Integrated Detection of Anomalous Behavior of Computer Infrastructures

Federico Maggi, Stefano Zanero

Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italia
{fmaggi,zanero}@elet.polimi.it

Abstract—**Our research concentrates on anomaly detection techniques, which have both industrial applications such as network monitoring and protection, as well as research applications such as software behavioral analysis or malware classification. During our doctoral research, we worked on anomaly detection from three different perspective, as a complex computer infrastructure has several weak spots that must be protected. We first focused on the operating system, central to any computer, to avoid malicious code to subvert its normal activity. Secondly, we concentrated on web applications, which are the main interface to modern computing: Because of their immense popularity, they have indeed become the most targeted entry point of intrusions. Last, we developed novel techniques with the aim of identifying related events (e.g., alerts reported by intrusion detection systems) to build new and more compact knowledge to detect malicious activity on large-scale systems. During our research we enhanced existing anomaly detection tools and also contributed with new ones. Such tools have been tested over different datasets, both synthetic data and real network traffic, and lead to interesting results that were accepted for publication at main security venues.**

I. Introduction

Network connected devices such as personal computers, mobile phones, or gaming consoles are extremely popular. The Web and the humongous amount of services it offers have certainly became the most ubiquitous tools of all the times. Facebook counts more than 800 millions active users of which 350 millions are using it on mobile devices; not to mention that more than 250 million photos are uploaded to Facebook *every month* [4]. And this is just one, popular website. Two years ago, Google estimated that the approximate number of unique URL is 1 trillion [1], while `YouTube` has stocked more than 70 million videos as of March 2008, with 112,486,327 views just on the most popular video as of January 2009 [20]. And people from all over the world inundate the Web with more than 3 million tweets *per day*. Thinking that on December 1990 the Internet was made of *one* site and today it counts more than 100 million sites is just astonishing [25].

The Internet and the Web became the most advanced workplace. Almost every industry connected its own network to the Internet and relies on these infrastructures for a vast majority of transactions; most of the time monetary transactions. As an example, every year Google looses approximately 110 millions of US Dollars in ignored ads because of the *"I'm feeling lucky"* button. The scary part is that, during their daily work activities, people typically pay poor or no attention at all to the risks that derive from exchanging any kind of information over such a complex, interconnected infrastructure.

978-1-4673-0269-2/12/$31.00 © 2012 IEEE

Unfortunately, the Internet is all but a safe place [15], with more than 1,250 *known* data breaches between 2005 and 2009 [2] and an estimate of 263,470,869 records stolen by intruders. Today's security issues are, basically, caused by the combination of two phenomena: a large amount of software vulnerabilities and an effective exploitation strategy adopted by the cyber criminals. The fact that software is affected by vulnerabilities is not a new story. Incidentally, however, web-related software (e.g., browsers and 3rd-party extensions, and web applications), are the most vulnerable ones. For instance, in 2008, Secunia reported around 115 security vulnerabilities for Mozilla Firefox, 366 for Internet Explorer's ActiveX [19]. Office suites and e-mail clients, that are certainly the must-have-installed tool on every workstation, hold the second position [22]. In parallel, attackers and the underground economy have quickly learned that a sweep of exploits run against *every* reachable host have more chances to find a vulnerable target and, thus, is much more profitable compared to a single effort to break into a high-value, well-protected machine. In addition, today's underground economy run a very proficient market: A person with malicious intents can easily buy (stolen) credit card numbers for as low as $0.06–$30, full identities for just $0.70–$60 or rent scam hosting for $3–$40 per week plus $2-$20 for the design [23].

These circumstances have started a vicious circle that provides the attackers with a very large pool of vulnerable targets. Vulnerable client hosts are compromised to ensure virtually unlimited bandwidth and computational resources to attackers, while server side applications are violated to host malicious code used to infect client visitors. And so forth. An old fashioned attacker would have violated a single site using all the resources available, stolen data and sold it to the underground market. Instead, a modern attacker adopts a "vampire" approach and exploit client-side software vulnerabilities to take (remote) control of million hosts. In the past the diffusion of malicious code such as viruses was sustained by sharing of infected, cracked software through floppy or compact disks; nowadays, the Web offers unlimited, public storage to attackers that deploy their exploit on compromised websites. Thus, not only the type of vulnerabilities has changed, posing virtually every interconnected device at risk. The exploitation strategy created new types of threats that take advantage of known malicious code patterns but in a new, extensive and effective way.

Every year, new threats are discovered and attackers take advantage of them until effective countermeasures are found. Then, new threats are discovered, and so forth. Symantec quantifies the amount of new malicious code threats to be 1,656,227 as of 2008 [23], 624,267 one year earlier and only 20,547 in 2002. The main underlying technology actually employs a classic type of software called *bot* (jargon for *robot*), which is not malicious *per sé*, but is used to remotely control a network of compromised hosts, called *botnet* [6]. Remote commands can be of any type and typically include launching an attack, starting a phishing or spam campaign, or even updating to the latest version of the bot software by downloading the binary code from a host controlled by the attackers (usually called *bot master*) [21]. The exchange good has now become the botnet infrastructure itself rather than the data that can be stolen or the spam that can be sent. These are mere outputs of today's most popular service offered for rent by the underground economy.

II. The Role of Anomaly Detection

The aforementioned picture may lead to think that the malicious software will eventually proliferate at every host of the Internet and no effective remediation exists. However, a more careful analysis reveals that, despite the complexity of this scenario, the problems that must be solved by a security infrastructure can be decomposed into relatively simple tasks that, surprisingly, may already have a solution. This is how a typical exploitation works:

- *Injection:* A malicious request is sent to a vulnerable web application with the goal of corrupting all the subsequent pages served to legitimate clients from that moment on. For instance, more than one releases of the popular WordPress blog application are vulnerable to injection attacks[1] that allow an attacker to permanently include arbitrary content into pages. Typically, such an arbitrary content is malicious code (e.g., JavaScript, VBScript, ActionScript, ActiveX) that, whenever a user requests the infected page, executes on the client host.
- *Infection:* Assuming that the compromised site is frequently accessed—this might be the realistic case of the WordPress-powered ZDNet news blog[2]—a significant amount of clients visit it. Due to the high popularity of vulnerable browsers and plug-ins, the client may run Internet Explorer—that is the most popular—or an outdated release of Firefox on Windows. This create the perfect circumstances for the malicious page to successfully execute. In the best case, it may download a virus or a generic malware from a website under control of the attacker, so infecting the machine. In the worst case, this code may also exploit specific browser vulnerabilities and execute in privileged mode.

[1]http://secunia.com/advisories/23595
[2]http://wordpress.org/showcase/zdnet/

- *Control & Use:* The malicious code just download installs and hides itself onto the victim's computer, which has just joined a botnet. As part of it, the client host can be remotely controlled by the attackers who can, for instance, rent it, use its bandwidth and computational power along with other computers to run a distributed denial of service attack. Also, the host can be used to automatically perform the same attacks described above against other vulnerable web applications. And so forth.

This simple yet quite realistic example shows the various kinds of malicious activity that are generated during a typical drive-by exploitation. It also shows the requirements and assumptions that must hold to guarantee success. More precisely, such an attack manifests itself in at least three layers:

- *Network Activity:* Clearly, the whole interaction relies on a network connection over the Internet: the HTTP connections used, for instance, to download the malicious code as well as to launch the injection attack used to compromise the web server.
- *Host Activity:* Similarly to every other type of attack against an application, when the client-side code executes, the browser (or one of its extension plug-ins, and even the kernel) is forced to behave improperly. If the malicious code executes till completion the attack succeeds and the host is infected. This happens only if the platform, operating system, and browser all match the requirements assumed by the exploit designer. For instance, the attack may succeed on Windows and not on Mac OS X, although the vulnerable version of, say, Firefox is the same on both the hosts.
- *HTTP Traffic:* In order to exploit the vulnerability of the web application, the attacking client must generate malicious HTTP requests. For instance, in the case of an SQL injection—that is the second most common vulnerability in a web application—instead of a regular "`GET /index.php?username=myuser`" the web server might be forced to process the following request

```
GET /index.php?username=' OR 'x'='x'--\&content=<
    script src="evil.com/code.js">
```

that causes the `index.php` page to behave improperly.

It is now clear that existing protection tools that analyze the network traffic, the activity of the client's operating system, the web server's HTTP logs, or any combination of the three, have chances of recognizing patterns of known attacks. However, one of the problems that may arise with these classic, widely adopted solutions is if a zero day attack is used. A zero day attack or threat exploits a vulnerability that is unknown to the public, undisclosed to the software vendor, or a fix is not available; thus, protection mechanisms that merely rely on blacklists of known attacks immediately become ineffective.

Ideally, an effective and comprehensive countermeasure can be achieved if all the protection tools involved (e.g., client-side, server-side, network-side) can collaborate together. For instance, if a website is publicly reported to be malicious, a client-side

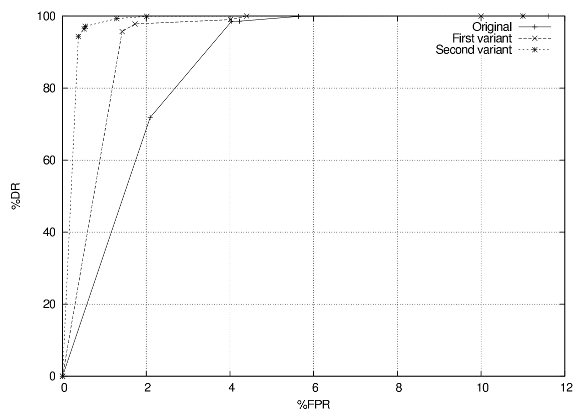


Figure 1: Detection capabilities of different variants of our host-based IDS.

protection tool should block all the content downloaded from that particular website. This is only a simple example.

To conclude, the main challenge that still need to be tackled today is twofold. First, the anomaly detection techniques themselves need new contributions, because the activity that contain evidence of intrusions is much more different than from the past: The spread of web services has moved the has brought much more semantic up to the HTTP layer, which has indeed became the transport layer of the future. In addition, the ever-changing nature of web applications pose further challenges, because anomaly detection techniques hypothesize that the "baseline" activity does not change, and changes are actually attacks. Secondly, there has always been a gap between the output of detection systems and the "picture" of what is really happening in a system under attack. In the past, this problem was called "alert correlation"; today, the same problem is called "advanced persistent threats", which are nothing but slow and stealthy intrusion that can be successfully detected only if good alert correlation techniques were available.

III. Original Contributions and Results

In our dissertation—publicly available at [7]—we concentrate on anomaly-based approaches to detect malicious activities. Since today's threats are complex, a single point of inspection is not effective. A more comprehensive monitoring system is more desirable to protect both the network, the applications running on a certain host, and the web applications (that are particularly exposed due to the immense popularity of the Web). Our contributions focus on the mitigation of both host-based and web-based attacks, along with two techniques to correlate alerts from hybrid sensors.

A. Host-based Anomaly Detection

Typical malicious processes can be detected by modeling the characteristics (e.g., type of arguments, sequences) of the system calls executed by the kernel, and by flagging unexpected deviations as attacks. Regarding this type of approaches, our contributions focus on hybrid models to accurately characterize the behavior of a binary application. In particular:

	DR	FPR	
Granularity:	Sequence	Sequence	Call
Context	`bsdtar`		
Y	100%	1.6%	0.1%
N	88%	1.6%	0.1%
	`eject`		
Y	100%	0%	0%
N	0%	0%	0%

Table I: DR and FPR on two test programs, with (Y) and without (N) context modeling.

- we enhanced, re-engineered, and evaluated a novel tool for modeling the normal activity of the Linux 2.6 kernel. Compared to other existing solutions, our system shows better detection capabilities, as shown in Figure 2, and good contextualization of the alerts reported, as shown in Table I. These results have been published in the IEEE Transactions on Secure and Dependable Systems [9].
- We engineered and evaluated an IDS to demonstrate that the combined use of (1) deterministic models to characterize a process' control flow and (2) stochastic models to capture normal features of the data flow, lead to better detection accuracy. Compared to the existing deterministic and stochastic approaches separately, our system shows better accuracy, with almost zero false positives, as summarized in Table II. These results have been published in the proceedings of the DIMVA international conference [5].
- We adapted our techniques for forensics investigation. By running experiments on real-world data and attacks, we show that our system is able to detect hidden tamper evidence although sophisticated anti-forensics tools (e.g., userland process execution) have been used. These results have been published in the ACM Operating Systems Review [13].

B. Web-based Anomaly Detection

Attempts of compromising a web application can be detected by modeling the characteristics (e.g., parameter values, character distributions, session content) of the HTTP messages exchanged between servers and clients during normal operation. This approach can detect virtually any attempt of tampering with HTTP messages, which is assumed to be evidence of attack. In this research field, our contributions focus on training

	`sing`	`mt-daapd`	`profdtpd`	`sudo`	`BitchX`
Traces	22	18	21	22	15
Syscalls	1528	9832	18114	3157	107784
Stochastic	10.0%	0%	0%	10.0%	0.0%
Deterministic	5.0%	16.7%	28%	15.0%	0.0%
Hybrid	0.0%	0%	0%	10.0%	0.0%

Table II: Comparison of the FPR of the stochastic IDS vs. deterministic IDS vs. stochastic+deterministic IDS. Values include the number of traces used.

data scarcity issues along with the problems that arise when an application changes its legit behavior. In particular:

- we contributed to the development of a system that learns the legit behavior of a web application. Such a behavior is defined by means of features extracted from 1) HTTP requests, 2) HTTP responses, 3) SQL queries to the underlying database, if any. Each feature is extracted and learned by using different models, some of which are improvements over well-known approaches and some others are original. The main contribution of this work is the *combination* of database query models with HTTP-based models. The resulting system has been validated through preliminary experiments that shown very high accuracy, as detailed in the main publication that is included in the proceedings of EC2ND [3].
- We developed a technique to automatically detect legit changes in web applications with the goal of suppressing the large amount of false detections due to code upgrades, frequent in today's web applications. We run experiments on real-world data to show that our simple but very effective approach accurately predict changes in web applications and can distinguish good *vs.* malicious changes (i.e., attacks). Changes are predicted by parsing HTTP responses, and extracting the links that a legit user may follow. In this way we derive a list of potential HTTP requests that we use to train our anomaly detection models in an online fashion. The ROC curves in Figure 3 show that without HTTP response modeling (a), changes in the web applications trigger many false positives, whereas the vast majority of these are treated as legitimate changes (not attacks) if response modeling is enabled (b). These results have been published in the proceedings of the RAID international conference [11].
- We designed and evaluated a machine learning technique to aggregate IDS models with the goal of ensuring good detection accuracy even in case of scarce training data available. Our approach relies on clustering techniques and nearest-neighbor search to look-up well-trained models used to replace under-trained ones that are prone to overfitting and thus false detections. As summarized in Figure 2, experiments on real-world data have shown that almost every false alert due to overfitting is avoided with as low as 32-64 training samples per model, reaching detection precision only achievable with thousands of training samples. These results have been published in the proceedings of NDSS [17].

Although these techniques have been developed on top of a web-based anomaly detector, they are sufficiently generic to be easily adapted to other systems using learning approaches.

C. Alert Correlation

IDS alerts are usually post-processed to generate compact reports and eliminate redundant, meaningless, or false detections. In this research field, our contributions focus on unsupervised techniques applied to aggregate and correlate alert events

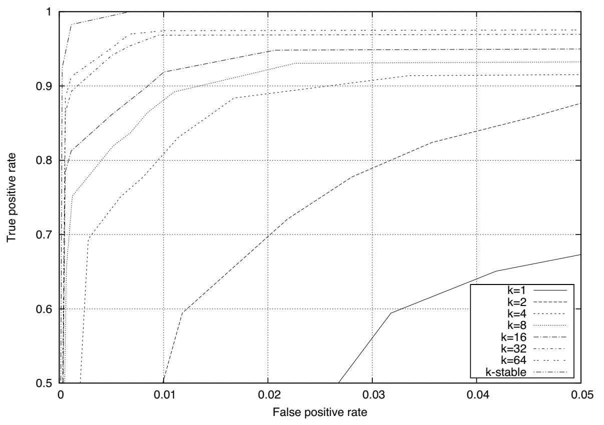


Figure 2: Global profile ROC curves for varying κ (proportional to the number of training samples). In the presence of severe undertraining ($\kappa \ll \kappa_{\text{stable}}$), the system is not able to recognize most attacks and also reports several false positives. However, as κ increases, detection accuracy improves, and approaches that of the well-trained case ($\kappa = \kappa_{\text{stable}}$).

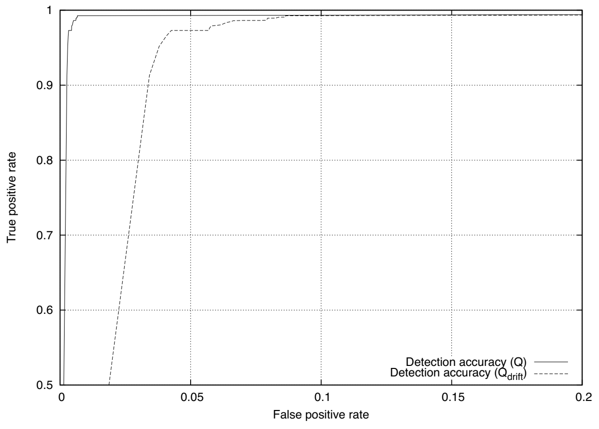
with the goal of reducing the effort of the security officer. In particular:

- We developed and tested an approach that accounts for the common measurement errors (e.g., delays and uncertainties) that occur in the alert generation process. Our approach exploits fuzzy metrics both to model errors and to construct an alert aggregation criterion based on distance in time. This technique has been show to be more robust compared to classic time-distance based aggregation metrics, because it allows for substantial reduction of false alerts at the price of slight reduction of detection capabilities, as summarized in Figure 4. These results have been published in an international journal [10].
- We designed and tested a prototype that models the alert generation process as a stochastic process. This setting allowed us to construct a simple, non-parametric hypothesis test that can detect whether two alert streams are correlated. Besides its simplicity, our approach has the advantage of requiring no parameters. These results have been published in the proceedings of the RAID international conference [12].

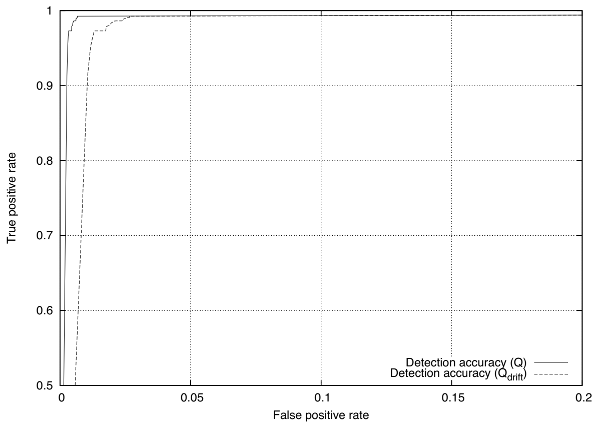
IV. Conclusions

The main question that we attempted to answer with our research work is, basically, *to what extent classic intrusion detection approaches can be adapted and integrated to mitigate today's Internet threats*. Examples of such threats include attacks against web applications like SQL injections, client-side malware that force the browser to download viruses or connect to botnets, and so forth. Typically, these attempts are labeled as *malicious activities*. The short answer to the question is the following.

As long as the technologies (e.g., applications, protocols, devices) will prevent reaching a sophistication level such that malicious activity is seamlessly camouflaged as normal activity, then anomaly detection techniques will constitute an effective

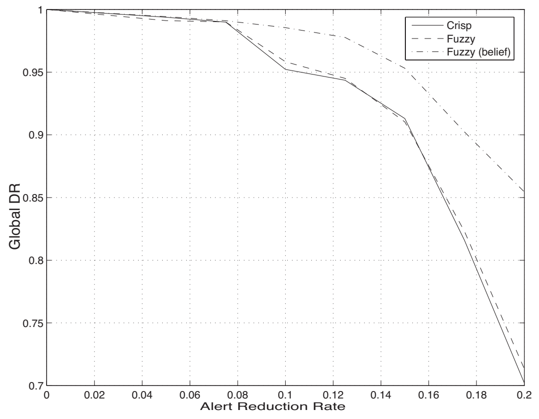


(a) Response modeling disabled.

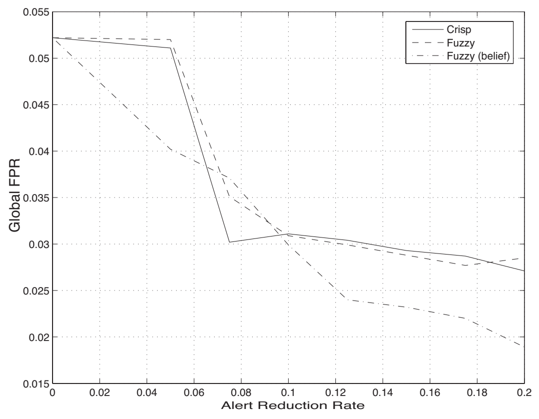


(b) Response modeling enabled.

Figure 3: Detection and false positive rates measured on Q and Q_{drift}, with HTTP response modeling enabledin (b).



(a)



(b)

Figure 4: Plot of the $DR_{\mathbb{A}'}$ (a) and $FPR_{\mathbb{A}'}$ (b) vs. ARR. "Crisp" refers to the use of the crisp time-distance aggregation; "Fuzzy" and "Fuzzy (belief)" indicates the simple fuzzy time-distance aggregation and the use of the `attack_belief` for alert discarding, respectively.

countermeasure. This is because IDS—in particular those that leverage anomaly-based techniques—are specifically designed to detect unexpected events in a computer infrastructure. The crucial point of anomaly-based techniques is that they are conceived to be generic. In principle, they indeed make no difference between an alteration of a process' control flow caused by an attempt to interpret a crafted JavaScript code and one due to a buffer overflow being exploited. Thus, as long as the benign activity of a system is relatively simple to model, then intrusion detection techniques are the building block of choice to design effective protections.

We also provide a longer and more articulated answer to the aforementioned question. A common line to the works presented in this thesis is the problem of false detections, which is certainly one of the most significant barriers to the wide adoption of anomaly-based systems. In fact, the tools that are available to the public domain already offer superb detection capabilities that can recognize all the known threats and, in theory, are effective also against unknown malicious activities. However, a large share of the alerts fired by these tools are negligible; either because they regard threats that do not apply to the actual scenario (e.g., unsuccessful attacks or vulnerable software version mismatches) or because the recognized anomaly does not reflect an attack at all (e.g., a software is upgraded and a change is confused with an threat). False detections mean time spent by the security officers to investigate the possible causes of the attack, thus, false detections mean costs.

We demonstrated that most of the false detections, especially false positives, can be prevented by carefully designing the models of normal activity. For example, we have been able to suppress many false positives caused by too strict model constraints. In particular, we substituted the "crisp" checks performed by deterministic relations learned over system calls' arguments with "smoother" models (e.g., Gaussian distributions instead of simple ranges) that, as we shown in [10], preserve the detection capabilities and decrease the rate of false alerts.

We also have shown that another good portion of false positives can be avoided by solving training issues. In particular, our contributions on detection of attacks against web applications have identified that, in certain cases, training is responsible for more than 70% of the false positives. We proposed to solve this issue by dynamically updating models of benign activity while the system is running. Even though our solution may pose new, limited risks in some situations, it is capable of suppressing all the false detections due to incomplete training; and, given the low base-rate of attacks used in our experiments, the resulting system offers a good balance between protection and costs of false positives.

Another important, desirable feature of an IDS is its capability of recognizing logically-related alerts. Note that, however, this is nothing but a slightly more sophisticated alert reduction mechanism, which once again means decreasing the effort of the security officer. In fact, as we have shown in the last chapter of this work, alert aggregation techniques can be leveraged to reduce the amount of false detections, not just for compressing them into more compact reports. However, alert correlation is a very difficult task and many efforts have been proposed to address it. Our point is that the research on this topic is still very limited and no common directions can be identified.

A. Future Directions

The main future directions of the research described in our dissertation (finished in January 2010) are, at the time of writing (November 2011), our current research topics.

Regarding host-based intrusion detection, given the spread of rough anti-malware campaigns, we have studied the different malware naming schemes adopted by anti-malware software, toward creating a global map of the current knowledge about malware [8].

Regarding web anomaly detection, by leveraging cooperative negotiation to compute the anomaly score, as opposed to naive methods such as a weighted average, we showed that this system is resilient to attacks in the training dataset and avoids false positives caused by naive model-aggregation strategies [24].

Regarding alert correlation, given the variety and significance of Internet threats (e.g., botnets, phishing, malware), we contributed in the fields of security visualization, with the twofold goal of increasing user awareness and providing experts with usable investigation tools [18] (which received a best paper award).

References

[1] J. Alpert and N. Hajaj. We knew the web was big... Available online at http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html, Jul 2008.

[2] P. R. Clearinghouse. A chronology of data breaches. Technical report, Privacy Rights Clearinghouse, July 2009.

[3] C. Criscione, F. Maggi, G. Salvaneschi, and S. Zanero. Integrated detection of attacks against browsers, web applications and databases. In *European Conference on Computer Network Defence - EC2ND 2009*, 2009.

[4] Facebook. Statistics. Available online at http://www.facebook.com/press/info.php?statistics, 2009.

[5] A. Frossi, F. Maggi, G. L. Rizzo, and S. Zanero. Selecting and Improving System Call Models for Anomaly Detection. In U. Flegel and M. Meier, editors, *DIMVA*, Lecture Notes in Computer Science. Springer, 2009.

[6] T. Holz. A short visit to the bot zoo. *IEEE Security & Privacy*, 3(3):76–79, 2005.

[7] F. Maggi. *Integrated Detection of Anomalous Behavior of Computer Infrastructures*. PhD thesis, Politecnico di Milano, 2010. http://home.dei.polimi.it/fmaggi/downloads/phd_thesis.pdf.

[8] F. Maggi, A. Bellini, G. Salvaneschi, and S. Zanero. Finding non-trivial malware naming inconsistencies. In *ICISS*, 2011.

[9] F. Maggi, M. Matteucci, and S. Zanero. Detecting intrusions through system call sequence and argument analysis (preprint). *IEEE Transactions on Dependable and Secure Computing*, 99(1), 2009.

[10] F. Maggi, M. Matteucci, and S. Zanero. Reducing False Positives In Anomaly Detectors Through Fuzzy Alert Aggregation. *Information Fusion*, 2009.

[11] F. Maggi, W. Robertson, C. Kruegel, and G. Vigna. Protecting a moving target: Addressing web application concept drift. In E. Kirda and D. Balzarotti, editors, *RAID*, Lecture Notes in Computer Science. Springer, 2009.

[12] F. Maggi and S. Zanero. On the use of different statistical tests for alert correlation. In C. Kruegel, R. Lippmann, and A. Clark, editors, *RAID*, volume 4637 of *Lecture Notes in Computer Science*, pages 167–177. Springer, 2007.

[13] F. Maggi, S. Zanero, and V. Iozzo. Seeing the invisible - forensic uses of anomaly detection and machine learning. *ACM Operating Systems Review*, April 2008.

[14] Miniwatts Marketing Grp. World Internet Usage Statistics. http://www.internetworldstats.com/stats.htm, January 2009.

[15] Ofer Shezaf and Jeremiah Grossman and Robert Auger. Web Hacking Incidents Database. http://www.xiom.com/whid-about, January 2009.

[16] M. J. Ranum. The six dumbest ideas in computer security. http://www.ranum.com/security/computer_security/editorials/dumb/, Sept. 2005.

[17] W. Robertson, F. Maggi, C. Kruegel, and G. Vigna. Effective Anomaly Detection with Scarce Training Data. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2010.

[18] F. Roveta, L. di Mario, F. Maggi, G. Caviglia, S. Zanero, and P. Ciuccarelli. Burn: Baring unknown rogue networks. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security (VizSec)*, pages 6:1–6:10, New York, NY, USA. ACM. **BEST PAPER AWARD**.

[19] Secunia. Secunia's 2008 annual report. Available online at http://secunia.com/gfx/Secunia2008Report.pdf, 2008.

[20] A. Singer. Social media, web 2.0 and internet stats. Available online at http://thefuturebuzz.com/2009/01/12/social-media-web-20-internet-numbers-stats/, Jan 2009.

[21] B. Stone-Gross, M. Cova, L. Cavallaro, B. G. t, M. tin Szydlowski andRichard Kemmerer, and C. K. andGiovanni Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *CCS 2009*, Chicago, November 2009. ACM.

[22] The SANS Institute. The twenty most critical internet security vulnerabilities. http://www.sans.org/top20/, Nov. 2005.

[23] D. Turner, M. Fossi, E. Johnson, T. Mark, J. Blackbird, S. Entwise, M. K. Low, D. McKinney, and C. Wueest. Symantec Global Internet Security Threat Report – Trends for 2008. Technical Report XIV, Symantec Corporation, April 2009.

[24] A. Volpatto, F. Maggi, and S. Zanero. Effective multimodel anomaly detection using cooperative negotiation. In *Proceedings of the First international conference on Decision and game theory for security (GameSec)*, volume 6442 of *GameSec'10*, pages 180–191. Springer-Verlag.

[25] R. H. Zakon. Hobbes' internet timeline v8.2. Available online at http://www.zakon.org/robert/internet/timeline/, Nov 2006.