Applied Soft Computing 25 (2014) 1–14

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Of daemons and men: A file system approach towards intrusion detection

G. Mamalakis*, C. Diou, A.L. Symeonidis, L. Georgiadis

Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 7 August 2013
Received in revised form 16 May 2014
Accepted 29 July 2014
Available online 17 September 2014

Keywords:
Intrusion detection systems
Information security
Machine learning
Data mining
File system
Anomaly detection

ABSTRACT

We present *FI²DS* a file system, host based anomaly detection system that monitors Basic Security Module (BSM) audit records and determines whether a web server has been compromised by comparing monitored activity generated from the web server to a normal usage profile. Additionally, we propose a set of features extracted from file system specific BSM audit records, as well as an IDS that identifies attacks based on a decision engine that employs one-class classification using a moving window on incoming data. We have used two different machine learning algorithms, Support Vector Machines (SVMs) and Gaussian Mixture Models (GMMs) and our evaluation is performed on real-world datasets collected from three web servers and a honeynet. Results are very promising, since *FI²DS* detection rates range between 91% and 95.9% with corresponding false positive rates ranging between 8.1×10^{-2} % and 9.3×10^{-4} %. Comparison of *FI²DS* to another state-of-the-art filesystem-based IDS, FWRAP, indicates higher effectiveness of the proposed IDS in all three datasets. Within the context of this paper *FI²DS* is evaluated for the web daemon user; nevertheless, it can be directly extended to model any daemon-user for both intrusion detection and postmortem analysis.

1. Introduction

1.1. About intrusion detection

In contemporary computer and communication networks almost everybody uses the Internet backbone to exchange personal or sensitive information in a daily basis. Through cellphones, laptops, net-pads and smart sensors people exchange data on top of various types of applications like email services, web transactions, social networks, file transfers, etc. In tandem with this extreme growth of information exchange via the Internet, cyber-threats evolve to exploit the expanding attack surface. In order to protect end-users, specialized software and hardware solutions have been deployed (firewalls, antiviruses, spam detectors, Intrusion Detection Systems, sandboxes, etc). It is common knowledge, though, that none of these solutions alone is enough to offer absolute protection; usually a combination of them is essential for providing an adequate level of cyber-protection.

Intrusion detection systems (IDS), known as the "computer world's burglar alarm", are based on the idea of identifying attacks when or after they occur, and fire an alarm or take some action – *Intrusion Response Systems* (*IRS*) – according to their configuration. Different types of IDS have been proposed during the last two and a half decades. Network IDS (NIDS) [3,10,13,28,36] monitor network activity, while host-based IDS (HIDS) monitor data generated within a host, including command histories [15], system calls [2,6], function calls [26] and file system data [34]. Hybrid IDS [16,40] monitor both network and host activity. Misuse IDS [8,14] are trained with malicious data to identify attacks, while anomaly-based IDS [6,7] raise an alarm whenever monitored activity diverges from a normal usage profile. It is not uncommon to use distributed architectures of different IDS types in order to enhance the security perimeter of large computer networks.

1.2. Our approach

We have built a system that is able to distinguish the man from the daemon on a running server. In the following paragraphs we describe how such a system can be used to allow intrusion identification of compromised daemons.

As most experienced system administrators know, no matter how well configured, hardened and up-to-date a running system

* Corresponding author. Tel.: +30 2310 99 4379; fax: +30 2310 99 4370.
E-mail addresses: mamalos@eng.auth.gr, mamalos@gmail.com (G. Mamalakis), diou@mug.ee.auth.gr (C. Diou), asymeon@eng.auth.gr (A.L. Symeonidis), leonid@auth.gr (L. Georgiadis).

http://dx.doi.org/10.1016/j.asoc.2014.07.026
1568-4946/

offering network services is, it still has a chance of being compromised (it only takes a clumsy PHP developer to create an easily exploitable vulnerability even on the most protected web server). On a server system the attack surface consists mainly of the public services it offers (HTTP, FTP, SSH, etc.) and furthermore, attackers exploiting a vulnerability on a service are usually rewarded with a remote shell running with the privileges of the exploited daemon-user. Once this type of access is granted, attackers are allowed to run arbitrary commands as this daemon user. Hence, one way to identify when a daemon has been compromised is to monitor the daemon process for abnormal behaviour.

We formulate our objective as a machine learning problem and try to identify novel features that are informative enough for various HIDS anomaly detection algorithms. Within the context of this work we argue that this distinction of daemon-human behaviour can be accomplished, since daemons usually behave in a very specific and repetitive way, serving web pages from certain paths or delivering mails to pre-configured mailboxes. Attackers, on the other hand, after compromising the vulnerable daemon usually perform actions like searching the system or network for further vulnerabilities, inspecting the system to identify its architecture and OS version, cleaning log files from their trails, downloading additional utilities to gain more privileges, defacing a web-page and generally performing actions that deviate from the way a daemon usually behaves. We argue that this divergence is reflected on the overall behaviour of the daemon and therefore an IDS should be able to identify the few diverging actions of an attacker against the daemon's normal usage profile.

So, in this paper we present *FI²DS*, a multiprocessing, python-based, file system IDS, that is able to identify attacks by reading BSM audit records both on-line and off-line (from `/dev/auditpipe` and BSM binary files, respectively), generating feature vectors from monitoring file system activity and, ultimately, by employing alternative machine learning techniques on the generated feature vectors. *FI²DS* utilizes file system data because: (a) the file system stores most attackers' traces on all OS's, and (b) this storage is permanent (contrary to the system's memory, or CPU registers, for example). The best and easiest source for monitoring file system activity on our FreeBSD[1] servers that have been used for our experiments, was FreeBSD's Audit System that generates BSM records.[2] Except from FreeBSD, BSM is also available for Solaris and Mac OS X, hence our proposed IDS can run on those systems as well. Furthermore, MS Windows Advanced Auditing and Linux audit are mechanisms that generate audit records analogous to BSM, so *FI²DS* can be ported to support those OS's too. It is important to stress at this point that *FI²DS* can be used in parallel with other types of IDS (HIDS, NIDS etc.) that are capable of identifying attacks not reflected on the file system. *FI²DS* practically adds an additional level of awareness regarding the usage of the file system from the running daemons.

In the context of this paper *FI²DS* reads audit records generated from the httpd daemon user (*www*), builds a normal usage profile and then monitors incoming activity for divergence through a moving window mechanism. We have employed both one-class Support Vector Machines and Gaussian Mixture Models (see Section 4) for novelty detection and have carried out experiments on datasets originating from three real-world web servers. The malicious activity dataset is inferred from commands gathered from a honeynet we have deployed just for this purpose. Our experimental results indicate that *FI²DS* achieves high detections rates with low corresponding false positive rates and when compared to FWRAP [34], *FI²DS* outperforms it.

[1] http://www.freebsd.org.
[2] http://www.freebsd.org/doc/en/books/handbook/audit.html.

1.3. Related work

In her seminal paper [4], Denning expresses the idea that intrusions against computers and networks may be detected if we assume that computer and/or network usage activity can be automatically profiled, and that the trails of intrusions are present in this activity. Therefore, if a security-benign usage profile can been created for a monitored system, then all subsequent profiles created by the system's later activity can be compared with this baseline using some meaningful metric and if great divergence is found, the system may fire an alarm to inform about the incident. Denning was the first to talk about how anomaly detection could be used for computer systems and specifically for host-based intrusion detection. Her initial idea came to life with the deployment of *Multics Intrusion Detection and Alerting System* (*MIDAS*) [30], an expert system whose rule-base used *Production Based Expert System Toolset*(*P-BEST*) and was developed by the *National Computer Security Center* (*NCSC*).

A few years later, Forrest et al [5,6,9,39] demonstrated the notion of *immune systems*, inspired by her studies in *natural immune systems*. The key concept in her work was how to define the sense of *self* in the UNIX processes so as to detect intrusions by identifying abnormalities through processes' deviations from *self*. Her representation of *self* in UNIX was through sequences of system calls. The approach of analysing system calls has been adopted by many other researchers [18,24,38,41], but as Wagner et al describe in [37], HIDS based on sequences of system calls seem to be susceptible to mimicry attacks.

Even though a lot of research has been held with the subject of intrusion detection, only a few papers have been involved with anomaly based IDS that identify attacks based on file system data and none of them seems to use the BSM audit mechanism to collect them, as we do. Some of the HIDS that used MIT's Lincoln Laboratories' DARPA's '98 and '99 BSM datasets [20], have trained SVM's [2,42] or other one class classifiers [12] for their detection mechanisms but, contrary to our proposed IDS, they have not used file system semantics on their features. Lastly, the legacy, discontinued EMERALD's [27] eXpert-BSM [19] monitor implements an expert system IDS that uses BSM data as its input. But due to the fact that expert-BSM is a misuse IDS, instead of being trained with benign data to form a norm and try to identify attacks on newly arriving data by inspecting how related they are with respect to the norm, it uses a malicious rule base in its expert system that probes newly arriving data against it.

One more interesting work, although more relevant to permanent data storage, is the one proposed by Stanton et al. [33]. Even though they follow a very different modeling approach and use different types of data (block level data combined with file system data), their idea illustrates some features that are worth mentioning. They propose a 3-tiered anomaly based IDS, *File and Block Surveillance System* (*FABS*), that monitors file system and device controller data for abnormal behaviour. FABS builds a normal usage profile by studying sequences of events (disk accesses at the file and block level) and classifies as malicious all events that deviate from it. It uses C-miner's [17] rule based engine to build their IDS and proposes a GUI prototype (VisFlowConnect-SS) for visualisation.

Another interesting approach that uses file system information is that of Stolfo et al. [34], where the proposed system, *File Wrapper Anomaly Detector* (*FWRAP*), uses features extracted from the file system of a Linux system, through the use of a kernel module the authors wrote, in order to build a normal usage profile; *FWRAP* uses the information extracted from this module to detect attacks based on a Bayesian estimation technique. Although our general idea uses the same source of information as Stolfo's – the file system – our technique differs significantly. First of all, the feature set and the algorithms used are entirely different. Secondly, Stolfo et al. introduce a command entry in each feature vector which