Effective Analysis, Characterization, and Detection of Malicious Web Pages

Birhanu Eshete
Supervised by: Adolfo Villafiorita

Fondazione Bruno Kessler
Trento, Italy
{eshete, adolfo.villafiorita}@fbk.eu

ABSTRACT

The steady evolution of the Web has paved the way for miscreants to take advantage of vulnerabilities to embed malicious content into web pages. Up on a visit, malicious web pages steal sensitive data, redirect victims to other malicious targets, or cease control of victim's system to mount future attacks. Approaches to detect malicious web pages have been reactively effective at special classes of attacks like drive-by-downloads. However, the prevalence and complexity of attacks by malicious web pages is still worrisome. The main challenges in this problem domain are (1) fine-grained capturing and characterization of attack payloads (2) evolution of web page artifacts and (3) flexibility and scalability of detection techniques with a fast-changing threat landscape. To this end, we proposed a holistic approach that leverages static analysis, dynamic analysis, machine learning, and evolutionary searching and optimization to effectively analyze and detect malicious web pages. We do so by: introducing novel features to capture fine-grained snapshot of malicious web pages, holistic characterization of malicious web pages, and application of evolutionary techniques to fine-tune learning-based detection models pertinent to evolution of attack payloads. In this paper, we present key intuition and details of our approach, results obtained so far, and future work.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Invasive Software(e.g., Viruses, Worms Trojan Horses)

Keywords

malicious web pages, web-based attacks, effective detection, static analysis, dynamic analysis, machine learning

1. PROBLEM

The Web has become substantially instrumental in facilitating day-to-day activities of users online. Unfortunately, the Web is also a place where miscreants exploit vulnerabilities in the browser, in web applications, and above all human weakness to steal sensitive data or compromise systems.

One prominent vehicle miscreants use to carry around malicious content on the Web through a web page that upon a visit exploits vulnerabilities of the client and launches attacks. Such a page is called *malicious web page.* Before tricking victims to visit malicious web pages, miscreants purposely craft them using attack kits sold in the underground economy [6] or compromise vulnerable legitimate web pages and inject them with malicious code. They then attract traffic to malicious web pages using a multitude of tricks including spam campaign, black-hat search engine optimization, and social engineering [26]. Lured by such tricks, an innocent victim visits a malicious web page and in case of exploitable vulnerabilities, in the browser or its extensions, the attack payload is executed without the victim's knowledge. Malicious web pages manifest in a number of ways depending on the goal of the attack. Next, we highlight the types of malicious web pages we focus on in this work.

Drive-by-Downloads. Web pages that when visited, download, install, and execute malware on a victim's machine without the knowledge of the victim are called drive-by download pages [16]. In a typical drive-by-download attack (see Figure 1), a victim with a vulnerable browser visits a malicious (compromised) page that automatically loads a remote page that, after a series of redirections, lands on a page with the actual exploit. Then the victim's environment (e.g., the browser, browser plugins) is fingerprinted and inspected for known vulnerabilities based on which a presumably effective exploit is crafted. Finally, the exploit binary is downloaded and executed on the realm of the victim's environment.

Phishing. Fraudulent pages that impersonate trusted website are called phishing pages [32]. They imitate the look-and-feel of a legitimate website (e.g., login page of victim's favorite bank) to lure the victim to give away sensitive credentials (e.g., passwords). In addition to crafting a misleading UI, phishers perform a great deal of URL manipulation to reduce the level of suspicion when victims see a slightly mis-spelled URL.

Malware-serving. These are family of web pages where binary executables are hosted and traffic is directed to them via a range of techniques such as rogue software download link, search engine referrers, and links from compromised legitimate sites.

Malvertisements. Are advertisements on the Internet that infect the viewer's machine with malware. The malware makes the compromised machine a member of a Botnet, which is then used to orchestrate more organized cybercrime (e.g., spam campaign). Malvertisements are placed on a website via legitimate advertisements or pop-up ads which deliver the malware (e.g., in a form of Scareware) as

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

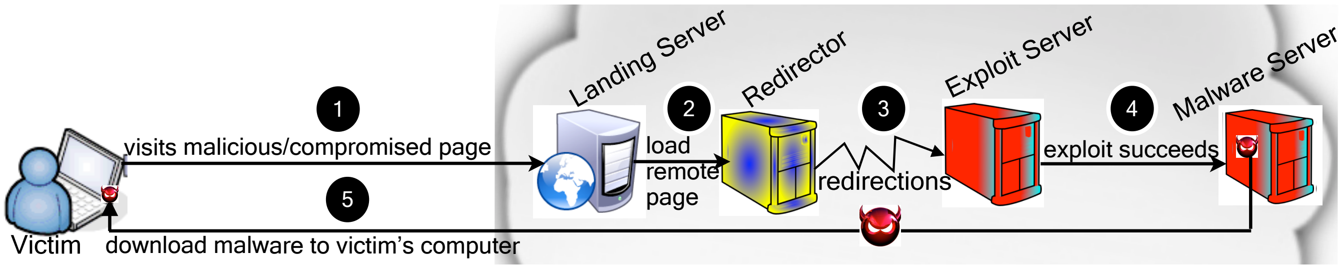

Figure 1: Typical drive-by-download attack.

soon as the ad shows up on the viewer's screen. In some cases, the malware executes when the user clicks the close button on the pop-up window.

In fact, the infection chains of different types of malicious web pages have notable overlaps. For instance, a victim may receive a spam email that lures him to give away his bank credentials or tricks him to click on a link whereby a drive-by-download attack is initiated and the ultimate landing page is a malware hosting server. Similarly, malicious advertisements could be linked to a drive-by-download attack or exploit server that fingerprints the client and downloads malware on the victim's machine.

Given the alarming prevalence of malicious web pages [26] and the constantly evolving tactics of adversaries to spawn new variants of attacks [6], existing reactive approaches have limitations in effectively and efficiently detecting malicious web pages. While existing approaches are pretty effective at detecting one prominent attack (such as drive-by-downloads), they fall short of coping with recently-emerging [17] complex web threats. Moreover, the constant evolution of attack payloads and artifacts of web pages challenges the effectiveness and efficiency of existing defenses against malicious web pages. Envisioning a proactive (instead of reactive), holistic (instead of partial), and evolution-aware (instead of statically updated) approach to thwart malicious web pages, we split our research problem into the following two problems:

Partial characterization of attack payloads. While there are numerous and complexly interwound attacks by malicious web pages, most existing approaches base the intuition of their detection techniques on specific attacks. However, an attacker crafts virtually any possible combination of existing attacks or blend existing attack payloads with newly spawned threats and embeds it into web pages. As a result, the majority of the techniques not only overlook a different type of attack but also capture course-grained features which loosely characterize malicious web pages. In effect, malicious web pages which escape detection techniques result in false negatives.

Constant evolution of threats and web page artifacts. Artifacts of benign web pages, on which existing analysis and detection techniques are based, are under continuos evolution. Old artifacts become useless over time while new ones emerge due to changes in: hosting infrastructure, web page source, functionality, web protocols, browser components and its extensions, and usage policies. Paralleled with these changes is emergence of attack vectors that exploit new vulnerabilities introduced in the course of these changes. For instance, the ongoing transition from HTML4 to HTML5 is a relevant example for changes in some crucial features of HTML (e.g., inline multimedia inclusion, local storage) [30]. Consequently, the major challenge in existing techniques is how to proactively and automatically cope with such an evolution in artifacts of web pages.

In the rest of this paper, we discuss the state of the art in the next section. Then we present in Section 3 our proposed approach with a methodology. In Section 4, we shade light on results obtained so far. Finally, in Section 5 we conclude the paper by outlining items in our to-do list for future research.

2. STATE OF THE ART

In this section, we first review blacklisting and heuristic-based approaches. Then, we discuss static analysis and dynamic analysis techniques.

2.1 Blacklisting

Blacklists maintain a list of known malicious URLs, IP addresses, and domain names collected by manual reporting, honeyclients, and custom analysis techniques. For example, Google Safe Browsing service [10] maintains a blacklist of phishing and malware URLs against which it checks requests from browsers to alert users if the requested URL happens to be in the blacklist.

Building a blacklist not only requires fresh, exhaustive, and trustworthy list but also demands a dedicated infrastructure for continuous patrolling and inspection of potentially malicious URLs. In practice, it is expensive, error-prone, and above all infeasible to keep on hunting for malicious URLs to update blacklists as attackers sleeplessly craft millions of such web pages [6] every single day. Moreover, attackers use automated tools to query blacklists and if they find their URLs in the blacklist, they simply change the URL, IP address, or domain names accordingly.

2.2 Heuristics-Based Approaches

Heuristics-based techniques [7, 23] rely on signatures of known attack payloads to be used by antiviral systems or intrusion detection systems to scan a web page and flag it as malicious if its heuristic pattern matches signatures in the database. Unfortunately, such signatures are easily evadable by attackers (e.g., using obfuscation) and the heuristics fails to detect novel attack vectors.

More importantly, the rate at which the signature database of heuristic-based systems is updated is way slower than the pace at which attackers overwhelm the Web with novel attacks, resulting in zero-day exploits.

2.3 Static Analysis Approaches

Static analysis techniques [4,5,13,14,24,27] inspect a web page without rendering it in the browser. The inspection involves extraction of distinguishing features from the URL string, host identity, HTML, JavaScript code, and reputation metadata of the page. The feature values are then encoded to train machine learning techniques to build classifiers based on which unknown web pages are classified. The main premise in static analysis is that the statistical distribution of features in malicious URLs (e.g., phishing pages) tend to differ from that of benign pages. A notable strength of such approaches is the quick extraction of features without executing the page in the browser.

Ma et al. [13] propose a technique based on lightweight extraction and analysis of lexical aspects of URL string and host details to derive a model that learns a classifier for spam and phishing URLs. The core assumption is that URL tokens and host-based details of malicious URLs tend to be significantly different from benign URLs.

Ma et al. [14] enhanced their approach [13] by applying online learning techniques on URL and host features via live feed of URLs and feature enhancement on the fly. The major argument behind using online learning algorithms is the fact that batch learning techniques fail to cope with the continuously changing distribution of features over time.

Hou et al. [33] propose an obfuscation-resilient approach based on machine learning to detect malicious web content (specifically malicious DHTML) for classifying web pages. They use mainly page content related features to evaluate different machine learning algorithms on a sample dataset and the authors report that their classification technique outperformed signature-based tools.

Canali et al. [4] propose a fast filtering technique called *Prophiler* based on machine learning to optimize resource consumption of an expensive dynamic analysis and detection backend [28]. While reusing most URL and host information features from [13], the authors introduced effective features which resulted in very low false positive rate over a large test dataset.

In general, static analysis is limited in detecting attacks that take action when the page is rendered. A disadvantage in using page source is the high risk of obfuscated JavaScript and overlooking of malicious JavaScript that exploits vulnerabilities of browser plug-ins as attacks are initiated when plugins are invoked. As for lexical URL features, an attacker may evade the features by carefully crafting URLs to be statistically indistinguishable from the benign ones.

2.4 Dynamic Analysis Approaches

Dynamic analysis [3, 12, 15, 16, 27] inspects the execution dynamics while a page is executed. A common deployment scenario of such techniques is at a proxy-level [1] where a page request(response) is intercepted and analyzed for malicious activities(e.g., unusual process spawning, repeated redirects). An alternative is client-side sandboxing [7, 22] of critical page content (e.g., JavaScript) to log critical actions (e.g., plugin invocation) and match logs with known patterns of malicious activities.

Honeyclients [21] are widely adopted systems that mimic a human visitor and use a dedicated sandbox environment to visit a web page. Execution dynamics of a web page is captured and analyzed to infer evidences for malicious activities. Low-interaction honeyclients (e.g., HoneyC [19]) monitor traces of activities during the interaction against static signatures –as a result are not able to detect zero-day exploits. Whereas, high-interaction honeyclients (e.g., Capture-HPC [20], MITRE HoneyClient [18], Microsoft HoneyMonkey [29]) check integrity changes in system states which requires monitoring file system, registry entries, processes, network connection, and physical resources like memory and CPU consumption anomalies.

Dynamic analysis techniques including honeyclients are powerful at uncovering daunting malicious web pages but at high computational cost. They need to load and execute pages and modern web pages are rich in client-side code and multimedia. This makes the analysis resource-intensive. Moreover, not all attacks are executed when a page is visited. For attacks that demand user interaction or wait for time/logic bombs, honeyclients are inflexible. Yet another downside of honeyclients is that their IP addresses may be blacklisted by malicious servers, their virtual machines be fingerprinted by miscreants, or they may also be victims of turing verification on pages with CAPTCHAs.

3. APPROACH AND METHODOLOGY

Our main claim in the proposed approach is that by capturing as much artifacts as possible for both static and dynamic aspects of web pages, using proven statistical learning methods, and applying evolutionary searching and optimization, it is possible to more precisely and efficiently analyze and detect malicious web pages in a proactive manner. The operational framework of our approach is depicted in Figure 2. In what follows, we describe the details of the approach.

3.1 Dataset Collection and Validation

Collection. To precisely analyze and detect malicious web pages, it requires to continuously patrol the Web, spot toxic regions, and collect web pages. For this dataset collection task, we mainly rely on a web-scale crawler [2] to frequently crawl and collect potentially malicious web pages based on seeds from trending topics on the Web. The crawler is equipped with anti-cloaking measures not to end up with misleading crawl results. In addition, we enrich the dataset using publicly-endorsed and constantly-updated blacklists (e.g., Google, PhishTank) and whitelists (e.g., Alexa, DMOZ).

Validation. After collecting the dataset, validating the dataset is an essential in order to end up with a trustworthy ground truth. In particular, we have to carefully validate the maliciousness of web pages we use in our dataset. To this end, we have at our disposal public services (e.g., Wepawet [28]) with which to validate our dataset and also the effectiveness of our approach. Alternatively, we setup a home-made honeyclient that disguises itself as a vulnerable client to monitor series of actions when a malicious web page is rendered to examine how far the actions deviate from rendering a benign web page.

3.2 Holistic Attack Characterization

Using the validated dataset, we extract features spanning static aspects, dynamic aspects, and metadata of URLs. While reusing effective features from prior work, we also equally rely on novel features that we introduce to improve the characterization of web pages. The new features are introduced based on thier contribution to improve the granularity of distinguishing existing attacks and their significance in capturing new threats. To this end, our focus is on HTML features that are affected by the current transition from HTML4 to HTML5 and features to characterize delayed attacks. In particular, we propose to emulate click-

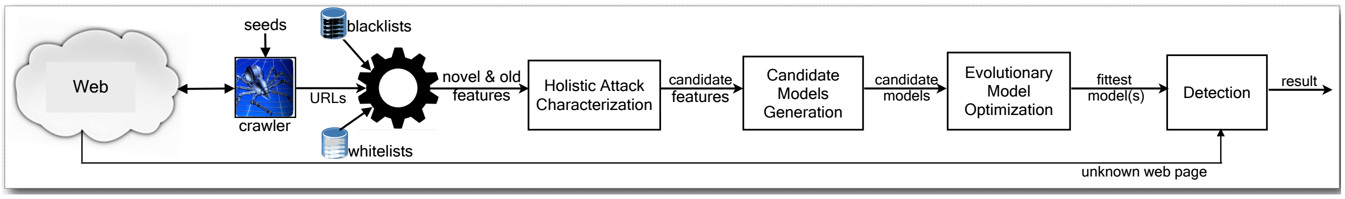

Figure 2: Overview of the proposed approach.

ing on links by customizing headless browsers [25] to nspect and characterize actions invoked when links are clicked by a user.

3.3 Candidate Models Generation

Once the characterizing features are extracted, we have at our disposal a number of established supervised learning algorithms [11] from which we drive candidate models. The main challenge in this phase is to determine best combination of features at hand and learning algorithms as there are multiple good models.

Unlike existing work [4, 13, 33] which pick the best performing model, in our approach we extend this important step using evolutionary techniques to search and optimize the best detection model automatically and in a scalable fashion. For instance, it should be scalable enough to add more features or learning algorithms as needed.

3.4 Evolutionary Searching and Optimization

In our approach, candidate models correspond to chromosomes in Genetic Algorithm (GA) [31]. The goal of a GA is to start with initial population of candidate models, iterate over generations by applying genetic operations (selection, crossover, and mutation) to search for the best combination of features and learning algorithms to produce the fittest model.

To make the best out of the GA-guided searching and optimization, a fitness function that determines how good a model is defined based on standard quality metrics such as classification accuracy and false signals. After the fittest model is selected by the GA, unknown web pages are detected by querying the fittest model.

In summary, the following are the major contributions of our approach:

- *Holistic Characterization* of malicious payloads by leveraging static aspects, dynamic aspects, and metadata in order to capture fine-grained artifacts web pages.
- *Introducing novel features* to enhance learning-based detection techniques to be able to detect novel attacks not yet addressed by existing work (e.g., attacks that require time or logic bombs).
- *Evolutionary searching and optimization* to improve the precision of detection models using genetic algorithms so as to align detection techniques with the evolution of the underlying web page artifacts.

4. RESULTS

We briefly describe two of our works, [8] and [9], as part of achieving the goals in our approach. We refer the reader to consult the full papers for details.

4.1 Holistic Analysis and Detection

To address partial characterization of attack payloads, in our work [8], we leveraged static analysis and minimalistic emulation to apply supervised learning in detecting malicious web pages pertinent to drive-by-downloads, phishing, injection, and malware distribution. The major contribution of this work is the introduction and large-scale evaluation of 10 new features (3 on URL string, 3 on HTML, 1 on JavaScript, and 3 on social reputation of URLs).

The new features are indeed effective at enhancing accuracy of classifiers by up to 3% in a large-scale dataset. Moreover, unlike previous work which evaluate and select the best performing classifier, we leveraged confidence-weighted voting of classifier outputs to classify web pages. The prototype implementation of this work is currently under improvement as part of our long-term plan to deploy it in real-life setting.

4.2 Evolution-Guided Analysis and Detection

To address constant evolution of web page artifacts, in our very recent work [9], we extended our previous work [8] by taking the models it generates as candidate models to initialize a GA. The goal in this work is to demonstrate the effectiveness of evolutionary searching and optimization of detection models using a GA. We implemented and evaluated our approach and the results show that it is possible to improve the effectiveness of analysis and detection of malicious web pages while aligning the detection models with the continuous evolution of web page artifacts. An evaluation of the approach on a fairly large-scale dataset shows that the GA significantly reduced false negatives.

5. CONCLUSIONS AND FUTURE WORK

We posed research challenges in holistically characterizing web page artifacts to facilitate analysis and detection of malicious web pages. Moreover, we described an equally pressing problem of constant evolution of web page artifacts which is being experienced by existing analysis and detection techniques for malicious web pages. At the center of these two challenges is this ever-evolving and polymorphic nature of the threat landscape on the Web.

To address these challenges, we presented a proactive, holistic, and evolution-aware approach that leverages static analysis, dynamic analysis, statistical learning, and evolutionary searching and optimization to more precisely and efficiently analyze and detect malicious web pages. In addition, we highlighted the promising results obtained so far to achieve all the goals in our proposed approach. Beyond the promising results of our approach, there still remains work to do towards effective and efficient detection of malicious web pages. Future work includes:

Further Enhancement of Attack Payloads: Using link-clicking functions of emulated browsers to capture novel features for attacks that require user interaction (e.g., malicious advertisements) with browser plugins enabled. In addition, improving the characterization of web pages by revisiting HTML4 features with respect to HTML5.

Large-Scale Evaluation: To evaluate the scalability of our approach and compare it with industry and research benchmarks using detection accuracy, false signals, and average delay to analyze a web page.

Public Deployment: We plan to deploy our approach as a public service with query interfaces for users and software services. The deployment mode (e.g., browser toolbar, proxy) is an open issue we will investigate in the future.

6. REFERENCES

[1] M. Alexander, B. Tanya, D. Damien, S. D. Gribble, and H. M. Levy. Spyproxy: execution-based detection of malicious web content. In *Proceedings of 16th USENIX Security Symposium*, pages 3:1–3:16, 2007.

[2] I. Archive. Heritrix. `http://crawler.archive.org/index.html`, July 2012.

[3] K. Byung-Ik, I. Chae-Tae, and J. Hyun-Chul. Suspicious malicious web site detection with strength analysis of a javascript obfuscation. In *International Journal of Advanced Science and Technology*, pages 19–32, 2011.

[4] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *Proceedings of WWW*, pages 197–206, 2011.

[5] H. Choi, B. B. Zhu, and H. Lee. Detecting malicious web links and identifying their attack types. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 11–11, 2011.

[6] S. Corporation. Symantec web based attack prevalence report. `http://www.symantec.com/business/threatreport/topic.jsp?id=threat_activity_trends&aid=web_based_attack_prevalence`, July 2011.

[7] A. Dewald, T. Holz, and F. C. Freiling. Adsandbox: sandboxing javascript to fight malicious websites. In *ACM Symposium on Applied Computing*, pages 1859–1864, 2010.

[8] B. Eshete, A. Villafiorita, and K. Weldemariam. Binspect: Holistic analysis and detection of malicious web pages. In *Proceedings of Security and Privacy in Communication Networks*, 2012.

[9] B. Eshete, A. Villafiorita, and K. Weldemariam. Einspect: Evolution-guided analaysis and detection of malicious web pages. Technical report, Fondazione Bruno Kessler, 2012.

[10] Google. Google safe browsing api. `http://code.google.com/apis/safebrowsing/`, August 2011.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[12] A. Ikinci, T. Holz, and F. Freiling. Monkey-spider: Detecting malicious websites with low-interaction honeyclients. In *Proceedings of Sicherheit, Schutz und ZuverlĽssigkeit*, pages 407–421, 2008.

[13] M. Justin, S. L. K., S. Stefan, and V. G. M. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of KDDM*, pages 1245–1254, 2009.

[14] M. Justin, S. L. K., S. Stefan, and V. G. M. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of ICML*, pages 681–688, 2009.

[15] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifer. Rozzle: De-cloaking internet malware. Technical report, Microsoft, 2011.

[16] C. Marco, K. Christopher, and V. Giovanni. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of WWW*, pages 281–290, 2010.

[17] T. Micro. Web threats. `http://apac.trendmicro.com/apac/threats/enterprise/web-threats/`, November 2012.

[18] MITRE. The mitre honeyclient project. `http://search.cpan.org/~mitrehc`, November 2011.

[19] H. Project. Honeyc. `https://projects.honeynet.org/honeyc`, July 2011.

[20] T. H. Project. Capture-hpc. `https://projects.honeynet.org/capture-hpc`, October 2011.

[21] M. Qassrawi and H. Zhang. Detecting malicious web servers with honeyclients. *Journal of Networks*, 6(1), 2011.

[22] K. Rieck, T. Krueger, and A. Dewald. Cujo: efficient detection and prevention of drive-by-download attacks. In *Proceedings ACSAC*, pages 31–39, 2010.

[23] C. Seifert, I. Welch, and P. Komisarczuk. Identification of malicious web pages with static heuristics. In *Proceedings of the Australasian Telecommunication Networks and Applications Conference*, 2008.

[24] C. Seifert, I. Welch, P. Komisarczuk, C. Aval, and B. Endicott-Popovsky. Identification of malicious web pages through analysis of underlying dns and web server relationships. In *33rd IEEE Conference on Local Computer Networks*, 2008.

[25] G. Software. Htmlunit. `http://htmlunit.sourceforge.net/`, March 2012.

[26] Symantec. Symantec report on attack kits and malicious websites. `http://symantec.com/content/en/us/enterprise/other_resources/b-symantec_report_on_attack_kits_and_malicious_websites_21169171_WP.en-us.pdf`, July 2011.

[27] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and Evaluation of a Real-Time URL Spam Filtering Service. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2011.

[28] UCSB. Wepawet. `http://wepawet.cs.ucsb.edu`, July 2011.

[29] Y.-M. Wang, D. Beck, X. Jiang, and R. Roussev. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *Proceedings of the NDSS*, 2006.

[30] A. Weiss. Top 5 security threats in html5. `http://www.esecurityplanet.com/trends/article.php/3916381/Top-5-Security-Threats-in-HTML5.htm`, October 2011.

[31] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4:65–85, 1993.

[32] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phishing pages. In *Proceedings of the NDSS*, 2010.

[33] H. Yung-Tsung, C. Yimeng, C. Tsuhan, L. Chi-Sung, and C. Chia-Mei. Malicious web content detection by machine learning. *Expert Syst. Appl.*, 37(1):55–60, 2010.