Ruling the Rules: Quantifying the Evolution of Rulesets, Alerts and Incidents in Network Intrusion Detection

Mathew Vermeer
m.vermeer@tudelft.nl
Delft University of Technology

Michel van Eeten
m.j.g.vaneeten@tudelft.nl
Delft University of Technology

Carlos Gañán
c.hernandezganan@tudelft.nl
Delft University of Technology

ABSTRACT

Notwithstanding the predicted demise of signature-based network monitoring, it is still part of the bedrock of security operations. Rulesets are fundamental to the efficacy of Network Intrusion Detection Systems (NIDS). Yet, they have rarely been studied in production environments. We partner with a Managed Security Service Provider (MSSP) to gain more insight into the evolution of rulesets, the alerts that they trigger and the incidents that get investigated. We analyze a combined ruleset –including both commercial and proprietary rules– that consists of 130 thousand rules and was used to monitor hundreds of networks. We find that these rulesets keep growing over time but there is almost no overlap among them in terms of detection options or what indicators of compromise they contain. The combined ruleset triggered more than 62 million alerts and led to 150 thousand incident investigations by SOC analysts, though the vast majority of rules never triggered a single alert. We find that just 0.5% of all rules are responsible for more than 80% of the alerts and incidents and only 1.2% of all alerts were deemed to merit closer investigation. Of all incidents, 16% were labeled as false positives and 9% carried significant risk to the client organization. Independently of the type of rule, updating rules is a minor activity. Most rules are never modified and only a fraction is deleted, except for periodic purges in some sets. Seven in-depth interviews with rule developers corroborate the patterns we found in our analysis. Finally, we identify several rule management practices that influence rule and ruleset efficacy, such as supplementing commercial rules with your own and making rules as specific as possible.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; *Network security*.

KEYWORDS

NIDS, intrusion detection, rules, network security, SOC, alerts

ACM Reference Format:
Mathew Vermeer, Michel van Eeten, and Carlos Gañán. 2022. Ruling the Rules: Quantifying the Evolution of Rulesets, Alerts and Incidents in Network Intrusion Detection. In *Proceedings of the 2022 ACM Asia Conference on Computer and Communications Security (ASIA CCS '22), May 30–June 3, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3488932.3517412

ASIA CCS '22, May 30–June 3, 2022, Nagasaki, Japan

ACM ISBN 978-1-4503-9140-5/22/05.
https://doi.org/10.1145/3488932.3517412

1 INTRODUCTION

Over the last decade, the number of network attacks has steadily increased, putting even more emphasis on the efficacy of network monitoring [8]. Recent research in network intrusion detection has focused on statistical and machine-learning methods—e.g., [12, 23, 25]. The underlying motivation is that conventional rule- and signature-based methods are deemed unable to keep up with the fast-evolving threats and therefore they will become increasingly obsolete. As early as the turn of the century, industry reports were predicting the demise of signature-based network monitoring [3, 29]. Yet, two decades later, signature-based monitoring is still part of the bedrock of organizational security. Walk into any security operations center (SOC) and what you will see is analysts triaging alerts generated by network intrusion detection systems (NIDS) that still rely heavily on rulesets. Most, if not all, Managed Security Service Providers (MSSP) purchase and develop rulesets in order to detect relevant events on their client networks. Despite important advances in host-based detection and anomaly detection, there are few signs that this is changing any time soon.

To put it a bit facetiously, rules work in practice, but not in theory. Rulesets are incredibly important for protecting organizations everywhere and yet they are barely researched in real-world production settings. Prior work has focused mostly on analyzing the impact of rulesets on the system performance of an NIDS in a test setting, measuring sensor-based metrics such as CPU load, memory usage and packet loss [1, 7, 22, 32]. These studies looked at the sensors, not at the rulesets themselves, nor how they are managed to optimize detection capabilities. Other work has focused on improving the signatures used in NIDS [9, 24, 30, 31, 33]. The closest work we could find was a recent study by Gashi and Asad [2], who compared four different open and commercial rulesets, their changes over a period of five months and the alerts they generated in a test on a part of a university network. They concluded that there was barely any overlap in alerts, suggesting that organizations should combine rulesets to optimize detection.

To the best of our knowledge, we present the first study that opens up the black box of NIDS rulesets and their management in a real-world production setting. We have partnered with an MSSP that monitors hundreds of client networks using various commercially-acquired rulesets as well as a ruleset it develops in-house. We present an analysis of four rulesets that collectively consist of 130 thousand rules, span 13 years (2008–2021), and functioned in a production environment to monitor hundreds of networks. The rulesets received more than one million modifications, triggered more than 62 million alerts and led to 150 thousand incident investigations by SOC analysts.

How do rulesets evolve over time? How fast do they get revised and updated? How many rules actually trigger an alert in practice?

How many of these alerts represent real threats? We are especially interested a critical underlying tradeoff: How are rulesets managed in order to maximize detection of intrusions (true positives) and minimize the number of alerts that need to be investigated (false positives)? You can only detect what you create a rule for and yet the larger the (combined) ruleset, the harder it will be to keep it accurate and up to date and the higher the likelihood of overwhelming the SOC analysts with irrelevant alerts.

We analyze the changes in the rulesets, the scale and type of modifications and the relation between the rules and the alerts and incidents that are triggered by them. We complement this analysis by a small, but in-depth, user study of seven rule developers within the MSSP. We conducted semi-structured interviews to understand the process of rule writing and management.

In sum, we make the following contributions:

- We present the first longitudinal analysis of NIDS rulesets spanning up to 13 years (2008–2021), the alerts they generated and incidents that they helped detect.
- We created a tool to quantitatively analyze NIDS rule changes. This tool allows tracking the evolution of a rule over its lifespan, extract metadata from the rule's syntax and classify the different types of modifications. We open source this tool to the community.
- While in use, around 23% of all rules are updated in terms of detection capability. We quantify the factors that trigger changes within rulesets and find two major causes: (i) changes in the threat landscape; and (ii) reducing the number of alerts by making rules more specific. These factors are then cross-validated through semi-structured interviews with seven experts in the area of rule developing and security incident response.
- We find that 80% of all alerts were triggered by a mere 0.5% of all rules. The overwhelming majority of rules never trigger a single alert. This helps explain why rulesets can grow into the tens of thousands over time, without overwhelming SOC analysts. Still, only 1.2% of all alerts were deemed to require closer investigation, and only 0.3% of all alerts carried significant risk to the organization.
- We identify a number of rule management practices employed by rule developers and their influence on the efficacy of the rules. Specifically: (i) multiple rulesets are used simultaneously due to the lack of exhaustive coverage of the threat landscape by any single ruleset; (ii) proprietary rules are updated more often than commercial rules leading to a higher incident detection rate; and (iii) false positive incidents have a noticeable impact on rule updates.

2 RELATED WORK

Research on 'traditional' signature-based NIDSs and NIDS rulesets is relatively rare. Modern research focuses primarily on creating statistical or machine learning-based NIDSs in lieu of studying or improving upon signature-based approaches, which remain an industry standard to this day. Examples include Srivastav and Challa[25], Shone et al. [23], and Mirsky et al. [12].

In the rare instances where research has indeed looked at signature-based NIDSs, the effort has gone into studying the performance of the system itself and what the effect is of ruleset volume changes, instead of the processes driving rule changes. Soumya Sen [21] examines the Snort IDS and its performance when varying bandwidth and, perhaps unsurprisingly, that as bandwidth increases, the size of the ruleset must decrease in order to keep Snort's error rate below a certain threshold. This effect is magnified for a given bandwidth when the size of IP payloads decreases, as Snort will have to process more packets during the same amount of time, resulting in more packet loss.

Thongkanchorn et al. [28] perform an analysis similar to Sen [21]. The authors take the Snort, Suricata, and Bro (now Zeek) NIDSs and analyze its performance when varying attack type, ruleset sizes, and network traffic rates. For the experiment, they use the Emerging Threats (ET) Open ruleset for all NIDSs. They select a set of different types of attacks and generated the corresponding packets to test the different NIDSs. Results from the analysis include low packet loss and low CPU usage for TCP traffic, and that a higher traffic rate leads to higher CPU usage, higher packet loss, and a higher number of generated alerts. Finally, the authors also find that using more rulesets also causes a higher number of generated alerts. However, the study does not examine the underlying causes for the performance detriments, such as potential inefficiencies in the rules.

Gashi and Asad [2] perform a study more closely related to our work, as they analyze the evolution of and similarities between four different rulesets: Snort's official Community, Registered, and Subscribed rulesets, as well as the ET Open Suricata ruleset [20]. The analysis was performed on five months of rule updates. They examine the diversity and overlap between both blacklisted IPs and rule content options for the four different rulesets. The authors find that there is minimal overlap between ET's and Snort's IP blacklists and rule content options, with only 1% of rules containing matching content options. As a result, the alert triggering behavior of the rulesets from both sources also has minimal overlap, potentially indicating that both may be useful to provide more defense to a network.

While the work from Gashi and Asad [2] is somewhat related to the analysis we carry out in this work, there is only marginal overlap. To the best of our knowledge, there has been no research into the nature of rule changes and the rule development process as a whole. Neither has this rule data been combined with alert and incident data from production settings – in our case: an MSSP – to investigate the evolution of the rulesets in a changing threat landscape.

3 BACKGROUND

At the core of this work are two closely related topics, namely intrusion detection systems and the rulesets they employ to detect potentially malicious traffic. In this section we explain these concepts and place them within the context of an MSSP.

3.1 Intrusion detection systems

IDSs are classified by their placement and by the techniques that are used for detection itself. As for the placement of an IDS, there are network-based, host-based, and application-based intrusion detection systems. Furthermore, depending on its method of detection,

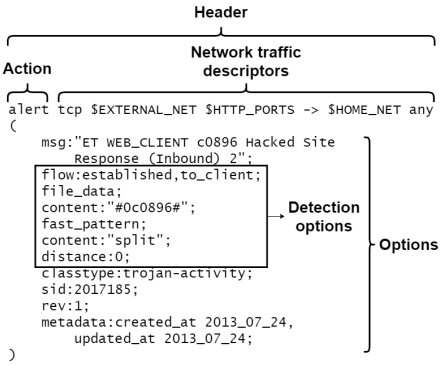


Figure 1: Rule syntax used by Snort and Suricata NIDSs.

an IDS can be signature-based or anomaly-based [11]. The type of IDS that is the focus of this work is the signature-based NIDS.

A network-based IDS is placed at a strategic point within a network and analyzes all network packets it receives to detect attacks. Signature-based IDSs find malicious activity by matching it with a predefined set of patterns or events that are characteristic of known attacks. Examples of such IDSs include Snort [4], Suricata [26], and Zeek (formerly Bro) [15].

3.2 NIDS rules and rulesets

Signature-based IDSs detect threats in a network with rules that inform the system what to look for in network traffic. Figure 1 illustrates the syntax of such a rule. An NIDS rule consists of two main parts: (1) the header, and (2) the options. The header can also be split up into two different parts: (1a) the action, and (1b) the network traffic descriptors. The action is the portion of the header that tells the NIDS what to do in case the rule is triggered (e.g., raise an alert, drop the packet, log the packet, among others). The second portion of the header that we identify is the network traffic descriptors, which specify the origin and destination IP addresses and ports, as well as the protocol of the packet that the rule looks for. The secondary section of the rule is the options, which can be split up into two parts: (2a) the detection options, and (2b) the non-detection options. They determine how a packet is analyzed, and contain fields for matching packet headers, or sections of the payload content, among others. Non-detection options provide additional information regarding the type of traffic the rule wants to detect, such as the category of threat, references to documentation about the threats, as well as rule ID and version number.

Rule developers create rules that are as specific to the threat as possible, but also generalizable to different versions of the same threat present on the Internet. Regardless of how this trade-off is made, it will inevitably imply some fraction of alerts being false positives, due to the sheer amount of network traffic flowing through the Internet. A false positive is defined as normal or legitimate traffic that is mistaken as malicious. Conversely, malicious activity can be missed by an IDS if there is no rule present that explicitly detects it. In order to keep this risk at a minimum, organizations need to continuously keep up with the latest threat intelligence and ensure their NIDS is up to date. Organizations can do this by either purchasing a subscription to a commercial ruleset created by a third party, or develop the rules in-house.

Figure 2 illustrates the typical life cycle of an NIDS rule. Rules developed by the MSSP are either directly added to the production environment, or are first added to a test environment to evaluate accuracy and false positive rates. Commercial rulesets are added to production directly, as it is assumed that these rules have already undergone an acceptable level of quality control. Once in production, the MSSP only updates its own proprietary rules; the commercial rules are left alone and are only filtered out through other methods. Rules in the production environment trigger alerts if the logic within the rule matches with an incoming packet. Finally, only if an alert is deemed severe enough by the analysts in the SOC, the alert is grouped together with a number of related alerts into an *incident*. This incident is then thoroughly investigated by the security analysts.

4 DATA & METHODOLOGY

Our collaboration with an MSSP has enabled us access to a number of datasets surrounding their NIDS and SOC operations. In this section we discuss these datasets and how we use them to perform this work.

4.1 Data collection

NIDS rulesets The MSSP uses rulesets from different origins on the network sensors that they install on their customers' networks. In addition to commercial rulesets purchased from Emerging Threats [19] and VRT [5], they also employ a proprietary ruleset that is created and maintained by an in-house team of developers. All rulesets are hosted on internal Git repositories. The MSSP has been using its own ruleset since 2008, and the repository has been tracking the changes to that ruleset ever since. Commercial rulesets have also been used since that time, but have only been mirrored on a local repository since 2018 or 2019, depending on the ruleset, meaning that the commercial ruleset data are limited to those years. Both VRT (now Talos) and Emerging Threats are market leaders in NIDS rulesets, with the former being the maker of the official Snort ruleset, and the latter being one of the three paid services specifically mentioned by OISF, Suricata's maintaining organization, in their 2018 SuriCon conference [14]. We feel this is a representative sample of the NIDS market, and we therefore limit our study to the two commercial rulesets.

Alert data The rules installed on the network sensors produce alerts when they detect threats, and all the alerts are logged when they arrive at the SOC. We have access to alert logs that date from mid-2009 to mid-2018. The logs are in CSV format and contain the following information: (i) alert timestamp; (ii) rule ID, revision, and priority; (iii) rule category; (iv) protocol of packet (TCP, UDP, ICMP, IP, numerous application layer protocols); and (v) potential corresponding incident. Between 2017 and 2018, the MSSP was gradually migrating to a different logging platform, causing this logging data to be incomplete throughout that last year.

Incident data One or more alerts deemed critical enough by SOC analysts to merit closer investigation are grouped together into an *incident*. The analysts then investigate all related alerts to determine the cause and severity of this incident and ultimately label it accordingly. These labels are *Undetermined*, *False positive*, *Not interesting*, *Interesting*, *Low risk*, *High risk*, and *Successful hack attacks*.

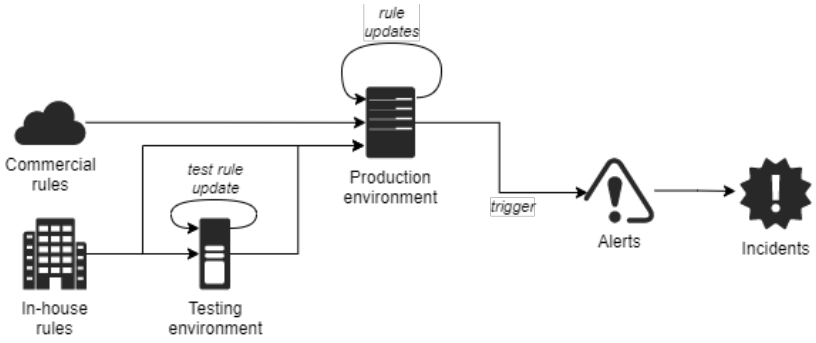


Figure 2: Life cycle of an NIDS rule. Commercial rules are added to production immediately, while many rules developed in-house are tested within a testing environment before they are added to production. Once in production, rules are updated or removed from the ruleset. Rules in the production environment produce alerts if they detect packets that match its detection logic. If deemed severe enough, related alerts are grouped together into an incident and thoroughly investigated by analysts.

We consider all but Undetermined and False positive incidents as true positives. The difference between a *High risk* incident and a *Successful hack attack* is that the former involves activities that will directly lead to network compromise, while the latter is an incident where such network compromise has already occurred without the SOC being able to halt the attempt. *Undetermined incidents* (0.9%) are excluded, as we cannot establish a reliable label for them.

The incident logs contain the following: (i) incident open, response, and close timestamps; (ii) corresponding customer; (iii) category label; (iv) whether the incident was escalated to the customer. This data also dates from mid-2009 to mid-2018 and is also missing entries from 2018 due to the platform migration process.

Interviews We carried out semi-structured interviews with seven analysts and rule developers employed by the MSSP to better understand their heuristics and work processes. We recorded, transcribed, and coded the interviews to extract the relevant information.

4.2 Quantifying rulesets' evolution

Although changes to the rules are tracked by the Git repository, the evolution of the ruleset as a whole is not. For the MSSP's proprietary ruleset, there are no set guidelines when it comes to managing rules and rulesets. New rules are created on the basis of new threat intelligence, and from analysis of potentially malicious software. Furthermore, we learn from the interviews that there are, in general, two instances where ruleset updating occurs: (i) when analysts or rule writers have no other work activities planned and go through the ruleset out of their own initiative; (ii) when a rule triggers alerts more often than is deemed acceptable by the analysts. While the creators of the commercial rulesets might also have such guidelines, the reasons behind rule updates are not logged (publicly) either.

We make a distinction in the methodology that we apply to the two different datasets that we have access to. The first is the repositories of both the commercial and proprietary NIDS rules. The second is the datasets containing the triggered alerts and incidents, which will be discussed in the next subsection. We analyze the repositories of the commercial and proprietary rulesets separately, since the two types of rulesets differ significantly not only in size, but also in purpose.

First, we implement a tool that is able to track the evolution of the ruleset over time. This tool checks out every commit made to the repository and keeps track of the additions and deletions made to the ruleset. It does this by traversing the repository in a reverse chronological order and parsing the complete ruleset for every commit in pairs of two. A *new rule* is a rule that is present in the latter ruleset and not present in the former. A *deleted rule* is a rule that is present in the former ruleset and not present in the latter; or a rule that is not commented out in the former ruleset and is so in the latter ruleset; or a rule that has been moved to a specific file designated for deleted rules. An *updated rule* is a rule that is present in both former and latter rulesets and differ in text. However, not all updates are made equal. Only updates to the header and detection options will influence the performance of a rule. Therefore, the tool also differentiates between the types of updates made to a rule. The result of this analysis is a list of all the rules that were ever added to the repository, paired with every modification performed on each rule over the lifetime of the repository.

To the best of our knowledge, a free and open-source NIDS ruleset analysis tool has not been developed and released before. We realize that such analyses might be valuable for other researchers and professionals. As a contribution to not only the scientific community, but the numerous users of this type of NIDSs in industry, we have published the source code on Github [1].

The tool's output allows us to calculate a number of statistics, both in total and longitudinally, that we can be used for further analysis of the rulesets. Such statistics include rule changes occurred in total and their type, the lifespan of individual rules, and size of the ruleset over time. All statistics are discussed in Section 5.1.

Manual inspection of rule changes was also performed. Specifically, we randomly sampled 50 rule updates and determined the reasons behind them (i.e., was the rule made more specific/efficient/etc.).

All rules, commercial and proprietary, have their own unique ID, and the MSSP ensures that there is no overlap between the IDs of proprietary rules and commercial rules. Every alert record includes a reference to the unique ID of the rule that triggered it. This enables us to link an alert to a specific version of its corresponding rule. By examining the changes to the Git repository, and combining this information with alert data and statistics, we can learn not only how rules change, but also ascertain the reasons behind those changes and identify trends in the ruleset's evolution (e.g., moving

[1] https://github.com/mathewvermeer/ruling-the-rules

away from very specific to more generic rules, more rule updates than rule deletions, etc.). This data also allows us to examine the relationships between rules and alerts.

The proprietary ruleset itself consists of two parts: (1) the rules in production, and (2) the rules still in testing. In this study, we focus primarily on alerts that influence the workflow of the SOC and its analysts. For that reason, we focus only on the ruleset in production and discuss the evolution of rules in the testing environment separately.

4.3 Interviews

To complement our data on the rulesets, alerts and incidents, we also conducted seven interviews with security professionals within the MSSP who write or manage NIDS rules, or have done so previously in their career. The participants were recruiting though snowballing—i.e., asking each participant for a list of names of other people who would be relevant to interview on the company's NIDS rule development and management. This process was halted when the only names we received were professionals who we had already interviewed before and two participants who did not respond to our invitations. The semi-structured interview protocol consists of 37 questions (Appendix A.1). Each interview was conducted by two interviewers, recorded and transcribed.

These interviews are part of a larger, separate study with similar interviews in other MSSPs. Here, we only include the data from interviews inside the partner MSSP for two narrow purposes: (i) to reconstruct the rule development process and understand the datasets at the MSSP (Sections 3.2 and 4.1); and (ii) to help interpret and validate our findings from the analysis of the rules, alerts and incidents. In the Section 8) (Discussion), we compare our findings to what we heard in the interviews.

4.4 Ethics

To conduct the user study, we received formal approval from the Human Research Ethics Committee at our institution. All interviewees explicitly consented to the recording and transcription of the interview and to the usage of quotes. We minimize the risks of data leaks by anonymizing all data gathered during the interviews. The recordings were stored for the duration of this research on an encrypted hard drive. All answers given were confidential and were only available to the other researchers involved in this project.

5 RULE EVOLUTION IN PRACTICE

The rulesets employed by MSSPs are very large and, at the surface level, we find that changes are made to these rulesets constantly. Upon further inspection, though, we discover a number of different phenomena within the rulesets. Firstly, many of the created rules are never meaningfully updated while remaining in use over a long period of time, though this differs per ruleset: 84% and 68% for the ET Pro Snort and Suricata rulesets, 97% for the VRT and 65% for the MSSP rulesets. Then, there is the subset of rules that are deleted without ever being updated. This subset makes up 12% of the MSSP ruleset, and at most 0.6% of the commercial rulesets. Finally, there are rules that are updated throughout their lifetime. The updates themselves can be divided into four different groups: updates made

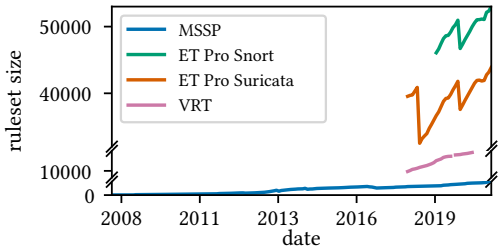


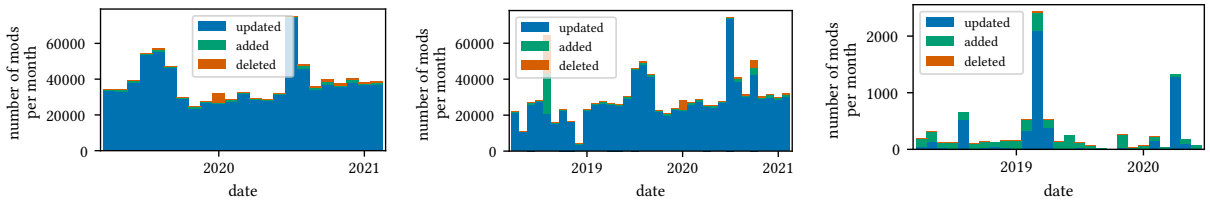
Figure 3: Size of the proprietary and commercial rulesets employed by the MSSP over time. The following are the ruleset sizes at the start and end of their respective curves: MSSP: 20-2,065; ET Pro Snort: 46,074-53,000; ET Pro Suricata: 39,546-43,863; VRT: 9,908-12,760.

to a rule's (1) action, (2) network traffic descriptors, (3) detection options, and (4) non-detection options. The second and third types of updates are what we will mainly focus on, since those are the types that affect detection. The ET Pro rulesets overwhelmingly perform network traffic descriptor updates over all the other types, while the VRT ruleset receives much more non-detection option updates, each signalling different priorities as to their ruleset management strategies. The subsections elaborate on the deeper examination we performed to shed light on the unique behavior of each of the rulesets.

5.1 Ruleset evolution

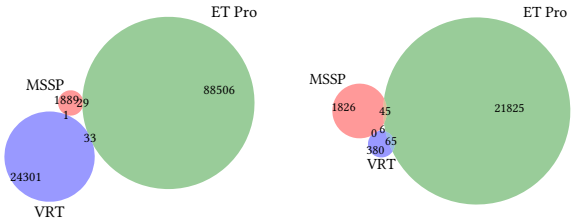
Ruleset size. Figure 3 illustrates the sizes of all the different rulesets employed by the MSSP. It seems to show a general increasing trend for all rulesets. However, the data we have of the commercial rulesets are limited to the last two to three years, trends across longer time spans are not visible. The evolution of the ET Pro rulesets are punctuated by significant purges of rules, with a linear increase between the purges. The third commercial ruleset used is the one created by VRT, now Talos, which is the organization that maintains the official Snort rulesets [5]. Though the size of this ruleset is significantly smaller than the ET Pro rulesets, it exhibits similar behavior to the ET Pro rulesets in the apparent steady addition of new rules without concurrent deletion of old and potentially outdated ones. Figure 4 illustrates this. This could mean that they either prefer to have the most number of threats covered by their rules regardless of quality, or deem most newly created rules of a high enough quality to remain (unchanged) in the ruleset indefinitely. The MSSP's ruleset is significantly smaller than the commercial rulesets, which can be explained by the fact that ruleset creation is not the core of the business. Similar to the ET Pro rulesets, the MSSP also has occasional purges of rules. As evidenced by the rising curve in Figure 3, though, this deletion of rules does not happen regularly, nor does it happen enough to maintain the ruleset at a (near) constant size, similar to the VRT ruleset. This indicates that efficiency through ruleset size limits is not a priority for rule management, though it certainly was in the past [21].

Next, we examine the overlap between the rulesets using two different measures, illustrated in Figures 5b and 5a, respectively. The first method used here is comparing the detection options extracted from all rules in a manner similar to that performed by



(a) ET Pro Snort ruleset (b) ET Pro Suricata ruleset (c) VRT ruleset

Figure 4: Number of additions, modifications, and deletions performed on the different commercial rulesets.



(a) Detection options overlap (b) Unique IP addresses and domain names overlap

Figure 5: Detection options, and unique IP and domain overlap, respectively between the different rulesets.

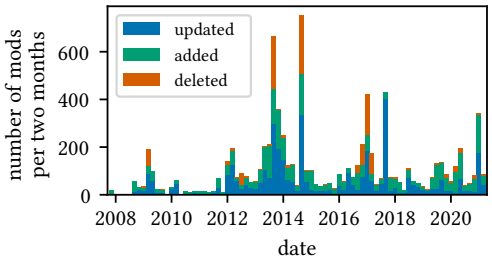


Figure 6: Number of updates in the proprietary ruleset.

Gashi and Asad [2], which aims to measure the functionality of the rules themselves. We supplement this overlap measure with the overlap in unique IP addresses and domains present in a rule. This measure is an indicator for a subset of threat intelligence-based rules that act as a blacklist (i.e., raise an alert solely due to the presence of a specific IP or domain). While the former measure has limitations, since different implementation of the same detection logic would result in no match, taking both measures together yields a general overview of the threat coverage of the different rulesets Irrespective of these limitations, both figures show that despite the large sizes of the commercial datasets and the large coverage they provide, the overlap is nearly negligible.

Rule updates. Of all rules currently active in the different rulesets, many have never been updated after their creation. This percentage stands on roughly 14% for both ET Pro rulesets, 42% for the MSSP ruleset, and 69% for VRT. There is also set of rules that are never touched until their deletion. This phenomenon occurs more often in the MSSP ruleset than the commercial rulesets: of all rules ever added to this ruleset, this number stands at 12%. For all commercial rulesets, this percentage is at most 0.6%.

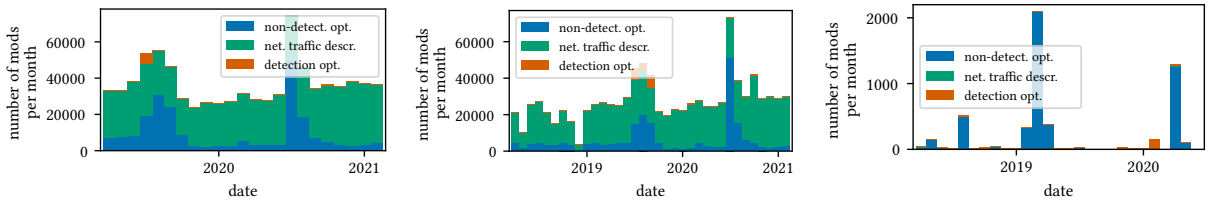
Then, there is the set of rules that are updated throughout their lifetime. We disregard trivial updates to the non-detection options, as these have no effect on the working of a rule. To investigate the nature of these rule updates, we sampled 50 rule updates that alter either the rule header or the rule's detection options from the different rulesets and grouped them into a number of broad categories:

- *Efficiency improvements*: Changes to a rule that make the rule faster or less resource intensive, for instance using the `fast_pattern` keyword or optimizing byte comparisons or regular expressions, while keeping the rest of the rule unchanged.
- *More specific*: Narrowing the rule's search space by, for instance, increasing the length of the content string to match or explicitly specify the subset of ports on which to look for the threat.
- *More general*: Widening the search space in a manner contrary to the previous.
- *Bug fixes*: Updates made to a rule due to the rule not working as intended or improper usage of certain rule options. Examples include detecting buffer overflows using `isdataat` instead of the `dsize` keyword, as usage of the latter can lead to false negatives.
- *Threat intel updates*: Implement into a rule discoveries made through threat intel such as changes to malicious domains or IPs or changes in how certain malware operates.
- *Other*: Any other type of modification such as change in rule action, editing of flowbits, or fixing typos.

This categorization is illustrated in Table 1. Interestingly, for the ET Pro rulesets, all of the sampled rule updates fall under the "threat intel update" category. The sampled updates from both the proprietary and VRT rulesets are distributed in a more balanced manner across the different categories. It seems that there is a general trend

Table 1: Rule update categories for the 50 randomly sampled updates from every ruleset.

Type of update	MSSP	ET Pro Snort	ET Pro Suricata	VRT
Efficiency improvement	10%	-	-	6%
More specific	36%	-	-	30%
More general	10%	-	-	24%
Bug fix	4%	-	-	30%
Threat intel update	22%	100%	100%	-
Other	18%	-	-	10%



(a) ET Pro Snort ruleset (b) ET Pro Suricata ruleset (c) VRT ruleset

Figure 7: Number of updates to existing rules per month and the type of modification for the different commercial rulesets. Updates to non-detection options are shown in blue, network traffic descriptors in teal, and detection options in red.

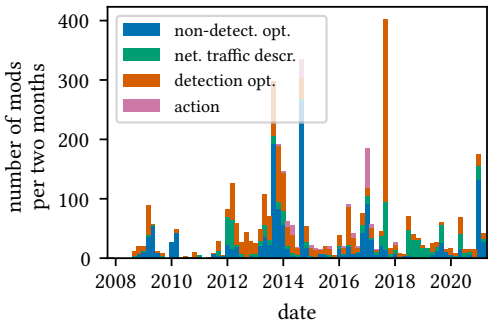


Figure 8: Number of modifications to existing rules in the proprietary ruleset per two months and the type of modification made. Updates to non-detection options are shown in blue, network traffic descriptors in teal, detection options in red, and to the action in pink.

towards making rules more specific, perhaps to reduce the chances of false positive alerts.

We can also split the non-trivial updates into two broad categories, the first of which is network traffic descriptor updates. Emerging Threats is, essentially, the only organization that focuses largely on network traffic descriptor updates (see Figures 7 and 8). The MSSP also performs such updates, although hardly to the extent as Emerging Threats. VRT, on the other hand, performs a negligible number of such updates. Both this, and the findings in Table 1, are consistent with Figure 5b.

The second category of non-trivial updates are those made to a rule's detection options. We first examine these changes made by the updates by means of the Levenshtein ratio between two consecutive versions of a rule's detection options, shown in Figure 9, which is a measure of their similarity. The ratio computed with the following formula: $Lev_{ratio} = \frac{\text{sum of string lengths} - 2 \times Lev_{distance}}{\text{sum of string lengths}}$. Clearly, this metric is not suitable for the analysis of rule update semantics, since single-character changes could be the difference between detecting a threat or not, and large changes could be an attempt at efficiency improvement without affecting detection at all. Therefore, we simply use it to identify update trends and mass update events. We see that the ET Pro rulesets exhibit notably different behavior: the Suricata ruleset has a spike at around the 0.5-mark, and both spike at around the 0.75-mark. The latter spike corresponds to nearly 4,000 nearly identical rule updates in each ruleset, each with a Levenshstein distance of 46. Most of the updated rules detect DNS lookups for potential trojans, ransomware, malicious URLs, etc., and the updates themselves change the manner in which the first 12 bytes of the UDP packet are matched. Interestingly, the spike at the 0.5-mark in the ET Pro Suricata ruleset corresponds to updates made to the same rules as the latter spike, and again altering the way these DNS lookups are detected. In this case, the detection options are updated with the Suricata-specific `dns_query` keyword, which explains why this spike is absent in the ET Pro Snort curve. Other minor spikes in both the ET Pro Snort and Suricata rulesets are similar in that they involve the introduction of case-specific keywords due to efficiency improvements [10]. Interestingly, though, these updates were not the direct effect of keyword release or deprecation, since these updates were made many years after the fact [13], and many rules using the deprecated keywords still remain in the ruleset. In fact, Emerging Threats announced support for these new features back in late 2019 [27], but noted that the implementation of the features were still a "work in progress and under active development," explaining why such updates are still made periodically.

The fact that these spikes are so evident when looking at the ratio indicates that many of the updates in the ET Pro rulesets are made in a single wave to all of the rules of that type. And although the curve of the VRT ruleset appears much smoother than the ET Pro curves, only 355 updates are made to the detection options during the two years we have records of for that ruleset — a minuscule amount compared to the nearly 13,000 rules in this set. Finally, we see that the MSSP ruleset also produces a very smooth graph, indicating that there are no mass update events, and that the updates that do occur are done on a more case-by-case basis. Thus, most detection option changes are made to the syntax of a rule without changing detection itself in a meaningful way, such as the aforementioned DNS lookup rules.

Altogether, it seems a minority of updates are actually made due to rule performance issues. Instead, most updates involve either changes in metadata, syntax, or swapping out threat indicators. Most rules are, evidently, of high enough quality once they enter production. Furthermore, rule development is primarily input driven, with the focus lying not so much on the eventual efficacy or precision of rules as observed by an organization or SOC, but on creating these rules for new threats in the first place.

Rule deletions. Aside from ruleset management through the updating of existing rules, sometimes rules are also deleted from the ruleset. Looking at the deletion behavior in all of the analyzed rulesets, there seems to be little reason to throw out rules. As is the case

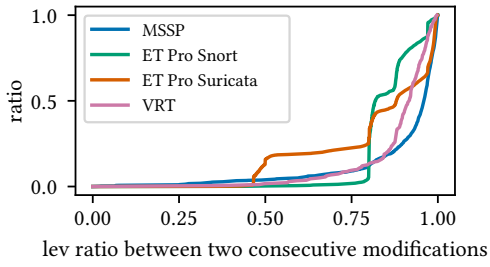


Figure 9: CDF of the Levenshtein ratio between the detection options of a rule and its previous version. A higher ratio indicates a high similarity between both versions.

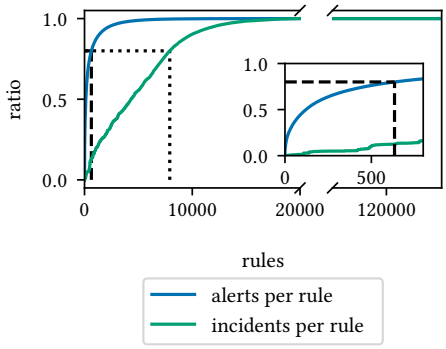


Figure 10: CDF of the number of alerts and incidents triggered over the rules employed by the MSSP. Includes all distinct rules that have ever been added to both the proprietary and commercial rulesets. The dashed and dotted lines indicate the 80%-mark for alerts and incidents, respectively.

with update behavior in the ET Pro rulesets, deletions also revolve around changes in threat intelligence. For instance, a large portion of the deleted rules in the major ET Pro rule purges (see Figure 3) involve outdated C&C servers and SSL certificates [16–18]. Thus, the absence of threat intel-based rules from the VRT ruleset could explain the lack of deletions for this particular ruleset. The MSSP ruleset somewhat shares the deletion behavior of the ET Pro ruleset, since it contains a number of threat intel-based rules, although not to the degree of ET Pro. Many of the deleted MSSP rules, however, fall outside of this category. This unique behavior can be explained by the direct feedback loop that the rule developers have from the MSSP's SOC.

5.2 Rule testing

We must not overlook the fact that rules are tested before being enabled in a production environment. Interestingly, we see that the overwhelming majority of rules are added fairly quickly. It seems that the MSSP's priority is to have a rule pushed to production as quickly as possible, and only optimize the rule after the fact if necessary. That said, 26% of rules from the testing environment never make the cut and are deleted from testing before they make it to the production environment. While many of these rules are deleted relatively early on (roughly a third within the first week), the remaining rules are deleted in a random fashion.

6 ALERTS TRIGGERED OVER TIME

The MSSP with which we collaborate has a particular manner in which alerts arriving at the SOC are processed. Under normal circumstances, all alerts that arrive are processed by the analysts working in the SOC. Deviation from these normal circumstances occurs in cases of false positive floods, for instance. In such cases, the alerts from the responsible rules may be manually suppressed to prevent overburdening the analysts and taking out the SOC back-end systems. In case that one or multiple alerts are suspicious and merit further investigation, these suspicious alerts are grouped together into an *incident*. Incidents are composed of one or multiple alerts that are potentially part of the same threat. Every incident is individually investigated. From this investigation, the analysts determine whether the incident is a false positive, and if so, it is labeled as such. In case of a true positive, the analysts assess the severity of the incident and label the incident If an incident is deemed severe enough, an escalation takes place whereby the customer itself is informed of the incident in order to carry out the necessary defensive and mitigation measures.

We have obtained nine years of alert and incident data for our analysis, from mid-2009 to mid-2018. We first analyze the raw alert data. It simply contains every alert triggered by any active rule present on the probes. Naturally, not every rule will trigger as often as the rest, as some malicious activities are more common than others. We expect this distribution to follow a power-law-esque distribution. Indeed, this is what we find if we plot the data. Figure 10 illustrates this, and we see that 672 rules are responsible for 80% of all alerts. Additionally, out of all the proprietary and commercial rules that we have records of, over 110, 000 rules—85%—did not trigger a single alert. However, we do not have access to the entire lifespan of the commercial rulesets. As a result, we have no records of commercial ruleset modifications before May 2018. Some rules may have been created and subsequently deleted before the MSSP began mirroring the ruleset on its local repository. The actual number of rules that never generated an alert could, therefore, be much larger.

To further investigate the nature of these highly productive rules, we randomly sample and manually examine 50 of the 672 rules. This examination allows us to identify characteristics of the rules that makes them trigger the number of alerts that they do, as oftentimes rules or rule descriptions are unclear, ambiguous, or mislabeled. Of the sampled rules, 52% consists of reconnaissance activity detection and detection of known vulnerability exploitation attempts. This explains the large number of alerts, since these activities are carried out by many a malicious actor on a daily basis across the entire IPv4 space. The remaining 48% not in the two aforementioned categories consists of activities that are not as common an occurrence. Examples include internal network policy violations, DNS requests for malicious domains, or usage of vulnerable software. Therefore, a high level of alerts maintained over a longer period of time is not realistic explanation for the productivity of these types of rules. Indeed, what we find is that 28% of the total sample population are, in essence, quiet rules until a single event causes the rule to trigger up to thousands of times in a single day (see Figure 11).

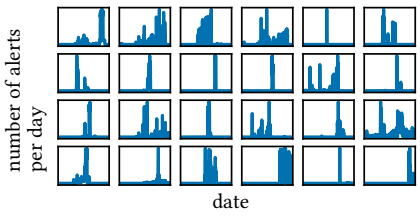


Figure 11: Alerts over time of 24 of the uncommonly triggered rules in our 50-rule sample. Notice the sudden spikes above the quiet baseline for many of them.

As the size of the proprietary ruleset increased since 2008, so have the number of rules that trigger alerts: from 3,758 after the first year to 6,787 just before the MSSP started migrating to a new logging platform (mid-2017). Not only that, but the proportion of the total alerts that are triggered by proprietary rules has also increased from 1% to 6%. Nevertheless, the number of rules that compose 80% of all the alerts per year remains fairly stable every year: roughly between 200 and 250 rules. Between 18% and 50% of these rules overlap year on year.

Of the aforementioned 672 rules responsible for 80% of alerts, 164 (26%) are proprietary rules, even though proprietary rules make up just 3% of all rules employed by the MSSP that we have a record of. Additionally, of the roughly 20,000 distinct rules responsible for all of the alerts, around 1,000 rules are from the MSSP's proprietary ruleset. Taking into account the smaller size of the proprietary ruleset compared to the commercial rulesets (see Figure 3), it is apparent that the proprietary ruleset performs better in terms of rule utility.

Since different rules detect different threats, and, therefore, exhibit different alerting behavior, we examine if such differences in behavior are also reflected in a rule's updates. Table 3 shows that rules in both the commercial and proprietary rulesets that trigger alerts are over eight times as likely to receive an update than rules that do not trigger any alert, and almost thrice as likely when looking solely at the MSSP rules. We stated in Section 5.1 that most of the commercial ruleset updates are threat intelligence-based, where, for instance, out of date indicators of compromise are swapped out for new ones. These are updates that occur regularly, and it makes sense that they be not affected by the the number of alerts that these rules trigger. Looking solely at detection options, the number of updates drops drastically in both alert and non-alert subsets, and the update frequencies per rule drop to 0.18 and 0.16, respectively, meaning that rules that trigger alerts are 12.5% more likely to receive updates than rules that do not trigger.

7 SECURITY INCIDENTS

Most of the alerts produced by the rulesets are not deemed relevant or severe enough by the SOC. Of the millions of alerts that the SOC has processed over the years, only a relative handful are investigated more thoroughly: 735 thousand (or 1.2%), which are produced by 6,720 different rules. This subset of alerts add up to 150 thousand incidents. The precise numbers are specified in Table 2.

Of the 6,720 rules that are present in the incidents, 4,806 (71%) of them are present in 80% of the rules; a very uniform distribution that is in stark contrast to the 672 that produce 80% of all alerts. Interesting here is that of these 672 rules, only 89 are investigated by the SOC, indicating that high alert-producing rules do not necessarily provide usable security information.

The incident labels described in Section 4.1 allows us to group the incidents into two larger groups: risky and non-risky incidents. The risky group contains incidents labeled *Low risk*, *High risk*, and *Successful hack attack*, while the non-risky group is made up of *False positive*, *Not interesting*, and *Interesting* incidents. Due to the granularity of the labeling, this alternate categorization perhaps better represents the relative severity of the different incidents.

Figure 12 illustrates that the number of alerts and (risky) incidents have, not only increased over time, but are also highly correlated. All three curves in the figure are also very much correlated with the increase in the MSSP's customer base. However, we must exclude the exact customer data from the paper due to reasons of confidentiality. The correlations suggests that organizations remain exposed to external threats to a constant degree as time goes on. We encounter the same phenomenon when observing the rate of triggered incidents by rules of a certain age, illustrated in Figure 13. The figure shows that the newest rules produce the most incidents per week, but this activity levels out as the rule gets older. Within several weeks, the rule ceases to produce as much incidents as in the moments closer to its inception. This can indicate an evolution of threats and vulnerability to novel threats, as well as an adaptation to older threats by organizations as to no longer remain vulnerable to them.

Table 2: Number of alerts that compose the different types of incidents, and the corresponding number of distinct rules that triggered the alerts.

	#	Associated alerts	Distinct associated rules
Alerts	-	62,321,663	19,744
Incidents	150,437	735,262	6,720
True positive incidents	69,471	674,177	4,731
Risky incidents	13,589	157,388	2,618
Successful hack attacks	106	734	80

With the commercial rulesets being significantly larger (over 20 times as large), one could likely expect the alerts and incidents to be caused by the different rulesets in similar proportions. This is not the case, however, as we actually find that MSSP rules are overrepresented in many of the true positive incidents, highlighting type of "quality over quantity" philosophy. We calculate the precision of all rules in the different rulesets by dividing the number of true positive incidents of that rule by the total number of incidents in which that rule is present. The ET Pro and VRT rulesets have an average rule precision of 0.68 and 0.65, respectively, and the proprietary ruleset bests both with an average rule precision of 0.74, which clearly shows the higher utility of the MSSP's proprietary ruleset. Indeed, despite making up a fraction of all the rules employed by the MSSP, the proprietary rules are present in 27% of all true positive incidents. Thus, a smaller but contextualized ruleset adds significant detection capability, which supports the economic reasons that were mentioned in the interviews to develop the ruleset.

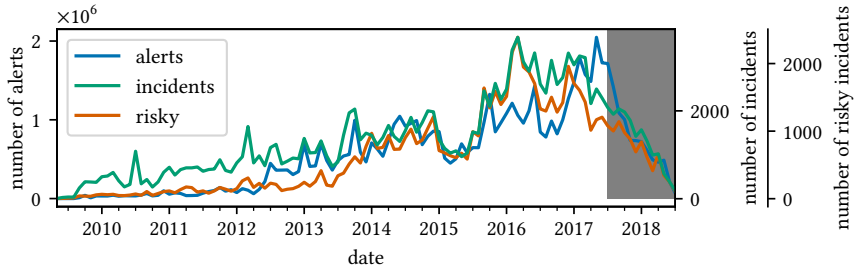


Figure 12: Number of alerts, incidents, and risky incidents handled by the SOC. The grayed out portion on the right-hand side of the graph indicates the MSSP's period of migration to a new logging platform. Note the secondary and tertiary axes.

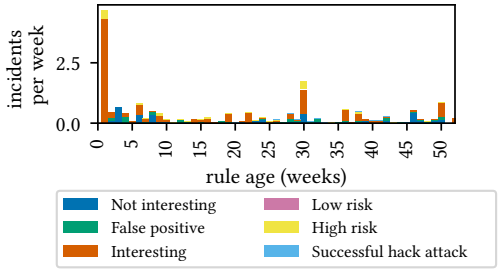


Figure 13: Average number of incidents per week triggered by MSSP rules of a certain age within the first year.

Table 3: Average updates per rule (to header or detection options) for different rule subsets.

Rules subset	# Rules	# Updates	Updates per rule	# Updates (detection options)	Updates per rule (detection options)
No triggered alerts	115,482	276,339	2.39	18,738	0.16
Triggered only alerts	9,988	370,417	37.1	1,768	0.18
Triggered only incidents	6,513	31,635	4.88	1,910	0.29
No triggered alerts (MSSP rules)	2,506	1,097	0.44	626	0.25
Triggered only alerts (MSSP rules)	447	576	1.29	432	0.97
Only TP incidents (MSSP rules)	276	244	0.88	124	0.45
At least one FP incidents (MSSP rules)	293	512	1.74	405	1.38

Most rules that cause false positive and low risk incidents are updated more quickly than the rest of rules. Specifically, with a median of 13 and 15 days, respectively, as opposed to the 30+ days for other incident categories. After the occurrence of false positive incidents, updates mainly fall into three different categories, namely bug fixes, threat intel updates, and making rules more specific (see Table 1 for the full list of categories), as is expected in case of false positive incidents. As for the low risk incidents, most of the rules updated within the first two weeks are updates made to the non-detection options, indicating that low risk incidents are no cause for concern and its corresponding rules are working as intended.

Since the MSSP also tests many of its own rules before adding them to the production environment, we examine its effects on the incidents triggered by these rules. The benefit of testing rules over a longer period of time is not immediately clear from the data. For all rules that trigger incidents and are tested before being added to the production environment, they remain in testing for a median of 11 days, with a notable outlier being the true positive-only rules that are tested for a median of 16 days. The fact that this subset of rules is tested for a longer period seems to indicate that this action has a positive effect on the quality of rules. However, only 21% of this true positive-only subset was tested at all before being added to the production environment.

In addition to examining the incidents themselves, this section will also investigate the effect of incidents on ruleset update behavior. Section 6 points out that the update behavior of alert-producing rules differs significantly from rules that never trigger throughout their lifetime. This section will expand on those findings and analyze the effect of incidents and their type on rule update behavior. In a similar vein as alert-producing rules, Table 3 shows that rules that trigger incidents are more likely to receive updates than rules that do not trigger at all. This holds for both types of updates shown in the table.

Updates made to the commercial rulesets are independent from the MSSP's alert and incident statistics; the MSSP does not influence the update behavior of the commercial rules. Furthermore, the datasets provided to us give us no way of knowing which commercial rules have been suppressed manually, due to, for instance, alert floods or false positives. When looking at the effect of incidents on update behavior, we therefore also focus on the MSSP's proprietary ruleset.

We find in Table 3 that false positive incidents are a much higher driver of updates. Rules that appear in at least one false positive incident are over twice as likely to receive updates than rules that only appear in true positive incidents. You don't change a winning team, and it seems the same goes for NIDS rules. This also makes sense from the perspective of a SOC. If a rule is causing valuable man-hours to be wasted on a fruitless investigation, it may be well worth it to make sure that rule performs as intended.

8 DISCUSSION AND LIMITATIONS

Our analysis has generated several takeaway findings. First, rulesets have grown over time and now a single sensor is running a combined set of over 65k rules. The focus in earlier research on the performance impact of large rulesets on an NIDS, seems to have disappeared as a concern. Only sporadically are rules purged from the set.

Second, notwithstanding the size of each ruleset, there is almost no overlap among them in terms of detection options or what indicators of compromise they contain. This suggests each ruleset covers a different and very limited fraction of the threat landscape, even those from leading vendors like Emerging Threats and Talos (formerly VRT). Thus, to maximize detection, all rules are combined

and enabled by default, regardless of official recommendations against this practice [6]. This finding also underlines the economic reasons for the MSSP to invest also in in-house rule development. It is tailored to high-end threats relevant to their client base which the MSSP felt were not sufficiently captured by the commercial rulesets. The results bear out their intuition. While these proprietary rules make up just a tiny fraction of the entire collection of rules, they contribute disproportionately to the detection of incidents that are evaluated as posing a real threat.

Third, maintenance of the rulesets to improve detection is a minor activity. Most rules are produced once and then either remain untouched or they are modified in bulk for technical reasons, like changes in syntax or replacing the indicators from threat intelligence. In the small fraction of updates where the detection options are changed, the primary reason for the update is to make the rule more specific to the threat it is trying to detect, possible in response to alerts flooding the SOC. Rules that trigger alerts more often are also more likely to receive updates. In rule management, as elsewhere, it seems that the squeaky wheel gets the grease.

Fourth, most rules do not contribute directly to detection. In fact, 85% of all rules never trigger a single alert in their lifetime. Only a minuscule fraction of these tens of thousands of rules are responsible for tens of millions of alerts: 0.5% of all rules generate 80% of all alerts. The overwhelming majority of these are noise. Only 1.2% of all alerts – generated by 5% of all rules – are deemed worthy of investigation by the SOC as incidents. And only a fraction of these investigated alerts – 0.3% of all alerts – turn out to carry legitimate risk to an organization.

8.1 Interviews

Our findings are supported and informed by the interviews with the security analysts who wrote or managed the rulesets. In terms of growing rulesets, the interviewees expressed that they did not really see a downside to keeping all the rules enabled by default. Even though older rules are much less active than newer ones, they are rarely deleted: "*Most of the rules don't really get out of production anymore, because it was proven that the rule works. [...] Maybe the [malicious] tool has new version and the old rule doesn't trigger on this new version, but you cannot say for certain that an actor wouldn't use the old version, so then it's better to have it active.*"

Most participants did not really see experience concerns for the impact of growing rulesets and continually adding rules to the NIDS: *"Unless it starts to flood [...] or becoming slow, maybe we don't even notice it. Then there's no process to evict this rule. [...] I probably think this rule will still be in the ruleset."* That said, some interviewees did express some concern, perhaps carried over from an earlier time, when performance impact might have been a factor. When asked about a computationally-expensive rule for the detection of a banking trojan, one interviewee stated: *"yeah, that's the trade-off, [...] if they already say the performance impact is high, I wouldn't deploy it."*

The lack of overlap – and thus threat coverage – of each ruleset was indeed a concern that led to the development of in-house rules: *"I do think you also need to have rules for [...] more advanced malware that maybe Emerging Threats doesn't really look into."* One analyst remarked that their proprietary rules make up *"thirty to forty percent of the cases we escalate, so these rules are so much better. [...] They are a really small part of our rule set but they have enormous impact."*

In terms of rule maintenance and modifications, one security analyst noted that: *"we tend to only delete rules if they're actually, like, flooding."* This fits with out finding that substantive changes to rules are often to make them more specific to the threat. Another analyst pointed to the fact that rules would be tested, before they are taken into production, thus reducing the need for later modifications: *"usually you give it like one or two days, or preferably a week, to run this rule [...] in testing."*

Finally, no interviewee had remarked on the fact that most rules never contribute directly to detection – i.e., never generating a single alert. This is understandable to some extent. No one would expect all rules to trigger alerts, so they would not evaluate rulesets that way. Lack of coverage would be much more critical than redundant rules. Thus, there is an intrinsic drive towards more rules, even if they never get triggered. All interviewees did remarked on false positives. One analyst said that an ideal rule is one that is *"extremely specific for a specific type of malware [...] with as low amount of false positives as possible, while maximizing true positives."* This is an important trade-off, since most interviewees agree that a false positive alert flood is the most severe situation that can occur in a SOC: *"rule-wise, I think that is the worst thing that could happen: flooding of the SOC. Not only because you cannot handle the alerts anymore, but it could take the back-end down because the database is not responding."* There are no rules on how to make trade-offs like these. Instead, they depend on the experience of a developer. And the resulting outcome is much skewed than the interviews led us to believe: nearly 99% of all alerts are never even investigated and only 0.3% are ever evaluated as indicating actual risk. This corresponds to 16% of all rules.

8.2 Limitations

The ruleset repository tool we developed contains a limitation that is inherent to its design. While it is able to accurately track additions, updates, and deletions from the ruleset, it is possible to introduce errors in its analysis. Since the tracking of rules across commits is based on a rule's ID, altering the ID of an already existing rule is counted as an addition of a new rule with the new ID and a removal of the rule with the old ID. While this does affect the computation of total additions and deletions over time, it does not affect the ruleset's total rule count. Through manual examination, we are able to estimate that of the approximately 8,800 modifications made to the proprietary ruleset, only around 100 have been ID alterations. Rule IDs are meant to be unique identifiers, and thus this limitation is an unfortunate symptom of ruleset mismanagement.

The manner in which the alert data is collected affects the result of the analysis. Specifically, the size of the customer base affects the distribution of alerts over rules. As the customer base of the MSSP grows, the same rules will trigger more often, and the more skewed the distribution will be (see Figure 10). This does not give a realistic view into the alert behavior within a single organization. However, we have approached this analysis from the perspective of the MSSP, not a single organization. Such an issue is inherent to an MSSP and undoubtedly plays a role in the processes that go into the management of the different rulesets.

9 CONCLUSION

Traditional NIDSs and their rulesets are an often overlooked portion of network security. This work presents the first study that aims to shed light on this cornerstone of network security.

After analyzing four different NIDS rulesets containing around 130 thousand rules, we find that the vast majority of rules fail to produce a single alert, i.e., 80% percent of all alerts were triggered by a mere 0.5% of all rules. However, this does not pose a problem for the SOC analysts as rule developers keep adding new rules and barely modifying the existing ones. In fact, only around 23% of all rules are updated in terms of detection capability, with primarily two objectives: (i) adapt to changes in the threat landscape; and (ii) reducing the number of alerts and false positives by making rules more specific. Hence the possibility of using large rulesets without overwhelming SOC analysts. Just 1.2% of all alerts were deemed important enough to be investigated by the SOC, and only 0.3% of all alerts carried significant risk to the organization.

We also identified a set of common rule management practices that include: (i) using multiple rulesets simultaneously due to the lack of exhaustive coverage of the threat landscape by any single ruleset; (ii) creating proprietary rules to cover client-specific threats and updating these with higher frequency than commercial rulesets; and, (iii) reducing false positive incidents by updating the rules that triggered the corresponding alerts.

Finally, our analyses allow us to conclude that the rumours of the death of signature-based monitoring were greatly exaggerated. Contrary to what appeared to be popular opinion, the findings in this paper seem to indicate that that signature-based NIDSs and its rulesets remain vital in network security. Future work is needed to compare this with different approaches, such as host-based detection in the form of end-point monitoring. With such analysis, we move closer to determining the fate of traditional signature-based systems: is it an archaic and obsolete technology or does it remain an indispensable part of a secure network?

ACKNOWLEDGMENTS

This research was partially supported by the Air Force Research Laboratory (AFRL) under agreement number FA8750-19-1-0152. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL or the U.S. Government.

REFERENCES

[1] Eugene Albin and Neil C Rowe. 2012. A realistic experimental comparison of the Suricata and Snort intrusion-detection systems. In *2012 26th International Conference on Advanced Information Networking and Applications Workshops (AINAW)*. IEEE, 122–127.

[2] Hafizul Asad and Ilir Gashi. 2018. Diversity in Open Source Intrusion Detection Systems. In *Proceedings of the 37th International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, Barbara Gallina, Amund Skavhaug, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 267–281.

[3] Bricata. 2021. IDS is Dead! Long Live IDS! An Analyst Prediction from 2003 Remains Relevant. https://bricata.com/blog/ids-is-dead/

[4] Cisco. 2021. Snort - Network Intrusion Detection & Prevention System. https://www.snort.org/

[5] Cisco. 2021. Talos - Author of the Official Snort Rule Sets. https://www.snort.org/talos

[6] Cisco. 2021. Why are rules commented out by default? https://www.snort.org/faq/why-are-rules-commented-out-by-default

[7] David Day and Benjamin Burns. 2011. A performance analysis of Snort and Suricata network intrusion detection and prevention engines. In *Proceedings of the 5th International Conference on Digital Society (ICDS)*. 187–192.

[8] Jason Firch. 2021. 2021 Cyber Security Statistics: The Ultimate List Of Stats, Data & Trends. https://purplesec.us/resources/cyber-security-statistics

[9] Christopher Kruegel, Davide Balzarotti, William Robertson, and Giovanni Vigna. 2007. Improving signature testing through dynamic data flow analysis. In *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC)*. IEEE, 53–63. https://doi.org/10.1109/ACSAC13565.2007

[10] Yaser Mansour. 2021. Rules Authors Introduction to Writing Snort 3 Rules. https://www.snort.org/documents/rules-writers-guide-to-snort-3-rules

[11] P. Mell. 2003. Understanding Intrusion Detection Systems. In *IS Management Handbook*. Auerbach Publications, 409–418.

[12] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. (2018).

[13] OISF. 2021. Announcing Suricata 5.0.0. https://suricata.io/2019/10/15/announcing-suricata-5-0-0/

[14] OISF. 2021. Suricata Update. https://suricon.net/wp-content/uploads/2019/01/SuriCon2018_Ish.pdf

[15] The Zeek Project. 2020. The Zeek Network Security Monitor. https://zeek.org/

[16] Proofpoint. 2021. Daily Ruleset Update Summary 2020/02/24. https://www.proofpoint.com/us/daily-ruleset-update-summary-20200224

[17] Proofpoint. 2021. Daily Ruleset Update Summary 2020/02/25. https://www.proofpoint.com/us/daily-ruleset-update-summary-20200225

[18] Proofpoint. 2021. Daily Ruleset Update Summary 2020/02/26. https://www.proofpoint.com/us/daily-ruleset-update-summary-20200226

[19] Proofpoint. 2021. Emerging Threats Pro Ruleset | Proofpoint. https://www.proofpoint.com/us/threat-insight/et-pro-ruleset

[20] Proofpoint. 2021. Proofpoint Emerging Threats Rules. https://rules.emergingthreats.net/

[21] Soumya Sen. 2006. Performance characterization & improvement of snort as an IDS. *Bell Labs Report* (2006).

[22] Syed Ali Raza Shah and Biju Issac. 2018. Performance comparison of intrusion detection systems and application of machine learning to Snort system. *Future Generation Computer Systems* 80 (2018), 157–170.

[23] N Shone, T.N. Ngoc, V.D. Phai, and Q. Shi. 2018. A Deep Learning Approach to Network Intrusion Detection. *IEEE Trans. on Emerging Topics in Computational Intelligence (TETCI)* 2, 1 (2018), 41–50.

[24] Robin Sommer and Vern Paxson. 2003. Enhancing byte-level network intrusion detection signatures with context. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, 262–271. https://doi.org/10.1145/948109

[25] N. Srivastav and R.K. Challa. 2013. Novel Intrusion Detection System Integrating Layered Framework with Neural Network. In *Proceedings of the 2013 IEEE Advance Computing Conference (IACC)*. IEEE, IEEE, 682–689.

[26] Suricata. 2021. Suricata | Open Source IDS / IPS / NSM engine. https://suricata-ids.org/

[27] Emerging Threats Research Team. 2021. Emerging Threats: Announcing Support for Suricata 5.0. https://www.proofpoint.com/us/corporate-blog/post/emerging-threats-announcing-support-suricata-50

[28] Kittikhun Thongkanchorn, Sudsanguan Ngamsuriyaroj, and Vasaka Visoottiviseth. 2013. Evaluation studies of three intrusion detection systems under various attacks and rule sets. In *Proceedings of the 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013)*. 1–4. https://doi.org/10.1109/TENCON.2013.6718975

[29] Giovanni Vigna. 2010. Network Intrusion Detection: Dead or Alive?. In *Proceedings of the 26th Annual Computer Security Applications Conference* (Austin, Texas, USA) *(ACSAC '10)*. Association for Computing Machinery, New York, NY, USA, 117–126. https://doi.org/10.1145/1920261.1920279

[30] Giovanni Vigna, William Robertson, and Davide Balzarotti. 2004. Testing network-based intrusion detection signatures using mutant exploits. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 21–30. https://doi.org/10.1109/WAINA20374.2012

[31] Hao Wang, Somesh Jha, and Vinod Ganapathy. 2006. NetSpy: Automatic generation of spyware signatures for NIDS. In *2006 22nd Annual Computer Security Applications Conference (ACSAC)*. IEEE, 99–108.

[32] Joshua S. White, Thomas Fitzsimmons, and Jeanna N. Matthews. 2013. Quantitative analysis of intrusion detection systems: Snort and Suricata. In *Cyber Sensing 2013*, Igor V. Ternovskiy and Peter Chin (Eds.), Vol. 8757. International Society for Optics and Photonics, SPIE, 10 – 21. https://doi.org/10.1117/12.2015616

[33] Vinod Yegneswaran, Jonathon T Giffin, Paul Barford, and Somesh Jha. 2005. An Architecture for Generating Semantic Aware Signatures.. In *Proceedings of the 14th USENIX Security Symposium (USENIX Security)*. 97–112.

A APPENDIX

A.1 Interview protocol

The conversation will be recorded and before each interview verbal permission will be sought form each participant using the following text:

1. Do you give permission to record the interview? The recordings will only be used for transcription. It will not be shared beyond the research team.
2. Do you confirm that you are participating voluntarily and aware that you may stop at any time?
3. Do you give permission for the processing of personal data in this study?

Questionnaire:

PROCEDURE

1) Welcome
2) Short overview of the study
3) Explanation of the interview
4) Informed consent
5) Start interview
6) Debriefing

ANALYST RECONSTRUCTION OF WORKFLOW

1) Could you describe to me your workflow?
    a. Probe about routine and non-routine tasks
2) What do you see as the main objectives of your work?
    a. Probe on incentives that they have to reach this objective
3) Can you walk me through the process of the acquiring, creating, changing, and deactivating rules?
    a. Probe on when a rule is added into the sensor
    b. Probe on all the testing that is done before rules are added into the sensor
4) How many rules do you investigate every day?
    a. Probe on how they perceive this task (creative / procedure / workload)
    b. Do you have any tasks that you don't have enough time for?
    c. Probe rule evaluation

MANAGEMENT

5) How do you work together with other colleagues on making or changing rules?
    a. Probe on working together or separation of tasks in specific client's rulesets?
    b. Probe if they ever had a disagreement on certain rules
6) How do you seek additional information in order to assess a rule?
    a. Probe on who they asked, what advice they received.
    b. What kind data they were looking for.
7) Did someone ever follow up with you after you made or changed a rule?
8) In your experience, what is the most severe thing that could go wrong with the rulesets you are using?
    a. Probe on fear of FN
    b. What are potential consequences of a ruleset that isn't functioning properly
    c. Probe on the amount of risk that they perceive on missing TP
    d. Probe on the amount of FP and their perception and definition of a FP
    e. Probe on how likely they think consequences might happen
9) Have there been made any mistakes while adapting rulesets?
    a. Probe on how this came to light.
10) What procedures does the organization have on making or changing rules?

11) Who is responsible for the quality of the rules?
    a. Probe on differences between the responsibility of individual, senior, manager.
12) How is a client involved in the creation of rules?
    a. Probe on relationship with clients
13) Can you give an example of feedback that you received from clients?
14) In your opinion, what could be improved in the management of rules?

OBJECTIVES

15) In your opinion, what is a good ruleset?
    a. Probe on the influence of the volume of a ruleset
16) How do you optimize a ruleset as a whole?
17) What is the best ruleset that is achievable in practice?

EVALUATING RULES

18) Can you walk me through the process on how you determine whether a rule is good or bad?
19) Can you give an example of a good rule?
    a. Probe on why this is a good rule
20) Can you give an example of a bad rule?
    a. Probe on why this is a bad rule
21) Which data do you use for evaluating rules?
    a. Probe on what they think is the most important data
22) Is there additional data that you would like to have?
23) What do you do when you have doubts on a rule?
24) How do you deal with rules that do not or no longer generate any alerts?
25) Is there anything else you would like to tell me that could benefit our research?

[EXPLANATION]

Now we are going to show you two rules that are used in a ruleset of one of your clients. I would like to ask you to take a look at the rule and give your evaluation I encourage you to say everything that comes to your mind. Afterwards I will show you the alerts this rule has captured and then will ask you to evaluate the rule again with this information in mind.

26) RULE EXAMPLE 1: How would you evaluate this rule?

[START TIME MEASUREMENT IN SECONDS]

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET POLICY Vulnerable Java Version
1.8.x Detected"; flow:established,to_server; content:" Java/1.8.0_"; http_user_agent;
content:!"251"; within:3; http_user_agent; flowbits:set,ET.http.javaclient.vulnerable;
threshold: type limit, count 2, seconds 300, track by_src; metadata: former_category
POLICY; reference:url,www.oracle.com/technetwork/java/javase/8u-relnotes-2225394.html;
classtype:bad-unknown; sid:2019401; rev:30; metadata:affected_product Java,
attack_target Client_Endpoint, deployment Perimeter, deployment Internal,
signature_severity Informational, created_at 2014_10_15, performance_impact Low,
updated_at 2020_04_27;)
```

27) The rule you just saw generated the following alerts: 0 TP, 4497 FP, how would you evaluate this rule knowing this information?

[END TIME MEASUREMENT]

28) RULE EXAMPLE 2: How would you evaluate this rule?

[START TIME MEASUREMENT IN SECONDS]

```
alert tcp $HOME_NET any -> any any (msg:"ET EXPLOIT Possible OpenSSL? HeartBleed?
Large HeartBeat? Response (Client Init Vuln Server)"; flow:established,to_client;
content:"|18 03|";depth:2;byte_test:1,<,4,2;flowbits:isset,ET.HB.Request.CI;
flowbits:isnotset,ET.HB.Response.CI;flowbits:set,ET.HB.Response.CI;
flowbits:unset,ET.HB.Request.CI; byte_test:2,>,150,3; threshold:type limit,track
by_src,count 1,seconds 120; metadata: former_category CURRENT_EVENTS;
reference:cve,2014-0160; reference:url,blog.inliniac.net/2014/04/08/detecting-openssl-
heartbleed-with-suricata/; reference:url,heartbleed.com/;reference:url,blog.fox-
it.com/2014/04/08/openssl-heartbleed-bug-live-blog/; classtype:bad-unknown;
sid:2018377; rev:4; metadata:created_at 2014_04_09, updated_at 2014_04_09
```

29) The rule you just saw generated the following alerts: 17 TP, 0 FP, how would you evaluate the rule knowing this information?

[END TIME MEASUREMENT]

30) RULE EXAMPLE 2: How would you evaluate this rule?

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg: "ET TROJAN [PTsecurity]
Tinba (Banking Trojan) Check-in";flow: established,
to_server;content:!"Referer|3a|";http_header;content: "|0d0a0d0a|"; depth:
2000;
byte_extract: 2, 0, byte0, relative;
byte_extract: 2, 0, byte1, relative;
byte_test: 2, =, byte1, 6, relative;
byte_test: 2, !=, byte1, 7, relative;
byte_test: 2, =, byte1, 10, relative;
byte_test: 2, !=, byte1, 11, relative;
byte_test: 2, !=, byte1, 23, relative;
byte_test: 2, !=, byte0, 25, relative;
byte_test: 2, !=, byte1, 27, relative;
byte_test: 2, =, byte0, 40, relative;
byte_test: 2, =, byte1, 42, relative;
byte_test: 2, =, byte0, 44, relative;
byte_test: 2, =, byte1, 46, relative;
byte_test: 2, =, byte0, 48, relative;
byte_test: 2, =, byte1, 50, relative;
content:!"|0000|";depth:30; http_client_body;
content: "|0000|";offset:34;depth:2; http_client_body; fast_pattern;
content: "|0000|";distance:2;within:2; http_client_body;
content: "|0000|";distance:2;within:2; http_client_body;
metadata: former_category TROJAN;
reference:md5,be312fdb94f3a3c783332ea91ef00ebd; classtype:trojan-activity;
sid:10003433; rev:1; metadata:affected_product
Windows_XP_Vista_7_8_10_Server_32_64_Bit, attack_target Client_Endpoint,
deployment Perimeter, tag Banker, signature_severity Major, created_at
2018_08_07, malware_family Tinba, performance_impact High; )
```

31) The rule you just saw generates just a single true positive during the span of one month/year. (Take performance impact into account)

DESCRIPTIVES

32) What is your name?

33) What is your job title?

34) How old are you?

35) What is your educational level?

36) How many years have you been doing this work?

37) Do you know anyone else who we could interview for this research?