POLITECNICO DI MILANO

Artificial Intelligence and Robotics Lab
Dipartimento di Elettronica, Informazione e Bioingegneria

MSc. in Computer Science and Engineering

Implementation of a 6 DoF EKF-SLAM for Leonardo Drone Contest

Author:
Diego Emanuel Avila

Supervisor:
Prof. Matteo Matteucci

December 2020

Preface

Some preface

Diego Emanuel Avila
Milano
October 2020

Abstract

Abstract

Contents

List of Figures

List of Tables

List of Acronyms

Introduction 1

Network connected devices such as personal computers, mobile phones, or gaming consoles are nowadays enjoying immense popularity. In parallel, 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 250 millions active users of which 65 millions are using it on mobile devices; not to mention that more than 1 billion photos are uploaded to the site *each month* (Facebook, 2009). And this is just one, popular website. One year ago, Google estimated that the approximate number of unique *Uniform Resource Locators* (URLs) is 1 trillion (Alpert and Hajaj, 2008), 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 (Singer, 2009). And people from all over the world inundate the Web with more than 3 million tweets *per day*. Not only the Web 2.0 has became predominant; in fact, thinking that on December 1990 the Internet was made of *one* site and today it counts more than 100 million sites is just astonishing (Zakon, 2006).

The Internet and the Web are huge (Miniwatts Marketing Grp., 2009). The relevant fact, however, is that they both 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. This is demonstrated by the effectiveness of social engineering (Mitnick, 2002) scams carried over the Internet or the phone (Granger, 2001). Recall that 76% of the phishing is related to finance. Now, compare this landscape to what the most famous security quote states.

> "The only truly secure computer is one buried in concrete, with the power turned off and the network cable cut". —*Anonymous*

In fact, the Internet is all but a safe place (Ofer Shezaf and Jeremiah Grossman and Robert Auger, 2009), with more than 1,250 *known* data breaches between 2005 and 2009 (Clearinghouse, 2009) and an estimate of 263,470,869 records stolen by intruders. One may wonder why the advance of research in computer security and the increased awareness of governments and public institutions are still not capable of avoiding such incidents. Besides the fact that the aforementioned numbers would be order of magnitude higher in absence of countermeasures, todays' security issues are, basically, caused by the combination of two phenomena: the high amount of software vulnerabilities and the effectiveness of todays' exploitation strategy.

software flaws — (un)surprisingly, software is affected by vulnerabilities. Incidentally, tools that have to do with the Web, namely, 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 (Secunia, 2008). Office suites and e-mail clients, that are certainly the must-have-installed tool on every workstation, hold the second position (The SANS Institute, 2005).

massification of attacks — in parallel to the explosion of the Web 2.0, 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.

These circumstances have initiated 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 classic malicious code patterns but in a new, extensive, and tremendously effective way.

1.1 Todays' Security Threats

Every year, new threats are discovered and attacker 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 (Turner et al., 2009), 624,267 one year earlier and only 20,547 in 2002. Thus, countermeasures must advance at least with the same grow rate. In addition:

> [...] the current threat landscape — such as the increasing complexity and sophistication of attacks,

Figure 1.1: Illustration taken from (Holz, 2005) and ©2005 IEEE. Authorized license limited to POLITECNICO DI MILANO.

> the evolution of attackers and attack patterns, and malicious activities being pushed to emerging countries — show not just the benefits of, but also the need for increased cooperation among security companies, governments, academics, and other organizations and individuals to combat these changes (Turner et al., 2009).

Todays' underground economy run a very proficient market: everyone can buy credit card information for as low as \$0.06–\$30, full identities for just \$0.70–\$60 or rent a scam hosting solution for \$3–\$40 per week plus \$2-\$20 for the design (Turner et al., 2009).

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* (Holz, 2005). 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*) (Stone-Gross et al., 2009). 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 todays' most popular service offered for rent by the underground economy.

1.1.1 The Role of Intrusion Detection

The aforementioned, dramatic big 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. Let us look at an example.

Example 1.1.1 *This is how a sample exploitation can be structured:*

injection — *a malicious request is sent to the vulnerable web application with the goal of corrupting all the responses sent 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 to the pages. Typically, such an arbitrary content is malicious code (e.g., JavaScript, VBSCrip, ActionScript, ActiveX) that, every time a legitimate 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.*

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* (DoS) *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

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

exploitation. It also shows its requirements and assumptions that must hold to guarantee success. More precisely, we can recognize:

network activity — clearly, the whole interaction relies on a network connection over the Internet: the *HyperText Transfer Protocol* (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) 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 *Structured Query Language* (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 a

```
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 protection mechanisms 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 that something malicious is happening in the network. For instance, if the *Internet Service Provider* (ISP) network adopt Snort, a lightweight *Intrusion Detection System* (IDS) that

analyzes the network traffic for known attack patterns, could block all the packets marked as suspicious. This would prevent, for instance, the SQL injection to reach the web application. A similar protection level can be achieved by using other tools such as ModSecurity (Ristic, 2008). 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 blacklist known malicious activity immediately become ineffective. In a similar vein, if the client is protected by an anti-virus, the infection phase can be blocked. However, this countermeasure is once again successful only if the anti-virus is capable of recognizing the malicious code, which assumes that the code is known to be malicious.

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

Thus, countermeasures against todays' threats already exist but are subject to at least two drawbacks:

- they offer protection only against known threats. To be effective we must assume that all the hostile traffic can be enumerated, which is clearly an impossible task.

> Why is "Enumerating Badness" a dumb idea? It's a dumb idea because sometime around 1992 the amount of Badness in the Internet began to vastly outweigh the amount of Goodness. For every harmless, legitimate, application, there are dozens or hundreds of pieces of malware, worm tests, exploits, or viral code. Examine a typical antivirus package and you'll see it knows about 75,000+ viruses that might infect your machine. Compare that to the legitimate 30 or so apps that I've installed on my machine, and you can see it's rather dumb to try to track 75,000 pieces of

> Badness when even a simpleton could track 30 pieces of Goodness (Ranum, 2005).

- they lack of cooperation, which is crucial to detect global and slow attacks.

This said, we conclude that classic approaches such as dynamic and static code analysis and IDS already offer good protection but industry and research should move toward methods that require little or no knowledge. In this work, we indeed focus on the so called anomaly-based approaches, i.e., those that attempt to recognize the threats by detecting any variation from a system's normal operation, rather than looking for signs of known-to-be-malicious activity.

1.2 Original Contributions

Our main research area is *Intrusion Detection* (ID). In particular, we focus on anomaly-based approaches to detect malicious activities. Since todays' 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.

1.2.1 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:

- 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 and good contextualization of the alerts reported.

- 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.

- 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.

1.2.2 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 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.

- 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 todays' 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).

- 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. 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.

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.

1.2.3 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 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.

- 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 or not. Besides its simplicity, the advantage of our approach is to not requiring any parameter.

The aforementioned results have been published in the proceedings of international conferences and international journals.

2.1 A Table

Feature	Misuse-based	Anomaly-based
Modeled activity:	Malicious	Normal
Detection method:	Matching	Deviation
Threats detected:	Known	Any
False negatives:	High	Low
False positives:	Low	High
Maintenance cost:	High	Low
Attack desc.:	Accurate	Absent
System design:	Easy	Difficult

Table 2.1: Duality between misuse- and anomaly-based intrusion detection techniques. Note that, an anomaly-based IDS can detect "Any" threat, under the assumption that an attack always generates a deviation in the modeled activity.

2.2 Code

```
/* ... */ cd['<'] = {0.1, 0.11} cd['a'] = {0.01,
    0.2} cd['b'] =
{0.13, 0.23} /* ... */

b = decode(arg3_value);

if ( !(cd['c'][0] < count('c', b) < cd['c'][1]) ||\
     !(cd['<'][0] < count('<', b) < cd['<'][1]) ||\
     ... || ...)  fire_alert("Anomalous content
         detected!");
/* ... */
```

2.3 A Sideways Table

Approach	Time	Header	Payload	Stochastic	Determ.	Clustering
(Mahoney and Chan, 2001)		•				•
(Kruegel et al., 2002)		•	•	•		
(Sekar et al., 2002)		•		•	•	
(Ramadas, 2003)			•			•
(Mahoney and Chan, 2003)	•		•		•	
(Zanero and Savaresi, 2004)		•	•			•
(Wang and Stolfo, 2004)			•	•		
(Zanero, 2005)		•	•			•
(Bolzoni et al., 2006)		•	•			•
(Wang et al., 2006)			•	•		

Table 2.2: Taxonomy of the selected state of the art approaches for network-based anomaly detection.

2.4 A Figure

Figure 2.1: `telnetd`: distribution of the number of other system calls among two `execve` system calls (i.e., distance between two consecutive `execve`).

2.5 Bulleted List

- O ="Intrusion", $\neg O$ ="Non-intrusion";
- A ="Alert reported", $\neg A$ ="No alert reported".

2.6 Numbered List

1. O ="Intrusion", $\neg O$ ="Non-intrusion";
2. A ="Alert reported", $\neg A$ ="No alert reported".

2.7 A Description

Time refers to the use of *timestamp* information, extracted from network packets, to model normal packets. For example, normal packets may be modeled by their minimum and maximum inter-arrival time.

Header means that the *Trasmission Control Protocol* (TCP) header is decoded and the fields are modeled. For example, normal packets may be modeled by the observed ports range.

Payload refers to the use of the payload, either at *Internet Protocol* (IP) or TCP layer. For example, normal packets may be modeled by the most frequent byte in the observed payloads.

Stochastic means that stochastic techniques are exploited to create models. For example, the model of normal packets may be constructed by estimating the sample mean and variance of certain features (e.g., port number, content length).

Deterministic means that certain features are modeled following a deterministic approach. For example, normal packets may be only those containing a specified set of values for the *Time To Live* (TTL) field.

Clustering refers to the use of clustering (and subsequent classification) techniques. For instance, payload byte vectors may be compressed using a *Self Organizing Map* (SOM) where class of different packets will stimulate neighbor nodes.

2.8 An Equation

$$d_a(i,j) := \begin{cases} K_a + \alpha_a \delta_a(i,j) & \text{if the elements are different} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

2.9 A Theorem, Proposition & Proof

Theorem 2.9.1 $a^2 + b^2 = c^2$

Proposition 2.9.2 $3 + 3 = 6$

Proof 2.9.1 *For any finite set* $\{p_1, p_2, ..., p_n\}$ *of primes, consider* $m = p_1 p_2 ... p_n + 1$. *If* m *is prime it is not in the set since* $m > p_i$ *for all* i. *If* m *is not prime it has a prime divisor* p. *If* p *is one of the* p_i *then* p *is a divisor of* $p_1 p_2 ... p_n$ *and hence is a divisor of* $(m - p_1 p_2 ... p_n) = 1$, *which is impossible; so* p *is not in the set. Hence a finite set* $\{p_1, p_2, ..., p_n\}$ *cannot be the collection of all primes.*

2.10 Definition

Definition 2.10.1 (Anomaly-based IDS) *An* anomaly-based IDS *is a type of IDS that generate alerts* $\mathbb{A}$ *by relying on normal activity profiles.*

2.11 A Remark

Remark 1 *Although the network stack implementation may vary from system to system (e.g., Windows and Cisco platforms have different implementation of TCP).*

2.12 An Example

Example 2.12.1 (Misuse *vs.* Anomaly) *A misuse-based system M and an anomaly-based system A process the same log containing a full dump of the system calls invoked by the kernel of an audited machine. Log entries are in the form:*

```
<function_name>(<arg1_value>, <arg2_value>, ...)
```

2.13 Note

Note 2.13.1 (Inspection layer) *Although the network stack implementation may vary from system to system (e.g., Windows and Cisco platforms have different implementation of TCP), it is important to underline that the notion of IP, TCP, HTTP* packet *is well defined in a system-agnostic way, while the notion of* operating system activity *is rather vague and by no means standardized.*

Bibliography

Jesse Alpert and Nissan Hajaj. We knew the web was big... Available online at `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`, Jul 2008.

Damiano Bolzoni, Sandro Etalle, Pieter H. Hartel, and Emmanuele Zambon. Poseidon: a 2-tier anomaly-based network intrusion detection system. In *IWIA*, pages 144–156. IEEE Computer Society, 2006. ISBN 0-7695-2564-4.

Privacy Rights Clearinghouse. A chronology of data breaches. Technical report, Privacy Rights Clearinghouse, July 2009.

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

Sarah Granger. Social engineering fundamentals, part i: Hacker tactics. Available online at `http://www.securityfocus.com/infocus/1527`, Dec 2001.

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

Christopher Kruegel, Thomas Toth, and Engin Kirda. Service-Specific Anomaly Detection for Network Intrusion Detection. In *Proceedings of the Symposium on Applied Computing (SAC 2002)*, Spain, March 2002.

Matthew V. Mahoney and Philip K. Chan. Learning rules for anomaly detection of hostile network traffic. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, page 601, 2003. ISBN 0-7695-1978-4.

M.V. Mahoney and P.K. Chan. Detecting novel attacks by identifying anomalous network packet headers. Technical Report CS-2001-2, Florida Institute of Technology, 2001.

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

Kevin Mitnick. *The art of deception*. Wiley, 2002.

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

M. Ramadas. Detecting anomalous network traffic with self-organizing maps. In *Recent Advances in Intrusion Detection 6th International Symposium, RAID 2003, Pittsburgh, PA, USA, September 8-10, 2003, Proceedings*, Mar 2003.

Marcus J. Ranum. The six dumbest ideas in computer security. `http://www.ranum.com/security/computer_security/editorials/dumb/`, Sept. 2005.

Ivan Ristic. mod_security: Open Source Web Application Firewall. http://www.modsecurity.org/, June 2008.

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

R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based anomaly detection: a new approach for detecting network intrusions. In *CCS '02: Proceedings of the 9th ACM Conference on Computer and communications security*, pages 265–274, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-612-9.

Adam 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.

Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Mar tin Szydlowski andRichard Kemmerer, and Christopher Kruegel andGiovanni Vigna. Your botnet is my botnet:

Analysis of a botnet takeover. In *CCS 2009*, Chicago, November 2009. ACM.

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

Dean Turner, Marc Fossi, Eric Johnson, Trevor Mark, Joseph Blackbird, Stephen Entwise, Mo King Low, David McKinney, and Candid Wueest. Symantec Global Internet Security Threat Report – Trends for 2008. Technical Report XIV, Symantec Corporation, April 2009.

Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*. Springer-Verlag, September 2004.

Ke Wang, Janak J. Parekh, and Salvatore J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, Hamburg, GR, September 2006. Springer-Verlag.

Robert H'obbes' Zakon. Hobbes' internet timeline v8.2. Available online at `http://www.zakon.org/robert/internet/timeline/`, Nov 2006.

Stefano Zanero. Analyzing tcp traffic patterns using self organizing maps. In Fabio Roli and Sergio Vitulano, editors, *Proceedings 13th International Conference on Image Analysis and Processing - ICIAP 2005*, volume 3617 of *Lecture Notes in Computer Science*, pages 83–90, Cagliari, Italy, Sept. 2005. Springer. ISBN 3-540-28869-4.

Stefano Zanero and Sergio M. Savaresi. Unsupervised learning techniques for an intrusion detection system. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 412–419. ACM Press, 2004. ISBN 1-58113-812-1.