2013 IEEE Security and Privacy Workshops

Understanding Network Forensics Analysis in an Operational Environment

Elias Raftopoulos
ETH Zurich
Communication Systems Group
Zurich, Switzerland
rilias@tik.ee.ethz.ch

Xenofontas Dimitropoulos
ETH Zurich
Communication Systems Group
Zurich, Switzerland
fontas@tik.ee.ethz.ch

***Abstract*— The manual forensics investigation of security incidents is an opaque process that involves the collection and correlation of diverse evidence. In this work we conduct a complex experiment to expand our understanding of forensics analysis processes. During a period of four weeks we systematically investigated 200 detected security incidents about compromised hosts within a large operational network. We used data from four commonly-used security sources, namely Snort alerts, reconnaissance and vulnerability scanners, blacklists, and a search engine, to manually investigate these incidents. Based on our experiment, we first evaluate the (complementary) utility of the four security data sources and surprisingly find that the search engine provided useful evidence for diagnosing many more incidents than more traditional security sources, i.e., blacklists, reconnaissance and vulnerability reports. Based on our validation, we then identify and make available a list of 138 good Snort signatures, i.e., signatures that were effective in identifying validated malware without producing false positives. In addition, we compare the characteristics of good and regular signatures and highlight a number of differences. For example, we observe that good signatures check on average 2.14 times more bytes and 2.3 times more fields than regular signatures. Our analysis of Snort signatures is essential not only for configuring Snort, but also for establishing best practices and for teaching how to write new IDS signatures.**

***Keywords*-Network forensics, IDS, Malware, Infections**

I. Introduction

Security analysts are overwhelmed by massive data produced by different security sources. Investigating security incidents is an opaque "art" that involves 1) carefully extracting and combining evidence from the available security sources; 2) thoroughly understanding how suspected malware operate; and 3) exploiting information about the infrastructure and configuration of the affected network. In this arena, security analysts are restricted to using slow manual and often ad-hoc forensics analysis processes.

Towards understanding and improving forensics analysis processes, in this work we conduct a complex experiment in which we systematically monitor the manual forensics analysis of live suspected infections in a large production university network that serves tens of thousands of hosts. In particular, over a period of four weeks, we manually investigated in coordination with the IT department of our university 200 security incidents about compromised hosts detected by an IDS alert correlator. The security investigation combined data from four security sources: 1) Snort alerts, 2) reports from four scanning and vulnerability assessment tools, 3) five independent blacklists, and 4) a search engine (Google).

Based on our experiment, we describe a number of lessons we learned from the validation of security incidents. In particular, we make three contributions. First, we describe how to leverage four different security data sources to remotely diagnose live infections in a large production network. Second, to delineate the manual investigation process, we evaluate the (complementary) utility of the four data sources. Surprisingly, we find that a search engine was one of the most useful sources in deciding if a suspicious host was infected, providing useful evidence that led to a positive diagnosis in 54.5% of the cases. Reconnaissance and vulnerability reports were useful in fewer cases, but helped diagnose more sophisticated malware, whereas blacklists were useful only for 10.5% of the incidents. In addition, we report which combinations of sources helped diagnose different types of malware.

Third, we make available a list of 138 Snort signatures that were effective in detecting validated malware without producing false positives. We analyze the differences between good and regular Snort signatures and find, for example, that good signatures check on average 23.5 Bytes in 2.8 different fields, while regular signatures check on average only 11 Bytes in 1.2 fields. In addition, we observe that good signatures tend to use offsets, regular expressions, fixed packet sizes, and specific destination ports much more often than regular signatures. Based on these observations, we highlight good signature characteristics useful for configuring Snort and for establishing signature writing best practices.

The remaining of our paper is structured as follows. In the next section we describe the data we used. Then, in Section III we describe the investigation of four representative malware cases. We present our findings regarding the utility of the data sources and the effective signatures in Section IV and V, respectively. Finally, in Sections VI and VII we outline related work and conclude our paper.

DOI 10.1109/SPW.2013.12

II. Data Collection

Our monitored infrastructure is the main campus network of the Swiss Federal Institute of Technology at Zurich (ETH Zurich). The campus is a "zoo" of diverse systems, like critical servers, desktops, laptops, and other lab and office devices. The traffic mix we observe is rich since there are almost no restrictions on the software users can install on their devices and on the services they can use. Our monitoring period lasted for approximately four weeks between the 1st and the 28th of April 2011, during which we observed in total 28,665 unique active internal hosts.

We select four data sources that provide complementary views into the security status of a host from different vantage points covering aspects, like its traffic, the running services, and their vulnerabilities. These sources are commonly used for security assessment. First, we collect IDS alerts generated by an IDS, which is located between the primary border router and the network firewall of the monitored network. The IDS monitors all upstream and downstream traffic and generates an alert when a malicious signature is triggered. The IDS gives a view of malicious activities from the gateway of the network.

To collect information related to services and applications running on internal hosts, we perform reconnaissance and active scanning using NIC *whois* querying and Nmap. After mapping the accessible network services of a host, we investigate for known vulnerabilities using the Nessus [9] and OpenVas [10] vulnerability scanners. These four tools, to which we collectively refer as reconnaissance and vulnerability scanners, give a more detailed view of the internal hosts. The last two data sources help extract information about remote hosts by leveraging publicly available data. In particular, we use blacklists from five blacklist providers covering different types of malicious hosts, like active attackers and spammers, and we query the Google search engine for suspected remote hosts and domains. The search engine indexes a variety of sources, including public forums, bulletins, and banlists, which we exploit in an automated way to find the roles, i.e., server type, or actions of remote hosts. The detailed description of each feature can be found in the accompanying technical report [18].

A. IDS Alerts

The IDS deployed in our infrastructure is Snort [22], which is commonly considered the open source IDS solution with the largest number of registered users and the most active community [8], [4]. Snort is configured to use signatures from the two most widely-used publicly available rulesets, namely the Vulnerability Research Team (VRT) ruleset and the Emerging Threats (ET) ruleset [6]. As of April 2011 the two rulesets have in total 37,388 distinct signatures.

We use IDS alerts in two ways. First, we apply an effective IDS alert correlator we have introduced in our previous work [17] to derive a small set of incidents, which we comprehensively analyze to evaluate their validity. Second, during the forensics analysis we manually examine the alerts of an investigated host and extract a number of features regarding the type and severity of the alerts, the size and duration of an incident, and the involved services and hosts. These features are presented to the analyst along with additional information extracted from the other three data sources and are used for the manual diagnosis of incidents. In addition, in many cases the analyst manually examined the specific signatures of Snort rules to assess their trustworthiness.

We had to address two challenges when analyzing Snort alerts. First, the amount of alerts is overwhelming making the analysis very time-consuming. In total we collected 37 million alerts over a period of four weeks, the majority of which are policy alerts that are not directly related to security incidents of interest. Second, the alerts are dominated by false positives, which makes it very hard to have any certainty about the existence of a malware based solely on a single alert. Our Snort alert correlator distills a small number of events that exhibit a recurring multi-stage malicious pattern involving specific types of alerts. It first aggregates similar alerts that exhibit temporal proximity, it then classifies each aggregate alert as *Direct attack*, *Compromised host*, or *Policy violation*, and finally it infers active infections by identifying internal hosts that exhibit a recurring multi-stage network footprint involving alerts of the first two classes. Applying our correlator on raw Snort alerts results into a small number of highly suspicious events. In particular, during the four weeks of our experiment, the correlator distilled 200 aggregate events from 37 million raw alerts.

B. Reconnaissance and Vulnerability Reports

In order to measure the exposure of a studied host to external attacks, we additionally collect host-based information about the running services, the patch level of deployed software, and the presence of vulnerabilities. This information is complementary to the network behavior patterns we get from Snort alerts, since it helps identify the threats that a host is susceptible to.

We use a combination of network scanning and vulnerability assessment techniques. In particular, we first use basic reconnaissance techniques, like IP sweeps, NIC whois querying, and TCP/UDP port-scanning, in order to identify if a host is reachable and exposed to external attacks. In addition, these techniques help determine the role of host, e.g., web, mail, or DNS server, within the infrastructure.

Secondly, we perform targeted network scanning using Nmap in order to retrieve information regarding the TCP and UDP network services running on suspected hosts, details about the type and version of their operating system, and information regarding the types of ICMP messages a host responds to, which reveal its filtering policies and firewall

effectiveness. After having detected the accessible network services, we investigate a host for known vulnerabilities. For this purpose, we use two well-known vulnerability scanners, namely Nessus [9] and OpenVas [10], in order to build a comprehensive profile of the vulnerability status of a host.

C. Blacklists

In order to examine if an examined host within our network frequently initiates connections to known malicious domains, we use a set of independent blacklists. We leverage five public blacklists [3], [11], [7], [5], [2], which are partly labeled indicating the type of malicious activity exhibited by a blacklisted host, e.g., bot activity, active attack, and spamming.

We then investigate whether the remote hosts contacted by an internal host are listed in a blacklist. If there is a hit then we tag the investigated host with the label of the corresponding blacklist. This method is useful to identify if suspicious internal hosts frequently establish communication with known malicious domains. Such communication typically occurs when a user visits malicious websites or when a piece of malware installed on the infected host attempts to perform unsolicited actions, e.g., redirecting to third-party domains, updating its binary, sharing stolen confidential data, or getting instructions from a remote controller.

D. Search Engine

Apart from using intrusion detection alerts, active scans, and blacklists in order to characterize remote hosts exhibiting frequent communication with local investigated hosts, we also exploit security-related information residing on the web. When a host exhibits a malicious activity it will leave traces in different forms in publicly available sources such as DNS lists, website access logs, proxy logs, P2P tracker lists, forums, bulletins, banlists, and IRC lists. To retrieve this information we query the Google search engine using as input the IP address and the respective domain name of the local and remote hosts. In an automated manner we parse the output looking for a set of pre-selected tags such as *malware*, *spam*, *bot*, *trojan*, *worm*, *pop3*, *netbios*, *banlist*, *adaware*, *irc*, *undernet*, *innernet*, *torrent*. A similar approach has also been used in [23]. These tags reveal specific actions a host has taken, e.g., receiving instructions from a known botnet C&C, or roles it had for extended periods of time, e.g., operating as a mail server.

III. Evidence Correlation Studies

In this section, we first describe the manual malware validation experiment we conducted and then we outline example evidence correlation cases for four different types of malware.

Our experiment used Snort alerts that were pushed in an hourly basis to an alert archival infrastructure we are operating since 2009. We configured our alert correlator with the default configuration parameters [17]. We restricted our analysis to static IP addresses, which are widely-used in the monitored network. Detected incidents about infected hosts were annotated with information about the IP address of the host, the involved alerts, and a timestamp. A security analyst on a daily basis (during week days) manually investigated the latest detected incidents. Our experiment was conducted in cooperation with the IT department of ETH Zurich. To expedite the manual analysis process we developed a number of tools that took as input the IP address of an investigated host, collected the data described in the previous section, extracted a large number of associated features, and displayed all the relevant information on a dashboard. The manual analysis process was complex and very time-consuming as it often required processing and analyzing a huge amount of IDS alerts, checking the quality of their signatures, collecting information about malware, and most importantly cross-correlating all this information. We investigated in total 200 consecutive incidents over approximately four weeks from the 1st until the 28th of April 2011. In the following paragraphs we present representative cases of the manual investigation performed for four different types of malware.

Case 1: Torpig infection. Torpig is one of the most sophisticated trojans in the wild. The typical method of infection is Drive-by-Downloads. The victim visits a vulnerable legitimate web site that requests Javascript code from a malicious webserver. The code is then executed and the trojan attaches itself to popular applications on the victim's system. Torpig is a typical data harvesting trojan using both passive monitoring and active phishing techniques to get valuable confidential data from an infected machine. Our active scanning for Torpig-infected hosts shows that Windows is the operating system of the host, whereas the services HTTP, Skype, and FTP are typically open. HTTP is the protocol used by the malware to contact the C&C servers and also to redirect the user to malicious domains, whereas Skype and CuteFTP are standard applications Torpig injects itself into. Torpig periodically contacts the C&C servers to get instructions and to update its binary triggering IDS alerts with IDs 2002762 and 2003066. Also, frequently it will attempt to report stolen user data using a POST command on known malicious domains triggering alerts with IDs in the range [2404000:2404300]. The domains we saw that were used for the POST operations were vgnyarm.com, rajjunj.com and Ycqgunj.com. The dominant tags produced by our search engine profiling method were *trojan* and *bot*.

Case 2: SdBot infection. W32/SdBot is a backdoor used by cyper-criminals to gain unauthorized access to victim machines. It can be used to launch active attacks or to harvest sensitive user data from an infected host. We observe that W32/SdBot-infected hosts are typically running MS network services such as WebDav, RPC, and LSASS. The malware exploits the MS-LSASS buffer overflow, the MS-RPC

malformed message buffer overflow, and the MS-WebDav vulnerability to compromise a victim. W32/SdBot uses a typical IRC communication method to contact its C&C servers to update its configuration triggering Snort alerts with IDs in the range [2500000:2500500]. Also, the malware will often attempt to propagate using extensive port scanning to detect vulnerable hosts, mostly on port 445, triggering alerts with IDs in the range [2011088:2011089]. Finally, the W32/SdBot backdoor will attempt to fetch additional badware from remote web sites to install them on the local host. The infected hosts we analyzed attempted to connect via FTP to the domains joher.com.tw and service.incall.ru, which are typically used to download W32/Koobface and Trojan.FakeAV, respectively. The tags we extracted from our search engine query results for these domains indicated that they are involved in *malware* and *bot* related activities.

Case 3: Hotbar infection. Win32/Hotbar is the most prominent infection detected in our infrastructure. Typically it comes as a web-browser add-on, like the Asksearch or Mysearch toolbar, providing seemingly legitimate functionality. However, in the background it tries to harvest user activity patterns including browsing habits and application usage. This spyware does not attempt to hide and it will often identify itself using the User-Agent field of an HTTP request using the string "AskTB". Snort uses alerts with IDs in the range [2003305:2003500] to detect this behavior. Moreover, the spyware will regularly attempt to redirect a user to the domains bestdealshost.biz and zangocash.biz by generating clickable pop-ups. These domains are typically tagged by our search engine analysis for hosting malware.

Case 4: Koobface infection. W32.Koobface is a worm using social networking sites such as Facebook and MySpace to propagate. It typically links to popular videos, in order to convince a user to install an executable that appears to be a necessary codec update but is in fact the Koobface executable. After successful infection it will attempt to propagate by sending messages to the victim's contact list. This is done by issuing HTTP POST requests, which are detected by the Snort signatures with IDs 2009156, 2010335, and 2010700. Typical landing pages used for the redirection in order to fetch the Koobface executable are prospect-m.ru and pari270809.com, which are tagged as suspicious by our Google profiling method generating the tags *bot* and *worm*.

IV. Complementary Utility and Ranking of Security Sources

In this section we present our results on the complementary utility of the four security data sources for validating different types of malware. In Table I we list the malware variants that were identified. We classify malware into four categories, namely backdoors, spyware, worms, and trojans, and for each malware we indicate in the second column the relevant category. Note that the behavior of modern malware often combines multiple characteristics, which seriously perplexes the process of putting real-world malware into a taxonomy. For this reason, in few cases malware could also be assigned to a different category.

For 30 out of the 200 analyzed incidents, our investigation did not lead to a definite assessment even when combining evidence from all sources. The remaining 170 validated incidents include 85 trojans, 59 spyware, 18 backdoors, and 8 worms. The most popular malware family was AskSearch followed by FakeAV and Simbar. In the last four columns of Table I we identify the combination of sources that were useful for identifying the corresponding type of malware. In 41.5% of the cases two sources and in 14% of the cases at least 3 sources had to be combined to have high certainty about the diagnosis of an investigated host. The correlation of multiple sources was particularly useful for the diagnosis of more sophisticated malware, like Torpig, SdBot, and Polybot. On the other hand, a single source was useful to identify AskSearch and MySearch.

In our previous short paper [16], which complements this work, we additionally mapped the forensics analysis process into a decision tree, and used the results of the manual investigation to train the C4.5 tree classifier. In this way we encoded the knowledge we derived from the manual assessment of security incidents presented here, to a decision support tool, which can be a significant aid in the diagnosis of future incidents.

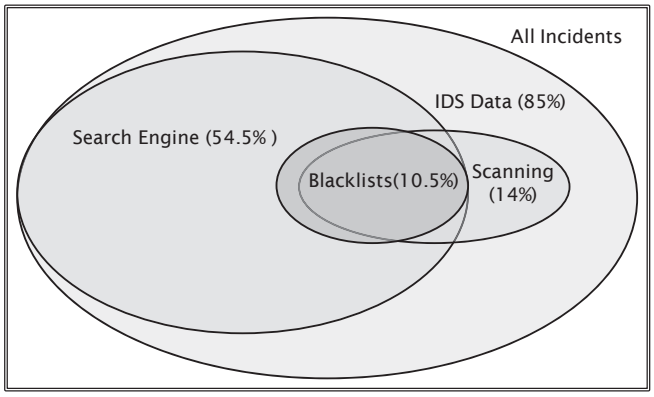

Figure 1. Complementary utility of security data sources for the diagnosis of 200 incidents

In Figure 1 we depict how often a source or a combination of sources were useful for diagnosing an infection, which illustrates the complementary utility of the four data sources. Based on our experiments, we rank the four sources as follows:

1) **IDS alerts:** IDS alerts are the most fruitful source of information since they were the basis for our assessment. In 170 out of the 200 investigated incidents, the further manual examination of IDS alerts observed near the time of a detected incident provided useful evidence for validation.

Table I
PREVALENCE OF DIFFERENT MALWARE TYPES AND VARIANTS IN THE 200 INVESTIGATED INCIDENTS. THE LAST FOUR COLUMNS MARK THE DATA SOURCES THAT PROVIDED USEFUL EVIDENCE FOR DIAGNOSIS.

Malware Type (#incidents)	Variant (#incidents)	IDS Logs	Search Engine	Blacklist Data	Active Scans
Trojans(85)	FakeAV(27)	✓	✓		
	Simbar(26)	✓	✓		
	Monkif(18)	✓	✓		
	Torpig(10)	✓	✓	✓	✓
	Nervos(4)	✓	✓		
Spyware(59)	AskSearch(50)	✓			
	MySearch(9)	✓			
Backdoors(18)	SdBot(5)	✓	✓	✓	✓
	ZBot(5)	✓	✓		✓
	Blackenergy(4)	✓	✓	✓	✓
	Parabola(2)	✓	✓		✓
	Ramsky(2)	✓			✓
Worms(8)	Koobface(6)	✓	✓		
	Conficker(2)	✓	✓		✓

2) **Search Engine:** The second most useful data source in our arsenal was the output of our automated search engine queries. In 54.5% of the cases, query results were a valuable resource providing critical information for the analysis. This information was in most cases complementary to the content of IDS alerts, providing knowledge about active C&C hosts, botnet communication channels, malware landing pages, redirections to malicious domains, malware download pages, and phishing forms.
3) **Reconnaissance and Vulnerability Reports:** On the other hand, the information we collected by scanning suspicious hosts helped us to reach a definite assessment in approximately 14% of the cases. However, these were the most demanding inferences involving sophisticated malware that exhibit a very complex behavior like *Torpig* or *SdBot*.
4) **Blacklists:** Finally, blacklists were the least valuable source of security information. Their output partially overlapped with information we already extracted from IDS alerts or from Google. Moreover, they contain a very large number of false positives, we speculate due to slow responsiveness in enlisting new malicious hosts [21].

Below we summarize the *main lessons* we learn from our evaluation of the utility of different security sources:

Insight 1. No single sensor provides conclusive evidence about an investigated incident. Relying on a single defensive mechanism might be sufficient to detect automated threats with deterministic and predictable activity. However, *most modern threats exhibit complex behaviors that require a multi-vantage point security architecture for effective detection.*

Insight 2. IDS sensors have been heavily criticized for producing a large number of false positives, rendering them unreliable if used in isolation for identifying active infections. In this work we highlight that by *exploiting statistical and temporal correlations of the alert stream, our IDS data became very useful, since they helped identify a few actionable cases that exhibited a high likelihood of infection.*

Insight 3. Alternative security sources, such as the output we get from the profiling method using the Google search engine, proved to be more helpful for making an assessment than traditional security sources, such as vulnerability reports and blacklists. This highlights the *importance of endpoint profiling for domains visited by infected hosts. The reputation and typical activity of these remote hosts provides invaluable pieces of evidence that can drive the investigation process.* Moreover, *blacklists should only be used as a low quality indicator of domain reputation. Search engine content seems to be more reactive and up-to-date regarding changes in a host's activity compared to the information contained in blacklists.*

V. WHAT A GOOD IDS SIGNATURE LOOKS LIKE?

Next, we provide a list of Snort signatures that were found by our alert correlator useful for detecting confirmed malware without generating false positives. These signatures are useful for configuring the widely-used Snort IDS and for teaching good signature writing practices. They are based on the 170 validated incidents and are effective in detecting the malware types listed in Table I.

Our alert correlator finds tuples of aggregated alerts that occur frequently together. For each tuple involved in a validated incident, we extracted the corresponding Snort signatures. We found in total 138 Snort Signature IDs (SID) that can be summarized into 25 aggregate signatures as several affiliate SIDs detect small variations of the same pattern. In Table II we provide the 138 SIDs and a short

Table II
EFFECTIVE SNORT SIGNATURES IN IDENTIFYING MALWARE INFECTIONS THE 200 INVESTIGATED INCIDENTS.

SID	Signature Description
[C&C Communication] Update malicious binary instruction set.	
2007668	ET TROJAN Blackenergy Bot Checkin to C&C
2010861	ET TROJAN Zeus Bot Request to CnC
2404138:2404156,2404242:2404247,2404335:2404347	ET DROP Known Bot C&C Server Traffic TCP/UDP
16693	SPYWARE-PUT Torpig bot sinkhole server DNS lookup attempt
2802912	ETPRO TROJAN Backdoor.Nervos.A Checkin to Server
2011365, 2010267	ET TROJAN Sinowal/sinonet/mebroot/Torpig infected host checkin
[Reporting] Share stolen user confidential data with controller.	
2008660	ET TROJAN Torpig Infection Reporting
2011827	ET TROJAN Xilcter/Zeus related malware dropper reporting in
2009024	ET TROJAN Downadup/Conficker A or B Worm reporting
2002728	ET TROJAN Ransky or variant backdoor communication ping
2010150	ET TROJAN Koobface HTTP Request
[Egg download] Update malicious binary/ Download additional malware.	
2010886	ET TROJAN BlackEnergy v2.x Plugin Download Request
2802975	ETPRO TROJAN Linezing.com Checkin
1012686	ET TROJAN SpyEye Checkin version 1.3.25 or later
2010071	ET TROJAN Hiloti/Mufanom Downloader Checkin
[Redirection] Redirect user to malicious domain.	
2011912	ET CURRENT_EVENTS Possible Fake AV Checkin
2003494:2003496	ET USER_AGENTS AskSearch Toolbar Spyware User-Agent
2003626,2007854	ET USER_AGENTS Suspicious User Agent (agent)
2009005	ET MALWARE Simbar Spyware User-Agent Detected
2406001:2406012,2406147:2406167,2406361:2406383,2406635:2406649	ET RBN Known Russian Business Network IP TCP/UDP
[Propagation] Detect and infect vulnerable hosts.	
2008802	ET TROJAN Possible Downadup/Conficker-A Worm Activity
2003068	ET SCAN Potential SSH Scan OUTBOUND
2001569	ET SCAN Behavioral Unusual Port 445 traffic
2000347	ET ATTACK_RESPONSE IRC - Private message on non-std port
12798:12802	SHELLCODE base64 x86 NOOP

description of the 25 aggregate signatures classified in five classes based on the behavior they detect.

A signature is labeled as good if the following conditions are met: it is contained in a tuple generated by our correlator, it is used as evidence in the subsequent manual assessment process, and the respective incident is validated by a security expert as an infection. Note that signatures which generate a large number of false positives and might coincidentally get triggered during a security incident, will be in most cases filtered out by our correlator [17]. Otherwise they will be discarded as irrelevant during the manual assessment process.

The malware we studied typically undergo a sequence of stages after the initial infection. In many cases, they attempt to contact their C&C to receive new instructions and/or to report stolen data. Periodically they may attempt to update their binary or to fetch additional malware, which are installed on infected hosts. Some types of malware (e.g. clickbots) often attempt to redirect a user to known malicious pages by changing search engine results, by generating pop-ups, or by redirecting HTTP requests. Finally, most trojans and worms in our study also attempt to propagate by scanning for vulnerabilities. Based on these, we classify signatures in one of the following five categories (see Table II): C&C communication, reporting, egg download, redirection, and propagation.

In Figure 2 we show an example of a good signature that looks for a beacon sent by a Blackenergy bot using an HTTP POST command to its controller. The signature checks for specific byte sequences in four different fields of the HTTP packet. In addition, it attempts to match the identification string sent by the infected host providing information about the build of the malicious binary. Additional conditions regarding the maximum packet size, the target ports, the state of the connection, and the exact format of the identification string are introduced to increase its specificity.

We next compare key characteristics of our good signatures to the average of the signatures that are triggered in our infrastructure, which uses the default VRT and ET rulesets. We created two groups of signatures. A group with the extracted 138 good signatures and its complement group of 3,198 regular signatures that are triggered during our experiment. In Table II we compare different features of the two groups. We observe that good signatures are much more complex requiring the satisfaction of multiple conditions. A good signature attempts to match on average 23.5 Bytes in 2.8 different fields, while a regular signature checks only 11 Bytes in 1.2 fields. In addition, 28% of the good signatures set the offset of a sought byte sequence; 50% provide a regular expression to further specify a search

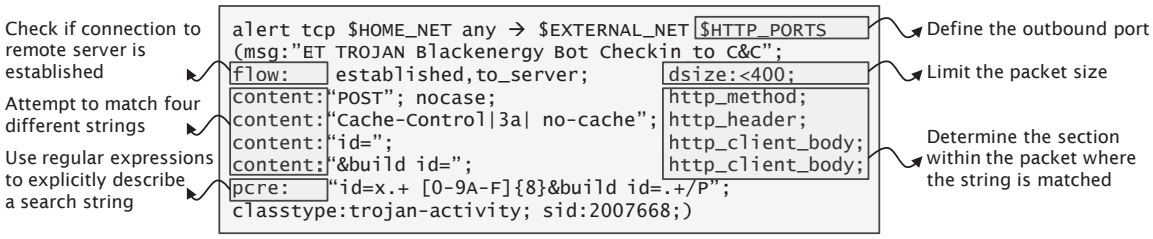

Figure 2. Example of good Snort signature used to detect a beacon frame sent by a Blackenergy bot to its controller

Table III
COMPARISON OF GOOD AND REGULAR SNORT SIGNATURES. STATISTICS ARE COMPUTED OVER 138 GOOD AND 1398 REGULAR SIGNATURES.

	Bytes Checked	Fields Checked	Byte Offset is Set	Regular Exp. is Set	Destination Port is Set	Packet Size is Set
Regular Signatures	11	1.2	8%	15%	17%	7%
Good Signatures	23.5	2.8	28%	50%	22%	15%
Increase	2.14 ×	2.3 ×	3.5 ×	3.3 ×	1.29 ×	2.14 ×

pattern; 22% fix the destination port(s); and 15% give a specific packet size. In sharp contrast, the corresponding numbers for regular signatures are only 8%, 15%, 17%, and 7%, respectively. In the case of *HTTP*, we observe that good signatures tend to define the section of an HTTP packet, e.g., with the Snort key-words *header*, *method*, *uri*, and *revision*, where pattern matching is performed. Overall, we observe that the computed statistics exhibit a consistent increase for good signatures by a factor that takes values between 1.29 and 3.5. We conclude that good Snort signatures tend to incorporate the entire arsenal of available features. Complexity in the case of signature writing is a virtue. Note, however, that this complexity typically comes at the cost of higher processing overhead for matching signatures.

Insight 4. IDS signature writing best practices often suggest that signatures should be kept short and simple. The primary reason for this is performance, since signature length and the usage of additional features handled by software modules, such as regular expressions, have a negative impact on the packet processing delay. Secondly, malware will often slightly change their behavior and communication patterns in order to evade detection. This results in a large number of similar network footprints generated from different flavors of the same malware. IDS signature writers cope with this problem by generalizing existing signatures, so that they effectively detect all different variants of a malware family. However, our work suggests that *reducing signature complexity will also reduce its effectiveness, since more generic signatures will often get triggered by benign traffic. Highly specific signatures exhibit higher true positive rate and generate in most cases a low number of alerts that can be easily handled by a security analyst.*

VI. RELATED WORK

Several studies have detected security incidents in traffic traces from production networks, e.g., [19], [24], [14], without providing though a systematic validation of detected incidents. Closer to our work, Sharma *et al*. [20] analyzed security incidents in a supercomputing center. The incidents were verified based on forensics analysis that exploited data from five security sources. The authors highlighted a number of best practices for the perimeter security of an organization. However, they do not provide insights about the effectiveness of the different security sources. In our recent short paper [16] we built a decision support tool that correlated evidence from different security sensors to expedite manual forensics analysis of compromised systems. [16] focused on the automation of the security assessment process. In contrast, in this paper we study in detail how the manual investigation of a wide range of incidents was carried out. In addition, we evaluate the complementary utility of the available security sources and highlight good practices for writing IDS signatures.

Besides, a group of studies on automating digital forensics analysis have focused on discovering and correlating evidence primarily from end hosts [1], [12], [25]. This line of work is orthogonal to ours, since their goal is to detect unauthorized user activity by combining the available host-level sources. Substantial part of these studies are centered around optimizing data representation so that evidence integrity and chain of custody is ensured [13], [15]. Our work, on the other hand is to the best of our knowledge the first that systematically documents network forensics analysis practices in an operational environment.

VII. CONCLUSIONS

In this paper we conducted a complex manual investigation of 200 suspected malware in a live operational

infrastructure. We exploited four commonly-used security data sources and a number of derived features to validate suspected infections. Based on our experiment, we analyze the complementary utility of the different data sources and the characteristics of the IDS signatures that were associated with confirmed incidents. Notably, we observe that a search engine, which is a less well-established security data source, was much more useful in the diagnosis of security incidents than other more traditional security sources, namely blacklists, active scanners, and vulnerability analysis tools.

Furthermore, we learn that a single data source is typically not sufficient to validate an incident and that multiple sources should be combined. In more than 10% of the cases, no single source, but the overall behavior of a host as seen from multiple sources helped to validate an infection. In addition, multiple sensors are needed in 70.5% of all cases when the easier to detect spyware are excluded. These results highlight the importance of a holistic approach in security assessment that combines multiple data sources. In order to detect elaborate pieces of malware, as the ones shown in Table I, we need to combine local information about the exposure level and the behavioral patterns of studied hosts with public knowledge about the maliciousness of contacted domains. In this context future work could also use a variety of tools to complement some of the sensors used in our study. For example, security inspectors (e.g. SecuniaPSI, QualysGuard), IDSs (e.g. BRO, Surricata), and NetFlow anomaly detectors can also be used to detect malicious activity, whereas spamtraps, blocklists and DNS-based reputation engines can be exploited to build profiles of contacted domains.

Finally, we extracted and made available a list of 138 Snort signatures that were triggered on hosts with validated malware. We compare the characteristics of good and regular signatures and report a number of interesting statistics that provide essential guidelines for configuring Snort and for teaching good signature writing practices. In particular, we find that good signatures attempt to match on average 23.5 Bytes in 2.8 different fields and tend to specify regular expressions, whereas regular signatures are much less complex.

VIII. Acknowledgements

The authors wish to thank Prof. Bernhard Plattner and Dr. Vincent Lenders for their invaluable help and fruitful discussions. Furthermore, we would also like to thank Stephan Sheridan and Christian Hallqvist at ETH for their help in the collection and archival of the data used in this paper. This work was supported by the Armasuisse Secmetrics [2-7841-09] project.

References

[1] Andrew Case, Andrew Cristina, Lodovico Marziale, Golden G. Richard, and Vassil Roussev. Face: Automated digital evidence discovery and correlation. *Digit. Investig.*, 5:S65–S75, September 2008.

[2] Advanced automated threat analysis system. http://www.threatexpert.com.

[3] Anonymous postmasters early warning system. http://www.apews.org.

[4] Best IDS/IPS solution. http://www.scmagazine.com/best-idsips-solution/article/130871/.

[5] Cooperative Network Security Community - Internet Security. http://www.dshield.org.

[6] Emerging Threats. http://www.emergingthreats.net.

[7] Shadowserver Foundation. http://www.shadowserver.org.

[8] The Case for Open Source IDS. http://www.itsecurity.com/features/the-case-for-open-source-ids-022607/%.

[9] The Nessus vulnerability scanner. http://www.tenable.com/products/nessus.

[10] The Open Vulnerability Assessment System. http://www.openvas.org.

[11] The Urlblacklist web page. http://www.urlblacklist.org.

[12] Simson L. Garfinkel. Forensic feature extraction and cross-drive analysis. *Digit. Investig.*, 3:71–81, September 2006.

[13] S.L. Garfinkel. Automating disk forensic processing with sleuthkit, xml and python. In *IEEE SADFE '09*.

[14] Gregor Maier, Anja Feldmann, Vern Paxson, Robin Sommer, and Matthias Vallentin. An assessment of overt malicious activity manifest in residential networks. In *DIMVA*, 2011.

[15] J. Mena. *Investigative Data Mining for Security and Criminal Detection*. Butterworth-Heinemann Limited, 2003.

[16] Elias Raftopoulos and Xenofontas Dimitropoulos. Shedding light on log correlation in network forensics analysis. In *DIMVA'12*.

[17] Elias Raftopoulos and Xenofontas Dimitropoulos. Detecting, validating and characterizing computer infections in the wild. In *ACM SIGCOMM IMC*, Berlin, Germany, November 2011.

[18] Elias Raftopoulos and Xenofontas Dimitropoulos. Technical report : Shedding light on data correlation during network forensics analysis. Technical Report 346, 2012.

[19] Stefan Saroiu, Steven D. Gribble, and Henry M. Levy. Measurement and analysis of spyware in a university environment. In *NSDI*, pages 141–153, 2004.

[20] Aashish Sharma, Zbigniew Kalbarczyk, James Barlow, and Ravishankar K. Iyer. Analysis of security data from a large computing organization. In *DSN*, 2011.

[21] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of grey: On the effectiveness of reputation-based blacklists. In *MALWARE'08*, pages 57–64, Fairfax, Virginia, USA.

[22] A free lightweight network intrusion detection system for UNIX and Windows. http://www.snort.org.

[23] Ionut Trestian, Supranamaya Ranjan, Aleksandar Kuzmanovi, and Antonio Nucci. Unconstrained endpoint profiling (googling the internet). In *ACM SIGCOMM'08*, NY, USA.

[24] Ting-Fang Yen and Michael K. Reiter. Traffic aggregation for malware detection. In *Proceedings of the 5th international conference on*, DIMVA, Berlin, Heidelberg, 2008.

[25] Yuanyuan Zeng, Xin Hu, and K.G. Shin. Detection of botnets using combined host- and network-level information. In *DSN'10*, pages 291 –300.