages import exploits from remote resources and hide exploit code, static attributes characterizing these actions can be used to identify a majority of malicious web pages. Combining high-interaction client honeypots and this new classification method into a hybrid system leads to significant performance improvements.

**Keywords:** Security, Client Honeypots, Drive-by-downloads, Intrusion Detection

## I. INTRODUCTION

As traditional attack vectors are barred by defenses, malicious users seek out new, unprotected paths of attack. One of these is the client-side attack, which targets client applications. As the client accesses a malicious server, the server delivers the attack to the client as part of its response. A web server that launches so-called drive-by-download attacks on web browsers is a common example. As the web browser requests content from a web server, the server returns an exploit embedded in the web pages that allows the server to gain complete control of the client system. Traditional defenses are ineffective against these new threats because they cannot deal with zero-day attacks.

To develop new defense mechanisms, it is necessary to study malicious servers. Client honeypots represent an emerging technology for detecting malicious servers on a network. So-called high-interaction client honeypots use a dedicated, vulnerable computer system to interact with potentially malicious servers [1]–[4]. As responses from these servers are consumed by the client honeypot, it monitors the system for any unauthorized state changes that indicate a successful attack (authorized changes are defined as changes to the system state associated with background system processes, such as the creation of log files).

Deployers of high-interaction client honeypots face some challenges. First, the honeypots require considerable resources, in that a dedicated system – a physical machine or a virtualized

environment – is necessary for them to function, which has a direct negative impact on the performance of these systems.

Second, the honeypots have a tendency to fail at identifying malicious web pages that require client-side input either in the form of user interaction or system events. Underlying the client honeypot approach is an assumption that simply accessing a web page will trigger an attack. However, if, for instance, user activity is needed for the exploit to trigger, then the attack will not happen and there will be no malicious state changes for the honeypot mechanism to detect.

This paper presents a novel classification method that identifies malicious web pages based on statically analyzing the initial HTTP response denoted by the URL. This classification method does not require the entire web page to be retrieved nor does it require a dedicated environment that client honeypots do. Application of this method leads to significant performance improvements.

## II. MALICIOUS WEB PAGES

The classification method utilizes common elements of malicious web pages. In this section, a description of these elements is presented. There are three core elements contained on a malicious web page: the exploit itself, the delivery mechanism that brings the browser to the exploit, and mechanisms to hide the exploit or the delivery mechanism from detection.

*Exploit:* The exploit is the central part of the malicious web page and the core element that must be present for a web page to be considered malicious. The exploit is the attack code that targets a specific vulnerability of the browser, its plug-ins, or underlying operating system. It is specific to the vulnerability it is targeting and can make use of a variety of techniques. Less obvious exploits have been found in images. The most common exploits target vulnerabilities in scriptable ActiveX components. For example, a popular web exploitation kit, called IcePack, primarily targets vulnerabilities in ActiveX components with 75% of the supported exploits being related to ActiveX components. [5].

*Exploit Delivery Mechanism:* While the exploit is the central part of a malicious web page, the web page might not contain the exploit directly. Exploits might be "imported" from a central exploit server to provide a mechanism for a reusable

and modular design. There are two types of imports: direct includes of resources and redirects.

Direct includes of resources is a feature naturally supported by HTML. The `src` attribute, which exists on several HTML tags, is able to import resources from local and remote web servers. Even if a tag does not support the `src` attribute, scripts are able to effectively source any HTML element remotely because scripts can arbitrarily modify an HTML page via the document object model (DOM) and so import whole HTML elements from remote sources.

Alternatively, instead of importing an exploit, an attacker might instruct the browser to fetch a new page from a new location altogether. Redirects can be used to instruct the browser to perform this action. There are server and client-side redirects. Server-side redirects instruct the browser to fetch a page from a different location via the HTTP response code (3xx) and the location header field. Client-side redirects instruct the browser to fetch a page from a new location via HTML or JavaScript. Client-side redirects trigger after a HTML page is loaded.

*Obfuscation:* Hiding the exploit or the exploit delivery mechanism through obfuscation is a common mechanism used by malicious web pages. Script code is provided in obfuscated form alongside a custom de-obfuscation function, which can convert the obfuscated code snippet into its clear form. Once converted, the code can be executed. Signature-based detection algorithms are unable to analyze the obfuscated JavaScript.

All the elements described above have their legitimate purposes, but attackers also use them. As a result, a web page cannot be classified as malicious if one merely observes these elements contained in an HTTP response. A more able mechanism is needed, which is described next.

### III. METHODOLOGY

In this section, a description on how characteristics of HTTP responses and the contained HTML page can be taken advantage of to classify web pages is presented. In October and November 2007, several thousand potentially malicious web pages were inspected using the high-interaction client honeypot Capture-HPC v2.1 [4] configured with a clean installation of Windows XP SP2 and running the Internet Explorer 6 SP2 web browser. As the client honeypot inspected potentially malicious web pages, the network traffic was recorded. Web pages identified as malicious were marked as such and the corresponding HTTP response was saved.

Similarly, the HTTP response was recorded when interacting with benign web pages using an identically configured system. To collect benign web pages, English 5 N-grams (an N-gram is a selection of  $n$  words from a string) were randomly selected from the corpus of web pages linked by the DMOZ Open Directory Project [6]. Those N-grams were issued to the Yahoo! search engine [7], and the first 50 URLs on the results page were visited. The web pages that were not classified as malicious by the client honeypot were marked as benign and the corresponding HTTP response was saved.

Once the HTTP responses were collected, attributes of the HTTP response and embedded HTML code that aim at capturing the characteristics of a malicious web page as described in section II were extracted. The attributes include characteristics that capture indications of potential exploits, exploit delivery mechanism, and obfuscation attempts. The attributes extracted are described in Table I.

All the extracted attributes were fed into the J4.8 decision tree learning algorithm implementation of the Waikato University's Weka Machine Learning Library [8]. The predictive value of the generated classifier was evaluated on new web pages that were not used in the learning phase. First, a sample of 61,000 URLs randomly selected using the method described above was used. All URLs were classified with the classification method presented in this paper as well as with the high-interaction client honeypot Capture-HPC v2.1. The false negative and positive rates were determined. In addition, the classification method was evaluated with a set of 500,000 URLs provided by HauteSecure, a leader in web-based threat protection [9]. These URLs were already analyzed for drive-by-download attacks by HauteSecure's technology. Evaluation with these URLs reduces the risk of potential bias introduced by the high-interaction client honeypot technology and sampling method.

The sample of 61,000 URLs were also used to assess the performance gain of the presented classification method using an Amazon EC2 instance with 1.7GB of RAM, which is equivalent to a CPU capacity of a 1.0-1.2 GHz 2007 Xeon processor, on a 250Mbps connection. The duration of classifying the sample with the presented classification method was compared to the high-interaction client honeypot on another test machine with similar specifications (assuming a divide-and-conquer algorithm [10], a service time of approximately 10 seconds and the ability to run three client honeypot instances).

### IV. RESULTS

For this study, 5,678 instances of malicious and 16,006 instances of benign web pages were input into the machine learning algorithm. The generated classifier was used to classify a new sample. Of the 61,000 URLs included in the sample, 3,590 URLs were marked as malicious by the presented classification method. Inspection by a high-interaction client honeypot determined the false positive and false negative rates of the presented classification method on the sample. Seven malicious URLs were detected in the 3,590 URLs; six malicious URLs were detected in the remaining URLs marked as benign. This amounts to a false positive rate of 5.88% and a false negative rate of 46.15% for the new classification method. The evaluation against the 500,000 URLs provided by HauteSecure resulted in similar metrics.

An abbreviated version of the decision tree being generated is shown in Fig. IV. The decision tree can be used to classify unseen pages. Starting from the root node, the value of the attribute shown on the tree node is evaluated which leads to a specific child node. Attributes are recursively evaluated until a

TABLE I  
EXTRACTED ATTRIBUTES

Category	Attributes	Description
Exploit	Plug-ins	Count of the number of applet and object tags.
	Script Tags	Count of script tags.
	XML Processing Instructions	Count of XML processing instructions. Includes special XML processing instructions, such as VML.
Exploit Delivery Mechanism	Frames	Count of frames and iFrames including information about the source.
	Redirects	Indications of redirects. Includes response code, meta-refresh tags, and JavaScript code.
	Script Tags	Count of script tags including information about the source.
Hiding	Script Obfuscation	Functions and elements that indicate script obfuscation, such as encoded string values, decoding functions, etc.
	Frames	Information about the visibility and size of iFrames.

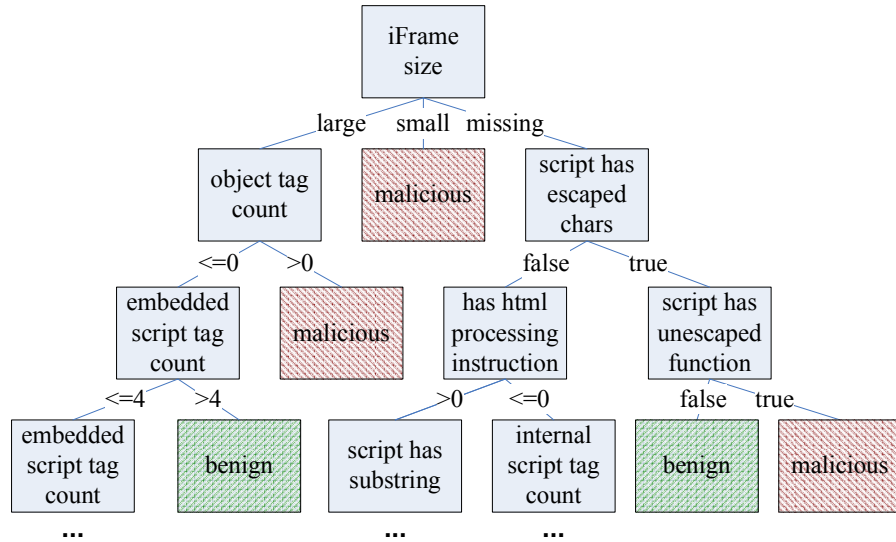


Fig. 1. Decision Tree

classification node that specifies whether a page is malicious or benign is reached.

The decision tree shows that the existence of iFrames is a good classifier on the malicious nature of a web page. Existence of a small iFrame on the web page causes a malicious classification. When the iFrame is missing, additional attributes are evaluated. The presence of escaped characters in JavaScript code and the existence of the unescape function is another good classifier of a malicious page. These are attributes that directly link to the exploit delivery mechanism and obfuscation.

The performance classifying a HTTP response with this classification method is greatly increased over the traditional high-interaction client honeypot. The test machine was able to retrieve and classify 61,000 URLs in 49 minutes. This is equivalent to 1.79 million web pages a day (approximately a service time of 0.05sec per URL). On the contrary, the high-interaction client honeypot classified 996 URLs in the

same period of 49 minutes; this is equivalent to approximately 29,270 URLs a day (approximately a service time of 2.95sec per URL). The presented method is able to inspect 61 times as many URLs as with high interaction client honeypots in the same period.

The presented classification method shows better performance over high-interaction client honeypots. At the same time, the classification method produces false positives and misses attacks. The usefulness of the approach becomes apparent if combined in a hybrid system, which is presented next.

## V. HYBRID SYSTEM

In order for the presented classification method to be useful, it need to be used in conjunction with a high-interaction client honeypot. A pipeline view of the system is shown in Fig. 2. At the start of the pipeline is the static heuristics approach detailed in this paper. It initially retrieves and classifies the web pages denoted by the input URLs. Because the false

positive rate is high, all URLs that have been classified as malicious are forwarded to the second part of the hybrid system. The high-interaction client honeypot retrieves the page once again and make a final classification. Since the high-interaction client honeypot has a false positive rate of zero, the false positives will be filtered out. The overall detection speed and accuracy of the hybrid system are presented next. A detailed description of the hybrid client honeypot system can be found in a technical report [11].

#### A. Detection Speed

The detection speed of a hybrid system is greatly influenced by the performance of the individual components. The main factor that influences speed is the number of web pages classified as being malicious by the static heuristics method and forwarded to the high-interaction client honeypot for a second inspection. Since the high-interaction client honeypot is significantly slower than the static heuristics method, a second inspection will have a great impact on performance of the hybrid system.

The number of malicious classifications of the presented classification method can be calculated as follows: If  $p_m$ , the percentage of malicious pages in the set  $N$  that is being inspected, is known, the number of malicious pages  $N_M$  and benign pages  $N_B$  can be determined with Equations 1 and 2. Taking into account the false positive and negative rate  $FP_C$ ,  $FN_C$  of the presented classification method, the number of malicious classifications  $alerts_C$  can be calculated with Equation 3. For example, if  $p_m = 0.05\%$  and the detection accuracy of the presented classification method as predicted by the evaluation on the test set is assumed, inspection of 100,000 URLs results in 5904 malicious classifications, or according to Equation 4 5.90% if expressed in percentage  $P_{alertC}$ .

$$N_M = p_m * N \quad (1)$$

$$N_B = N - N_M \quad (2)$$

$$alerts_C = FP_C N_B + (1 - FN_C) N_M \quad (3)$$

$$P_{alertC} = \frac{alerts_C}{N} \quad (4)$$

Adding the time to inspect set  $N$  with the presented classification method and the time to inspect  $alerts_C$  with the high-interaction client honeypot yields the overall time of the hybrid system  $T_{Hy}$  to inspect  $N$ . If a service time  $T_C$  for the presented classification method and  $T_H$  for the high-interaction client honeypot is assumed, the total time for the hybrid system is calculated according to Equation 5. The implementation of the hybrid system, with a service time  $T_C$  of 0.05 sec. and  $T_H$  of 2.95 sec., classified the sample of 61,000 URLs with 3590 malicious classifications in 227 minutes. This compares to 2999 minutes for an inspection of the URLs with a high-interaction client honeypot alone. The hybrid system

would be able to inspect approximately 13 times as many URLs as a high-interaction client honeypot in the same period.

$$T_{Hy} = T_C N + alerts_C T_H \quad (5)$$

#### B. Detection Accuracy

Inspection with a hybrid system allows for inspection of a much larger set of URLs compared to a high-interaction client honeypot system. Because the high-interaction client honeypot has a false positive rate of zero, the hybrid system also has this false positive rate. However, some attacks are missed. If only the  $alert_C$  are forwarded, the false negative rate of the hybrid system  $FN_{Hy}$  is identical to the false negative rate of the presented classification method  $FN_C$  union the false negative rate of the high-interaction client honeypot  $FN_H$  shown in Equation 6.

$$\begin{aligned} FN_{Hy} &= FN_C \cup FN_H \\ &= FN_C + FN_H - (FN_C, FN_H) \end{aligned} \quad (6)$$

For instance, if a set of URLs are inspected with the presented classification method and a high-interaction client honeypot that has a false negative rate of 10.0% and both methods fail detect 5.00% of all attacks, the resulting false negative rate of the hybrid system will be 51.15% (46.15% + 10.0% - 5.00%). If a false negative rate of zero is assumed for the high-interaction client honeypot, the false negative rate of the hybrid system falls to the false negative rate of the presented classification method, so 46.15%.

The sample of 61,000 pseudo-random URLs inspected with the hybrid system resulted in a total of seven malicious URLs. Six URLs were missed by the hybrid system resulting in a false negative rate of 46.15%. Since the false negative rate of the high-interaction client honeypot component could not be determined, the false negative rate of the high-interaction client honeypot needs to be added in order to determine the false negative rate of the hybrid system.

A false negative rate of 46.15% might not be acceptable for certain scenarios. In a hybrid system, the false negative rate can be improved if a portion of URLs for which no alert was initially raised, are randomly forwarded to the high-interaction client honeypot for classification. This percentage  $P_{noAlertL}$  URLs that have been missed by the presented classification method now have an opportunity to be classified correctly by the high-interaction client honeypot.  $FN_{Hy}$  falls.

$$\begin{aligned} FN_{Hy} &= FN_C + FN_H - (FN_C, FN_H) \\ &\quad - (P_{noAlertL} FN_C) \end{aligned} \quad (7)$$

If an additional 20% of URLs for which no alert was raised is forwarded to the high-interaction client honeypot, the total false negative falls from 46.15% to approximately 36.92% according to Equation 7. Forwarding additional alerts to the high-interaction client honeypot, however, has direct impact on performance. The time to classify the URLs rises from

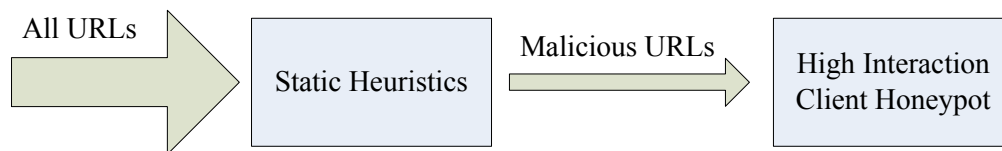


Fig. 2. Hybrid System

227 minutes to approximately 792 minutes as an additional 11,482 malicious classifications need to be inspected by the expensive high interaction client honeypot.

The trade-off between detection accuracy and performance gain of the hybrid client honeypot exists. As  $P_{noAlertL}$  value is increased, the false negative rate (excluding  $FN_H$ ) is decreased, but at the same time the performance gain decreases.

## VI. RELATED WORK

This section reviews related work in the field of client honeypots and the detection of malicious web pages and compares other detection methods to the classification method presented here.

HoneyC is a client honeypot that uses a static detection method to classify malicious web servers [12], [13]. It uses signatures of known attacks to directly inspect the content of the response and to make a classification. HoneyC therefore needs prior knowledge of the attacks to detect them and cannot correctly classify unknown attacks. That stands in contrast to the new classification method presented in this paper, which is able to detect both known and unknown attacks.

Caffeine Monkey engine is a tool that is targeted at collection, detection and analysis of malicious JavaScript [14]. It uses a combination of static and dynamic analysis techniques to classify JavaScript. Instead of looking at absolute numbers of JavaScript element, ratios of function calls are taken into account. Obfuscation that might hinder such an analysis is addressed by automatically deobfuscating the JavaScript code with an instrumented JavaScript engine. The presented classification method differs in that only absolute counts of JavaScript elements and indications of JavaScript obfuscation are taken into account. No attempt is made to deobfuscate. As such, the presented classification method is able to make a quick assessment, but at the same time is likely to miss attacks that would have been correctly identified with more thorough analysis technique as applied by Caffeine Monkey.

Moshchuk et al. identified malicious web servers by classifying executable downloads with antivirus engines [3]. However, this method identified malicious web servers that host malicious content, whereas high-interaction client honeypots and the classification method presented in this paper are both able to identify malicious web servers that launch drive-by-download attacks.

Provos et al. analyzed a large number of malicious web servers using client honeypots and presented four prevalent mechanisms used to inject malicious content onto popular web sites [15]. The researchers applied heuristics to reduce an extensive set of web pages to a smaller set of suspicious

web pages. Metrics on the detection speed and accuracy of this method were presented in later work [16]: One billion pages are classified daily and the error estimates using a five-fold cross validation predicts a false positive rate of  $10^{-3}$  and false negative rate of 0.4 for a chosen threshold. Details of the classification method have not been disclosed making a direct comparison the presented classification method infeasible.

Besides research that is targeted at detection of malicious web pages, numerous efforts are underway to directly protect the client application [17]–[20]. The work presented in this paper focuses on detection.

The hybrid system presented utilizes a high-interaction client honeypot to eliminate false positives. Anagnostakis et al. demonstrates the concept of a system that utilizes honeypots to reduce the false positive rate. Anomaly-based intrusion detection is used for an initial assessment of the data. Once suspicious data with a large false positive rate is identified, it is forwarded to a honeypot for a final assessment. Sidiroglou et al. present a hybrid system to protect against malicious email attachments [21]. Suspicious email attachments are opened in an instrumented virtual machine. If dangerous actions, such as writing to the Windows Registry are detected, the mail is deemed malicious and quarantined.

Pouget et al. suggest to combine honeypot technology to increase scalability. A so-called high-interaction honeypot could be combined with more light weight, so called low-interaction, honeypots to increase coverage, scalability while at the same time keeping the favorable detection properties of high-interaction honeypots [22]. The work presented in this paper takes this approach into the area of client honeypots.

## VII. CONCLUSION

This paper presents a simple yet effective classification method for detecting malicious web pages that requires assessing attributes of the initial HTTP response. The evaluation of the classification method on a sample of 61,000 URLs resulted in a false positive rate of 5.88% and false negative rate of 46.15%. Evaluation with 500,000 URLs provided by HauteSecure resulted in similar metrics. While these numbers initially appear not well suited for a client honeypot system, one needs to take into account the low base-rate of malicious web pages [23]. These error rates combined with a low base rate allow the classification method presented in this paper to discard the majority of benign web pages and identify the majority of malicious web pages that a high-interaction client honeypot would be able to identify. As a result, one is able to inspect more web pages and identify significantly more malicious web pages with identical resources. The extracted

knowledge in form of the decision tree captures two common concepts adopted by malicious web pages: modular design and obfuscation. Because this classification method is based on *common* attributes of malicious pages, attackers could structure malicious pages to evade detection by this method. They merely need to make use of uncommon features. For instance, an exploit that is not imported via an iFrame and does not make use of JavaScript could evade detection. The sample used in this study, for instance, contained a few such malicious web pages that contained such an exploit: the VML exploit [24]. However, if attackers commonly adopt such an approach, the knowledge acquisition, if reapplied would potentially adjust itself to capture these common attributes. The necessity and required frequency of new knowledge acquisition will be explored as part of future work.

Speed is the greatest advantage of the presented method. There are two reasons for the speed increase. First, the presented classification method does not require all components of a web page to be downloaded nor is support for rich functionality, such as JavaScript, required before an assessment can be made. Second, the presented classification method can be implemented as a threaded stand-alone application. This stands in contrast to the requirements of a high-interaction client honeypot that requires several seconds to classify a page whereas the presented method only requires a fraction of that time for classification.

With the performance gain comes a false positive rate that produces many false positive alerts in a real world setting. A hybrid system was presented in which the classification method and a high-interaction client honeypot are combined. In this hybrid system, the classification method initially classifies the URLs at high speed significantly reducing the set of URLs. The malicious URLs are forwarded to a high-interaction client honeypot for a final classification to eliminate the false positives that might have been generated.

## VIII. FUTURE WORK

As attackers change their techniques, the acquired knowledge needs to be updated periodically to ensure that the majority of malicious pages are detected with the presented classification method. In a hybrid system, a self-learning system could be devised in which classification of the high-interaction client honeypot would be fed back to the algorithm to update the underlying knowledge model. The effects of such a feedback loop on the detection accuracy need to be explored.

## ACKNOWLEDGMENT

We thank HauteSecure for providing us with a set of fully classified URLs that permitted evaluation of the classification method on an independent data set.

## REFERENCES

- [1] K. Wang, "Honeyclient," vol. 2007, no. 1/2/2007, 2005, p. available from <http://www.honeyclient.org/trac>; accessed on 2 January 2007.
- [2] 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 *13th Annual Network and Distributed System Security Symposium*. San Diego: Internet Society, 2006.
- [3] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy, "A crawler-based study of spyware on the web," in *13th Annual Network and Distributed System Security Symposium*. San Diego: The Internet Society, 2006.
- [4] C. Seifert and R. Steenson, "Capture - honeypot client," 2006, pp. available from <https://www.client--honeynet.org/capture.html>; accessed on 22 September 2007.
- [5] C. Seifert, "Know your enemy: Behind the scenes of malicious web servers." The HoneyNet Project, 2007, p. available from <http://www.honeynet.org/papers/wek>; accessed on 7 November 2007.
- [6] Netscape Communications Corporation, "DMOZ open directory project," 1998, p. available from <http://www.dmoz.org>; accessed on 13 October 2007.
- [7] D. Filo and J. Wang, "Yahoo! search engine," 1994, p. available from <http://www.yahoo.com/>; accessed on 20 November 2007.
- [8] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.
- [9] HauteSecure, "Home page," 2006, p. available from <http://www.hautesecure.com>; accessed on 13 September 2008.
- [10] C. Seifert, P. Komisarczuk, and I. Welch, "Application of divide-and-conquer algorithm paradigm to improve the detection speed of high interaction client honeypots," in *23rd Annual ACM Symposium on Applied Computing*. Ceara: ACM, 2008.
- [11] C. Seifert, "Improving detection accuracy and speed with hybrid client honeypots, phd proposal," vol. 2007. Victoria University of Wellington, 2007, pp. available from [http://www.mcs.vuw.ac.nz/~cseifert/publications/Cseifert\\_phd\\_proposal--Hybrid\\_Client\\_Honeypots.pdf](http://www.mcs.vuw.ac.nz/~cseifert/publications/Cseifert_phd_proposal--Hybrid_Client_Honeypots.pdf); accessed on 22 March 2007.
- [12] C. Seifert, I. Welch, and P. Komisarczuk, "Honey - the low-interaction client honeypot," in *NZCSRCS*, Hamilton, 2007.
- [13] N. Provos and T. Holz, "Client honeypots," in *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Upper Saddle River, NJ: Addison Wesley Professional, 2007, pp. 231–272.
- [14] B. Feinstein and D. Peck, "Caffeine monkey: Automated collection, detection and analysis of malicious javascript," in *Black Hat USA 2007*, Las Vegas, 2007.
- [15] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, "The ghost in the browser: Analysis of web-based malware," in *Hot-Bots'07*. Cambridge: Usenix, 2007.
- [16] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, "All your iframes point to us." Google Inc., 2008, pp. available from <http://googleonlinesecurity.blogspot.com/2008/02/all-your--iframe--are--point--to--us.html>; accessed on 15 February 2008.
- [17] C. Reis, J. Dunagan, H. J. Wang, O. Dubrovsky, and S. Esmeir, "Browsershield: Vulnerability-driven filtering of dynamic html," in *7th USENIX Symposium on Operating Systems Design and Implementation*. Seattle: Usenix, 2006.
- [18] M. Ernst, "Self-defending software: Collaborative learning for security," 2008.
- [19] K. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. Keromytis, "Detecting targeted attacks using shadow honeypots," in *14th USENIX Security Symposium*. Baltimore: Usenix, 2005.
- [20] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy, "Spyproxy: Execution-based detection of malicious web content," in *Proceedings of the 16th USENIX Security Symposium*, Boston, MA, August 2007.
- [21] S. Sidiroglou, J. Ioannidis, A. Keromytis, and S. J. Stolfo, "An email worm vaccine architecture," in *1st Information Security Practice and Experience Conference*, Singapore, 2005.
- [22] F. Pouget and T. Holz, "A pointillist approach for comparing honeypots," in *DIMVA*, Vienna, 2005.
- [23] S. Axelsson, "The base-rate fallacy and its implications for the difficulty of intrusion detection," in *6th ACM Conference on Computer and Communications Security*. Singapore: ACM Press, 1999.
- [24] Microsoft Cooperation, "Microsoft security bulletin ms06-055: Vulnerability in vector markup language could allow remote code execution," 2006, pp. available from <http://www.microsoft.com/technet/security/Bulletin/MS06--055.mspx>; accessed on 14 February 2007.