Information Engineering Express
International Institute of Applied Informatics
2024, Vol. 10, No. 1, IEE794

Automatic URL Signature Construction and Impact Assessment

Shota Fujii [*], Nobutaka Kawaguchi [*],
Tomoya Suzuki [†], Toshihiro Yamauchi [‡]

Abstract

In the more recent cyberattacks and malware, the servers of the attacker (e.g., C2 servers) play an important role. It is important to use network-based signatures to block malicious communications to reduce the impact. However, the signatures must not block harmless communications during normal business operations. Therefore, signature generation requires a high level of understanding of the business, and highly depends on individual skills. It is necessary to test and ensure that the generated signatures do not interfere with benign communications, which results in high operational costs. We propose *SIGMA*, a system that automatically generates signatures to block malicious communication without interfering with benign communication and then automatically evaluates the impact of the signatures. SIGMA automatically extracts the common parts of malware communication destinations by clustering them and generating multiple candidate signatures. Thereafter, it automatically calculates the impact on normal communication based on business logs, etc., and presents the final signature that has the highest blockability of malicious communication and non-blockability of normal communication to the analyst. We aim to reduce the human factor in generating the signatures, reduce the cost of the impact evaluation, and support the decision of whether to apply the signatures. In our evaluation, we showed that SIGMA can automatically generate a set of signatures that detect 100% of suspicious URLs with an over-detection rate of just 0.87%, based on the results of 14,238 malware analyses and actual business logs. This result suggests that the cost of generating signatures and evaluating their impact on business operations can be reduced; these are time-consuming and human-intensive processes.

Keywords: Malware, Malicious URL, Signature

1 Introduction

Cyberattacks and the malware they use are becoming more and more sophisticated, to the point that they now pose a serious threat to companies and nations. It is more important

[*] Research & Development Group, Hitachi, Ltd., Kanagawa, Japan
[†] Defense Systems Division, Hitachi, Ltd., Kanagawa, Japan
[‡] Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Okayama, Japan

than ever to analyze malware and take immediate countermeasures. In the more recent cyberattacks and malware, the servers of the attacker (e.g., C2 servers) played an important role in sending attack commands and receiving stolen information. To counter this, it is important to block communication to suspicious servers used in cyberattacks to curb the attacks. The signatures for blocking such communication must block malicious communication while simultaneously allowing the benign communication used in daily business. In other words, signature generation requires knowledge of malicious communications and understanding of normal business operations. Therefore, signature generation is not an easy task and requires high-level human resources. In addition, it is necessary to test and ensure that the generated signatures do not interfere with benign communication, and this drives up the operation cost.

In response to the above, we developed a SIgnature Generation and iMpact Assessment (*SIGMA*) system which automatically generates signatures to block malicious communication without interfering with benign communication and then automatically evaluates the impact of the signatures. Our objectives with this system are to reduce the human factor in generating the signatures, reduce the cost of the impact evaluation, and support the decision of whether to apply the signatures. In this paper, we describe the design and implementation of SIGMA and report the results of our evaluation using a prototype.

The contributions of this study are summarized as follows.

- We organized the tasks related to signature creation and impact assessment and then derived the requirements for automating and supporting these tasks.
- We designed SIGMA, a system that creates signatures to detect and block malicious communication without blocking benign communication, and conducts an impact assessment.
- We implemented a prototype of SIGMA in which it automatically generated 43,541 signatures for 14,238 samples and 69,571 URLs. The results showed that it could detect 100% of suspicious URLs with an over-detection rate of just 0.87%.
- We evaluated the processing time of the proposed system and we confirmed that the processing time of web access via the proposed method is within the practical range.

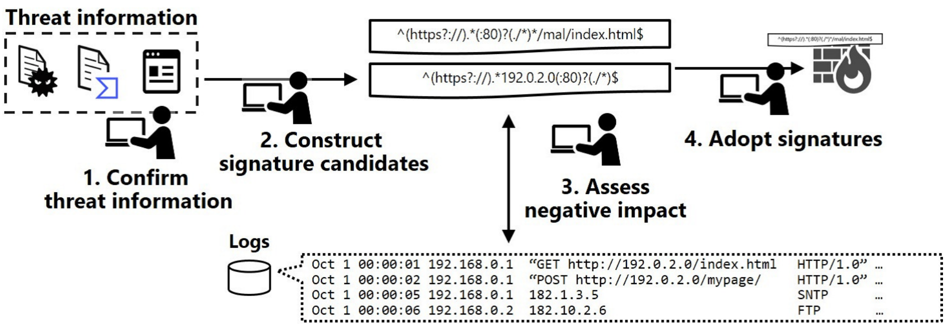
This paper is the extended version of the paper presented at IIAI-AAI 2022 [1]. A summary of the additional elements included in this paper is as follows:

- We clarified the computational complexity of the proposed method (§3.7).
- We evaluated the processing time of the proposed system. Through this evaluation, we confirmed that the processing time of web access via the proposed method is within the practical range (§4.3.3).
- We further clarified the contribution of this study by expanding and adding discussion of related studies (§6).

2 Background

2.1 Network-level Signature

As mentioned above, in recent years, many malware attacks accomplished by communicating and collaborating with the servers of the external attackers have been reported. For


Threat information
1. Confirm threat information
2. Construct signature candidates
^(https?://).*(:80)?(./*)*/mal/index.html$
^(https?://).*192.0.2.0(:80)?(./*)$
3. Assess negative impact
4. Adopt signatures
Logs

```
Oct 1 00:00:01 192.168.0.1  "GET http://192.0.2.0/index.html   HTTP/1.0" …
Oct 1 00:00:02 192.168.0.1  "POST http://192.0.2.0/mypage/     HTTP/1.0" …
Oct 1 00:00:05 192.168.0.1  182.1.3.5                          SNTP      …
Oct 1 00:00:06 192.168.0.2  182.10.2.6                         FTP       …
```

Figure 1: Example of operation flow for developing signatures and impact assessment.

example, *Emotet* uses HTTP Communication to upload files on infected terminals to an external server [2]. The malware used by the attack group *Lazarus* is known to attempt to download attack modules and receive attack commands using http communication [3]. It has also been shown that *xxmm* can receive attack commands from outside using HTTP(s) communication and upload files on infected terminals [4]. Under these circumstances, there is a growing need for damage control by blocking suspicious HTTP(s) communication at the network boundary using network-level signatures, especially outbound communication that involves information leakage and external attacks. One of the Security Operation Center (SOC) tasks is to create and apply signatures to block such communications. Below is the flow of this work (Figure 1).

1. Check the threat information: Check the threat information and get the information of suspicious URLs to be blocked.

2. Create signature candidates: Create signature candidates based on the acquired information and knowledge of the operator.

3. Evaluate the impact: Verify that the signature candidates do not adversely affect the business by comparing them with the business logs.

4. Signature determination: Based on the verification results, determine the signature to be adopted and apply it to various security devices.

2.2 Problems

As discussed in the previous section, it is necessary to come up with a signature candidate from the threat information and then to manually evaluate whether 1) it is possible for the candidate to block the attack and 2) it would have any influence on normal business operations. Since the communications that occur in normal operations differ from organization to organization, organization-tailored signatures need to be created for each organization. This process is therefore highly dependent on the knowledge of the operator who creates the signature. It is also necessary to compare a large number of logs to evaluate the impact of the created signatures. These requirements inevitably lead to a high implementation cost.

3 Automatic Signature Generation and Impact Assessment System

3.1 Objectives and Requirements

It is important to block malicious communications to prevent cyberattacks. However, both the creation of signatures to block malicious communications and the evaluation of the impact of the created signatures are labor-intensive and costly. Therefore, we developed a method that improves the efficiency of this task by automatically creating signatures for detecting malicious communication and evaluating their impact. The requirements to achieve this purpose are as follows.

Requirement 1: Automatically generate candidate signatures tailored to the target organization. To generate signatures to block malicious communications automatically, it is desirable that the generated signatures do not adversely affect benign communications related to normal operations as much as possible. Since the objective is to block malicious communication, we adopt a network-based signature. This is expected to achieve the detection of malicious communications at the network layer while reaping the benefits of the signature such as unification and manageability.

Requirement 2: Quantitatively calculate the possibility of blocking attacks and the impact on benign communication and evaluate the necessity of application. The system automatically evaluates whether the signature can block the attack and whether it has any impact on normal business operations, and then determines whether or not the signature should be applied. This reduces the dependence on human resources and the costs associated with the task.

Requirement 3: Robustness against detection evasion by attackers. When a signature is created for a URL or IP address used in HTTP Communication, a fixed signature is unlikely to cause over-detection, but if the URL, IP address, or path is slightly changed, the attack is likely to be overlooked. It is thus desirable to create signatures that are robust against such detection evasion. In doing so, we can detect the malicious host used in the attack even if the URL, IP address, or path is slightly changed.

In this study, we aim to meet the above requirements to support automatic impact assessment and customization considering the assessment results in a network-based domain.

3.2 Policy and Overview

To block malicious communications, we extract communications from malware analysis results and use them as information sources, and then specify the communications common to multiple malware as signature candidates (Requirement 1). At this time, the created signature candidates are compared with the communications from the malware analysis results and the business logs, and the possibilities of 1) blocking the attack and 2) the non-blocking of normal communication is automatically evaluated (*Requirement* 2). In addition, the system generates signature candidates that are robust against detection evasion by reducing the non-fixed parts common to multiple malicious communications into regular expressions (*Requirement 3*). The extraction of the common parts and regular expressions is repeated while adjusting various conditions and the signature with the highest blockability of malicious communication and non-blockability of normal communication is finally applied.

The overview of SIGMA is shown in Fig. 2. First, the threat information to make the signature is acquired. The malware analysis result obtained from VT (VirusTotal) is used

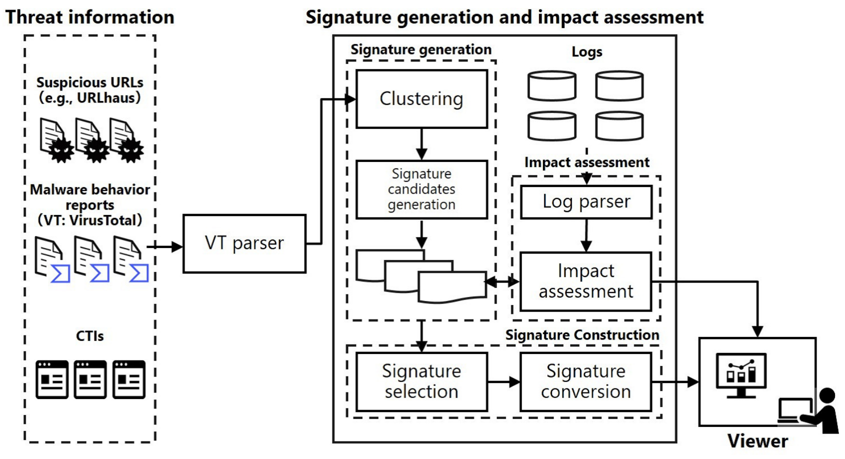


Figure 2: Overview of SIGMA.

as the threat information. Next, the acquired threat information is parsed and input to the signature generation mechanism, which satisfies *Requirement 1*. In parallel, the impact evaluation mechanism parses and stores the business log, compares the log with the signature candidate generated by the signature generation mechanism, and calculates the impact, which satisfies *Requirement 2*. If the impact is greater than a certain level, the signature candidate is created again after changing the parameters. If the impact is less than a certain level, the signature is changed to the format required by the intended security device and presented to the operator. After referring to the generated signature and its impact level, the operator decides whether to apply it.

Each mechanism of SIGMA is described in detail in the following sections.

3.3 Signature Candidate Creation

This mechanism clusters the communication extracted by the report parser. When hierarchical clustering is used, the abstraction level of the common parts (i.e., candidate signatures) to be extracted from the clusters is adjusted by changing the number of clusters while adjusting the threshold value. However, hierarchical clustering is computationally expensive, and when the amount of data increases, the computation time may not be within an operationally feasible range. Therefore, in our method, as shown in Fig. 3, coarse-grained non-hierarchical clustering is performed on all malware samples based on the similarity of communication destinations in the first stage. Then, in the second stage, fine-grained hierarchical clustering is performed on the communication destinations of malware that belongs to the clusters divided in the first stage. This allows us to reduce the amount of computation while still reaping the benefits of hierarchical clustering described above.

The features used in the first stage of clustering are listed in Table 1, following previous studies [5]. Since the number of clusters is unknown in advance, we use Variational Bayesian Gaussian Mixture Model (VBGMM) as the clustering algorithm, which does not require the number of clusters to be specified. Note that we used VBGMM, but other clustering algorithms can be used if they do not require the number of clusters to be specified in advance, such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

The second step, hierarchical clustering, is performed on each of the previously created

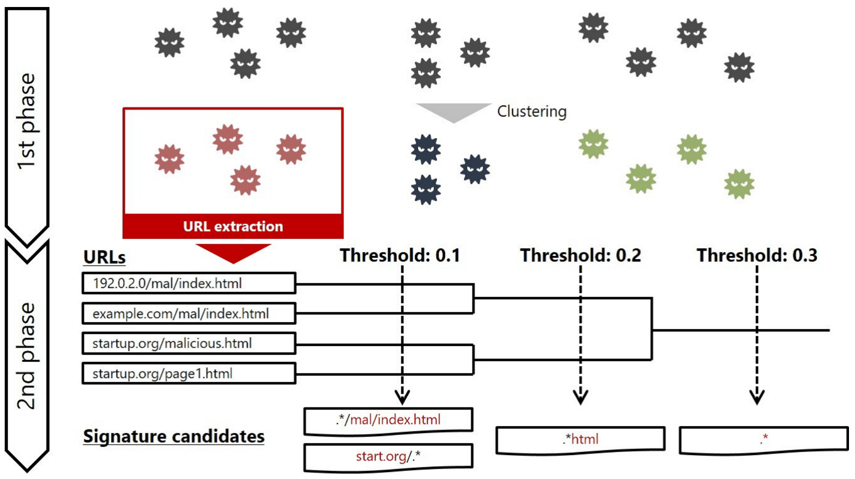


Figure 3: Overview of signature generation.

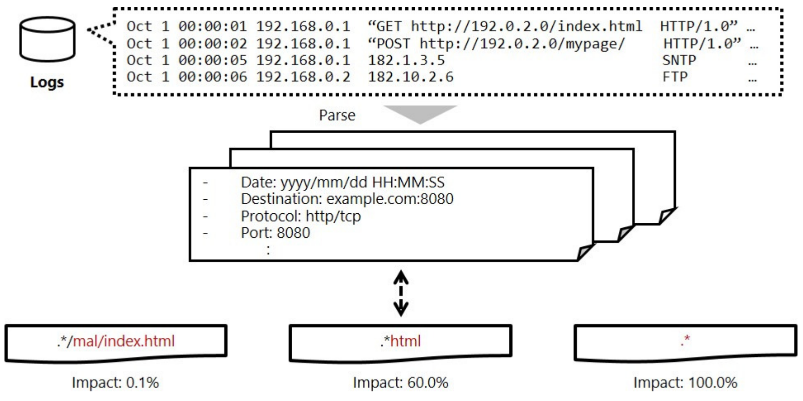
Table 1: Feature values for first clustering.

No.	Features
1	Number of HTTPs
2	Number of GET requests
3	Number of POST requests
4	Average length of URLs
5	Average number of parameters
6	Average data volume of POST requests
7	Average length of responses

clusters. Hierarchical clustering is performed on the URLs in each cluster, using the edit distance between URL strings. In this process, inspired by [6], the edit distance is calculated for each path divided by "/" and the average of the Levenshtein distance is used as the distance between communication destinations. We create clusters of multiple patterns by varying the threshold value of the URL and use the common parts in the clusters as signature candidates. We also generate signature candidates that are robust to detection avoidance by reducing numbers, hexadecimal numbers, and base64 in the signatures to regular expressions using the method in [7]. Specifically, numbers are converted to [0–9], hexadecimal numbers to [a–fA–F0–9], and base64 to [a–zA–Z0–9].

Although we use the result of malware analysis as input, malware may communicate with benign sites to confirm the network communication or to create a decoy against the analysis, and there is a possibility that the signatures generated as a result will include those that block benign communication. Therefore, we remove the benign communication from the signatures in the clustering phase. There are two major methods for this removal.

- *Statistical method.* This method is based on the hypothesis that the communication recorded in the analysis results of many malware samples is normal. It is assumed that much of the communication removed by this method is communication that occurs in the backend of the analysis environment (Windows Update communication, communications that occur when Office software starts up, etc.), rather than communication that is intended by the malware.
- *Allow list method.* This method judges a site that is at the top of the access number



```
Oct 1 00:00:01 192.168.0.1  "GET http://192.0.2.0/index.html  HTTP/1.0" …
Oct 1 00:00:02 192.168.0.1  "POST http://192.0.2.0/mypage/    HTTP/1.0" …
Oct 1 00:00:05 192.168.0.1  182.1.3.5                         SNTP      …
Oct 1 00:00:06 192.168.0.2  182.10.2.6                        FTP       …
```

- Date: yyyy/mm/dd HH:MM:SS
- Destination: example.com:8080
- Protocol: http/tcp
- Port: 8080

:

Figure 4: Overview of impact assessment.

ranking (Alexa Rank, etc.) as normal. The communication to be removed by this method is assumed to be communication to sites such as google[.]com.

In the two-step clustering performed by the proposed method, a set of similar samples is created by clustering malware in the first step. In this case, not only malicious sites but also benign sites may characterize the malware. For example, the *IcedID* campaign at the end of October 2020 accessed www[.]intel[.]com and support[.]microsoft[.]com, etc. as part of a connectivity test and detection evasion [8][9]. This suggests that it is undesirable to perform communication removal using allow list, at least in the first malware classification phase. Therefore, we perform communication removal by statistical methods during the first clustering phase and by allow listing during the second clustering phase.

3.4 Impact Assessment

This mechanism parses the business log and matches it with the signature generated in the previous step, and then calculates the degree to which the normal communication is blocked (Fig. 4). In this study, we used the access log of the forward proxy *squid* [10] as the business log. The formula to calculate the business impact is as follows.

Business Impact (%) = Number of business log URLs that match the signature / Total number of business log URLs

3.5 Signature Construction

This mechanism repeats the creation of candidate signatures and the evaluation of their impact while changing various parameters, and finally determines the signature that has the highest blockability of malicious communication and non-blockability of normal communication as the signature to be applied. The finalized signature is then converted into a format that can be applied to the target network device (Fig. 5). In this study, we assumed that the forward proxy blocks the communication; thus, SIGMA converts signature candidate into the access control format of *squid*.

3.6 Viewer

The final signature is presented to the operator through the viewer along with its impact and other information. An example of the viewer configuration is shown in Fig. 6. The

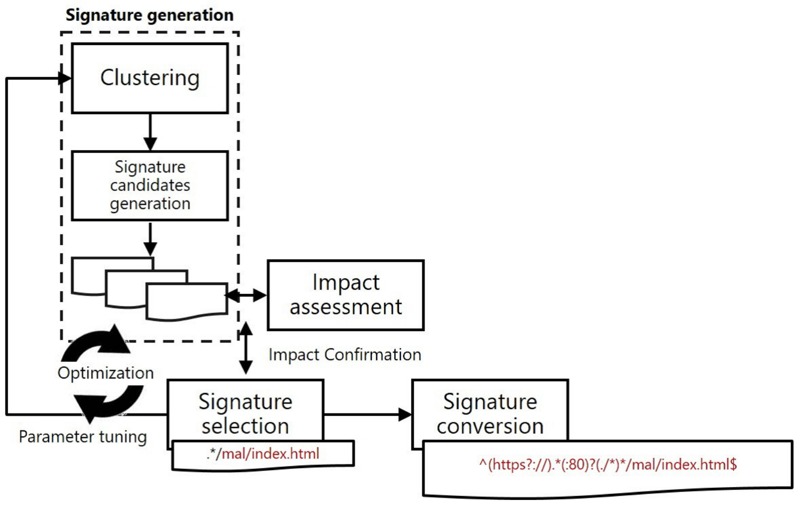


Figure 5: Overview of signature construction.

list screen in the upper row shows the generated signatures and their effects. In the detailed view, the impact of each signature and the affected destinations are shown to support the decision making of the operator on whether to apply the signature or not. For example, the left screen shows how many accesses that match the signature are included in all the access logs. The right screen shows a list of the URIs of the accesses that match the signatures. These were calculated during the impact assessment described above. By using this information together with the impact of the signatures, it is possible to support the decision regarding whether the signatures should be applied. Additionally, it is possible that an access matching a signature is not only a false positive, but also a true positive (i.e., an access to an actual malicious site). In such cases, the system can be used for tasks other than signature creation, such as incident response.

3.7 Computational Complexity

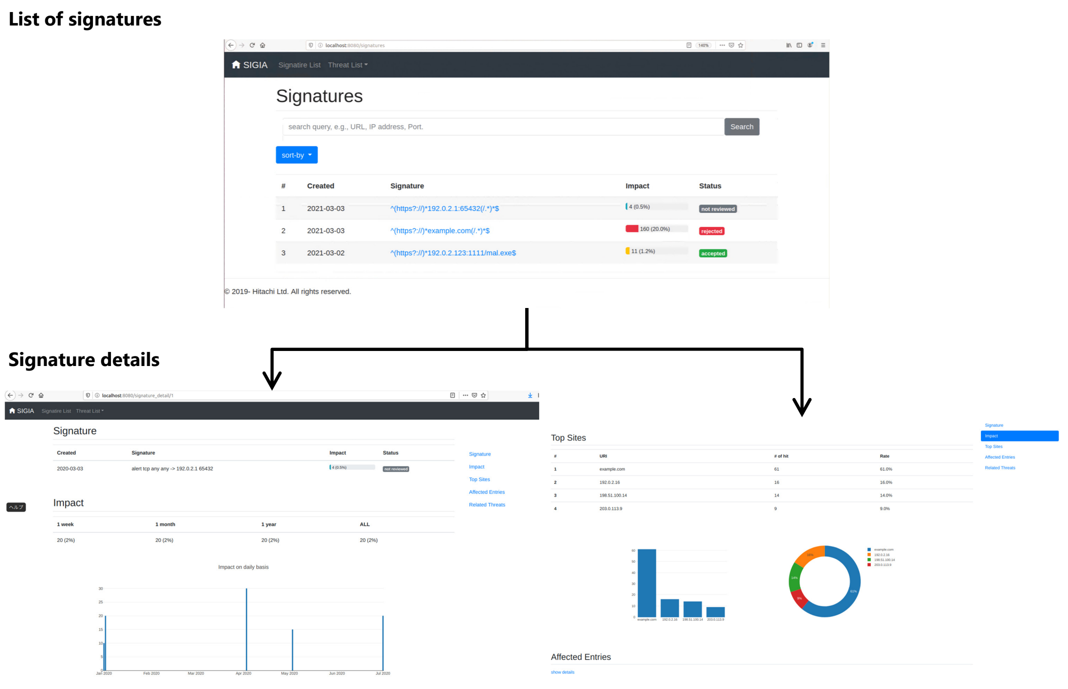
The proposed method mainly consists of clustering in VBGMM in the first stage, hierarchical clustering in the second stage, and signature matching with normal logs in the third stage. We show the computational complexity using Big O notation, with the number of malware as n and the normal logs of the organization as m.

Assuming that all communication logs of all malware are proportional to the number of malware n, the computational complexity of the first stage of clustering is $O(n^2)$ according to the VBGMM computational complexity.

The maximum computational complexity of the second stage of hierarchical clustering occurs when all communication logs are aggregated into a single cluster as a result of the first stage. In this case, the computational complexity is $O(n^2)$ because the number of logs is proportional to n when hierarchical clustering is applied.

The third stage of log matching has a cost of $O(n \cdot m)$, assuming that the number of signatures generated as a result of the second stage is proportional to n. If false positives occur as a result of the matching and the hierarchical clustering is traced back, the worst case is that the matching is repeated for the height of the tree ($log\ n$). Consequently, the order of computational complexity is $O(log\ n) \cdot O(n \cdot m) = O(n \cdot m \cdot log\ n)$.

Finally, the total computational complexity for all steps is below.



Signatures

search query, e.g., URL, IP address, Port.

#	Created	Signature	Impact	Status
1	2021-03-03	^(https?://)*192.0.2.1:65432(/.*)*$	4 (0.5%)	not reviewed
2	2021-03-03	^(https?://)*example.com(/.*)*$	160 (20.0%)	rejected
3	2021-03-02	^(https?://)*192.0.2.123:1111/mal.exe$	11 (1.2%)	accepted

© 2019- Hitachi Ltd. All rights reserved.

Signature

Created	Signature	Impact	Status
2020-03-03	alert tcp any any -> 192.0.2.1 65432	4 (0.5%)	not reviewed

Impact

1 week	1 month	1 year	ALL
20 (2%)	20 (2%)	20 (2%)	20 (2%)

Top Sites

#	URI	# of hit	Rate
1	example.com	61	61.0%
2	192.0.2.16	16	16.0%
3	198.51.100.14	14	14.0%
4	203.0.113.9	9	9.0%

Affected Entries

Figure 6: Overview of signature viewer.

$$O(n^2) + O(n^2) + O(n \cdot m \cdot log\ n) \in O(n \cdot m \cdot log\ n)$$

Therefore, the computational complexity of the proposed method is O($n \cdot m \cdot log\ n$).

4 Evaluation

4.1 Experimental Setup

We implemented a prototype of SIGMA according to the aforementioned design and conducted the following three evaluations.

1. **Impact assessment of benign communication removal.** SIGMA eliminates benign communications during the first stage of clustering by using statistical methods instead of a probable allow list, as the allow list may adversely affect the clustering of samples (as described above). We added *IcedID* to the dataset and compared the clustering results between cases where the statistical method and the Alexa Top 1,000 allow list were used.

2. **Accuracy of created signatures.** We performed a quantitative evaluation of the created signatures using detection and over-detection rates on the dataset.

3. **Processing Time.** The signatures generated by the proposed method were assumed to be applied as a forward proxy deny list. Thus, we performed web access with the signatures generated by the proposed method and verified the effect on processing time.

Table 2: Generated clusters, number of elements in each cluster, and affiliation of *IcedID* (Statistical method).

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
No. of elements	1	574	1	30	407	1	1	4	1	20	5	1	4	9	8	43	93	14	2
IcedID	0	1		0	2	0	0	0	0	0	0	0	0	0	0	27	3	0	0

4.2 Dataset

We prepared a set of malware to generate signatures and benign logs (access logs) to evaluate their impact, as follows.

- *Malware group.* We collected the analysis results for 14,238 samples obtained from VT under the following conditions. The collection period was roughly four months from September 2020 to January 2021.
  - More than ten anti-virus engines detected as malicious (to extract samples with high certainty as malware; established with reference to [11])
  - HTTP communication exists (to perform HTTP communication-based detection)
- *Access log.* We used the actual access logs of 14 employees at the same company. The collection period was from December 2020 to February 2021.

4.3 Evaluation Results

4.3.1 Evaluation 1: Impact Assessment of Benign Communication Removal.

This evaluation was based on *IcedID* samples because *IcedID* has benign sites commonly accessed, and as hypothesized, it is suitable for testing whether statistical methods are more suitable when clustering samples. A total of 1,107 samples from November 2020, when the *IcedID* samples were prevalent, were selected for verification. Tables 2 and 3 show the clustering results of the statistical and allow list methods, respectively. In both cases, the *IcedID* samples were classified into the same cluster. For the statistical method, 27 out of 43 samples in cluster number 15 were *IcedID*s, and the remainder were *IcedID* samples from the past. In contrast, in the allow list method, the majority of *IcedID*s (27 samples) were classified into miscellaneous clusters with 400 elements, which makes it difficult to determine whether they were properly classified. The allow list eliminated the communication that characterized the aforementioned *IcedID*s, so as a result, they were classified into miscellaneous clusters.

In conclusion, as hypothesized, the allow list tends to remove too much communication in the clustering; therefore, a communication removal method using the statistical method is more desirable in the first stage of clustering.

4.3.2 Evaluation 2: Accuracy of Created Signatures.

In this evaluation, we adjusted the over-detection rate to be as small as possible while maximizing the detection rate in consideration of actual work. We also assumed that all signatures were applied even if there was a negative impact on business. In other words, the

Table 3: Generated clusters, number of elements in each cluster, and affiliation of *IcedID* (Allow list method).

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
No. of elements	574	1	1	5	6	4	400	1	2	19	10	3	22	14	5	115	4	2	1
IcedID	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	5	1	0	0

Table 4: Detection and over-detection rates.

Detection rate	Over-detection rate	No. of malicious URLs	No. of signatures
100.0%	0.87%	69,571	43,541

over-detection rate and business impact were the same in this experiment. The results are shown in Table 4.

In the experimental setup, 43,541 signatures were generated from 69,571 URLs extracted from 14,238 samples. First, we confirmed that 100% of malicious communications were detected, while at the same time, 0.87% of normal communication was over-detected. This means that just 0.87% of normal communication is blocked when the generated signatures are automatically applied. Although this is a relatively low amount, it might interfere with business operations. However, we assumed that the signatures with critical impact will not be applied, as the analyst will be notified of the impact of the signatures and asked to decide whether they should be applied. In addition, compared to the case where all these signatures are manually created and the impact judged, SIGMA can significantly reduce human resources and operational costs.

4.3.3 Evaluation 3: Processing Time.

The evaluation environment is shown in Fig. 7. It consists mainly of user PC, proxy, and the pseudo-Internet. The signatures created by the proposed method are registered as a deny list of proxy, and when a user PC accesses the pseudo-Internet, if there is a connection attempt matching the signatures, the communication is blocked. We measured the time required for 10,000 accesses from a user PC to an HTTP server in the pseudo-Internet under the aforementioned environment, with and without the signature of the proposed method. The 10,000 accesses were executed using Python's for-loop and *requests* module. We then verified whether the proposed method increased the processing time, or if it did, whether it was within the practical range. Note that the IP addresses of the pseudo-Internet were assigned to all domains by the DNS in the pseudo-Internet; thus, all domains were connected to the HTTP service in the pseudo-Internet, not to the actual site. The user PC, proxy, and pseudo-Internet were each run as a VM on an Intel Core i9-9900K 3.60GHz host machine *ESXi* 6.7 [12] with *Ubuntu* 22.04 LTS OS [13], 2 virtual cores, and 8GB memory. *INetSim* 1.3.2 [14] was used for the pseudo-Internet. *Squid* 5.2 was used as a proxy. We used *squid*'s *url_regex* to represent signatures using regular expressions, and *http_access deny* to block communications that matched the signatures. The measurements was conducted with the *squid* cache function disabled.

Table 5 shows the evaluation result. First, to verify the processing time with and without signatures, we compared the case without proxy (i.e., # of signatures: 0) and with proxy (# of signatures: 1). The processing time with proxy (# of signatures: 1) was 0.506 seconds

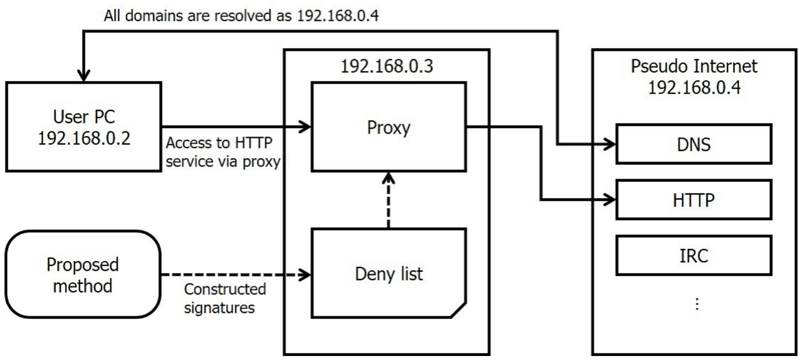


Figure 7: Overview of experimental environment.

Table 5: Processing time of web access with the proposed system.

Setting	Processing time	Overhead		Processing time per access
Proxy with signature (# of signatures: 0)	123.057s	-	-	0.0123057s
Proxy with signature (# of signatures: 1)	123.585s	+0.506s	(+0.413%)	0.0123585s
Proxy with signature (# of signatures: 43,541)	189.344s	+66.287s	(+53.867%)	0.0189344s

longer than that without proxy (+0.413%), but was within the practical range. Next, to verify the processing time when the number of signatures increased, we compared the case without proxy (# of signatures: 0) and the case with proxy (# of signatures: 43,541). The processing time with proxy (43,541 signatures) was 66.287 seconds longer (+53.867%). We also confirmed that the processing time increased with the number of signatures. On the other hand, the processing time per access was about 0.0189s (+0.00663s), and it was still within the practical range.

To summarize, when the signatures created by the proposed method were applied, the processing time increased, but it was still within the range of practical use.

5 Discussion

By using the mechanisms described so far, we aimed to achieve the automatic generation of signatures and automation of impact assessment, and to support and improve the efficiency of signature application in SOC/Computer Security Incident Response Team (CSIRT) operations. As discussed in Section 2.2, the current issue with most signature-generation methods is that the costs of both constructing the signatures and evaluating their impact are high.

Since SIGMA creates signatures automatically, we expect the cost of building signatures to be lowered. SIGMA also automatically generates organization-tailored signatures by adjusting them so that they do not affect the business logs.

In addition, we expect to mitigate the evaluation cost because the impact assessment is automatically performed by comparing the signatures and business logs, and the visualization screen is provided to support the evaluation of whether the application is necessary. One of the advantages of our method is that it can create a signature that detects unknown URLs with similar characteristics by extracting the common parts of suspicious URLs and converting some of them into regular expressions. When we checked the signatures created by SIGMA, we found several signatures in which the domain strings and paths used in multiple attacks were extracted as common parts, and the rest were regular expressions. This made it possible to detect malicious URLs that were not included in the dataset but had been reported separately, such as malicious URLs with only different subdomains and malicious URLs with similar path names. We were able to reduce the number of entries in the deny list to a single entry by using regular expressions, which would have resulted in multiple entries if a simple deny-list method had been adopted, thus reducing the size of the deny list as a secondary effect.

The above results demonstrate that SIGMA can handle different expressions that have not appeared, based on common parts of URLs, and in this respect, it is superior to the deny-list method, which can only detect known and specific URLs. On the other hand, this is only a qualitative example, and it would be desirable to evaluate SIGMA quantitatively using larger-scale data in the future.

The evaluation of processing time was conducted in a pseudo-Internet environment to minimize the impact on actual services. Therefore, it is possible that the actual access to the outside world takes a longer time. However, the signature overhead of the proposed method is the processing time required for URL validation at the proxy and does not affect the external access time described here. Therefore, the increase in processing time is within the practical range in the practical use case as well as in the conclusions stated in the evaluation.

6 Related Work

Many methods have used malicious logs such as malware analysis results to create rules for intrusion detection systems and signatures to block malicious communications, similar to SIGMA. Signatures can be classified into host-based and network-based signatures.

Reference [15] proposes a method for embedding malware API calls and classifying them as malware or not using Bi-LSTM (Bidirectional LSTM). DeepSign [16] is a method that uses deep learning to generate signatures for malware detection based on the results of dynamic analysis of malware. Reference [17] generates host-based signatures in a two-step process that performs coarse-grain clustering followed by fine-grain clustering. Other studies, such as HEAVEN [18], improve the performance of detection methods with hardware support. These methods target host-based signatures and are different in scope from the proposed method for network-based signatures.

Kitsune [19] is a NIDS with an ensemble of autoencoders that performs intrusion detection using the information contained in PCAP as features. Reference [5] developed an automatic signature generation method focusing on HTTP communication, which creates generic signatures by combining multiple clustering methods. Reference [20] also developed a clustering-based signature creation method. MalGene [21] extracted the similarity from the system call sequence of a malware dynamic analysis result and generated the signature. Although these methods are similar to SIGMA in that they create signatures from malicious logs, they do not consider normal communication. It is known that the results of malware analysis may include benign communication, and if the signature is created without considering normal communication, normal communication may be blocked, and business may be affected. In contrast, SIGMA takes not only malicious logs but also business logs as input, enabling it to create signatures that do not block normal communication.

Some of the existing research has also considered normal communication. Reference [22] proposed a method to create signatures from dynamic analysis logs of Android malware, which treats apps downloaded from Google Play as benign and does not affect benign apps in the evaluation. F-Sign [23] uses only the functions executed by the malware as a source for signature generation by filtering a list of functions used in benign files. FIRMA [24] is a method for creating signatures for various network protocols (HTTP, IRC, SMTP, TCP, and UDP) by clustering based on malware network traffic. In the process, signatures that detect benign communications are excluded by using communications from popular sites, etc. as an allow list. A method called EIGER [11] creates signatures based on dynamic analysis of logs of malware and is configured to have no effect on the behavior logs of public Windows applications. Although these methods consider the influence of normal communication, the same as SIGMA, normal communication is limited to general applications. In a business environment, the use of original applications and communication to the intranet is common; therefore, the signature that does not affect public applications may still affect business communication. In contrast, SIGMA creates signatures in such a way that they do not directly affect the business log, making the business impact of the signatures only minimal.

In practice, when applying signatures, it is necessary to test whether they affect the business and to evaluate whether they should be applied based on the test results. The above-mentioned methods do not consider that, and the testing and application decisions thus depend on human resources. In contrast, SIGMA automatically evaluates the impact and provides a visualization to support the decision. In this way, SIGMA supports the actual work, which we believe is a significant advantage.

7 Conclusion

In this paper, we proposed SIGMA, a system that automatically generates organization-tailored signatures that block malicious communication without interfering with benign communication and automatically evaluates the impact of the signatures. SIGMA then repeats the creation of candidate signatures and the evaluation of their impact while changing various parameters, and finally determines the signature that has the highest blockability of malicious communication and non-blockability of normal communication as the signature to be applied. Our analysis showed that SIGMA can automatically generate 43,541 signatures for 14,238 samples and 69,571 URLs and can detect 100% of suspicious URLs with an over-detection rate of just 0.87%. We also confirmed that the overhead of applying the proposed system is within the practical range (maximum +0.00663per access).

Future work will include quantitative evaluation using a larger amount of data. In addition, a more practical evaluation and verification of the practicality of this prototype by applying it to actual business operations will be conducted.

References

[1] Shota Fujii, Nobutaka Kawaguchi, Shoya Kojima, Tomoya Suzuki, and Toshihiro Yamauchi. Design and implementation of system for url signature construction and impact assessment. In *2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 95–100, 2022.

[2] Unit42. Case Study: Emotet Thread Hijacking, an Email Attack Technique, 2020. https://unit42.paloaltonetworks.com/emotet-thread-hijacking/.

[3] JPCERT/CC. Malware Used by Lazarus after Network Intrusion, 2020. https://blogs.jpcert.or.jp/en/2020/08/Lazarus-malware.html.

[4] You Nakatsuru. Understanding Command and Control - An Anatomy of xxmm Communication -, 2019. https://jsac.jpcert.or.jp/archive/2019/pdf/JSAC2019_8_nakatsuru_en.pdf.

[5] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI' 10, p. 26, 2010.

[6] Serita Susumu, Fujii Yasuhiro, Kakuta Tomo, Michiori Yoshitake, Tomoya Ohtori, Kishiro Takeyasu, and Terada Masato. Automatic generation of url regular expression for detecting malicious traffic. In *Computer Security Symposium 2014*, CSS '14, pp. 242–249, 2014 (in Japanese).

[7] Terry Nelms, Roberto Perdisci, and Mustaque Ahamad. ExecScent: Mining for new C&C domains in live networks with adaptive control protocol templates. In *22nd USENIX Security Symposium*, SEC'13, pp. 589–604, 2013.

[8] JUNIPER NETWORKS. COVID-19 and FMLA Campaigns used to install to new IcedID banking malware, 2020. https://blogs.juniper.net/en-us/threat-research/covid-19-and-fmla-\quadcampaigns-used-to-install-new-icedid-banking-malware.

[9] SANS ISC InfoSec Forums. More TA551 (Shathak) Word docs push IcedID (Bokbot), 2020. https://isc.sans.edu/forums/diary/More+TA551+Shathak+Word+docs+push+IcedID+Bokbot/26674/.

[10] Squid: Optimising Web Delivery, 2022. http://www.squid-cache.org/.

[11] Yuma Kurogome, Yuto Otsuki, Yuhei Kawakoya, Makoto Iwamura, Syogo Hayashi, Tatsuya Mori, and Koushik Sen. Eiger: Automated ioc generation for accurate and interpretable endpoint malware detection. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, pp. 687–701, 2019.

[12] VMware ESXi, 2022. https://www.vmware.com/products/esxi-and-esx. html.

[13] Enterprise Open Source and Linux — Ubuntu, 2022. https://ubuntu.com/.

[14] INetSim: Internet Services Simulation Suite, 2022. https://www.inetsim.org/.

[15] Ce Li, Qiujian Lv, Ning Li, Yan Wang, Degang Sun, and Yuanyuan Qiao. A novel deep framework for dynamic malware detection based on api sequence intrinsic features. *Computers & Security*, Vol. 116, p. 102686, 2022.

[16] Omid E. David and Nathan S. Netanyahu. Deepsign: Deep learning for automatic malware signature generation and classification. In *2015 International Joint Conference on Neural Networks*, IJCNN '15, pp. 1–8, 2015.

[17] Mohannad Alhanahnah, Qicheng Lin, Qiben Yan, Ning Zhang, and Zhenxiang Chen. Efficient signature generation for classifying cross-architecture iot malware. In *2018 IEEE Conference on Communications and Network Security*, CNS '18, pp. 1–9, 2018.

[18] Marcus Botacin, Marco Zanata Alves, Daniela Oliveira, and André Grégio. Heaven: A hardware-enhanced antivirus engine to accelerate real-time, signature-based malware detection. *Expert Systems with Applications*, Vol. 201, No. C, p. 117083, 2022.

[19] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. In *Network and Distributed System Security Symposium 2018*, NDSS '18, 2018.

[20] Roberto Paleari, Lorenzo Martignoni, Emanuele Passerini, Drew Davidson, Matt Fredrikson, Jon Giffin, and Somesh Jha. Automatic generation of remediation procedures for malware infections. In *19th USENIX Security Symposium*, SEC'10, 2010.

[21] Dhilung Kirat and Giovanni Vigna. Malgene: Automatic extraction of malware analysis evasion signature. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pp. 769–780, 2015.

[22] Yu Feng, Osbert Bastani, Ruben Martins, Isil Dillig, and Saswat Anand. Automated synthesis of semantic malware signatures using maximum satisfiability. In *Network and Distributed System Security Symposium 2017*, NDSS '17, 2017.

[23] Asaf Shabtai, Eitan Menahem, and Yuval Elovici. F-sign: Automatic, function-based signature generation for malware. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 41, No. 4, pp. 494–508, 2011.

[24] M. Zubair Rafique and Juan Caballero. Firma: Malware clustering and network signature generation with mixed network behaviors. In *Research in Attacks, Intrusions, and Defenses*, RAID '13, pp. 144–163, 2013.