Intrusion Detection in Distributed Systems using Fingerprinting and Massive Event Correlation

Florian Skopik and Roman Fiedler

AIT Austrian Institute of Technology, Safety & Security Department
{florian.skopik|roman.fiedler}@ait.ac.at

Abstract: New computing paradigms, such as mobile computing and cloud computing introduce considerable vulnerabilities to today's society. Systems do not only become more and more connected and interdependent, but multi-billion dollar markets have led to the emergence of new – economically motivated – forms of crime. Additionally, since most of today's critical infrastructures are controlled by complex ICT systems, service outages due to attacks can cause serious situations. However, because of the increasing scale and complexity of today's networked infrastructures, traditional protection mechanisms, such as firewalls and anti-virus software seem to become insufficient to guarantee an adequate level of security. In this paper, we present a novel intrusion detection concept which utilizes distributed monitoring and massive data correlation techniques to discover potential attack traces. This is crucial to establish situational awareness on a higher level and finally make informed decisions on mitigation strategies. In contrast to many others, this approach operates not on the network layer, but uses semantically rich service logs. We demonstrate the feasibility of our fingerprint-based anomaly detection approach in context of a real-world use cases and discuss its applicability.

1 Introduction

Today, mainly two forms of protective mechanisms for ICT networks are employed. First, proactive means aim at avoiding breaches and infections as far as possible, i.e., by deploying firewalls and filters, implementing efficient software update policies of corporate ICT systems, studying early warning pages (e.g., maintained by CERTs), and perform end user trainings regarding the secure handling of computer systems. Second, reactive mechanisms, such as anti-virus programs and intrusion detection systems, try to detect and contain already raging infections. Both forms have been proven useful, however, since attacks become more and more sophisticated, traditional approaches sometimes fail at protecting ICT systems. For instance, signature-based anti-virus systems cannot properly handle zero-day exploits and their heuristics can be by-passed with customized malware. Additionally, often there are not enough evidences on a single machine to discover an attack. Furthermore, firewalls and IDS/IPS software are knocked out through obfuscation methods – in short, many threats are not preventable. Thus, these days we observe a major paradigm shift from prevention and remediation-focused approaches to response and containment strategies. This shift also requires organizations to move from traditional policy-

based security approaches towards intelligent approaches, incorporating identification of anomalies, analysis and reasoning, and in-time response strategies. As recently pointed out [EMC12] basic properties of such approaches are: (i) *Risk-based:* Determine the most important assets to be secured, since an organization can never cost-efficiently secure all assets with a maximum strength. (ii) *Contextual:* Collect huge amounts of data and use analytics to identify relevant data sources for anomaly detection. A 'context space' is created by aggregating and correlating a wide variety of events, even if an attacker partially deleted his traces. (iii) *Agile:* Enable near real-time responses to minimize the exploitable attack window and to keep (financial) losses to a minimum.

Although attacks using customized tools are unique in each case, their manifestation and impact across the network is often very similar, for instance calling home functionalities might be visible in DNS logs, database accesses in SQL query logs, and probing causes events in Firewall or IDS logs. Thus, in this paper, we introduce a reactive approach that discovers these manifestations and fuzzily determines deviations from a healthy state. Here we do not investigate widely used net flow analysis on a network packet layer. Due to the increasing interconnections of network devices, the complexity of holistic monitoring on this layer is growing at an exponential rate. We rather deploy detection mechanisms on a much higher level of the network stack, in particular on the application layer, were we carefully select relevant service logs that promise to be most expressive in terms of anomaly detection. Using these log files, we are able to automatically classify log-data, correlate it, and thus, detect usage patterns across different ICT systems within an organization.

The main contributions of this paper are:

- **Agile Anomaly Detection in Distributed Service Logs.** We highlight organizational as well as technical requirements on a system implementation. This is particularly important to ensure the wide applicability in real environments. Then we outline the rationale behind our approach.
- **Architectural Blueprint and Prototype Implementation.** We discuss a scalable architecture and prototype implementation using state-of-the-art components, specifically designed for large-scale systems.
- **Evaluation and Discussion.** We discuss the applicability and functioning in a real world setting and discuss advantages and shortcomings.

The remainder of this paper is organized as follows. In Section 2 we survey related work. Section 3 outlines the big picture and basic principles. Section 4 shows the framework architecture and the prototype implementation. Section 5 covers a discussion on the applicability of our concepts, including a step-by-step use case. Then, Section 6 deals with an analytical evaluation of the system. Finally, Section 7 concludes the paper.

2 Background and Related Work

Anomaly detection approaches have been massively studied [CBK09] in recent years – especially in context of network intrusion detection systems for computer networks [GT+09, LEK+03, MHL94]. However, since systems become increasingly interconnected and complex novel approaches that can cope with today's large amount of data, aggregated from numerous organizational units, need to be developed. Recent advances inspired by approaches from the domain of bioinformatics [Mou04] use fingerprinting techniques for data reduction, and alignment mechanisms for fast and fuzzy similarity search. These techniques have been used for single systems in the past, such as for detecting illegitimate accesses in database systems [LLW02], and are now being applied on a much larger scale.

The input to above mentioned solutions are logging data collected from numerous system components being distributed over the whole network. Events reflected by log entries are correlated [KTK02] in order to realize sophisticated anomaly detection [ATS+03, HS09, LTP+04]. In this context, an important question to deal with is how to extract attack manifestations to determine log data requirements for intrusion detection [BJ04].

A wide range of protocols, technologies and supporting tools on the technical layer are available to build up sophisticated defense mechanisms for today's networks. From the wide range of network utilities, we make especially use of logging engines and protocols on a lower layer, such as syslog[1], and logging management solutions, providing aggregation and search capabilities on an intermediate layer, e.g., Splunk[2], Graylog[3] and Apache Solr[4]. The proposed anomaly detection approach in this paper relies on this intermediate layer. Security information and event management (SIEM) solutions are widely rolled out in large organizations. They offer additional services by aggregating and partly reasoning over logs to detect anomalies. In contrast to our approach presented in this paper, these systems typically rely on predefined rules to function properly, require additional configuration efforts, and often focus on service availability aspects only – which is only one dimension of the security triangle (confidentiality - integrity - availability). One particular well-known open source SIEM solution is OSSIM[5]. Once anomalies have been discovered and confirmed as intrusions, such information is typically shared between various departments or with a governmental cyber center in order to establish situational awareness on a national level [SMSB12]. In order to facilitate this kind of sharing, numerous protocols have been proposed so far, such as the Incident Object Description Exchange Format (IODEF)[6] or the Intrusion Detection Message Exchange Format (IDMEF)[7] – just to name a few.

[1] http://tools.ietf.org/html/rfc5424
[2] http://www.splunk.com/
[3] http://graylog2.org/
[4] http://lucene.apache.org/solr/
[5] http://code.google.com/p/ossiem/
[6] http://tools.ietf.org/html/rfc5070
[7] http://tools.ietf.org/html/rfc4765

3 Fingerprint-Based Anomaly Detection

The main challenge of future ICT protection systems will be to cope with highly sophisticated and potentially coordinated attacks that affect various systems in parallel within connected networks. Here, each of these single attack impacts are often below security thresholds, i.e., are not recognized by a single firewall or IDS, however point to a serious threat if they happen to occur simultaneously. Therefore we argue that looking at isolated systems or single points in a network is not sufficient any longer. Moreover, identity theft and social engineering is becoming a huge problem [BSBK09], where attacks are not performed using classic software tools only, but stolen identities to traverse systems.

Fundamental Approach: Our proposed solution to these challenges is the introduction of an additional security layer that spans the entire set of relevant services and systems even across organizational borders. This security layer harnesses logging information from firewalls, intrusion detection systems, application servers, performance monitors etc. in order to fulfill the following objectives:

- Detect coordinated attacks towards multiple targets.
- Detect attacks using multiple attack vectors.
- Detect advanced persistent threats (aberrant behavior).

In contrast to many common net flow analysis approaches, our anomaly detection mechanism relies on log file processing. Here, we do not only utilize one source, but a multitude of logging sources that are distributed across an organization. Additionally to attacks to single machines, this way, we aim at discovering distributed and coordinated attacks, especially if they manifest in different log sources, such as DNS lookups, firewall logs, and application server events. Roughly, our defense approach consists of the following steps:

1. Identification of service logs of critical assets.
2. Real-time collection and aggregation of logging data.
3. Discovery of attacks by applying foundational anomaly detection mechanisms spanning the whole organization.
4. Periodic adaptation of employed logging mechanisms to respond to new threats and to control risks.

Anomaly Detection in Bioinformatics - The Big Picture: The primary objective of our anomaly detection approach is *the detection of correlated events (based on distributed patterns) that (might) reflect anomalies in massive amounts of recorded data*. Notice, that due to the complexity of today's ICT networks and their continuously changing operating parameters, we apply an anomaly detection technique that does not rely on predefined search patterns and signatures, but can learn significant patterns at run-time. Essentially, our approach is inspired by algorithms used within the domain of bioinformatics, where tools have been designed to process large amounts of genetic material [Mou04] or complex time-series data [LZP+07].The approach consists of the following three steps:

1. Fingerprinting of current situations
2. Alignment and correlation/matching with reference sets
3. Establishment of situational awareness

Fingerprinting defines the process of pattern mining for data reduction; especially the mining of repetitive sequences and extraction of characteristic attributes in sequences. Hence, these fingerprints characterize, in our case, the technical situation of an ICT network based on log file mining, and are subsequently used to compare situations across networks or with historic data. In order to do this comparison, alignment mechanisms are applied to perform a near real-time discovery of patterns. Alignment "*[...] is a way of arranging the sequences to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences.*" [Mou04].

Similar to anomaly detection in biology, we use available high-performance algorithms for anomaly detection in large-scale computer networks. After the alignment, situational awareness is being established. This process is supported by rule-based data interpretation and correlation, and incorporates human intelligence to infer higher knowledge (information) from low-level data.

4 Design and Implementation

We implemented a system that follows the proposed approach[8] to learn about its feasibility and applicability in every-day situations. The architecture, as depicted in Figure 1, is structured in three layers. In order to detect anomalies based on logged data, its components are traversed from the bottom to the top.

1. **Log File Management and Refactoring** is about collecting log information from numerous sources using a wide variety of protocols and formats.
2. **Anomaly Detection** is about discovering events that differ from every-day situations with minimal human intervention and configuration effort.
3. **Reporting and Configuration** implements an administration interface that allows tuning the whole system on the one side, and receiving reports about significant events on the other side.

Log File Management: The basic idea is that most applications produce logging information about their internal state by default. However, typically this data is not created for security monitoring, but is used for traceability (Web access logs), statistics (network load) or trouble-shooting. Using this logging data for detection of security incidents, as current SIEM solutions do, requires the application of special parsers or error detectors for

[8] Notice, currently we focus on data mining to create a model of normal system behavior (learning), while we keep the comparison or synchronization (knowledge-exchange) of models for future work.

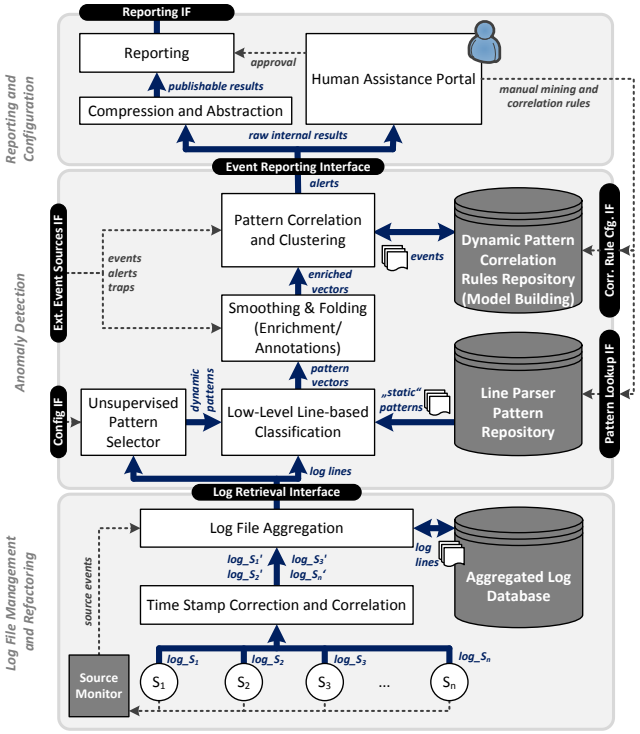


Figure 1: Framework architecture.

each type of data. Even with parsers available, it is hard to detect security-critical aberrant behavior of authorized clients with acceptable false-positive rates, e.g. a project member's notebook copying one file after another from the project file-space at night and uploading them to an Internet storage. Thus, we aggregate data from distributed logging facilities and store them in a single logging database to establish a coherent view – and later detect deviations from a regular mode of operation using our analysis approach. However, special care must be taken to (i) keep the temporal order of log entries intact; and (ii) to not artificially introduce a performance bottleneck. Here it is essential to pre-process logs to keep the temporal order, which is challenged through machines in different time zones and not being synchronized, e.g. using NTP.

Current Technical Realization: Numerous sophisticated log management products are available to meet these requirements (cf. Section II) which are able to collect log files from different sources, aggregate them and store them in a flexible format. We particularly picked rsyslog to transport log messages over the network, and Graylog2 as storage and management solution.

Anomaly Detection: Having all required information in one consistent database enables us to start the outlined anomaly detection approach. As a first step of this approach we require a classification of infrastructure events. Such events are for instance the instantiation

of a component/applet or similar on a Web server, the creation of a database connection, the login of a user etc. However, if the system works with a predefined catalog only, it might be inflexible for future extensions (or costly to maintain), and easy to crack. Thus periodic updates, supported by machine-learning concepts and heuristics are required to even cover reoccurring events that have not been pre-classified.

Referring to Figure 1, a log line is classified by determining matching patterns. Here, a *pattern vector* reflects if a log line contains certain patterns and thus describes the type of log line (and similarities to other lines). Patterns are taken from a predefined set (static patterns) and from an intelligent pattern selector (e.g., based on statistical techniques; see WEKA[9]). Smoothing and Folding enables the framework to enrich the pattern vector with additional components that are the result of basic averaging operations, noise suppression or similar refactoring techniques. Then, pattern correlation and clustering is applied to gain higher level knowledge, such as the co-occurrence of events and thus to capture complex situations. For example, assume a call to a particular site of a Web shop will always trigger the creation of a database connection and issue five database queries. All these events are captured in the log files, and log lines are thoroughly classified by appropriate set of patterns. In case of a system crash or an SQL injection attack where database calls are issued without loading Web shop pages in the same way, the correlation of these events with ratios (1:1:5) will differ, pointing out something abnormal. Not only the ratio, also time dependency will change: normal web requests will cause short database requests perhaps 3-50ms later, attack-induced database requests may have longer duration or different timing. Also notice, this disturbance alone may not be an ample indicator, however, if not only three indicators but tens to hundreds are monitored, the system is able to derive a sophisticated view on the system's health state and degree of abnormal usage behavior.

The *External Event Sources Interface* enables to feed in data from sources not under scrutiny to be correlated with events extracted from logging databases. An example is the input of news about capabilities of current malware and exploitations in other organizations – which should be used to also increase the sensitivity of the anomaly detection mechanisms regarding similar threats.

1. Define set of suitable patterns (pattern vector) for log−line classification.
   a. Human−assisted static definitions based on expert knowledge
   b. Machine−based dynamic detection of reoccurring log sequences
2. Parse available log entries on the fly and count defined patterns.
3. If number of log lines without matching patterns is greater than a configured threshold then reexecute step 1.
4. Classify log−lines using selected patterns, correlate classification results:
   a. Count number of occurrences per line, classify to get event type
   b. Randomly add new hypotheses to the correlation model, but mark them as untested
   c. Compare temporal event type occurrence with correlation model (hypothesis testing)
   d. Reject unverified hypotheses after some time, keep only those with good predictability
5. Trigger alerts when hypotheses with formerly good predictability fail:
   a. in case of missing or additional events violating a hypothesis
   b. In case of altered temporal behavior not predicted by the hypothesis

Listing 1: High-level definition of anomaly detection..

[9] `http://www.cs.waikato.ac.nz/ml/weka/`

Current Technical Realization: We have implemented a module in Java which retrieves and processes logging data using the outlined algorithm in Listing 1. Patterns are selected either manually or randomly. Results are passed on to the SIEM solution OSSIM via its plugin interface for sophisticated alert visualization.

Reporting and Configuration: This block deals with the customization of the anomaly detection approach, and the overall reporting of discovered incidents to higher levels. These levels can be on a department, organizational or national layer. Furthermore, such systems tend to output huge amounts of information which decision makers are unable to handle. For this purpose, we require appropriate compression and/or filtering techniques in order to forward only "relevant" messages. Relevant are those messages, whose content can contribute to establishing situational awareness (e.g., through a proper visualization) and, finally, support decision makers. As a matter of fact, forwarded messages become more abstract, aggregated and filtered the higher they move up the hierarchy.

Current Technical Realization: We utilize the popular SIEM solution OSSIM to visualize security-relevant events. The advantage of OSSIM is that multiple instances are easily stackable, which means the same solution can be used on different layers, i.e., departments, organizations, and national level. For that purpose OSSIM provides a sophisticated rule-engine to define detection and evaluation rules (human-assisted anomaly detection), as well as forwarding conditions (filters) for classifying security incidents.

5 Mode of Operation: A Real-World Scenario

Use Case Description: A company is running an internal Web service providing confidential information for their staff only. It is located in a separate network zone, isolated from the corporate LAN by a firewall. An attacker has managed to gain access to another server in that zone and tries to access the internal Web service from that server. All requests are syntactical correct and authorized, so that there are no errors or access violations in the log files. Finally, when the Web service is accessed the normal way, two machines will record logging information: the firewall fw-1 will report each TCP connection from the corporate LAN to the isolated zone and the Web service web-2 will log each request.

Basic Data Processing: The first step in our approach is to collect the logging data from the two different sources fw-1 and web-2 using standard remote logging tools, e.g. rsyslog, and then combine the two streams to one ordered stream of log entries. The combined stream for normal access might look as depicted in Listing 2.

```
1 Dec 17 17:13:11 fw-1 iptables:ACCEPT IN=eth2 OUT=eth0 MAC=00:50... SRC=192.168.0.11 DST
      =192.168.100.5 PROTO=TCP SPT=55325 DPT=443
2 Dec 17 17:13:11 192.168.0.11 web-2 -- "GET /" 200 "Mozilla/4.0 (compatible; MSIE 7.0; ...
3 Dec 17 17:13:11 192.168.0.11 web-2 -- "GET /images/background.gif" 200 "Mozilla/4.0 (compa..
4 Dec 17 17:13:12 192.168.0.11 web-2 -- "GET /main-style.css" 200 "Mozilla/4.0 (compatible; MSI ...
5 Dec 17 17:13:19 fw-1 iptables:ACCEPT IN=eth2 OUT=eth0 MAC=00:50... SRC=192.168.0.11 DST
      =192.168.100.5 PROTO=TCP SPT=55327 DPT=443
6 Dec 17 17:13:19 192.168.0.11 web-2 -- "POST /login" 200 "Mozilla/4.0 (compatible; MSIE 7.0; ...
7 Dec 17 17:13:19 192.168.0.11 web-2 -- "GET /service-overview" 200 "Mozilla/4.0 (compatible; ...
```

Listing 2: Log snippet representing normal access behavior.

Listing 3 and 4 show collected logging data when the attacker is accessing the service from within the zone. Hence, there are no corresponding log entries from the firewall fw-1, since the attacker's connections do not traverse the firewall. Also the Web service logs may vary, depending on the tools the attacker is using. Access patterns with a normal Web browser will be quite similar to normal access, where e.g. a text-based browser or automated tool might skip loading of images and style sheet data, since it cannot process them anyway.

```
1 Dec 17 17:13:11 192.168.100.7 web-2 -- "GET /" 200 "Mozilla/4.0 (compatible; MSIE 7.0; ...
2 Dec 17 17:13:11 192.168.100.7 web-2 -- "GET /images/background.gif" 200 "Mozilla/4.0 (c...
3 Dec 17 17:13:12 192.168.100.7 web-2 -- "GET /main-style.css" 200 "Mozilla/4.0 (compat...
4 Dec 17 17:13:19 192.168.100.7 web-2 -- "POST /login" 200 "Mozilla/4.0 (compatible; MSIE 7.0; ...
5 Dec 17 17:13:19 192.168.100.7 web-2 -- "GET /service-overview" 200 "Mozilla/4.0 (com...
```

Listing 3: Log snippet representing access from within the isolated zone.

```
1 Dec 17 17:13:19 192.168.100.7 web-2 -- "POST /login" 200 "wget"
2 Dec 17 17:13:19 192.168.100.7 web-2 -- "GET /service-overview" 200 "wget"
```

Listing 4: Log snippet representing access from within the isolated zone with wget.

The correlation system tries to learn from logging information under normal operation, so that it can then detect the occurrence of new, aberrant behavior, both in the data domain, e.g. new or missing pattern strings, but also in the time domain, e.g. different sequences or frequencies of patterns. When feeding of the normal, repeated baseline logging information from Listing 2 to the correlation system will act in the following way: First, all logging items are structurally unknown or are unlikely to match any correlation rules already learned. The system cannot make any prediction on the frequency of those items. Hence it will generate alerts and notify the administrator (or a member from the security response team) about unexpected data. When the responsible staffs confirm that the new input is acceptable, the system starts to learn from it. Therefore it draws random samples from the input data marked in bold in Listing 5.

```
1 Dec 17 17:13:11 fw-1 ip tables:ACCEPT IN=eth2 OUT=eth0 MAC=00:50... SRC=192.168.0.11 DST=192.168.10
      0.5 PROTO=TCP SPT=55325 DPT=443
2 Dec 17 17:13:11 192.168.0.11 web-2 -- "GET /" 200 "Mozilla/4.0 (compatible; MSIE 7.0; ...
```

Listing 5: Randomly extracted patterns (in bold font) in the start-up phase.

The bold sample patterns from Listing 5 are then used to classify the log lines as shown in Listing 6.

```
1 Pattern vector: ("w-1 ip", "h2 OU", "0.5 P", "T=553", "PT=44", "2.168" "GET /", "IE 7.")
2 Match vector on first line: (1, 1, 1, 1, 1, 1, 0, 0)
3 Match vector on next line: (0, 0, 0, 0, 0, 1, 1, 1)
```

Listing 6: Instances of the pattern vector extracted from Listing 5.

As one can see, the pattern "w-1 ip" is suitable to classify a firewall log line, while the pattern "T=553", meaning source port is 553..., will match only occasionally. The classification algorithm will continuously try to improve the current set of search patterns, e.g. by picking more patterns from log lines for evaluation, removing patterns that just match randomly or fusing adjacent pattern parts. As a result the pattern vector may evolve to that given in the first line in Listing 7. Using this vector, all logging items of the initial log

snippet in Listing 2 can be encoded by the classifier (cf. Figure 1) to binary vectors (see Listing 7).

```
1 Pattern vector: ("fw-1..MAC=", "DST=..SPT=", "DPT=443", "192.168.0.", "GET /", "images", "IE 7.")
2 Dec 17 17:13:11: v0=(1, 1, 1, 1, 0, 0, 0)
3 Dec 17 17:13:11: v1=(0, 0, 0, 1, 1, 0, 1)
4 Dec 17 17:13:11: v2=(0, 0, 0, 1, 1, 1, 1)
5 Dec 17 17:13:12: v3=(0, 0, 0, 1, 1, 0, 1)
6 Dec 17 17:13:19: v4=(1, 1, 1, 1, 0, 0, 0)
7 Dec 17 17:13:19: v5=(0, 0, 0, 1, 0, 0, 1)
8 Dec 17 17:13:19: v6=(0, 0, 0, 1, 1, 0, 1)
```

Listing 7: Evaluation of the initial log snippet with the evolved pattern vector.

Using the timestamp of the log entry plus the classification vector, the correlation algorithm tries to create random hypotheses (modeled as rules in our system) and subsequently test them. For example, the algorithm discovers that all vectors v1 to v6 always follow a vector v0. It can also find out, that v5 (HTTP POST) can never be the first request, because the client has always to request the login page first before sending the login data via POST. Hence a regular GUI-based browser without caching[10] would also cause a log item with classification v2 (background image) when loading the login page. For each hypothesis, the correlation algorithm tries to determine the statistical significance. So, for example, the significance of the rule "firewall-log item at most 5sec before Web server log entry" will have a high significance. Hence an attacker connecting from within the trusted network will cause a mismatch of this correlation rule. The correlation system will then either generate an alert immediately or wait for more of these unlikely events to increase the significance of its prediction.

Extended Analysis: Depending on the input, the correlation algorithm will find multiple hypotheses with high significance – some of them a human administrator who configures a SIEM solution might not think about. For example, if a company uses only Microsoft products on the machines usually accessing the Web service, then additional hypotheses can be generated, e.g. each "GET" Web server request is sent from Internet Explorer (User-Agent signature "IE7"). When a staff member or WLAN-attacker connects a device to the network and accesses the service, the user-agent signature will probably not match and trigger an alert. Just using the simple seven-component vector from this real-world use case allows the creation of a multitude of different hypotheses. Some of them are given here, but not elaborated in detail for the sake of brevity:

- Each firewall log line with a connection to the server IP usually contains the destination port 443. Access to a different port, e.g. a maintenance port or the database port is unusual.
- Usually login attempts (POST) are rare, compared to other service requests. An increase of these requests indicates brute-forcing of passwords.

[10]Caching behavior of browsers is not a severe problem for the algorithm: if it cannot find suitable highly reliable hypotheses for characterizing the access to cached resources, then those log-lines will have little to no influence on the anomaly-score. However, if caching also introduces new patterns, e.g. all company browsers use caching and ask for the resource just once, receiving a "304 not modified" response, then unusual access behavior (e.g., caused by wget) may again violate a hypothesis on this behavior and hence triggers anomaly alerts.

- Usually the number of firewall log items correlates to the number of Web service log items: a human user will send a request (firewall reports new connection), receive some Web service responses, e.g., a page with up to 15 different images and then human will read the received information before clicking on the next link. This will open the next connection. An automated tool extracting all the information from the system, e.g. "wget", will send requests with much higher frequency and different ratios of firewall to Web service requests.

Use Case Conclusion: This example shows, that a correlation system can potentially detect significant anomalies, a human might not have thought about yet. It is then up to the network administrator and the security response team respectively to evaluate the logging information and rules that caused an alarm and to decide, if the discovered anomaly was indeed caused by an incident or just a rare error, e.g., the result of system maintenance actions. From this example one can also easily see that a hypothesis-based correlation algorithm can be trained to detect different forms of attacks by learning from logging information. The only chance for a traditional SIEM solution to detect the described abnormal behavior is to have a specific rule set that triggers on e.g. the atypical source IP or the user-agent string in the Web server logs. These specific rules have to be added manually by the administrators (which causes high efforts) and can only detect anomalies the administrator has considered in advance.

6 Evaluation and Discussion

We run a scenario similar to that described in the previous section, however, now we let multiple instances of a hypothetical document exchange system copy files. All these services are under scrutiny, thus events from all services are collected, aggregated and finally correlated in this closed environment. We expect our system to find anomalies, if an attacker manages to get access to this closed environment from outside.

Pattern Frequencies: Our first experiment deals with the extraction of significant patterns from logging-data collected via rsyslog. Here, we apply a one second time-window (due to the precision of the timestamps) to evaluate raw pattern frequencies. The investigated patterns are "GET", "LIST", "Connecting to", "Connection from", "iptables * IN" and "iptables * OUT". Figure 2 depicts the individual pattern counts over time slots, i.e., the raw frequencies of the patterns. Notice that the raw frequencies themselves are highly variable because service interactions vary over time. Nevertheless, one can already recognize – if s/he knows what to look for – the attack between second 651 and 661 which leads to significant deviations between pattern-E and pattern-F – which is not the case in normal operation.

Basic Correlation: In order to properly visualize correlation deviations, the system implements a basic correlation of patterns, here: "Connecting to" (pattern-E) and "iptables" (pattern-F). The timely evaluation of the hypothesis "Connecting to implies iptables" discovers connection attempts that bypass the firewall (and thus reflect unusual connection attempts from a potentially infected machine in the isolated network segment). Figure 3

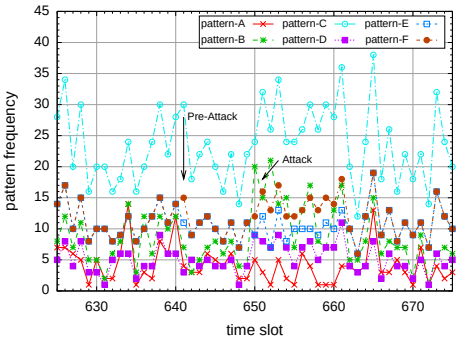
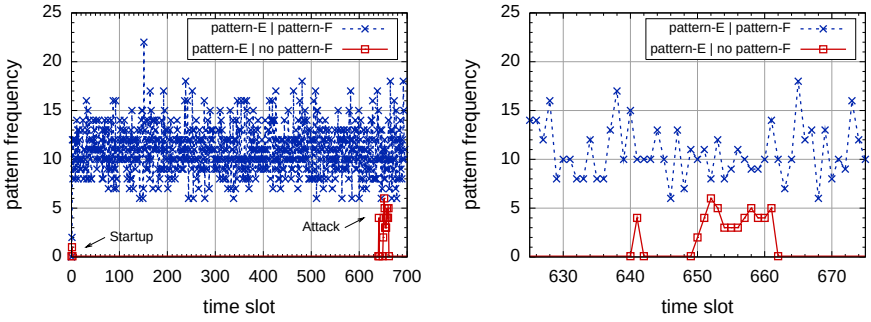


Figure 2: Raw pattern frequencies showing the relevant time around an attack.

shows how often this hypothesis evaluates to true (pattern-E | pattern-F: usual case), and how often this correlation rule evaluates to false (attack case). The attack can be easily identified in this obvious case. Notice that Figure 3 is based on the same data as Figure 2.



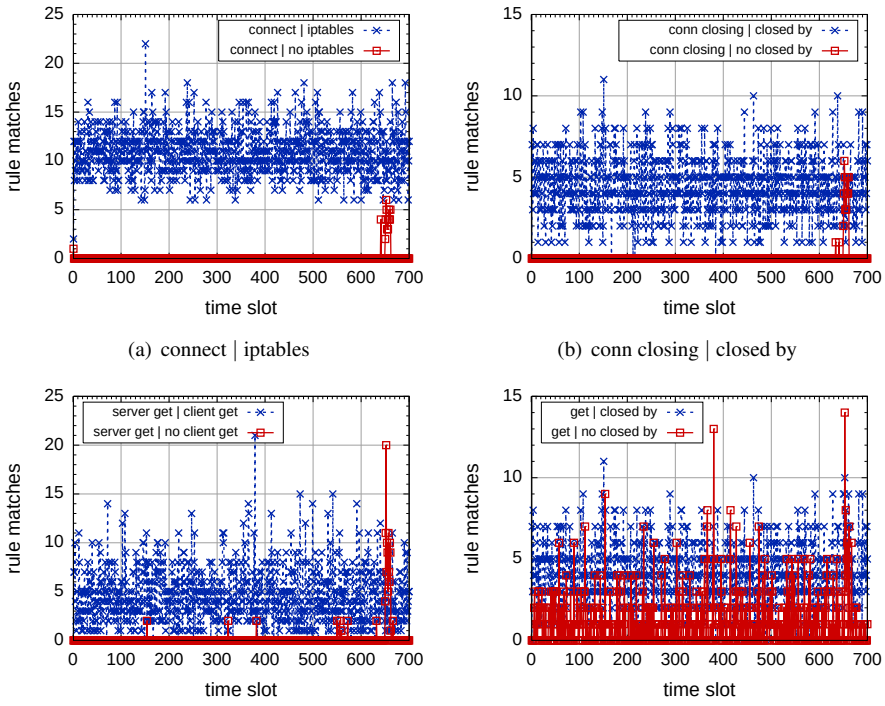
(a) over the whole run-time.

(b) around the attack.

Figure 3: Basic correlation.

Extended Correlation: Next, we demonstrate the application of a set of manually defined correlation rules. Even in the quite simple example we can demonstrate various dependencies between pattern vector components (see Figure 4). Similar to the example before, we identified further correlated patterns, such as ”connection closing” (server) and ”closed by” (client) or ”server get” and ”client get” in case of document exchanges. The discovery of these rule sets effectively allows building up a system model to learn about the common usage behavior of the ICT system. As a counter example, one case (”get” and ”closed by”) is given, where events are not correlated, neither in the normal case, nor in the case of an attack. This experiment shows that – even for small use cases – a set of correlation rules can be found to characterize abnormal behavior. The combination of all these correlation rules enables us to determine significantly abnormal behavior with high probability.

Self-Learning Operation: The manual definition of search patterns and correlation hy-



(a) connect | iptables

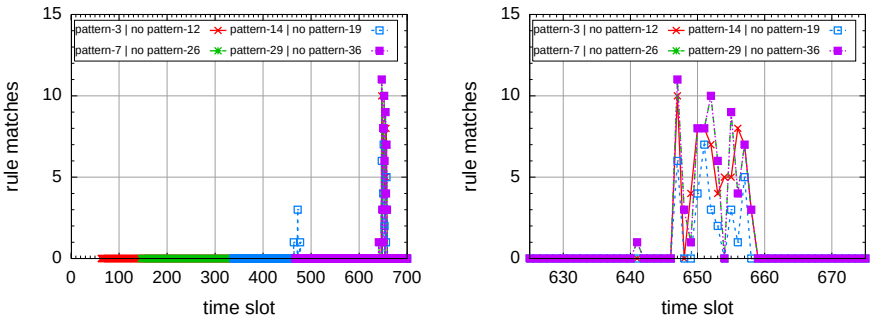
(b) conn closing | closed by

(c) server get | client get

(d) get | closed by

Figure 4: Parallel application of manual correlation rules: three feasible cases of highly correlated patterns (top left and right, and bottom left) and one counter example (bottom right).

potheses, as well as testing of these hypotheses would be a time-consuming task. Therefore the anomaly detection system continuously tries to optimize its model using self-learning techniques. This process, as currently used and one of many applicable techniques, works as follows: For each processed log-line one token is added at a virtual bucket. New random log-line search patterns are selected as soon as enough tokens are available. If no pattern matches a log-line and more than $t_{unmatched\_line}$ tokens are in the bucket or if a log-line was matched n-times and the bucket contains more than $t_{unmatched\_line} + n * t_{match}$ tokens, a new pattern of 3-11 bytes length is selected. In our setup $t_{unmatched\_line} = 100$ and $t_{match} = 300$ was used. To limit resource consumption, patterns not included in any higher-level correlation rule should be removed from time to time. The pattern vectors are then used to classify the log-lines. The classification rules are randomly generated using the same token-bucket approach (100/300) and contain a random selection of 1-to-n bits (log-distributed) from the pattern-vector. When applied, the classification rules determines to which classes a log-line be-longs to. The timestamp of the log-line and the class type are combined to form a log-event element. These log-events are then correlated using randomly generated hypotheses. A similar token-bucket approach is used, requiring 200 tokens to add a rule for a new log-event, and 300 tokens more for an event which is already part of a rule. A rule is generated as such that occurrence of event-A implies



(a) over the whole run-time.

(b) around the attack.

Figure 5: Statistically outstanding sequence of matches per timeslot of 4 different correlation rules (38 selected patterns are correlated with 200 generated rules). Notice the different start times of the evaluation rules in the left figure. Compare the right figure with Figure 3 (manual selection).

that also event-B is found in the time window [-1sec, 0sec] (p=3/8) [0sec, 1sec] (p=3/8), [-10sec, 0sec] (p=1/8) or [0sec, 10sec] (p=1/8). All currently stated hypotheses are evaluated using the log-events. It is counted, how many times event-A occurred and implied event-B was also found within a time window and also how many times event-A occurred with event-B missing. For each rule, this match/mismatch count is statistically analysed to find significant changes: the number of matches/mismatches per 10 second are recorded for 300 seconds. The average and standard deviation is then used to evaluate, if the counts observed in the next 10x10 seconds are within a 6*sigma range around the mean of the 300 seconds before. If not, then an anomaly concerning this rule is reported. This approach is applied to the same data as used in the previous studies, and is still able to detect outstanding sequences of matches in case of an attack (Figure 5). Thus, we conclude that (for the given scenario) the learning approach is able to detect an attack with a similar accuracy, however with considerably less system configuration effort.

7 Conclusion and Future Work

In this paper, we proposed a novel anomaly detection system that is based on concepts from the domain of bioinformatics to be applied in future large-scale ICT networks. Its design objectives are to provide a holistic perspective on a multitude of services (and their logs/events respectively), to learn about higher level events through massive data correlation, and to detect patterns reflecting malicious behavior across service boundaries. The main advantage of our system is that it is able to learn from unstructured data without the need to understand the actual content, by applying a range of statistical methods to detect significant patterns on its own. Thus, the system has the potential to automatically adapt to changing situations. For that purpose, we outlined the basic application of a self-learning mechanism for pattern selection. However, the application of sophisticated machine-based learning techniques is up to future work. Other future work deals with the

actual integration of this system in a larger setting, especially where computer networks are often extended and reconfigured. This way, we are able to discover the actual performance of our approach, its configuration efforts (e.g., additional pattern selection and correlation rule design through human intervention), and of course its limitations.

Acknowledgment

This work was partly funded by the Austrian security-research program KIRAS (operated by the FFG) and by the Austrian Ministry for Transport, Innovation and Technology (BMVIT) in course of the project CAIS. This work was further supported by the EU FP7 research project PRECYSE (Reference FP7-SEC-2012-1-285181).

References

[ATS+03] Cristina Abad, Jed Taylor, Cigdem Sengul, William Yurcik, Yuanyuan Zhou, and Kenneth E. Rowe. Log correlation for intrusion detection: A proof of concept. In *ACSAC*, pages 255–264. IEEE Computer Society, 2003.

[BJ04] Emilie Lundin Barse and Erland Jonsson. Extracting attack manifestations to determine log data requirements for intrusion detection. In *ACSAC*, pages 158–167, 2004.

[BSBK09] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks. In *WWW*, pages 551–560. ACM, 2009.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41:1–58, July 2009.

[EMC12] EMC Press Release. Rsa chief rallies industry to improve trust in the digital world, after year filled with cyber attacks. http://www.emc.com/about/news/press/2012/20120228-03.htm, 2012.

[GT+09] Pedro Garcia-Teodoro et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, 2009.

[HS09] A. Hassanzadeh and B. Sadeghiyan. A data correlation method for anomaly detection systems using regression relations. In *ICFIN*, pages 242–248, 2009.

[KTK02] Christopher Kruegel, Thomas Toth, and Clemens Kerer. Decentralized event correlation for intrusion detection. In *ICISC*, pages 59–95. 2002.

[LEK+03] Aleksandar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *SDM*. SIAM, 2003.

[LLW02] Sin Yeung Lee, Wai Lup Low, and Pei Yuen Wong. Learning fingerprints for a database intrusion detection system. In *ESORICS*, pages 264–280. Springer, 2002.

[LTP+04] Zhenmin Li, Jed Taylor, Elizabeth Partridge, Yuanyuan Zhou, William Yurcik, Cristina Abad, James J. Barlow, and Jeff Rosendale. Uclog: A unified, correlated logging architecture for intrusion detection. In *ICTSM*, 2004.

[LZP+07] Peng Li, Chaoyang Zhang, Edward J. Perkins, Ping Gong, and Youping Deng. Comparison of probabilistic boolean network and dynamic bayesian network approaches for inferring gene regulatory networks. *BMC Bioinformatics*, 8(S-7), 2007.

[MHL94] Biswanath Mukherjee, L. Todd Heberlein, and Karl N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, 1994.

[Mou04] David W. Mount. *Bioinformatics: sequence and genome analysis*. CSHL press, 2004.

[SMSB12] Florian Skopik, Zhendong Ma, Paul Smith, and Thomas Bleier. Designing a cyber attack information system for national situational awareness. In *Future Security*, pages 277–288, 2012.