Raftopoulos and Dimitropoulos *EURASIP Journal on Information Security* 2013, **2013**:7
http://jis.eurasipjournals.com/content/2013/1/7

EURASIP Journal on Information Security
a SpringerOpen Journal

RESEARCH Open Access

A quality metric for IDS signatures: in the wild the size matters

Elias Raftopoulos* and Xenofontas Dimitropoulos*

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 first conduct a complex experiment to expand our understanding of forensics analysis processes. During a period of 4 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 publicly available a list of 165 good Snort signatures, i.e., signatures that were effective in identifying validated malware without producing false positives. In addition, we analyze the characteristics of good signatures and identify strong correlations between different signature features and their effectiveness, i.e., the number of validated incidents in which a good signature is identified. Based on our experiment, we finally introduce an IDS signature quality metric that can be exploited by security specialists to evaluate the available rulesets, prioritize the generated alerts, and facilitate the forensics analysis processes. We apply our metric to characterize the most popular Snort rulesets. Our analysis of signatures is useful not only for configuring Snort but also for establishing best practices and for teaching how to write new IDS signatures.

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 4 weeks, we manually investigate, in coordination with the IT department of our university, 200 security incidents about compromised hosts detected by an IDS alert correlator. The security investigation combines 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

*Correspondence: rilias@tik.ee.ethz.ch; fontas@tik.ee.ethz.ch
Communication Systems Group, Computer Engineering and Networks Laboratory, ETH Zurich, Rämistrasse 101, Zurich 8092, Switzerland

incidents. In addition, we report which combinations of sources helped diagnose different types of malware. Third, we make available a list of 165 Snort signatures that were effective in detecting validated malware without producing false positives. We analyze the differences between good and regular Snort signatures and observe that good signatures tend to use offsets, regular expressions, fixed packet sizes, and specific destination ports much more often than regular signatures. This shows that unlike common best practices for keeping IDS signatures simple, complexity in the case of signature writing is a virtue. In addition, we find that different signature features exhibit strong correlations with the effectiveness of a good signature, i.e., the number of validated incidents it detected. Based on this, we introduce a novel signature quality metric that can be used by security specialists to evaluate the available rulesets, prioritize the generated alerts, and facilitate the forensics analysis processes. Finally, we apply our metric to the most popular signature rulesets, i.e., to the *Vulnerability Research Team*, the *Emerging Threats* [1], the *Bleeding Edge* [2], and the *SRI Bothunter* [3] rulesets, and highlight their differences.

The remaining part of our paper is structured as follows. In the next section, we describe the data we used. Then, in Section 'Evidence correlation studies' 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 'Complementary utility and ranking of security sources' and 'What does a good IDS signature look like?', respectively. Finally, in Sections 'Related work' and 'Conclusions' we outline the related work and conclude our paper.

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 that users can install on their devices and on the services they can use. Our monitoring period lasted for approximately 4 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 and 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 [4] and OpenVas [5] 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 [6].

IDS alerts

The IDS deployed in our infrastructure is Snort [7], which is commonly considered the open source IDS solution with the largest number of registered users and the most active community [8,9]. 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 [1]. 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 [10] 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. In Table 1 we illustrate the IDS alert features extracted for a host that has been infected with the Torpig trojan. Note that IP addresses have been anonymized.

These features are presented to the analyst along with the 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 4 weeks, the majority

Table 1 Example features extracted from Snort alerts for an investigated internal host

Feature	Value
Contacted remote hosts	{a.b.c.d , k.l.m.n}
Target remote services	{80}
Target local services	{80,135,443}
Severe alert count	271
Infection duration (hours)	23
Severe alerts triggered (IDs)	{2801953, 2012642, 2912939, 240001}

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 4 weeks of our experiment, the correlator distilled 200 aggregate events from 37 million raw alerts.

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 Domain Name System (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 [4] and OpenVas [5], in order to build a comprehensive profile of the vulnerability status of a host.

Blacklists

In order to examine if a host within our network frequently initiates connections to known malicious domains, we use a set of independent blacklists. We leverage five public blacklists [11-15], 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. In Table 2 we show the different blacklist providers we used and the available labels per source.

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.

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*, and *torrent*. A similar approach has also been used in [16]. 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. This process is illustrated in Figure 1.

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.

Table 2 Public blacklist providers and blacklist data labels

	Ads	Attack	Bot	Chat	Drugs	Generic	Malware	Porn	Rbn	Religion	Spam
Apews											✓
Dshield		✓				✓					
Emerging Threats		✓	✓			✓			✓		
Shadowserver			✓								
UrlBlacklist	✓			✓	✓		✓	✓		✓	

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 [10]. 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 4 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

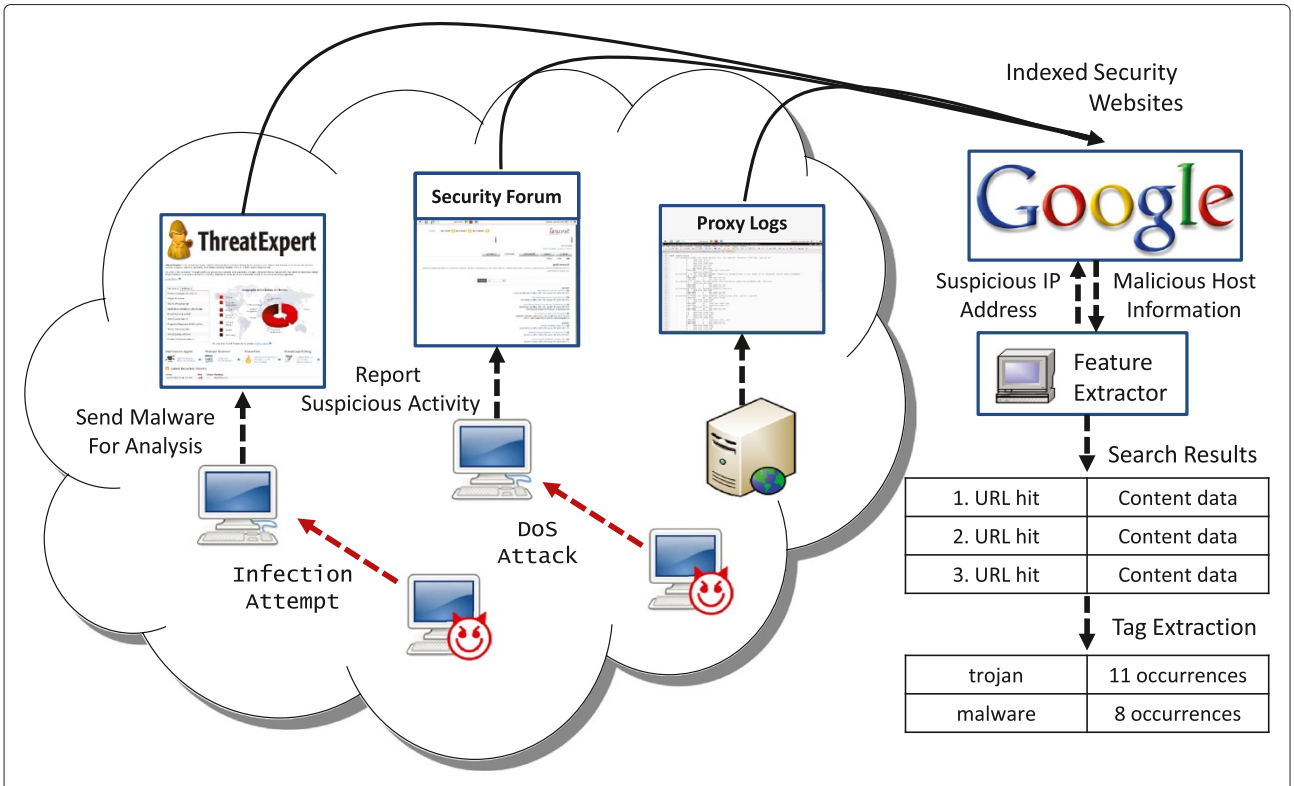

Figure 1 Extracting features from Google.

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*.

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 3 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 3, we identify the combination of sources that is useful for identifying the corresponding type of malware. In 41.5% of the cases, two sources, and in 14% of the cases, at least three 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

Table 3 Prevalence of different malware types and variants in the 200 investigated incidents

Malware type (number of incidents)	Variant (number of 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)	✓	✓		✓

The last four columns mark the data sources that provided useful evidence for diagnosis.

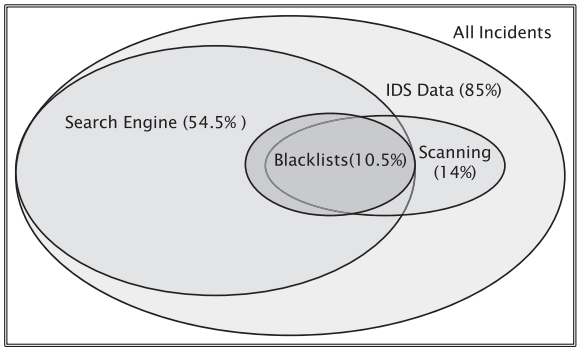

Figure 2 Complementary utility of security data sources for the diagnosis of 200 incidents.

hand, a single source was useful to identify AskSearch and MySearch.

In our previous short paper [17] 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.

In Figure 2 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.
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 [18].

Below we summarized the *main lessons* we learned 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.*

What does a good IDS signature look like?

Next, we provide a list of Snort signatures that were found by our alert correlator that are 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 3.

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 165 Snort signature IDs (SID) that can be summarized into 49 aggregate signatures as several affiliate SIDs detect small variations of the same pattern. In Table 4 we provide the 165 SIDs and a short description of the 49 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 [10]. 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 4): C&C Communication, reporting, egg download, redirection, and propagation.

In Figure 3 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 165 good signatures and its complement group of 1,371 regular signatures that are triggered during our experiment. In Table 5 we compare the 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 23.2 bytes in 2.4 different fields, while a regular signature checks only 11.2 bytes in 1.2 fields. In addition, 28% of the good signatures set the offset of a sought byte sequence; 51% provide a regular expression to further specify a search pattern; 22% fix the destination port(s); and 14.5% give a specific packet size. In sharp contrast, the corresponding numbers for regular signatures are only 8%, 15%, 17%, and 7%, respectively.

The regular expression size in good signatures is 11.3, compared to 4.2 in the case of regular signatures. To compute the regular expression size, we use the alphabetic width which is, to the best of our knowledge, the most commonly used metric [19]. The alphabetic width counts the total number of alphabetic symbols in the regular expression, taking into account multiplicity. However, we have adapted this metric to reduce the effect of the union operator, which attempts to match any of a list of input variables to a target string, by considering only the alphabetically widest variable of the union. For example, we iteratively reduce the regular expression $R(V_1|V_2)$ to $R(V_1)$ if $\text{Alphabetic}_{\text{width}}(V_1) \geq \text{Alphabetic}_{\text{width}}(V_2)$.

Flow options are used to provide flow level control. For example, signatures can state whether the processed packet belongs to an *established* connection and further explicitly define the direction (e.g., *to_server*, *to_client*). IP options are typically used to detect IP-level attacks, reconnaissance and network mapping attempts, and protocol based DoS attacks. For example, IP options check for the usage of the *Fragment* bit, the *Type of Service* option, and the *Time to Live* value.

Table 4 Effective Snort signatures in 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,	ET DROP Known Bot C&C Server Traffic TCP/UDP
2404242:2404247,2404335:240434	
16693	SPYWARE-PUT Torpig bot sinkhole server DNS lookup attempt
2011857	ET TROJAN SpyEye C&C Check-in URI
2013076	ET TROJAN Zeus Bot GET to Google checking Internet connectivity
2013348	ET TROJAN Zeus Bot Request to CnC 2
2013911	ET TROJAN P2P Zeus or ZeroAccess Request To CnC
2000348	ET ATTACK_RESPONSE IRC - Channel JOIN on non-std port
2014107	ET TROJAN Zeus POST Request to CnC - cookie variation
2015813	ET CURRENT_EVENTS DNS Query Torpig Sinkhole Domain
16140	BACKDOOR torpig-mebroot command and control 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
2802912	ETPRO TROJAN Backdoor.Nervos.A Checkin to Server
2002728	ET TROJAN Ransky or variant backdoor communication ping
2010150	ET TROJAN Koobface HTTP Request
2010885	ET TROJAN BlackEnergy v2.x HTTP Request with Encrypted Variable
2012279	ET CURRENT_EVENTS SpyEye HTTP Library Checkin
2002762	ET TROJAN Torpig Reporting User Activity
2008660	ET TROJAN Torpig Infection Reporting
2000347	ET ATTACK_RESPONSE IRC - Private message on non-std port
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
2011388	ET TROJAN Bredolab/Hiloti/ Mufanom Downloader Checkin 2
2014435	ET TROJAN Infostealer.Banprox Proxy.pac Download
2007577	ET TROJAN General Downloader Checkin URL
2016347	ET CURRENT_EVENTS Styx Exploit Kit Secondary Landing
2011365, 2010267	ET TROJAN Sinowal/sinonet/ mebroot/Torpig infected host 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,	ET RBN Known Russian Business Network IP TCP/UDP
2406361:2406383,2406635:2406649	
2016583	ET CURRENT_EVENTS SUSPICIOUS Java Request to DNSDynamic DNS

Table 4 Effective Snort signatures in the 200 investigated incidents *(Continued)*

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
2003292	ET WORM Allaple ICMP Sweep Ping Outbound
2011104	ET TROJAN Exploit kit attack activity likely hostile
2010087	ET SCAN Suspicious User-Agent Containing SQL Inject/ion, SQL Scanner
2006546	ET SCAN LibSSH Based Frequent SSH Connections BruteForce Attack!
2001219	ET SCAN Potential SSH Scan
2003	SQL Worm propagation attempt
3817	TFTP GET transfer mode overflow attempt
12798:12802	SHELLCODE base64 x86 NOOP

In the case of *HTTP*, we observe that good signatures tend to define the section of an HTTP packet, e.g., with the Snort keywords *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 1.1 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.*

Building a signature effectiveness metric

In Section 'What does a good IDS signature look like?' we identified a number of complexity features for the triggered signatures and highlighted that *good* signatures will tend to exhibit much higher values compared to the *regular* ones. We next ask the following question: is it possible to build a metric that would predict the effectiveness of a new signature based on its structural characteristics? Such a metric would be extremely useful to authors of new signatures, allowing them to evaluate the quality of the tailored rules before release, and also to security practitioners that wish to rank and prioritize signatures in the deployed IDS rulesets. To answer this question, we first evaluate the correlation between the collected complexity features and the effectiveness of a signature. We explore whether a single complexity metric is sufficient or whether a group of complexity features are needed to capture the signature effectiveness. Then, we tailor a metric based on the most prominent set of complexity features and evaluate its performance. Finally, we apply this metric

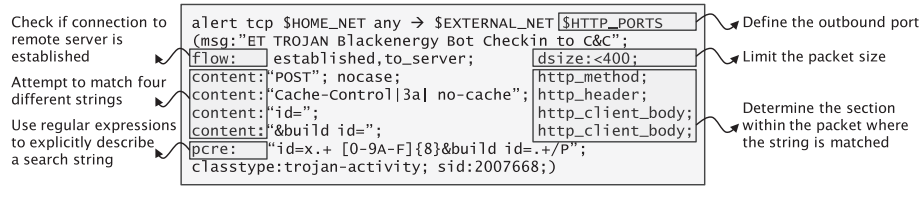

Figure 3 Example of good Snort signature used to detect a beacon frame sent by a Blackenergy bot to its controller.

Table 5 Comparison of good and regular Snort signatures

	Bytes checked	Fields checked	Byte offset is set	Regular exp. is set	Regular exp. size	Destination port is set	Packet size is set	Flow options are set	IP options are set
Regular signatures	11.2	1.2	8%	15%	4.2	17%	6.9%	30.2%	62.1%
Good signatures	23.2	2.4	28%	51%	11.3	22%	14.5%	38%	68.4%
Increase	2.1 ×	2 ×	3.5 ×	3.4 ×	2.7 ×	1.3 ×	2.1 ×	1.8 ×	1.1 ×

Statistics are computed over 165 good and 1,371 regular signatures.

to evaluate the signatures contained in popular publicly available rulesets.

Evaluating complexity features

To investigate the predictive strength of the collected complexity features, we first evaluate whether they are correlated to the signature effectiveness. Signature effectiveness for good signatures, as defined in Section 'What does a good IDS signature look like?', is equal to the number of validated infection incidents involving the signature, whereas it is equal to zero for regular signatures. Intuitively, an effective signature helps to validate many detected infections.

We use the Spearman correlation coefficient that measures the monotonic relationship between two input random variables. It is considered as a robust correlation technique [20], since it does not consider a linear relationship between the examined variables, such as in the case of Pearson correlation, and it is not sensitive to outliers. In Table 6 we present the correlation results for different signature classes. We highlight the significant correlation values, at the 0.05 significance level, in bold.

First, we see that there are significant correlations among all signature classes for several complexity features. These correlations exceed in some cases 0.90 at the 0.05 significance level, indicating that there is a very strong association between the respective complexity features and the measured signature effectiveness.

Second, we observe that there is no single set of features that correlates with signature effectiveness for all signature classes. *C&C Communication* signatures detect the communication attempts between a victim host and its controller. Typically, the victim machine will include in the sent packet several pieces of information regarding its state and ID, the version of the rootkit used, and the type of request. This information is dispersed within the different sections of the payload. Therefore, we see increased correlation values 0.94 and 0.95, for the *Bytes Checked* and *Fields Checked* features.

Reporting signatures are triggered whenever a victim host is attempting to leak sensitive data it has harvested from the victim host. Typically, this information is split into sections, which are preceded by a short string indicating the beginning of the section. Therefore, we see significant correlation for the Bytes Checked and *Fields Checked* features. Moreover, a communication flow needs to be established before the reporting takes place in most of the cases, which explains the 0.78 coefficient for the *Flow Options are Set* feature.

Egg Download signatures detect attempts made by the malware to update itself or to download additional badware. Typically, they try to match the string following the *GET* command of an HTTP request, corresponding to the target malicious binary. This request can slightly vary since the sender might include ID or timestamp information, and the requested binary might come in different versions with slightly different filenames. Therefore,

Table 6 Complexity features and their correlations with signature effectiveness for 165 good signatures

	Signature class					
Complexity feature	C&C Communication	Reporting	Egg Download	Redirection	Propagation	All signatures
Bytes checked	**0.95**	**0.52**	0.30	**0.88**	**0.65**	**0.84**
Fields checked	**0.94**	**0.60**	**0.55**	**0.87**	**0.61**	**0.83**
Byte offset is set	**0.48**	0.28	**0.54**	**0.62**	0.34	**0.51**
Regular expression is set	**0.48**	0.27	**0.59**	**0.73**	0.35	**0.52**
Regular expression size	**0.73**	0.24	**0.51**	**0.63**	0.30	0.37
Destination port is set	**0.49**	0.19	0.13	0.35	0.29	0.24
Packet size is set	0.19	0.12	0.19	0.16	**0.40**	0.12
Flow options are set	**0.91**	**0.78**	**0.60**	**0.41**	**0.48**	**0.68**
IP options are set	**0.53**	**0.42**	**0.48**	**0.53**	**0.73**	**0.56**

We highlight in bold correlation values higher than 0.4.

regular expressions are used to cope with this issue, explaining the increased correlation coefficient of 0.59 we observe for the *Regular Expression is Set* feature.

Redirection signatures detect requests towards potentially malicious domains. These signatures describe the target domain using either a string descriptor or a more flexible regular expression field. As a result, Bytes Checked-, *Fields Checked-*, and regular expression-related features exhibit an increased correlation value.

Finally, *Propagation* signatures involve scanning and network enumeration detection, which often uses IP specific flags, such as the fragmentation bit, therefore, we see that the correlation value for the *IP Flags are Set* feature is high and equal to 0.73.

In the last column of Table 6, we see the respective correlation values between the complexity features and the signature effectiveness, considering all good signatures. We see that Bytes Checked and Fields Checked are the most prominent features, with correlation values of 0.84 and 0.83, followed by the *Flow/IP options are Set* and the *Regular Expression is Set* feature, with correlation values of 0.68, 0.56, and 0.52, respectively. However, these aggregate correlation results exhibit significant difference from the per-class correlation values, indicating that there is no single feature set that captures the signature effectiveness across all classes.

Insight 5. For most classes of signatures, there exists a set of complexity features that strongly correlates with signature effectiveness. However, there is no single set of features that fits all classes.

Building a metric that captures signature effectiveness

Our goal is to exploit complexity features that correlate with signature effectiveness to build predictor variables. This can be done by using a regression model that takes as input the complexity features as the explanatory variables and the signature effectiveness as the response variable. However, an inherent problem that we need to deal with is the fact that the feature variables might exhibit inter-correlations. This is the known problem of feature multi-collinearity, which means that one feature can be predicted from other input features with a non-trivial degree of accuracy [21]. Such inter-correlations among predictor variables can lead to unreliable and unstable estimates of regression coefficients, and therefore an inflated variance of the response variable, which in our model is the signature effectiveness.

To overcome this problem of inter-correlation among input features, we employ *Principal Component Analysis* (PCA). PCA takes as input the original features and generates a new set of variables, which are called *principal components* (PCs). PCs are linear combinations of the original features and form an orthogonal basis, meaning that they are linearly uncorrelated. Moreover, PCs are ordered so that the first few components retain most of the variation present in the original features. Therefore, we can reduce the dimensions of the feature space by only retaining the first few PCs. Fewer parameters in the regression modeling translate in lower variance of the regression coefficient estimator or, in other words, an 'easier' and more effective regression.

In Figure 4 we illustrate the extracted PCs for the feature matrix consisting of all good signatures. We see that six PCs account for approximately 98% of the total variance of the dataset. This means that we can safely drop the remaining three PCs and, thus, reduce the feature space to six dimensions. Additionally, we perform the same transformation for five feature matrices derived from the remaining signature classes and one that corresponds to

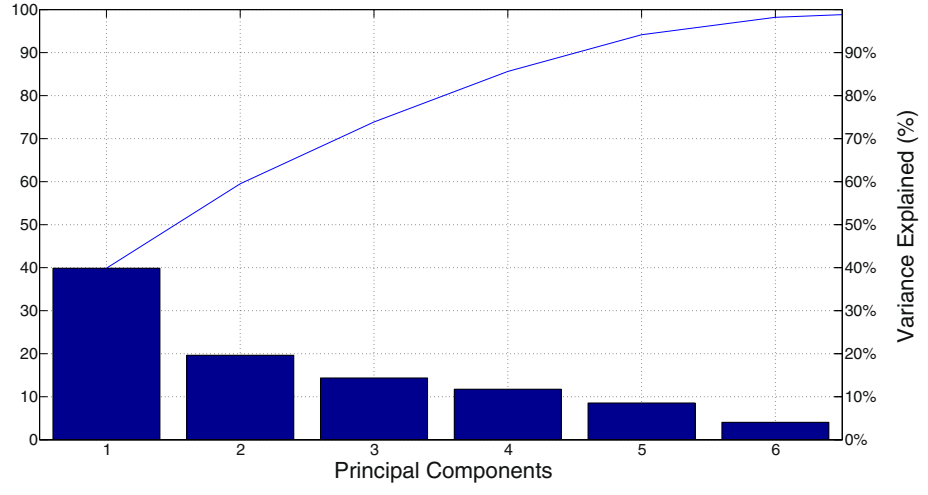

Figure 4 Variance explained by different PCs for all good signatures.

all good signatures. In all cases, six PCs account for at least 95% of the total variance and, therefore, suffice.

We then apply multiple linear regression using the PCs derived in the previous step as exploratory variables and the signature effectiveness as response variable. The goal of this process is to build a model that best describes the relationship between the exploratory and the response variables and can be used to perform prediction. In Table 7 we show the regression results for the derived models.

For each signature class, we list the R^2 value which is a statistic that measures the goodness of fit. It shows how closely the values generated by the model match the response variable. A value of 0.8 would mean that 80% of the variability in the original data is explained by the model, whereas an R^2 value of 1.0 suggests that this is a perfect fit. However, R^2 improves as exploratory variables are being added to the model, which might lead to artificially inflated R^2 values. Adjusted R^2 fixes this problem by penalizing the number of terms in the model and therefore is more appropriate when comparing how different models fit the original data. The F-statistic tests the hypothesis that the coefficients that associate the exploratory variables to the response variable are zero.

In essence, it tests whether the selected exploratory variables are useful in predicting the response variable. Values close to 1 suggest that there is no association, whereas values greater than 5 at statistically significant levels signify that the selected exploratory variables are good predictors of the response variable.

In Table 7 the R^2 value indicates that the PCs extracted from the complexity features explain in all cases between 73% and 96% of the original variance, meaning that the derived regression model fits well the data. The adjusted R^2 ranging from 0.6 to 0.9 denotes that these results are robust and are not subject to bias introduced by the number of predictors used in the model.

To assess the predictive strength of our models, we perform a stratified threefold cross validation. In this process the complete feature matrix, consisting of all good signatures, is split into three folds, two of which are used for training and one for testing. The folds are stratified so that each fold has on average the same mix of different signature classes. Based on the training data, we build the predictors described in Table 7. We then compute the Spearman correlation between the predicted values and the signature effectiveness. A high correlation indicates that the predictor sensitivity is high, signifying that it can be used to estimate the signature effectiveness given the respective features.

In Table 8 we show the results of the validation. In total we evaluate 18 models derived from the six listed classes and the three dataset splits introduced by the stratified validation. We see that we get high correlations on average, above 0.41 in all cases, at the 0.05 significance level. If we exclude the *Redirection* signature class, which appears to produce the worst prediction results, the remaining classes exhibit correlations ranging from 0.68 to 0.87, suggesting that the generated models have a very high prediction accuracy. Moreover, we see that the generic model that uses all good signature complexity features, without taking into account class information, performs reasonably well with an average correlation of 0.6893.

However, how can one use these models in a realistic setting? Assuming that we have a new set of developed signatures, we have extracted the respective complexity features. If we apply our regression model on this data, we will get a list of response values corresponding to the estimated signature effectiveness. These values should be interpreted in a comparative fashion. Our metric is essentially a method to rank the evaluated signatures and assign a score of effectiveness that only makes sense when compared to the same score computed for other signatures.

To illustrate this process we perform the following experiment. We use the predictors generated based on the training folds of the stratified validation illustrated in Table 8. We apply these predictors to the respective testing fold and additionally apply the generic *All Signatures* predictor to the PCs that we have extracted from the complexity features of regular signatures. The predictor is useful in practice if the good signatures of the testing fold rank above the regular signatures in terms of estimated effectiveness. We present the respective results in Table 9.

Table 7 Regression models goodness of fit

Signature class	R^2value	Adjusted R^2	F-test
C&C Communication	0.939	0.848	91.8, $p < 0.001$
Reporting	0.737	0.614	9.9, $p < 0.001$
Egg Download	0.957	0.864	12.9, $p < 0.0005$
Redirection	0.961	0.910	151.9, $p < 0.001$
Propagation	0.917	0.856	6.9, $p < 0.005$
All signatures	0.933	0.802	31.9, $p < 0.001$

Table 8 Regression model validation using threefold stratified validation

Signature class	Split A	Split B	Split C	Class average
C&C Communication	0.8628	0.8691	0.6652	0.7990
Reporting	0.6752	0.7269	0.9517	0.7846
Egg Download	0.9127	0.8282	0.8845	0.8751
Redirection	0.6940	0.1309	0.6713	0.4115
Propagation	0.8473	0.9443	0.5387	0.7768
Good signatures	0.7680	0.6211	0.6783	0.6893

Table 9 Predicted ranking of good signatures taking into account all 1,423 signatures

Top *N*%	Number of signatures in the Top N%	Percentage (and the absolute number in parenthesis) of the good signatures that are in the top N% of all signatures
N = *5*% (71)	71	47% (24)
N = *10*% (142)	142	79% (40)
N = *50*% (711)	711	94% (48)

Table 9 shows that the top 5% of the predicted most effective signatures included 24 (47%) of the good signatures in the dataset, whereas 94% of the good signatures are in the upper half of the ranking. This result practically tells us that if a security specialist would have a new ruleset to deploy consisting of these 1,423 signatures; then by using our predictor, he could derive an estimate about which are the most prominent and reliable ones that he should focus on. He could then use this information to prioritize alerts generated by the deployed IDS and facilitate the forensics analysis.

Insight 6. Complexity features can be used to build regression models that allow the estimation of signature effectiveness. These prediction models can constitute a signature quality metric that can be exploited by security specialists to evaluate the available rulesets, prioritize the generated alerts, and facilitate the forensics analysis processes.

Evaluating commonly used rulesets

Several publicly available IDS rulesets are used in the security community. However, partitioners have no tools available to evaluate the quality of these rulesets before deployment. Our effectiveness metric can be used in this context to provide an early indicator of the usefulness of a given signature set. In Table 10 we compare the four most popular rulesets, namely, the Vulnerability Research Team [22] ruleset which is the default ruleset bundled with SNORT, the *Emerging Threats* [1] ruleset, the *Bleeding Edge* [2] ruleset, and the *SRI Bothunter* [3] ruleset. Note that the latter ruleset also includes some selected Emerging Threats rules. We have excluded these overlapping signatures and only considered the SRI signatures.

We build a composite ruleset consisting of all available signatures. We then extract the complexity features for this ruleset and derive the respective PCs. We apply our generic predictor, i.e., the one produced considering all good signatures, on the PCs and produce the signature effectiveness vector. We sort the signature effectiveness values on descending order and evaluate how many signatures from different rulesets fall into different percentiles. For example, we see that the top 5%, which accounts for 988 signatures, contains 246 VRT, 703 ET, 13 BE, and 26 SRI signatures. These correspond to 4.4%, 6.2%, 1%, and 1.6% of the total VRT, ET, BE, and SRI signatures, respectively.

Table 10 suggests that the Emerging Threats ruleset is the one contributing the most effective signatures at all percentile levels, followed by the default *VRT* ruleset. The *ET* is indeed considered one of the most sensitive and responsive rulesets, since it is open-source and is fine-tuned by a high number of contributors. This also is in agreement with our own experience. As shown in Table 4 the signatures that we have listed as our good signatures originate in principle from the ET ruleset. The VRT is the most popular ruleset, since it comes bundled with the default SNORT installation. It is sensitive to a wide range of threats, accounting 5,559 signatures and is constantly fine-tuned by the Sourcefire team. Based on our ranking results, VRT is the second most fruitful source of signatures, having 246 and 290 signatures at the 5% and 10% levels, respectively.

The SRI ruleset is a collection of community rules and custom-tailored signatures. It is targeted to a narrow range of specific threats, including 1,606 signatures. Signature selection at SRI happens by deploying and evaluating rulesets on a custom honeynet. In our effectiveness ranking, SRI ruleset has only a minor impact, contributing 26 signatures at the 5% and 104 at the 10% level. These signature counts account for the 1.6% and 6.5% of the available SRI signatures. Finally, *Bleeding Edge* is a platform allowing security specialists to upload their own custom rules. The *Bleeding Edge* team organizes them into coherent rulesets and makes quality tweaks. Based on our ranking, the *Bleeding Edge* ruleset was the least effective, contributing only 13 and 98 signatures at the 5% and 10% levels, respectively.

Table 10 Predicted signature ranking for four popular SNORT rulesets

	Percentile level and in parenthesis absolute number of signatures		
Top *N*% (total number of signatures)	*N* = *5*% (988)	*N* = *10*% (1,976)	*N* = *50*% (9,881)
Ruleset (number of signatures)[a]			
Vulnerability Research Team (5,559)	4.4% (246)	5.2% (290)	47.6% (2,646)
Emerging Threats (11,344)	6.2% (703)	13.0% (1,484)	51.1% (5,797)
Bleeding Edge Team (1,254)	1.0% (13)	7.8% (98)	44.1% (554)
SRI Bothunter (1,606)	1.6% (26)	6.5% (104)	55.0% (884)

[a]Corresponding values in the second to the fourth columns are presented as percentage (number of ruleset signatures).

Related work

Several studies have detected security incidents in traffic traces from production networks, e.g., [23-25], without providing though a systematic validation of detected incidents. Closer to our work, Sharma et al. [26] 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 [17], we built a decision support tool that correlated evidence from different security sensors to expedite manual forensics analysis of compromised systems. Authors in [17] 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 has focused on discovering and correlating evidence primarily from end hosts [27-29]. 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 [30,31]. 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.

Prediction methods have been the subject of extensive study in the field of vulnerability and software defect prediction. The goal in this line of work is to detect vulnerabilities and defective software modules at design time, based on the structural or behavioral characteristics of the respective code. Ohlsson et al. used graph based complexity metrics to detect software prone components in telecom systems [32]. Gyimothy et al. [33] applied logical and linear regression methods to assess the applicability of object-oriented metrics to estimate the fault-proneness of Mozilla. Neuhaus et al. used diverse code features, including imports, function call structure, and bug history, to predict the vulnerabilities of future Mozilla components [34]. Nagappan et al. used a long list of module, function, and class structural features to build a predictor of five Microsoft systems, and found that predictors built for individual projects only work well for similar projects [35]. Zimmermann et al. exploited historical bug data and code complexity features to predict vulnerable components [36]. Our work uses a similar modeling approach to solve a different problem. Instead of predicting software vulnerabilities, we focus on building a metric that captures the effectiveness of IDS signatures.

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 3, 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 165 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 addition, we introduce a novel signature quality metric that can be exploited by security specialists to evaluate the available rulesets, prioritize the generated alerts, and facilitate the forensics analysis processes. Based on our metric, we compare the most popular signature rulesets and highlight differences.

Competing interests

The authors declare that they have no competing interests.

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.

Received: 2 August 2013 Accepted: 12 November 2013
Published: 20 December 2013

References

1. Emerging threats. http://www.emergingthreats.net, Accessed 17 December 2013
2. Bleeding edge snort rules. http://www.bleedingsnort.com. Accessed 17 December 2013
3. SRI Malware threat center. http://www.snort.org/vrt. Accessed 17 December 2013
4. The Nessus vulnerability scanner. http://www.tenable.com/products/nessus. Accessed 17 December 2013
5. The open vulnerability assessment system. http://www.openvas.org. Accessed 17 December 2013
6. E Raftopoulos, X Dimitropoulos, Technical report: shedding light on data correlation during network forensics analysis. Tech. Rep. 346 (2012)
7. A free lightweight network intrusion detection system for UNIX and Windows. http://www.snort.org. Accessed 17 December 2013
8. The case for open source IDS. http://www.itsecurity.com/features/the-case-for-open-source-ids-022607/. Accessed 17 December 2013
9. Best IDS/IPS solution. http://www.scmagazine.com/best-idsips-solution/article/130871/. Accessed 17 December 2013
10. E Raftopoulos, X Dimitropoulos, Detecting, validating and characterizing computer infections in the wild, in *ACM SIGCOMM IMC* (Berlin, 2011)
11. Anonymous postmasters early warning system. http://www.apews.org. Accessed 17 December 2013
12. The Urlblacklist web page. http://www.urlblacklist.com. Accessed 17 December 2013
13. Shadowserver Foundation. http://www.shadowserver.org. Accessed 17 December 2013
14. Cooperative Network Security Community - Internet Security. http://www.dshield.org. Accessed 17 December 2013
15. Advanced automated threat analysis system. http://www.threatexpert.com. Accessed 17 December 2013
16. I Trestian, S Ranjan, A Kuzmanovi, A Nucci, Unconstrained endpoint profiling (googling the internet), in *ACM SIGCOMM'08,* NY, USA. http://doi.acm.org/10.1145/1402958.1402991
17. E Raftopoulos, X Dimitropoulos, Shedding light on log correlation in network forensics analysis, in *DIMVA'12*
18. S Sinha, M Bailey, F Jahanian, Shades of grey: on the effectiveness of reputation-based blacklists, in *MALWARE'08* (Fairfax), pp. 57–64
19. K Ellul, B Krawetz, J Shallit, Mw Wang, Regular expressions: new results and open problems. J. Autom. Lang. Comb. **9**(2-3), 233–256 (2004). http://dl.acm.org/citation.cfm?id=1103362.1103368
20. A Lehman, N O'Rourke, L Hatcher, *Jmp for Basic Univariate and Multivariate Statistics: Methods for Researchers and Social Scientists, Second Edition.* (SAS Institute, Cary, 2013). http://books.google.ch/books?id=Zh4iD8V2sTsC
21. S Chatterjee, B Price, *Regression Analysis by Example*, 2nd edn. (Wiley) http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470905840.html
22. Sourcefire Vulnerability Research Team. http://www.snort.org/vrt. Accessed 17 December 2013
23. S Saroiu, SD Gribble, HM Levy, Measurement and analysis of Spyware in a university environment, in *NSDI*, (2004), pp. 141–153
24. TF Yen, MK Reiter, Traffic aggregation for Malware detection, in *Proceedings of the 5th International Conference on DIMVA* (Berlin, Heidelberg, 10–11 July 2008). http://dx.doi.org/10.1007/978-3-540-70542-0_11
25. G Maier, A Feldmann, V Paxson, R Sommer, M Vallentin, An assessment of overt malicious activity manifest in residential networks, in *DIMVA*, (2011)
26. A Sharma, Z Kalbarczyk, J Barlow, RK Iyer, Analysis of security data from a large computing organization, in *IEEE/IFIP 41st International Conference on DSN,* 27-30 June 2011
27. A Case, A Cristina, L Marziale, GG Richard, V Roussev, FACE: automated digital evidence discovery and correlation. Digit. Investig. **5**, S65–S75 (2008). http://dx.doi.org/10.1016/j.diin.2008.05.008
28. SL Garfinkel, Forensic feature extraction and cross-drive analysis. Digit. Investig. **3**, 71–81 (2006). http://dx.doi.org/10.1016/j.diin.2006.06.007
29. Y Zeng, X Hu, K Shin, Detection of botnets using combined host- and network-level information, in *IEEE/IFIP International Conference on DSN'10, Chicago, 28 June to 1 July* 2010, pp. 291–300
30. S Garfinkel, Automating disk forensic processing with SleuthKit, XML and Python, in *Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering 2009* (Berkeley, 21–21 May 2009)
31. J Mena J, *Investigative Data Mining for Security and Criminal Detection* (Butterworth-Heinemann Limited, 2003). http://books.google.ch/books?id=WbSNpYHoNWMC
32. N Ohlsson, H Alberg, Predicting fault-prone software modules in telephone switches. IEEE Trans. Softw. Eng. **22**(12), 886–894 (1996). http://dx.doi.org/10.1109/32.553637
33. T Gyimóthy, R Ferenc, I Siket, Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Trans. Softw. Eng. **31**, 897–910 (2005)
34. S Neuhaus, T Zimmermann, C Holler, A Zeller, Predicting vulnerable software components, in *Proceedings of the 14th ACM Conference on Computer and Communications Security* (ACM New York, 2007), pp. 529–540
35. N Nagappan, T Ball, A Zeller, Mining metrics to predict component failures, in *Proceedings of the 28th International Conference on Software Engineering,* ICSE '06 (ACM New York, 2006), pp. 452–461. http://doi.acm.org/10.1145/1134285.1134349
36. T Zimmermann, R Premraj, A Zeller, Predicting defects for eclipse, in *Proceedings of the Third International Workshop on Predictor Models in Software Engineering* (Washington, 20–26 May 2007)

doi:10.1186/1687-417X-2013-7
Cite this article as: Raftopoulos and Dimitropoulos: **A quality metric for IDS signatures: in the wild the size matters.** *EURASIP Journal on Information Security* 2013 **2013**:7.