Shedding Light on Log Correlation in Network Forensics Analysis

Elias Raftopoulos, Matthias Egli, and Xenofontas Dimitropoulos

ETH Zurich, Switzerland
{rilias,fontas}@tik.ee.ethz.ch, eglima@ee.ethz.ch

Abstract. Presently, forensics analyses of security incidents rely largely on manual, ad-hoc, and very time-consuming processes. A security analyst needs to manually correlate evidence from diverse security logs with expertise on suspected malware and background on the configuration of an infrastructure to diagnose if, when, and how an incident happened. To improve our understanding of forensics analysis processes, in this work we analyze the diagnosis of 200 infections detected within a large operational network. Based on the analyzed incidents, we build a decision support tool that shows how to correlate evidence from different sources of security data to expedite manual forensics analysis of compromised systems. Our tool is based on the C4.5 decision tree classifier and shows how to combine four commonly-used data sources, namely IDS alerts, reconnaissance and vulnerability reports, blacklists, and a search engine, to verify different types of malware, like Torpig, SbBot, and FakeAV. Our evaluation confirms that the derived decision tree helps to accurately diagnose infections, while it exhibits comparable performance with a more sophisticated SVM classifier, which however is much less interpretable for non statisticians.

Keywords: Network forensics, IDS, Malware, Infections

1 Introduction

Computer Security Incident Response Team (CSIRT) experts use a combination of intuition, knowledge of the underlying infrastructure and protocols, and a wide range of security sensors, to analyze incidents. Although, the low-level sensors used provide a source of fine-grained information, often a single source is not sufficient to reliably decide if an actual security incident did occur. The process of correlating data from multiple sources, in order to assess the security state of a networked system based on low-level logs and events is complex, extremely time consuming and in most parts manual. Although, thorough manual investigation is critical in order to collect all the required evidence for a detected breach and to make a definite assessment regarding the severity of an investigated incident, it would be highly beneficial for administrators to have tools that can guide them in the log-analysis process, helping them to diagnose and mitigate security incidents.

A large number of previous studies have analyzed how to aggregate, correlate, and prioritize IDS alerts. A survey can be found in [24]. However, aggregated IDS alerts are then passed to a security analyst for manual diagnosis, which is a complex, typically ad-hoc process that leverages multiple security sources, like blacklists, scanning logs, etc. In this work we focus on this process with the goal of understanding how to correlate *multiple* security data sources and how to expedite manual investigation.

For this purpose, we conduct a complex experiment turning a human security analyst into the subject of our analysis. We systematically monitor the used evidence and the decisions of an analyst during the diagnosis of 200 security incidents over a period of four weeks in a large academic network. Based on the analyzed incidents, we build a decision tree using the C4.5 algorithm that reflects how low-level evidence from the four security sources can be combined to diagnose different families of malware, like Torpig, SbBot, and FakeAV. The derived model is useful for expediting the time-consuming manual security assessment of security incidents. It accurately encodes a large part of the decisions of the analyst in correlating diverse security logs and can serve as a decision support tool helping an analyst identify the most critical features that suggest the presence of an infection. In addition, we show that using the decision tree for fully-automated classification correctly identifies infections in 72% of the cases.

Finally, we ask the question if other state-of-the-art classifiers exhibit better performance than a C4.5 decision tree, which is highly interpretable and therefore useful as a decision support tool. We compare its detection accuracy with a support vector machine (SVM), a Bayesian tree classifier (BTC), and a tree-augmented naive Bayes (TAN). We find that a C4.5 decision tree is better than BTCs and TANs and only slightly worse than the more sophisticated SVM, which however is much less interpretable.

In summary, in this work we make the following contributions:

- We outline a number of features useful for security assessment that can be extracted from four commonly-used data sources.
- We build a decision support tool that clearly depicts how evidence from four sources should be correlated to diagnose different types of malware. We show that our decision tree is 72% accurate in automatically classifying suspected infections.
- We compare our decision tree with other classifiers and show that state-of-the-art SVMs, which are more sophisticated but much less understandable, have only slightly better performance.

In the next section we describe in detail the data and features we used. In Section 3 we provide a brief summary of our experiment. Next, in Section 4 we present our decision support tool and in Section 5 we compare its performance to other alternatives. Finally, in Section 6 we discuss related work and we conclude in Section 7.

2 Data Sources and Feature Extraction

In this section, we review in detail the data we used and the features we extracted. We conducted all our experiments in the network of the main campus of the Swiss Federal Institute of Technology at Zurich (ETH Zurich). We use four data sources. IDS alerts provide a view of malicious activity from the gateway of the studied network. Reconnaissance and vulnerability reports provide fine-grained information about local hosts, like the client or server role of a host, the running services, and the associated vulnerabilities. Finally, blacklists and search engine queries provide two additional views that are particularly useful for remote hosts. More details about the extracted features can be found in [22].

2.1 IDS Alerts

Our IDS data is comprised of raw IDS alerts triggered by a Snort sensor [25] that monitors all the upstream and downstream traffic through the main border link of the network of the main campus of ETH Zurich. The sensor is configured with the official Snort signature ruleset and the Emerging Threats (ET) ruleset [7], which are the two most commonly-used Snort rulesets.

We use IDS alerts in two ways. First, IDS alerts form the input to an IDS alert correlator we developed in [21], which detects infected hosts that exhibit a recurring multi-stage alert pattern involving specific classes of alerts. In particular, the correlator first aggregates similar alerts, then it classifies aggregate alerts into three classes relating to an *Attack*, a *Compromised host*, or a *Policy*, and finally it uses alerts of the first two classes to detect internal hosts that exhibit a recurring multi-stage alert pattern. During our experiment, the alert correlator processed 37 million Snort alerts and detected 200 infected hosts, which were thoroughly analyzed further using data from four security sources.

Secondly, we further exploit IDS alerts during the manual investigation of suspected hosts. Given the IP address of an infected host and the timestamp of the infection, we retrieve the aggregate IDS alerts of the classes *Attack* and *Compromised host* that were observed within 24 hours before or after the timestamp. From the aggregate IDS alerts of these two classes we extract the following features:

– *Suspicious remote hosts*: If the communication to a remote host triggers more than 10% of the total number of aggregate alerts of a local host, we deem the remote host suspicious and mark its IP address for further investigation. We select the 10% threshold empirically based on the alert volume distribution for remote hosts. This feature is useful to identify common malicious domains used by infected hosts to receive instructions, share data, or update their malicious binary.
– *Suspicious remote services*: We aggregate the activity of all non-privileged ports (port numbers above 1024) into a single port with label *High*. If more than 10% of the aggregate alerts target a specific remote port, then we consider this service suspicious. This feature is useful to identify targetted

services, e.g., worms performing a distributed scan for vulnerable services or spamming bots.
- *Suspicious local services*: If a local service port is involved in more than 10% of the aggregate alerts, then we consider it suspicious. Again, we aggregate the activity of all non-privileged ports into a single port with label *High*. This feature is useful to detect malware that attach to popular software such as IE, Firefox, Skype, or CuteFTP.
- *Count of severe alerts*: We count the total number of aggregate alerts. This feature is important to detect malware that generate spurts of high severity alerts. It helps to distinguish high activity malware from more stealthy ones.
- *Infection duration*: We compute the time in hours that elapsed between the first and the last triggered alert within the observed daily interval. We only take into account hourly slots where at least one alert from any class, including policy alerts, was triggered. This feature enables to normalize the volume of alerts of a suspected host over time.
- *Common severe alerts*: If a specific alert accounts for more than 5% of total number of aggregate alerts, then we build a new feature for its alert ID. This feature targets malware that have a very consistent network footprint triggering always the same set of IDS alerts.

2.2 Reconnaissance and Vulnerability Reports

We next actively probe suspicious internal local hosts to collect information about running services and vulnerabilities. We use this information to evaluate if the software and operating system (OS) a node is susceptible to the malware reported in the corresponding IDS alerts. We first scan a host using Nmap, and then we use the Nessus [11] and OpenVas [12] vulnerability scanners to build a comprehensive profile of the vulnerability status of a node.

In summary, we extract the following features from reconnaissance and vulnerability reports:

1. *Host Reachability:* This binary feature indicates if a host is reachable. Nodes behind a firewall or a NAT will typically be unreachable.
2. *Host Role:* We exploit hostname keywords, such as `proxy-XX.ethz.ch` and `guest-docking-nat-YY.ethz.ch`, to determine the role of a host. This feature takes the values *client*, *dns-server*, *web-server*, *ftp-server*, or *unknown-server*.
3. *OS and active services:* For each open service in a host we set a corresponding bit in a bitmap of all observed services. Each bit is treated as a separate feature in our classification. In addition, we use Nmap OS fingerprinting and encode the most likely OS match into an additional feature.
4. *Vulnerability data:* We collect vulnerability reports from Nessus and OpenVas and use this information in the manual diagnosis performed in Section 3. However, we exclude vulnerability data from the feature space used in the classification scheme discussed in Section 4, since it drastically increases the

dimensionality of the input data. For example, we have seen that a host running an unpatched version of Windows 7 typically has more than 60 active vulnerabilities.

2.3 Blacklists

The third security source we exploit is blacklists. Blacklists are commonly-used to identify IP addresses and domains that have been reported to exhibit malicious activity. We use them to investigate hosts in the *suspicious remote hosts* feature, which is extracted from IDS alerts. We leverage five public blacklist providers [5, 13, 10, 6, 3]. The blacklists are partly labeled providing information about the reason a host was enlisted, including the type of malicious activity it was involved in, e.g., bot activity, active attack, and spamming. For each local host, we lookup the corresponding suspicious remote hosts in the blacklists and count the number of hits with a specific label. The count of each label forms an input feature for our classifier.

2.4 Search Engine

A lot of useful information about remote hosts resides on the web coming from several diverse sources such as DNS lists, proxy logs, P2P tracker lists, forums, bulletins, banlists, etc. In order to exploit this information, we query the Google search engine using as input string the IP address of an analyzed host and the respective domain name. For each suspected internal host we query for the contacted IP addresses in the *suspicious remote hosts* feature. Then, in an automated fashion we parse the output and extract tags, like *malware*, *spam*, *trojan*, *worm*, *bot*, *adaware*, *irc*, and *banlist*, based on the methodology of [26]. The list of tags we used can be found in [22].

3 Forensics Analysis Experiment

In this section we briefly describe our forensics analysis experiment. More information on the validated infections and the diagnosis process along with four example malware cases can be found in [22].

We used the Snort alert correlator we developed in our previous work [21] to detect infected hosts within the monitored infrastructure. During our experiment, we ran our correlator on the latest Snort alerts and we passed on a daily basis newly detected infections to an analyst for manual inspection and validation. Our experiment lasted for approximately four weeks between 01.04.2011 and 28.04.2011 during which we thoroughly investigated 200 consecutive infections. We limited our study to nodes with static IP addresses, which correspond to the majority of the active nodes within the monitored network. Besides, we built automated tools to extract the features discussed in Section 2 and to present them on a dashboard in order to facilitate the investigation process. The analyst would typically have to check the quality of collected signatures, examine the network footprint generated by the studied host, gather information about the expected behavior of the investigated malware, and most importantly cross-correlate this information. We validated the activity of a specific malware type for 170 out of the 200 infections. We use this set of validated infections to build our decision support tool in Section 4.

4 Decision Support Tool

In this section we introduce a decision tree that captures how to correlate the key evidence from the four security sources to identify different types of malware. Our decision tree is useful for expediting the complex and time-consuming manual correlation of multiple data sources for the diagnosis of infected hosts.

We use the C4.5 decision tree induction algorithm, which is a state-of-the-art tree-based classifier [20]. Studies have shown that its performance is better than BTCs and TANs, whereas it is comparable to SVMs [14]. Moreover, it is computationally efficient and an open-source implementation is publicly available. For our purposes, the most important aspect of C4.5 is the interpretability of its results. It is important that a security analyst can understand which feature contributed in every step of the process of a decision, without requiring expert statistical knowledge such as in the case of SVMs. The classification is performed using a tree, where each internal node corresponds to an intermediate decision based on one or more features and the leaf nodes correspond to the final decision.

We use the J48 implementation of the C4.5 algorithm [27]. It takes as input a sample of vectors that correspond to the manually classified hosts. Each vector captures the values of different security features of a host. We use in total 131 security features we described in Section 2. Secondly, C4.5 takes as input the class of each host.

In Figure 1 we show the derived decision tree. The tree accurately depicts the main decisions of the manual process followed in the investigation of security incidents highlighting the most important signs of infection that can drive forensics analysis. Using the decision tree we can easily identify critical signs of malicious behavior. In the following paragraphs, we provide examples of how to combine evidence to detect specific types of malware.

Zbot-infected hosts are prominent spammers. We see that they generate a high percentage of high severity alerts that are related to destination port 25. These correspond to spamming attempts for which Snort raises an alert. Moreover, we see that they typically attempt to share stolen confidential data with an HTTP POST request on a malicious domain. This typically triggers the IDS alert 2013976:"*ET TROJAN Zeus POST Request to CnC*". Periodically, the bot attempts to upgrade its binary triggering alerts with ID 2010448:"*ET MALWARE Potential Malware Download, trojan zbot*".

Besides, *SdBot*-infected hosts exhibit frequent communication with their C&C. The bot attempts to identify a valid communication channel from a set of predefined rendez-vous domains in order to update its instruction set and to potentially post valuable client data it has intercepted. These attempts trig-

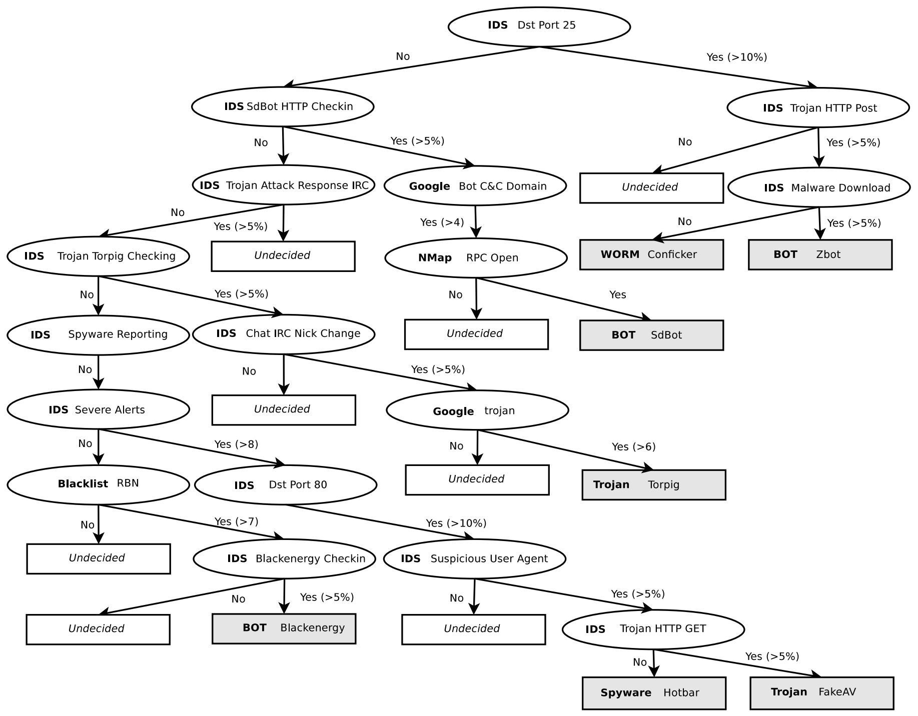

Fig. 1. Decision tree generated from the C4.5 algorithm using training data from 200 manually examined incidents

ger alerts with ID 2007914:"*ET WORM SDBot HTTP Checkin*". Moreover, the malware uses MS network shares to propagate and therefore we see that on most of the infected machines port 135 is open, which corresponds to the RPC service.

The *Torpig* trojan periodically attempts to post using HTTP the data it has stolen from a victim triggering the Snort alert "*ET TROJAN Sinowal/Torpig Checkin*" with ID 2010267. Also, Torpig typically uses IRC to receive updates resulting in frequent IRC nickname changes, detected by the Snort rule 542: "*CHAT IRC nick change*". The domains used to upload harvested user data, i.e., `vgnyarm.com`, `rajjunj.com` and `Ycqgunj.com`, were tagged with the keyword *trojan* by our search results.

Finally, *FakeAV* is a trojan that intentionally misinterprets the security state of the victim and generates pop-ups attempting to redirect the user to domains where the victim can purchase software to remediate the malware. This activity generates a high number of alerts with ID 2002400:"*ET USER_AGENTS Suspicious*" that are related to port 80 (HTTP) activity.

From our analysis for building the decision tree we highlight the following key observations:

- A small number of features is sufficient to make an assessment with high certainty. In most studied cases these features reflect different stages of the lifecycle of a malware. C4.5 decision trees retain a high level of interpretability assisting the analyst in making an assessment using a manageable number of intuitive features.
- Combinations of features yield more accurate results. Multiple security data sources are required to detect a wider range of malware types exhibiting complex behavioral patterns. In contrast, simple rules of thumb such as 'if the IDS triggers N alerts of type X then there is an infection' cannot be effectively used.

Finally, we note that C4.5 can be used in an adaptive fashion leveraging a feedback loop with the analyst, who can update the set of classified infections in order to enrich the derived tree and improve the classification results. Model induction is very efficient even for datasets involving a large number of features.

5 Automated Diagnosis

In this section we analyze the effectiveness of our decision support tool in fully automated diagnosis. We also compare how C4.5 performs against other state-of-the-art classifiers. Specifically, we evaluate the classification accuracy of two tree-based classifiers, namely BTCs and TANs. We use their WEKA implementation with the default parameters. We also evaluate the performance of an SVM, which is a state-of-the-art classifier. To configure the SVM parameters we use the sequential minimal optimization method [17].

Table 1. Performance of different classification algorithms.

Malware Type	C4.5		Bayesian Tree		TAN		SVM	
(#incidents)	TP (%)	FP (%)	TP (%)	FP (%)	TP (%)	FP (%)	TP (%)	FP (%)
Trojans (85)	83	10	80	12	82	10	89	6
Spyware (59)	85	4	85	5	85	4	88	4
Backdoors (18)	55	8	53	7	56	7	63	5
Worms (8)	75	1	75	1	75	1	77	1
Undecided (30)	60	10	48	14	51	13	63	9

In Table 1 we summarize our findings. C4.5 exhibits on average a true positive rate of 72% whereas the false positive rate does not exceed 7%. Bayesian networks and TANs are worse exhibiting a true positive rate of 68% and 70% and a false positive rate of 8% and 7%, respectively. On the other hand the SVM achieves slightly better classification results with a true positive rate of 76% and a false positive rate that does not exceed on average 5%.

6 Related Work

Previous studies have extensively studied the aggregation and correlation of IDS alerts with the goal of generating high-level inferences from a large number of low-level alerts. A group of studies exploit statistical correlation to perform causality inference and root cause analysis of detected incidents [23, 18, 19]. A second group of studies hardcode expert knowledge by introducing scenarios [16, 2, 15] or sets of rules [1, 21] that capture observed malicious behavior. These studies mine solely IDS alerts without taking into account complementary sources of security logs that are often available. They produce and prioritize inferences that at the end are passed on to a security analyst for manual inspection. Our work is complementary and focuses on the manual verification of aggregated IDS alerts by correlating data from multiple instead of a single source.

Besides, a number of commercial solutions, such as IBM Tivoli SCM [9], Alienvault [4], and GFI Languard [8], unify scattered security sensors within an enterprise and provide a single framework that can be used by security analysts to configure security sensors and to visualize logs. However, log correlation in these systems is based on simple rules that need to be determined by an administrator. In our work, we encode a number of classification rules in a C4.5 decision tree that can form the input to such systems.

Finally, in our previous work we developed a Snort alert correlator [21] and we characterized a number of aspects of approximately 9,000 infections we detected over a period of nine months in a large academic infrastructure. We also validated our correlator based on the experiment we describe in this paper. In this work, we describe a number of additional lessons we learned from our experiment and we build a decision support tool to facilitate network forensics analyses.

7 Conclusions

Network forensics analysis can be likely better described as art rather than science. It relies on a security expert combining his reasoning with background knowledge about malware, domain specific knowledge about the target environment, and available evidence or hints from a number of diverse security sensors, like IDS systems, vulnerability scanners, and other. We believe that in the future processes for handling and diagnosing security incidents should be largely automated diminishing (but not completely removing) the involvement of humans in the loop.

In this work we analyze the decisions of an analyst during the diagnosis of 200 security incidents over a period of four weeks. We show that a large part of the decisions can be encoded into a decision tree, which can be derived in an automated fashion. The decision tree can help as a decision support tool for future incident handling and provides comparable performance in fully automated classification with a more advanced classifier, an SVM, which however is much less interpretable.

8 Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments and suggestions. Furthermore, we wish to thank Prof. Bernhard Plattner and Dr. Vincent Lenders for their invaluable help and fruitful discussions. We would also like to thank Stephan Sheridan and Christian Hallqvist at ETH for their help in the collection and archiving of the data used in this paper.

References

1. F. Cuppens and A. Miège. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*.
2. H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Proceedings of the 4th International RAID Symposium*, UK, 2001.
3. Advanced automated threat analysis system. `http://www.threatexpert.com`.
4. AlienVault. `http://www.alienvault.com/`.
5. Anonymous postmasters early warning system. `http://www.apews.org`.
6. Cooperative Network Security Community. `http://www.dshield.org`.
7. Emerging Threats web page. `http://www.emergingthreats.net`.
8. GFI Languard. `http://www.gfi.com/network-security-vulnerability-scanner`.
9. IBM Tivoli SCM. `http://www-01.ibm.com/software/tivoli/`.
10. Shadowserver Foundation web page. `http://www.shadowserver.org`.
11. The Nessus vulnerability scanner. `http://www.tenable.com/products/nessus`.
12. The Open Vulnerability Assessment System. `http://www.openvas.org`.
13. The Urlblacklist web page. `http://www.urlblacklist.org`.
14. Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29:131–163, November 1997.
15. Benjamin Morin and Herv Debar. Correlation of intrusion symptoms: an application of chronicles. In *RAID03*, pages 94–112, 2003.
16. Peng Ning, Yun Cui, and Douglas S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th CCS ACM conference*, 2002.
17. John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.
18. Xinzhou Qin. *A probabilistic-based framework for infosec alert correlation.* PhD thesis, Atlanta, GA, USA, 2005. AAI3183248.
19. Xinzhou Qin and Wenke Lee. Statistical causality analysis of infosec alert data. In *RAID 2003*, pages 73–93, 2003.
20. J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1:81–106, March 1986.
21. Elias Raftopoulos and Xenofontas Dimitropoulos. Detecting, validating and characterizing computer infections in the wild. In *Proceedings of IMC '11*, NY, USA.
22. Elias Raftopoulos and Xenofontas Dimitropoulos. Technical report : Shedding light on data correlation during network forensics analysis. Technical Report 346, 2012.
23. Hanli Ren, Natalia Stakhanova, and Ali A. Ghorbani. An online adaptive approach to alert correlation. In *Proceedings of the 7th international DIMVA conference.*
24. Reza Sadoddin and Ali Ghorbani. Alert correlation survey: framework and techniques. In *Proceedings of PST '06*, pages 37:1–37:10, New York, NY, USA. ACM.
25. A free lightweight network IDS for UNIX and Windows. `http://www.snort.org`.
26. I. Trestian, S. Ranjan, A. Kuzmanovi, and A. Nucci. Unconstrained endpoint profiling (googling the internet). *SIGCOMM Comput. Commun. Rev.*, 2008.
27. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2 edition, 2005.