

Research on Sophistication and Improving Efficiency of Practical Security Operations

2023, March

Shota Fujii

Graduate School of
Natural Science and Technology
(Doctor's Course)
OKAYAMA UNIVERSITY

Contents

1	Introduction	1
1.1	Background	1
1.1.1	Increase in Number and Sophistication of Cyberattacks	1
1.1.2	Security Operations of SOC/CSIRTs	1
1.2	Problems and Objectives	3
1.3	Research Problems	4
1.3.1	Operations During a Non-Emergency	4
1.3.2	Operations During an Emergency	4
1.4	Research Strategies	6
1.4.1	Structuring CTI Using a Common Format	6
1.4.2	Tracing the Diffusion of Classified Information and Detecting Information Leakage	6
1.4.3	Survey of Support Mechanisms in Online Sandboxes and Derivation of Best Practices	7
1.4.4	Automatic URL Signature Construction and Impact Assessment	8
1.5	Related Work	9
1.5.1	Utilization of Cyber Threat Intelligence	9
1.5.2	Tracing Diffusion of Classified Information and Detecting Information Leakage	10
1.5.3	Support Functions in Dynamic Analysis of Malware in Sandboxes	12
1.5.4	Detection of Malicious Communications by Creation of Signatures	13
1.6	Outline of the Dissertation	14
2	Information Extraction from Unstructured CTI	15
2.1	Introduction	15

2.2	Background and Research Questions	18
2.2.1	Cyber Threat Intelligence	18
2.2.2	NLP	18
2.2.3	Challenges	19
2.3	Design and Implementation	19
2.3.1	Basic Idea and Overview	19
2.3.2	Information Gathering	21
2.3.3	Preprocessing	21
2.3.4	Pretraining	21
2.3.5	CTI Classification	23
2.3.6	Named Entity Recognition	23
2.3.7	Relation Extraction	23
2.3.8	STIX Generation	24
2.4	Evaluation	25
2.4.1	Experimental Setup	25
2.4.2	Dataset	26
2.4.3	Result	27
2.5	Analysis	30
2.5.1	Overview	30
2.5.2	IOC Coverage	32
2.5.3	Time-series	33
2.5.4	Information Source Relation	35
2.6	Discussion	35
2.6.1	Practicality	35
2.6.2	Limitation	37
2.6.3	Research Ethics	37
2.7	Conclusion	38
3	Tracing Diffusion of Classified Information on KVM	39
3.1	Introduction	39
3.2	OS-Based Function for Tracing Diffusion of Classified Information	41
3.2.1	About This Section	41
3.2.2	Classified Information Diffusion Path	41

3.2.3	Purpose	41
3.2.4	Overview of the OS-based Tracing Function	42
3.2.5	Problems with the OS-Based Tracing Function	43
3.3	VMM-Based Function for Tracing Diffusion of Classified Information	44
3.3.1	Requirements	44
3.3.2	Overview of the VMM-Based Tracing Function	45
3.3.3	Tasks	46
3.3.4	Collecting System Call Information with Virtual Machine Monitor . .	46
3.3.5	Collecting OS Information with Virtual Machine Monitor	47
3.3.6	Advantages	48
3.4	Implementation	49
3.4.1	Environment	49
3.4.2	File Operation	49
3.4.3	Child Process Creation	50
3.4.4	Inter-process Communication	51
3.5	Evaluation	54
3.5.1	Experimental Setup	54
3.5.2	Traceability	55
3.5.3	Lines of Code	57
3.5.4	Overheads	59
3.6	Conclusion	65
4	Survey and Analysis on ATT&CK Mapping Function	67
4.1	Introduction	67
4.2	Background and Research Questions	69
4.2.1	Online Sandbox	69
4.2.2	MITRE ATT&CK	70
4.2.3	Problems	70
4.2.4	Research Questions	71
4.3	Methodology	72
4.3.1	Design of Survey	72
4.3.2	Survey Subjects	72
4.3.3	Dataset	73

4.4	Results	76
4.4.1	Overview of Survey	76
4.4.2	RQ1: Are There Differences in ATT&CK Mapping Capabilities between Online Sandboxes?	76
4.4.3	RQ2: Are There Techniques that are Easy or Difficult to extract in Online Sandboxes?	78
4.4.4	RQ3: Are There Techniques that Tend to be Mapped to Benign Files?	81
4.4.5	RQ4: Are There Differences in Characteristic between Other Technique Detection Methods?	85
4.5	Discussion	87
4.5.1	Best Practice	87
4.5.2	Limitation	88
4.5.3	Research Ethics	89
4.6	Conclusion	90
5	Automatic URL Signature Construction and Impact Assessment	91
5.1	Introduction	91
5.2	Background	92
5.2.1	Network-level Signature	92
5.2.2	Problems	93
5.3	Automatic signature generation and impact assessment system	94
5.3.1	Objectives and Requirements	94
5.3.2	Policy and Overview	95
5.3.3	Signature Candidate Creation	95
5.3.4	Impact Assessment	98
5.3.5	Signature Construction	98
5.3.6	Viewer	99
5.4	Evaluation	101
5.4.1	Experimental Setup	101
5.4.2	Dataset	102
5.4.3	Evaluation Results	102
5.5	Discussion	105
5.6	Conclusion	106

6	Conclusions	108
6.1	Concluding Remarks	108
6.2	Future Directions	110
	Acknowledgments	113
	References	114
	Appendix A Information Extraction from Unstructured CTI	129
A.1	Source of CTI	129
A.2	Refang Rules	129
	Appendix B Survey and Analysis on ATT&CK Mapping Function	132
B.1	Detailed information on the validation of the ATT&CK Technique mapping function	132

List of Figures

1.1	Overview of SOC/CSIRT operations.	2
1.2	Relationship between each method and SOC/CSIRT operations.	7
2.1	Overview of CyNER.	20
2.2	Extraction method for noncontextual IOCs of CTI.	25
2.3	Confusion matrix of each named entity.	29
2.4	Number of AV detections for IOCs included in VirusTotal.	33
2.5	Lifetime for each type of IOC.	34
2.6	Number of sources including the IOC.	36
3.1	Overview of the OS-based tracing function.	42
3.2	Overview of the VMM-based tracing function.	44
3.3	Log generated by the VMM-based tracing function in (Assumed Scenario 1).	55
3.4	Log generated by the VMM-based tracing function in (Assumed Scenario 4).	57
4.1	Top 10 MITRE ATT&CK Technique for each sandbox.	74
4.2	More than one MITRE ATT&CK Technique was found in the sandbox analysis results.	80
4.3	MITRE ATT&CK Technique for the top 10 p-values	82
4.4	MITRE ATT&CK Technique for the lower 10 p-values	83
4.5	MITRE ATT&CK Technique mapped by each technique	84
5.1	Example of operation flow for developing signatures and impact assessment.	93
5.2	Overview of SIGMA.	96
5.3	Overview of signature generation.	97
5.4	Overview of impact assessment.	99
5.5	Overview of signature construction.	100

5.6	Overview of signature viewer.	101
5.7	Overview of experimental environment.	105

List of Tables

2.1	Named entity list.	22
2.2	Relation rule list.	23
2.3	NER accuracy of each model.	27
2.4	Recognition result of each named entity.	28
2.5	Processing time of proposed method per CTI.	30
2.6	Accuracy of relation extraction.	31
2.7	IOC coverage of each platform.	31
3.1	System call utilized for socket communication. The relevant parts are marked with “✓.” .	52
3.2	Evaluation environment.	54
3.3	Comparison of logical LOC and the number of files modified for the tracing function. . .	58
3.4	Overhead of system calls incurred by the VMM-based tracing function (μ s).	60
3.5	LMbench results (μ s).	62
3.6	Time and overhead for building bzImage. (s)	63
3.7	Average response time and overhead of Web server. (ms)	64
4.1	Data overview.	73
4.2	Similarity of MITRE ATT&CK Technique mapping results between sand- boxes by formula (1).	73
4.3	Analysis environment for each sandbox.	74
4.4	Number of observations and presence of significant differences among sand- boxes for each MITRE ATT&CK Technique (top 10 observations for each sandbox).	75
4.5	Usage of the deprecated MITRE ATT&CK Technique per sandbox.	78
4.6	Number and percentage of each MITRE ATT&CK Tactic present.	81
4.7	Extraction trend of MITRE ATT&CK Technique by each method.	83

4.8	Technique observed in multiple methods and presence/absence of significant differences between methods (excerpt).	86
5.1	Feature values for first clustering.	98
5.2	Generated clusters, number of elements in each cluster, and affiliation of <i>Ice-dID</i> (Statistical method).	100
5.3	Generated clusters, number of elements in each cluster, and affiliation of <i>Ice-dID</i> (Allow list method).	102
5.4	Detection and over-detection rates.	103
5.5	Processing time of web access with the proposed system.	105
A.1	Source websites of CTIs.	130
A.2	Refang and defang rules.	131
B.1	Number of observations and presence of significant differences among sand-boxes for each MITRE ATT&CK Technique (RQ1) (1/3).	134
B.2	Presence of significant differences between malware and benign files for each technique (RQ3) (1/2).	137
B.3	Technique observed in multiple methods and presence/absence of significant differences between methods (RQ4) (1/4).	139

Summary

There are more cyberattacks each year, with the number of communications subjected to such attacks observed per year between 2018 and 2021 increasing by approximately 2.4 times; this number is still increasing, and more of these attacks are becoming sophisticated. Circa 2000, the purpose of most attacks was self-expression, but since 2010, the intent is clearly more malicious, focusing on information theft and terrorism instead. Additionally, attacks not only indiscriminate but also targeted against specific organizations have emerged; such attacks are more difficult to detect. Organizations such as government offices, enterprises, and universities tend to suffer more damage from cyberattacks. More countermeasures against cyberattacks are being developed because these attacks not only cause economic damage but can also result in secondary damage such as damage to the reputation of the organization. To detect and respond to cyberattacks under such circumstances, most organizations have established a security operations center (SOC) or computer security incident response team (CSIRT).

SOCs/CSIRTs are reported to be short on human resources and need to be more efficient. Additionally, the abilities and maturity levels of operators vary, raising the need to systemize and automate SOC/CSIRT operations. Network-related cyberattacks tend to cause extensive damage; thus, it is particularly important to address them. The intruder of an organization's network can obtain access to various devices and information. Furthermore, there are cases wherein confidential information within an organization gets leaked through networks. Therefore, this study proposes various methods to improve the efficiency of SOC/CSIRT operations, especially those related to network security. Specifically, we have categorized these operations based on existing frameworks and derived problems that may hinder growth in the efficiency of these operations. Then, we have proposed methods to mitigate each problem.

First, we propose a method for structuring cyber threat intelligence (CTI). The increase

in the number and sophistication of cyberattacks has supported increased attention toward collecting and analyzing CTI to be familiar with the latest threat-related information. This information can be efficiently utilized to support the creation of efficient responses in case of emergencies. Since most CTI is written in natural language, analyzing it can be time-consuming and expensive. Additionally, various organizations publish information separately, making cross-sectional analysis difficult. To solve these problems, we propose CyNER, which supports analysis by automatically structuring CTI in the Structured Text Information eXpression (STIX) format, a format commonly used for CTI. CyNER extracts named entities in the context of CTI and converts them to the STIX format by extracting relationships between named entities and indicators of compromise (IOCs); this is expected to improve efficiency and realize cross-sectional analysis. In the evaluation, we showed that the model trained on a corpus for the cybersecurity domain could improve the F value of named entity recognition (NER) by up to 2.6 points. We also analyzed CTI cross-sectionally, showing that CyNER could extract IOCs not included in existing reputation sites, that more than 97% of IOCs were included in only a single source, and that CyNER could automatically extract IOCs that had been exploited over time and across multiple attack groups. These results demonstrated the potential of CyNER in contributing to the efficiency of CTI analysis.

Second, we propose a method to trace the diffusion of classified information on a guest operating system (OS) using a virtual machine monitor (VMM) and detect information leakage outside this guest OS; information leakage has increased in recent years. A function for tracing the diffusion of classified information within an OS has been proposed to address this problem. However, this function has two shortcomings. First, to introduce the function, the source code of the OS needs to be modified. Second, there is a risk that this function will be disabled when the OS is attacked. Thus, we have designed a function for tracing the diffusion of classified information in a guest OS by using a VMM, which allows the proposed function to be introduced in various environments without modifying the source code of the OS. It is also expected that attacks specifically targeting this proposed function will be difficult to carry out because the VMM is isolated from the OS. This dissertation describes the design and implementation of the proposed function for file operations, child process creations, and inter-process communication (IPC) in the guest OS through the kernel-based virtual machine (KVM), a type of VMM; demonstrates the traceability of classified

information diffused by file operations, child process creations, and IPC; and evaluates the logical lines of code required to implement the proposed function and performance overheads. The evaluation results showed that the implementation and usability of the proposed function were realistic.

Third, we have described the results of the investigation of the mapping function of the Adversarial Tactics, Techniques, and Common Knowledge (MITRE ATT&CK) technique, which is one of the analytical support functions in online sandboxes. Dynamic analysis, which automatically analyzes malware, has become the de facto method to deal with a large amount of malware. There is an analytical support function that maps malware behavior to each element of the MITRE ATT&CK technique, which has been adopted by many online sandboxes and contributes to the efficiency of analysis. On the other hand, this function depends on the implementation of mapping rules, which may affect the analysis results. Therefore, we conducted a survey of online sandboxes that had a technique-mapping function. Through this survey, we clarified the actual status of the mapping function, such as the differences in mapping among sandboxes and those in mapping trends compared to manual mapping; and we also derived the best practices for their use.

Lastly, we propose a method for automatically creating signatures to detect communications to suspicious destinations using the results of malware analysis. As mentioned above, malicious hosts play a significant role in cyberattacks. Therefore, blocking communications to these hosts is important. These signatures are required for blocking both malicious communications and benign communications in normal business operations. Therefore, generating these signatures requires a high degree of business understanding and is highly-personalized. Additionally, the generated signatures need to be tested for their impact on benign communications in actual operation, which is also expensive. To solve these problems, we propose a system that automatically generates signatures that block malicious communications without interfering with the benign ones and automatically performs an impact assessment of the signatures. The proposed system is expected to reduce the human resources required for signature generation, reduce the cost of impact assessment, and assist in deciding whether a signature is applicable or not. This dissertation describes the design and implementation of this system and its evaluation using a prototype. The evaluation showed that the system could reduce the cost of each task by automatically creating signatures and evaluating their impact; and it showed that the generated signatures could block commu-

nication to malicious hosts. The overhead incurred by the proposed system was within the practical range.

To summarize, SOC/CSIRT operations, especially those related to networks that are susceptible to extensive damage, can be made more effective and efficient using these proposed methods.

Chapter 1

Introduction

1.1 Background

1.1.1 Increase in Number and Sophistication of Cyberattacks

There are more cyberattacks each year, with the number of communication networks subjected to such attacks observed per year between 2018 and 2021 increasing by approximately 2.4 times; both the number and sophistication of these attacks are increasing [1]. Circa 2000, the purpose of attacks was self-expression. However, since 2010, the purpose has shifted to information theft and terrorism, which are clearly malicious in intent [1]. Additionally, attacks that are not only indiscriminate but those targeted against specific organizations have emerged; these attacks are more difficult to detect. Furthermore, organized and state-sponsored attackers have also emerged; organizations such as government offices, enterprises, and universities tend to suffer more damage from cyberattacks. The importance of countermeasures against cyberattacks is increasing year by year, as they not only cause economic damage but also lead to secondary damage such as damage to the reputation of an organization.

1.1.2 Security Operations of SOCs/CSIRTs

Under the aforementioned circumstances, each organization has established a Security Operation Center (SOC) or Computer Security Incident Response Team (CSIRT) to detect and respond to cyberattacks. Since it is sometimes difficult to incorporate such organizations

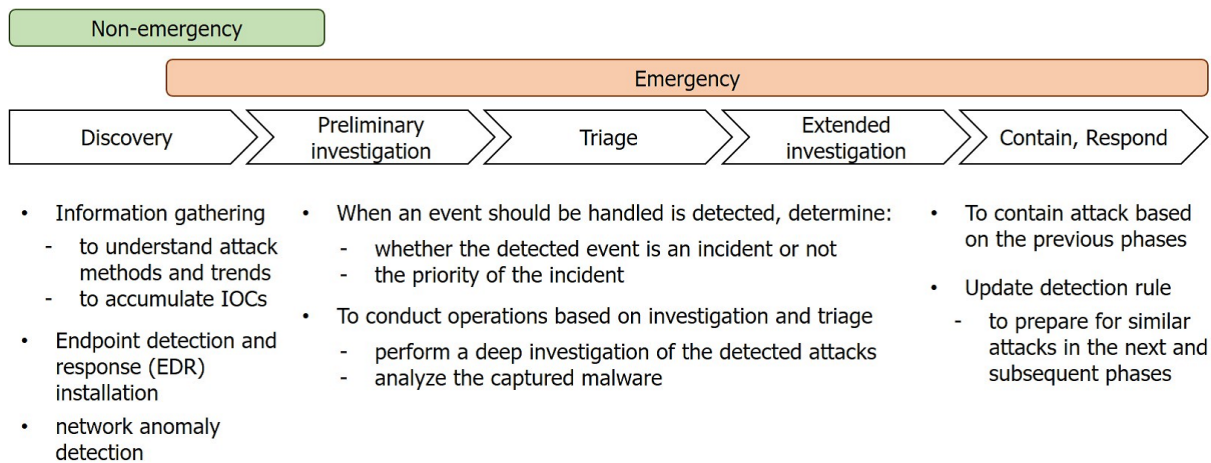


Figure 1.1 Overview of SOC/CSIRT operations.

into another organization, services that provide functions equivalent to those provided by SOC/CSIRTs have also emerged as businesses. The compound annual growth rate (CAGR) of cybersecurity businesses, including SOC/CSIRT services, is expected to be at a high standard of 12.5% from 2020 to 2028 [2], which shows the importance of cybersecurity measures.

The tasks performed by SOC/CSIRTs are called security operations; these operations are being systematized. For example, there are many activities available to develop a framework for a new SOC/CSIRT [3–6]. On the other hand, surveys show that there is still no de facto framework for SOC/CSIRTs [4], with many vendors and organizations working on their own definitions of such a framework [7, 8]. The tasks of SOC/CSIRTs can roughly be classified into the following five phases [7].

- (1) Discovery
- (2) Preliminary investigation
- (3) Triage
- (4) Extended investigation
- (5) Contain, Respond

Although there are differences in the definitions provided by organizations, vendors, and studies for SOC/CSIRT frameworks, all these definitions roughly follow the above sequence.

Therefore, we have used this corresponding definition in this dissertation. The overview of an SOC/CSIRT based on the above definition is shown in Fig. 1.1. SOC/CSIRT operations can be largely divided based on two categories: non-emergencies and emergencies. Regarding the above phases, stages 1–2 i.e., *Discovery–Preliminary investigation*, are generally considered non-emergencies, and stages 2–5 i.e., *Preliminary investigation–Contain, Respond*, are considered emergencies.

In non-emergencies, information is gathered in the *Discovery* phase. This phase involves understanding attack methods and trends, and accumulating indicators of compromise (IOCs) for detecting attacks by using cyber threat intelligence (CTI). When responding to emergencies, detecting events (e.g., possible information leakage) that need to be handled should be prioritized. For this purpose, the endpoint detection and response (EDR) solution is installed on each terminal, and anomaly detection is performed on network devices.

Next, there are a series of phases related to investigation and triage: *Preliminary investigation, Triage*, and *Extended investigation*. In this series of phases, when an event that should be handled is detected, the event is determined to be either an incident or not. Then, if it is an incident, the priority for handling this event is assessed. Based on this judgment, SOC/CSIRT analysts conduct a deep investigation of the detected attacks and analyze the captured malware.

Finally, in the *Contain, Respond* phase, the attack is contained based on the results under the previous phases. After the response, the detection rules are updated to prepare for similar attacks in the future.

SOC/CSIRT analysts respond to cyberattacks inside and outside the organization through the above operations.

1.2 Problems and Objectives

A shortage of human resources has been reported in SOC/CSIRT operations [9]; thus, an improvement in the efficiency of these operations is needed. The issue of variability in the abilities and maturity levels of operators has also been identified [10]; thus, SOC/CSIRT operations need to be systematized and automated to support security operations. The shortage of human resources and the importance of automation can be inferred from the high CAGR of 37.0% forecasted for the security orchestration, automation, and response

(SOAR) business in Japan from 2020 to 2024 [11].

Among cyberattacks, those related to networks tend to cause extensive damage; thus, it is particularly important to address these issues. For example, the intruder of an organization's network can receive access to various devices and information. Regarding this, cyberattacks have been reported to be approximately four times more likely to occur from the outside via networks than from the inside [12]. Additionally, there are cases wherein confidential information within an organization is leaked through a network.

Therefore, this study aims to improve the effectiveness and efficiency of SOC/CSIRT operations, especially those related to network security.

1.3 Research Problems

This chapter enumerates the research problems that should be addressed. As mentioned above, security operations can be roughly divided into two phases: non-emergencies and emergencies; each phase is described below.

1.3.1 Operations During a Non-Emergency

Problem 1: High Cost of Manually Utilizing Cyber Threat Intelligence

Information gathering by CTI is a normal-time security operation. CTI includes attack trends and detection indicators called IOCs (e.g., malware hash values, malicious uniform resource locators (URLs), etc.). Utilizing such information is expected to improve the detection rate of security events and facilitate responses in case of emergencies. On the other hand, CTI is often published in an unstructured form, making it difficult to machine-process, creating the need for manual verification. However, since CTI is so vast, it is not realistic to manually check all the information.

1.3.2 Operations During an Emergency

Problem 2: Difficulty in Tracing the Diffusion of Classified Information and Detecting Information Leakage

Before conducting an operation, it is necessary to first detect the type of emergency. Regarding the various emergencies, the leakage of classified information is an event that has

a high detection-priority due to its large impact. Additionally, it is not easy to determine whether information sent to the outside has been classified or not, because modern applications frequently exchange information with the outside world.

Problem 3: Lack of Clarity on Actual Support Mechanism for Dynamic Analysis of Malware

When cyberattacks involving malware are detected, malware analysis is sometimes performed for a more detailed analysis and to prevent future incidents. Particularly, dynamic analysis, wherein the behavior of malware is clarified through execution, is frequently used due to its scalability. Various analysis support mechanisms have been implemented in sandboxes that perform dynamic analysis, including a mapping function to the ATT&CK technique. The ATT&CK technique is a knowledge base published by MITRE that summarizes malware attack techniques. By mapping the results of dynamic analysis to the ATT&CK technique, the functionality of the malware can be understood more quickly. On the other hand, there are many ATT&CK techniques that do not provide specific detection rules or detection thresholds. Thus, the mapping function for these ATT&CK techniques in online sandboxes is implementation-dependent. Therefore, it is important to understand the actual mapping function of ATT&CK techniques in various sandboxes to perform security operations. This can be difficult because the mapping function in each sandbox is a black box, and the actual status of each function is unclear.

Problem 4: High Cost of Creating Signatures to Detect Malicious Communications

After completing the detection and response to a cyberattack, a signature is sometimes created to detect the attack and thereby improve the detection of future attacks. In cyberattacks, the attacker's server plays an important role in sending attack orders and receiving stolen information. Therefore, it is important to create signatures that can block the communication to such suspicious servers. Signatures generated to block such malicious communications should not block benign communications during normal business operations. Therefore, the generation of signatures requires a high degree of business understanding and is highly personalized. Additionally, the generated signatures need to be tested to ensure that they do not interfere with benign communications; this testing can be expensive in terms of operational cost.

1.4 Research Strategies

This chapter presents guidelines and methods for solving the research problems described in Chapter 1.3. In this research, each method was combined to enhance the effectiveness and efficiency of the security operations described above. The relationship between each method and the SOC/CSIRT operations described in Section 1.1.2 is shown in Fig. 1.2.

1.4.1 Structuring CTI Using a Common Format

Problem 1 is caused by most CTI being unstructured, and the cost of manually analyzing it is high. Therefore, this dissertation proposes CyNER, a method for structuring unstructured CTI.

The proposed method utilizes information-extraction techniques in natural language processing, such as NER and relation extraction (RE), to structure CTI. This research uses STIX 2.1 as a common format for structured CTI. STIX 2.1 consists of domain word objects i.e., STIX Domain Objects (SDOs) and the relationships among the objects i.e., STIX Relationship Objects (SROs), in the context of cybersecurity. Therefore, SDOs and SROs are extracted from CTI written in natural language using NER and RE, respectively, and converted to STIX. This enables the machine-learning processing of unstructured CTI, which was previously difficult to process this way, and it is expected to reduce the cost of utilization. Additionally, the use of CTI is also expected to improve the efficiency of the triage system in emergency situations. Specifically, by comparing the hash values and URLs of malware related to the detected event with those collected in advance by CTI, the threat level and priority level of countermeasures can be determined.

This dissertation presents the design, implementation, and evaluation results of this method.

1.4.2 Tracing the Diffusion of Classified Information and Detecting Information Leakage

To address Problem 2, this dissertation proposes a method to trace the diffusion of classified information within a computer and detect the transfer of classified information outside it.

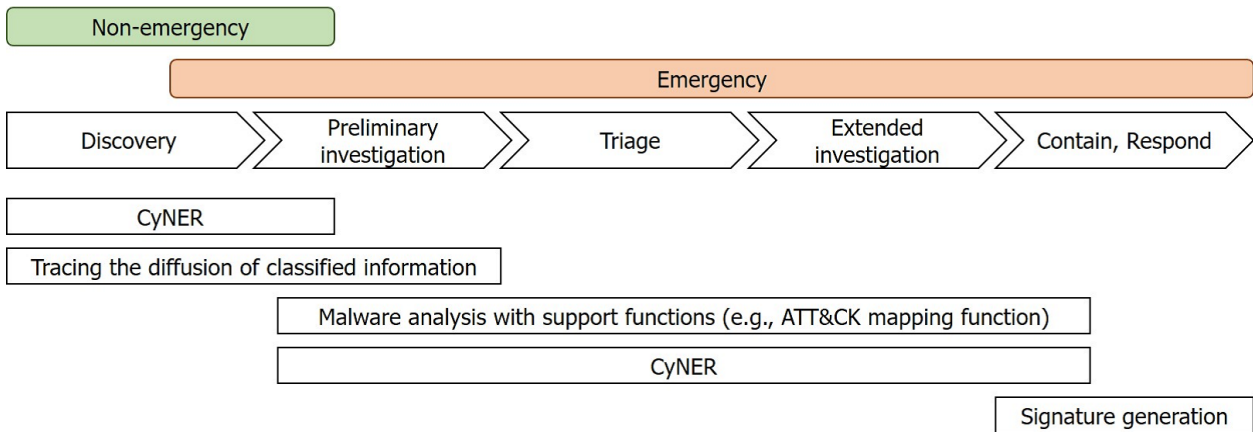


Figure 1.2 Relationship between each method and SOC/CSIRT operations.

To track the diffusion of classified information, system calls issued by the OS are hooked to detect and record the transfer and duplication of the classified information. When classified information is sent outside the computer, the system detects and considers the process related to the system call as a possible leakage of the classified information. Furthermore, by implementing this mechanism on the VMM instead of the OS, centralized management is enabled at the VMM layer, and the resistance to attacks is improved.

This dissertation reports the design and implementation of this method, the results of feasibility verification, and the results of performance evaluation.

1.4.3 Survey of Support Mechanisms in Online Sandboxes and Derivation of Best Practices

As described in Problem 3, the mapping of malware behavior analyzed by online sandboxes to ATT&CK techniques is useful, but the actual state of such sandboxes is unclear, making utilizing them difficult. Therefore, we systematically investigated the mapping characteristics of each online sandbox to clarify the actual situation.

We collected analysis reports of malware from each online sandbox and assessed them in terms of whether there were differences in the mapping functions among these sandboxes; whether there were techniques difficult to extract in the online sandboxes; and whether there were techniques easily mapped, even to benign files. Mapping to ATT&CK techniques is implemented not only in dynamic analysis in online sandboxes, but also in static analysis

tools and other situations such as those wherein reports are manually written. This dissertation quantifies and deepens our current understanding of the difference between the mapping of ATT&CK techniques in online sandboxes and that in other methods. Based on the survey results, this dissertation also provides the most excellent practices for practically implementing the mapping function in security operations.

1.4.4 Automatic URL Signature Construction and Impact Assessment

Problem 4 is caused by two reasons, namely the high cost of creating signatures to detect malicious communications and the high cost of assessing the impact of signatures on normal communications. To address the first and second problems, respectively, a signature was automatically generated to block malicious communications without disturbing benign communications, and the impact of the signature was automatically assessed.

The proposed method first generates candidate signatures based on the results of malware analysis. Then, clustering is performed based on the similarity of the malware destinations, and the common components among the destinations in a cluster are combined into a regular expression to be a signature candidate. In this process, multiple candidates are created by changing the parameters used for clustering. For each signature candidate, the blocking rate of malicious communications and the non-blocking rate of normal communications are calculated. After that, the signature intercepted most by malicious communications and intercepted least by normal communications is selected as the final signature to be applied. The proposed method creates signatures with a minimal impact on normal communications and automates the assessment of this impact. This method is expected to reduce the existing dependence on human resources for signature generation, reduce the cost of impact assessment, and assist in determining the applicability of signatures.

This dissertation reports on the design, implementation, and evaluation results of this method.

1.5 Related Work

1.5.1 Utilization of Cyber Threat Intelligence

Prior attempts have been made to structure unstructured data by creating dictionaries and ontologies [13, 14]. Cyber ontologies and their extensions for malware have been discussed [14], and the importance of developing a multi-layered CTI ontology has been argued [13]. However, in the security field, the continuous maintenance of dictionaries and ontologies is not easy because new words are often created due to the emergence of new malware, the discovery of vulnerabilities, and the assignment of code names. Also, it has been pointed out that existing ontologies lack expressiveness and coverage due to a lack of development [13].

To mitigate these issues, some studies have attempted to structure unstructured data by machine learning-based or probabilistic method-based natural language processing, similar to the proposed method [15–19]. Particularly, iACE [17] attempts to extract not only named entities but also contextual information related to IOCs by graph mining.

Many studies have also focused on the use of CTI. FeatureSmith [20] generates features by text mining CTI and automatically builds a model to detect Android malware. [21] is another method for automatically constructing threat detection rules from CTI. TTPDrill [22] conducts text mining for CTI and assigns the descriptions to Tactics, Techniques, and Procedures (TTPs) and cyber kill chains; and ChainSmith [23] estimates the roles of IOCs extracted from CTI. POIROT [24] performs threat hunting by graphing and comparing audit logs and CTI, respectively; Extractor [25] automates the graphing of CTI. However, the objectives of these studies did not involve structuring CTI.

There are also several studies that have attempted to perform a crossover analysis of CTI. IP addresses and domains in multiple blocklists have been investigated; many IOCs were found unique to a single list [26]. Similarly, multiple blocklists have been investigated [27].

As described above, many research studies have aimed to structure the unstructured CTI, with most of them focusing on term-extraction and few attempting to extract the relationships between words. In CTI, the relationships between terms, such as the relationship between threat actors and the vulnerabilities to be exploited and the relationship between malware and the malicious URLs to be accessed, are important and should be extracted.

There is also a problem in that CTI does not have contextual information because the distance between the subject and the object of the relationship is large; the details of the

malware are explained in the first half, and the URL that the malware accesses and the hash value of the malware are enumerated in the second half. However, the relationships with distant and non-contextual IOCs have not been extracted in existing studies.

1.5.2 Tracing Diffusion of Classified Information and Detecting Information Leakage

Many methods for tracing the diffusion of information within an OS have been proposed.

TaintDroid [28] is a method that traces the diffusion of classified information using a dynamic taint analysis (DTA). A DTA tracks information that has been tainted by other data. If the tainted data are written to another memory location, the destination is marked as tainted. Thus, we can follow the path of the classified information through a DTA. TaintEraser [29], which was implemented for smartphones, follows a similar procedure. It traces the diffusion of sensitive information within such devices and sends a notification to the user when an external leakage of information is detected. Taint-exchange [30] is a method for cross-host taint tracking. This method is applied by injecting tainted information into a data transfer. Argos [31] is a honeypot utilizing a DTA, and it is implemented on Quick EMUlator (QEMU) [32]. It marks the data received from the network and tracks the tainted data. Subsequently, when the data received from a network are utilized, Argos detects these data as an attack and obtains the information related to the attack (e.g., a memory dump). A taint-based protection system [33] implemented on Xen [34] has been described. This method tracks tainted data received from a network and prevents their utilization, preventing attacks based on malicious code injection. To mark the data using a taint-tag, a DTA requires additional storage called shadow memory. Therefore, additional non-trivial memory and disk space are required.

There are also other methods for dynamically tracing an information flow. CopperDroid [35] operates Android malware on QEMU and analyzes the behavior of this malware by hooking system calls. Android-specific IPC using Binder is analyzed by hooking `ioctl()`, which sends Binder the data. Aquifer [36] prevents unintended information leakage by limiting the applications that can handle sensitive data using a policy that restricts host exportation. AppIntent [37] detects the transmission of sensitive data using an Android application and notifies the user of this transmission. Subsequently, during an unintended

user operation, the application that executes the operation is judged to be malicious. The purpose of these methods is to analyze malware or prevent information leakage.

DroidSafe [38] provides a framework for analyzing static information-flows that may include sensitive data. This analysis can verify whether an Android application has the potential to leak sensitive data. DroidSafe detects the possibility of such a leakage statically. DroidSafe is specialized for Android because it detects the possibility of leakage from intents, which are Android-specific features.

TightLip [39] is a privacy management system that swaps an original process for a dummy process called a doppelganger, when a process that includes sensitive data attempts to write the data to a network. This protects the sensitive data from leakage because doppelgangers do not contain sensitive data themselves. Filesafe [40] protects sensitive files on a guest OS using a VMM. The user sets the security policy (read-only, not-accessible, etc) for the sensitive files beforehand. By enforcing the security policies using VMM, Filesafe can prevent sensitive files from unauthorized access. The secure virtual file system (SVFS) [41] operates a normal virtual machine (VM) running standard applications, an admin VM for system administration, and a Dalvik virtual machine (DVM) to store sensitive files for other VMs. These sensitive files can only be edited by the admin VM. Thus, these files can be protected even if the normal VM is compromised by an attacker. Additionally, the virtual organization file system (VOFS) [42] only permits the user to view sensitive files using the SVFS. The method introduced in [43] reduces the root privilege and prevents the modification of files that exceed the specified authorization, thereby protecting important files. This method, along with TightLip, Aquifer, and SVFS, are necessary for modifying the structure of an OS, showing the limited operational environment.

Cashtags [44] prevents information leakage that may occur through shoulder surfing in public places. To prevent such an information leakage, the Cashtags system replaces sensitive data elements with non-sensitive data elements before they are displayed on the screen. I-BOX [45] prevents information leakage using untrusted input method editor (IME) apps and intercepts and analyzes the user's input data. When sensitive data are included in these input data, I-BOX rolls back the execution of the state of an IME app, preventing an information leakage from an untrusted IME app. DroidTrack [46] traces the diffusion of classified information by hooking the information-gathering application programming interface (API). When DroidTrack detects the possibility of information leakage, it notifies the user. Finally,

the user can disallow the operation; the operation will be terminated as an error, thereby preventing information leakage.

When sensitive data are leaked, the assurance of log integrity is important for analyzing the cause. The method proposed in [47] gathers the logs by using the Linux Security Module (LSM). Additionally, the log integrity is guaranteed by using mandatory access control. The methods proposed in [48] and [49] gather the logging information generated by the OS and the access points (APs) working on the target VM by the VMM. To gather this logging information, the VMM hooks the system call that was invoked for sending logs from the user process to the syslog daemon; this method makes it difficult to tamper with a log by isolating it from the VM. NIGELOG [50] provides multiple backups of the log files. By using these backups, the log files can be restored even if the original file is altered or deleted by intruders. The log data integrity support scheme (LISS) [51] backs up the log files by using a mirroring technique. Then, it verifies the integrity of the log files by comparing the hash value between the original and backup files. Similar to [48] and [49], the VMM-based tracing function ensures the reliability of the monitoring information by sheltering the tracing function from the target OS. Like the methods proposed in [48] and [49], the VMM-based tracing function ensures the reliability of the monitoring information by sheltering the tracing function from the target OS.

In several existing studies, classified-information tracing and leakage detection functions have been realized as in-box applications within the OS to be protected. However, if the OS is compromised by an attacker, applications in the same OS may be disabled. FileSafe is more resilient to attacks because it is an out-of-the-box approach that tracks sensitive information from the VMM. However, FileSafe necessitates the setting of a policy for each file individually, which may cause a leakage of classified information in case of policy misconfigurations.

1.5.3 Support Functions in Dynamic Analysis of Malware in Sandboxes

Various online sandboxes have implemented functions for mapping malware to techniques; some studies have attempted to analyze these techniques. Hierarchical clustering has been used to derive correlations between advanced persistent threats (APTs) and software reported in ATT&CK [52]. A method and tool have been proposed to analyze the correla-

tion between MITRE ATT&CK, Common Attack Pattern Enumeration and Classification (CAPEC), Common Weakness Enumeration (CWE), and Common Vulnerabilities and Exposures (CVE) [53]. This study revealed that these methods could be used more effectively by improving the true positives of the techniques that were the inputs to each method.

While the present study focuses on a technique related to the functions of online sandboxes, other studies have been conducted from other perspectives. For example, the developers of SandPrint [54] investigated and demonstrated whether various online sandboxes could be detected by fingerprint technology; other studies have also investigated and verified whether online sandboxes can be detected [55, 56].

As mentioned above, there are several studies on the ATT&CK technique and studies that have investigated online sandboxes. However, the actual state of the mapping function of the ATT&CK technique in online sandboxes has not been investigated. Therefore, while the usefulness of this function is known intuitively, its actual behavior and best practices are not clear, hindering its practical applicability.

1.5.4 Detection of Malicious Communications by Creation of Signatures

Many methods have used malicious logs such as malware analysis results to create rules for intrusion detection systems and signatures to block malicious communications.

Kitsune [57] is a network intrusion detection system (NIDS) with autoencoders that performs intrusion detection using the information contained in packet capture (PCAP) files as features. An automatic signature generation method focusing on Hypertext Transfer Protocol (HTTP) communication, which creates generic signatures by combining multiple clustering methods, was developed [58]. A clustering-based signature creation method was also developed [59]. MalGene [60] extracted the similarities from the system-call sequence of a malware dynamic analysis result and generated the corresponding signature.

Some existing research has also considered normal communications. A method to create signatures from the dynamic analysis logs of Android malware, which treats apps downloaded from Google Play as benign and does not affect benign apps in the evaluation, was proposed [61]. A method called EIGER [62] creates signatures based on the dynamic analysis logs of malware and is configured to have no effect on the behavior logs of public Windows

applications.

Several existing methods do not consider the impact of the created signatures on normal operations. The results of malware analysis may include benign communication; if the signature is created without considering normal communication, normal communication may be blocked, and business may be affected. Several other methods consider the influence of normal communication, which is limited to general applications. In a business environment, the use of original applications and communication to the intranet is common; therefore, a signature that does not affect public applications may still affect business communication. In practice, when applying signatures, it is necessary to test whether they affect the business and evaluate whether they should be applied based on the test results. The methods proposed in the above-mentioned studies did not consider that. Thus, ultimately, the testing and application decisions depend on human resources.

1.6 Outline of the Dissertation

This dissertation is organized as follows:

Chapter 2 proposes a method for structuring unstructured CTI and a method for threat analysis that utilizes structured CTI.

Chapter 3 proposes a method to trace the diffusion of classified information on a guest OS using a VMM and detect information leakage outside the guest OS.

Chapter 4 describes the results of the investigation of the mapping function of the ATT&CK technique, which is one of the analytical support functions in the online sandbox.

Chapter 5 proposes a method for automatically creating signatures to detect communications to suspicious destinations using the results of malware analysis.

Chapter 6 presents the conclusions of the research and the future directions.

Chapter 2

Information Extraction from Unstructured Text of CTI Sources with Noncontextual IOCs

2.1 Introduction

With the increase in frequency and sophistication of cyber attacks, it is important to be up-to-date with threat information using cyber threat intelligence (CTI). For example, CTI contains information on new vulnerabilities, malware, attackers' methods, and countermeasures against them. Indicator Of Compromise (IOC) is often included as an indicator for detecting attacks, and consists of, for example, IP addresses, URLs of suspicious sites, and hash values of malware. By utilizing this information for the detection rules of firewalls and intrusion detection systems, attacks can be detected in advance. Thus, by appropriately extracting and utilizing the information of the malware, vulnerability, and IOCs contained in CTI, it is possible to construct detection rules and analyze attack trends.

CTI is often first distributed as unstructured data in media such as blogs, news sites, and social networking sites. There is a time lag between the release of such information and its structuring—sometimes up to a month or more [63]. Therefore, in order to keep up with the latest threat information, it is necessary to analyze and utilize unstructured data. However, more than 60,000 CTIs are published every month [64], and it is not realistic to analyze all of them manually. In addition, since many CTIs are written in natural language, it is difficult

to simply implement machine processing on them. In such circumstances, it is important to structure a CTI written in natural language into a form that can be processed by machines, and to support efficient analysis.

To resolve these challenges, some studies [13,14] have tried to analyze unstructured CTIs by constructing dictionaries or ontologies. However, in the security field, new words tend to be generated because of new malware or vulnerabilities, so their continuous maintenance is not easy. In addition, since IOCs such as URLs and IP addresses have a fixed format, they can be extracted by using regular expressions, but they often lack contextual information such as what kind of malware or attacker they are being used by. It is difficult to use such noncontextual information for analysis and to judge whether it is applicable as a detection rule or not. Therefore, it is important to add contextual information such as malware name and attacker name to the IOC.

Other research has examined machine learning, probabilistic method, and graph mining to perform robust information extraction for unknown words and to provide meaning to IOCs. For example, [65] matches Common Vulnerabilities and Exposures (CVE) summaries with Common Platform Enumeration (CPE) through machine learning-based Named Entity Recognition (NER) with high accuracy. In addition, iACE [17] attempts to extract contextual information about IOCs using graph mining. These studies mitigate the aforementioned difficulties with maintaining dictionaries and ontologies and the lack of contextual information. On the other hand, these methods, including the above-mentioned research, assumed that words related to IOCs appear in the neighborhood of IOCs. As for the semantics of IOCs, there are many cases in which IOCs are listed at the bottom of a CTI after the main topic is described. In such cases, the context of the IOCs is missing, and the existing methods, which assume that IOCs and their related words appear in the same neighborhood, cannot give proper context to the IOCs.

To solve these problems, we propose CyNER, a method for structuring CTIs using Natural Language Processing (NLP) techniques such as NER and Relation Extraction (RE). The proposed method aims to improve the efficiency of analysis by extracting named entities that should be focused on in the context of cybersecurity, such as malware names, vulnerability names, and IOCs. CyNER also aims to structure CTIs in a way that maintains contextual information by extracting relations between named entities. In addition, by estimating the topics mentioned in a CTI through key phrase extraction and associating them with IOCs, we

can give contextual information such as relevant malware names and vulnerability names to IOCs lacking context, which is not possible with the existing methods. This makes it possible to link noncontextual IOCs to relevant malware or vulnerability names, which is difficult to extract with existing methods [17,23] that assume that technical terms associated with IOCs will appear in the same sentence. Moreover, we aim to improve the usability of the data by structuring them in a general-purpose format of Structured Threat Information eXpression (STIX) [66]. This makes it possible, for example, to conduct crossover analysis for threat information that is dispersed across different information sources. Although there are studies that cross-analyze already structured information such as blocklists and threat feeds [26,27], it is difficult for existing studies to cross-analyze unstructured threat information due to the existence of IOCs separated from the main texts as mentioned above.

The contributions of this study are as follows:

- By extracting named entities and relations between named entities from unstructured CTIs, CyNER automatically structures them in the STIX 2.1 format. In addition, by extracting key phrases from CTI and associating them with noncontextual IOCs, we can extract relations that have no relation in the neighborhood, which is not done in existing RE methods.
- The evaluation with our dataset showed that both the named entities and noncontextual IOCs could be extracted. We found that the F-measure for NER can be improved by up to 2.4 points by using a language model trained on a domain corpus for structuring CTI, compared to using a general-purpose language model. In addition, we were able to link entities to noncontextual IOCs with an accuracy of up to 81.65%.
- Using CyNER, we structured 55,266 CTIs from 35 sources, extracted 297,101 IOCs, and conducted a crossover analysis. In this analysis, the following facts were revealed and the possibility of using CyNER was demonstrated.
 - We compared the coverage of the IOCs extracted by CyNER with that of existing reputation services, and showed that CyNER can extract IOCs that are not included in the existing services.
 - We found that 19,274 IOCs were reported continuously, and some were exploited by multiple attack groups for more than a year.

2.2 Background and Research Questions

2.2.1 Cyber Threat Intelligence

As mentioned earlier, threat information, called CTI, and especially structured CTI, has an important role to play in conducting security operations. In this context, various structured formats for cyber security have been developed for the purpose of machine-readable security information and information sharing in a common format. There is OpenIOC [67], which specializes in IOCs, and STIX [66] and MISP [68], which cover a wider range of information. For example, STIX consists of two parts: SDO (STIX Domain Object), which is an object of domain terms in the context of cyber security, and SRO (STIX Relationship Object), which is a relationship between SDOs.

The infrastructure for sharing such information is also being developed. Facebook ThreatExchange [69], the Defense Industrial Base Cybersecurity Information Sharing Program [70], and Automated Indicator Sharing [71] are frameworks for sharing reliable information among member organizations. There are also public frameworks for sharing IOCs, such as AlienVault OTX [72], OpenCTI [73], and MISP. However, a structured CTI for sharing in these frameworks needs to be created separately.

2.2.2 NLP

Information extraction is an NLP task in which structured data are extracted from unstructured documents. This task consists of various technologies such as NER, which extracts named entities from sentences, and RE, which extracts relations between named entities.

In recent years, high accuracy in information extraction has been achieved by using language models such as Word2Vec [74] to convert words or sentences into numerical expressions called distributed representations, which are then used as input for various tasks. In particular, Bidirectional Encoder Representations from Transformers (BERT) [75] and applied language models based on BERT have achieved high performance in a variety of tasks. There are also a number of later improved models, such as RoBERTa [76] for higher accuracy and ALBERT [77] for lighter weight.

2.2.3 Challenges

As mentioned above, structured CTI is useful and the infrastructure for sharing it is being developed. On the other hand, since most CTI is written in natural language, unstructured CTI needs to be structured. In this case, it is desirable to structure the CTI in a common format (as discussed in Section 2.2.1) so that we can utilize the various functions that have been developed, such as visualization and linkage with security appliances. Therefore, the goal of this research is to automatically convert CTI into a common format. In order to achieve this, the following issues need to be addressed.

Challenge 1: The complexity of terms. As mentioned above, new terms are developed every day, so extracting terms using a dictionary is not easy because it requires continuous maintenance of the dictionary. In addition, there are multiple terms that have the same meaning (e.g., “C&C” and “C2”, “APT10” and “menuPass”). In addition, some unique expressions overlap with common words (e.g., meltdown).

Challenge 2: Extraction of distant relationships. In natural language of the general domain, named entities with relations often co-occur in the same or neighboring sentences, and existing RE methods often use the same sentence or a few neighboring sentences as the search range of relations [78–81]. This is not true, however, for CTI. For example, a specific malware threat is described in detail in the text, and the IOCs related to the malware are listed at the bottom of the CTI. In this case, the IOCs at the bottom of the CTI should be associated with name of the malware as a named entity, but since the IOCs are located far from the name of the malware, it is difficult to extract this relation using existing RE methods. In fact, when we examined the 270,047 IOCs we collected, more than half of them (144,430) were separated from the main texts (e.g., bullet points). Thus, it is necessary to implement a method for extracting such distant relations and restore the context of “noncontextual” IOCs.

2.3 Design and Implementation

2.3.1 Basic Idea and Overview

As discussed, while CTI structured according to a common format has various advantages, the construction cost is high and it is not practical to manually structure all unstructured

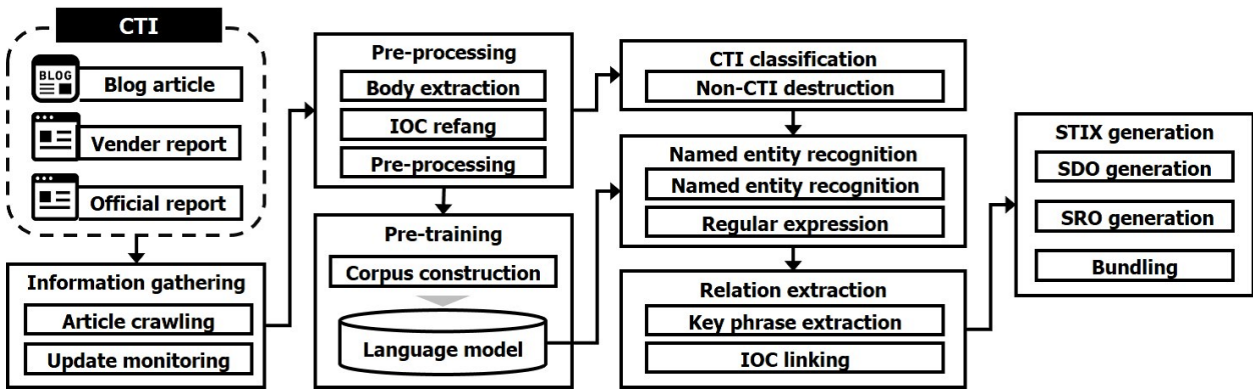


Figure 2.1 Overview of CyNER.

CTIs. Therefore, our proposed method aims to automatically structure CTIs in a common format to support efficient analysis. To accomplish this, we need to solve the two challenges mentioned in Section 2.2.3.

First, to solve *Challenge 1*, we use BERT and related methods for NER. BERT is a machine learning-based method that can extract a greater number of new words compared to dictionary-based methods. In addition, unlike previous word representation methods such as Word2Vec, BERT can construct word representations that take into account the context. This increases the likelihood of recognizing words with equivalent meaning even if they are different. In addition, BERT learns embedded representations on a sub-word basis, not on a word basis. This is expected to increase the likelihood of recognizing unique expressions by subword, even if the word is new.

In order to solve *Challenge 2*, we assume that the noncontextual IOCs are related to the words that represent the CTI in question, and extract the distant relationships related to the IOCs by using key phrase extraction. Specifically, key phrases are extracted from the CTI, and those that match the named entities extracted by the NER are judged to be the words that represent the IOC in question, and the relationship is established. In this way, we should be able to extract relations even when there is no word representing the IOC in the neighborhood.

Figure 5.2 shows the overview of the proposed method. First, articles are collected from sites that publish CTI, and then text in the collected CTIs are preprocessed for the later stage of processing. Then, only CTIs are extracted by classifying the articles, and non CTIs are rejected. After that, information that should be described as STIX is extracted by NER

and RE. Finally, the extracted information is formatted as STIX.

In the following sections, we describe the details of the processing for each step.

2.3.2 Information Gathering

First, CyNER gathers candidates of CTI from various websites such as blogs and official reports. CyNER initially crawls all CTI-related webpages and gathers all articles. In this dissertation, we chose several major websites and implemented a crawler and parser tailored for each. In addition, to prevent duplication of articles and overload of target websites, CyNER gathers only updated articles. To do so, updated articles are gathered by using RSS feeds in cases where target websites provide RSS. Otherwise, CyNER parses CTI-providing pages and verifies whether the articles are new or not. After that, CyNER gathers only new articles.

2.3.3 Preprocessing

In this step, CyNER carries out preprocessing for text in the gathered CTIs. Web articles often include non-CTI information such as html tags, advertisements, and navigation bars, so CyNER extracts body texts for deleting any unnecessary information.

Next, CyNER performs refang on the IOCs. *Refang* means to returning defanged IOCs to their original form, e.g., converting “example[.]com” into “example.com”. In doing so, IOCs can be extracted by regular expression. The refang mechanism is implemented by defining refanging rules (such as replacing “[.]” with “.”) in advance and then using rule-based search and replace. One of the refang rules is removing brackets. All refang rules are shown in the Appendix A.2.

After this preprocessing, the collected information is processed to make it suitable for the later stage of NLP. Specifically, the extracted text is divided into sentences so that it can be processed by the language model.

2.3.4 Pretraining

As mentioned above, BERT and other pre-trained language models are widely available and can be used for NER. On the other hand, it is known that pre-training on a domain-

Table 2.1 Named entity list.

STIX object	Extracted items	Description	Examples	Extraction method
Attack pattern	name	Attack pattern name	Spear Phishing	NER
Campaign	name	Campaign name	Operation Aurora	NER
Threat actor	name	Threat actor name	APT10	NER
Identity	name	Name	Hitachi, Ltd.	NER
Indicator	pattern	IOC	URL, hash, etc.	Regular expression
	labels	Malware type	Ransomware	NER
Malware	name	Malware name	WannaCry	NER
Tool	name	Tool name	Metasploit	NER
Vulnerability	name	Vulnerability name	CVE-2014-0160	Regular expression
			HeartBleed	NER

specific corpus in a specialized field improves the accuracy of various tasks based on the model in question [82]. Therefore, we aim to improve the accuracy of NER by constructing our own pre-training model using the domain corpus of the cyber security field.

In order to build a pre-training model for the cyber security domain, we first crawl web pages that publish CTI and collect them as candidates for the domain corpus to be used as training data for building the language model. Next, we remove unnecessary information from the collected CTI to extract sentences for training. Specifically, in order to extract the main text, we remove unnecessary information such as HTML tags and JavaScript. In addition, even in the body part, there are still some sentences such as headings and bullets that are not necessary for learning. Therefore, referring to the literature [83], we remove the unnecessary information by the following process to make a domain corpus.

- Pages with less than 5 sentences
- Lines with less than 3 words
- Lines that may be signatures such as *snort* (lines starting with "{" or "\$")

Finally, the domain corpus constructed so far is used for pre-training to build the language model. By using the above method, we aim to improve the accuracy of NER for structuring CTI.

Table 2.2 Relation rule list.

Subject	Object	Relation
Indicator_pattern	Attack_pattern_name	indicates
- Hash value	Campaign_name	
- File name	Malware_name	
	Threat_actor_name	
Malware_name	Indicator	communicates-with
	- URL	
	- IP address	

2.3.5 CTI Classification

Some of the blogs and official pages that provide CTIs include articles introducing products and seminars. Since these are not CTIs, we reject them by constructing a binary classifier to determine whether they are CTIs or not. Our binary classifier consists of the aforementioned pre-trained BERT and a fully-connected layer that outputs whether the input document is a CTI or not.

2.3.6 Named Entity Recognition

To modify the corpus that has been processed up to this point into a form suitable for NLP, NER is performed. We first define the items to be extracted as extended named entities (Table 2.1). As mentioned earlier, CyNER carries out structuring according to the STIX 2.1 format. Therefore, we define the named entities in a form corresponding to the objects (SDOs) in STIX. As already described, formatted named entities (e.g., IP addresses, URLs, and CVE numbers) are extracted by regular expressions. In addition, other named entities are extracted by the NER model, which is implemented by fine-tuning *huggingface* [84] pre-trained models (BERT, RoBERTa, and ALBERT) for NER, the same as CTI classification.

2.3.7 Relation Extraction

By extracting the relationships between the named entities extracted in the previous step, CyNER acquires the contextual information. The definition of the relationship between IOCs and named entities is provided in Table 2.2. This definition is aligned with the SRO of

STIX 2.1. CyNER firstly extracts relationships by existing method such as [17] for named entities located at same sentence. Here, on the basis of the policy described in Section 5.3.1, CyNER also attempts to extract the relationships between named entities and independent IOCs (Fig. 2.2). The specific process flow is as follows.

- (1) Among the IOCs extracted with regular expressions in the NER step, extract those independently listed, e.g., located alone at the bottom of a CTI, as candidates for RE.
- (2) Extract the top 10 key phrases that represent the CTI by using a key phrase extraction technique.
- (3) Among the extracted named entities, compare the named entities that can have a relationship with the IOCs with the key phrases, and associate the one that matches the top key phrase with the IOC as having the predefined relationship. The properties of the named entities that can have a relationship with the IOCs are `attack_pattern`, `campaign_name`, `malware_name`, and `threat_actor_name`. In other words, each IOC is assigned a relationship with up to four named entities.

With the above process, we can extract relationships between named entities and “non-contextual” IOCs, which are difficult to extract with existing methods.

As a key phrase extraction method, we used MultipartiteRank [85], which had the highest accuracy for our test data among the several methods we implemented and compared. The details are described in Section 5.4.

2.3.8 STIX Generation

STIX is generated using the named entities and their relationships extracted in the previous process. Specifically, first, the named entities extracted in the NER step are converted into the corresponding SDOs. Next, the relations extracted in the RE step are converted into SROs that define the relations among STIX objects. Finally, a STIX object is created for each CTI using a bundle object that groups STIX objects.

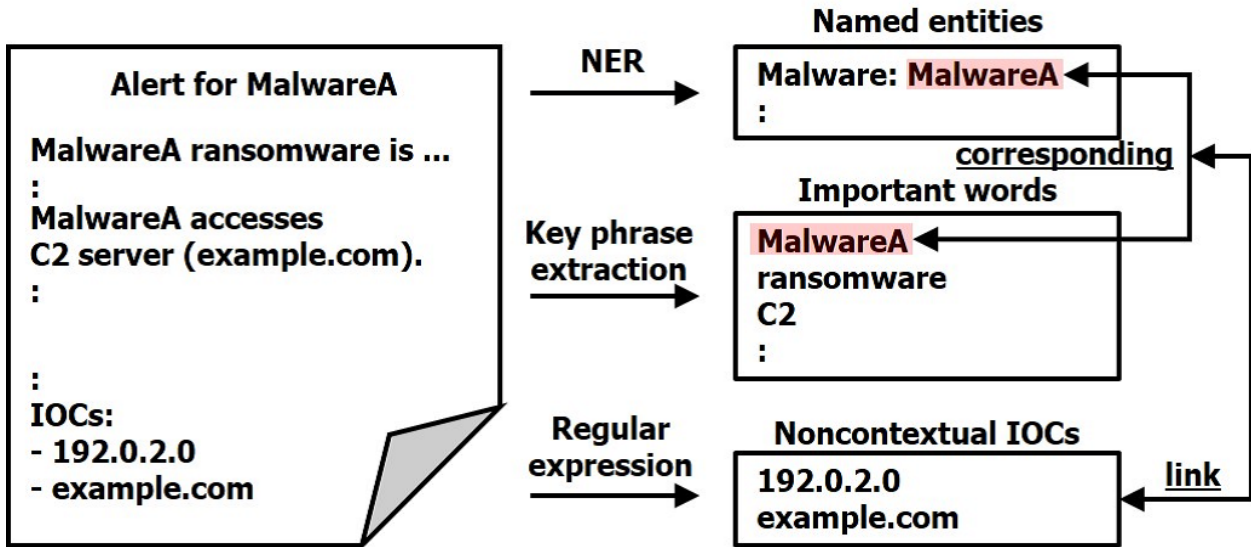


Figure 2.2 Extraction method for noncontextual IOCs of CTI.

2.4 Evaluation

2.4.1 Experimental Setup

We implemented a prototype of CyNER according to the design described above and conducted the following two evaluations.

1. **Named Entity Recognition Accuracy.** CyNER attempts to extract the named entities (shown in Table 1) using a fine-tuned language model for SDO extraction; therefore, the accuracy of the NER is evaluated in terms of the precision, recall, and F-measure. In doing so, we will also test whether the accuracy is improved by pre-training language models using domain corpora.
2. **Relation Extraction Accuracy.** CyNER extracts relations between named entities and noncontextual IOCs of CTIs by comparing the results of NER with those of key phrase extraction. We evaluate the correctness of this RE. We then implement several key phrase extraction methods and evaluate which one is most suitable for this task.
3. **Processing Time.** CyNER is intended for use in SOC/CSIRT daily analysis work. Therefore, we measure the processing time of each process and evaluate whether it is within the practical range in comparison with daily operations. The measurements were performed using the *time* module of Python under the GUI multi-user mode.

2.4.2 Dataset

To conduct each of the evaluations described in the previous section, we selected 35 sites that distribute CTIs on the basis of existing studies and interviews with practitioners (detailed in the Appendix A.1.) We implemented a crawler for each site and collected 78,868 CTI candidates published between June 2001 and September 2022, and then constructed the following datasets for evaluation. In all evaluations, data before 2019 were used for training and data after 2019 were used for testing. The labeling of the data was done independently by the author, who are experts in the field of cyber security. In addition, CTIs for labeling were randomly selected as described below, but those that contain few named entities were excluded.

- *Dataset for training language models.* The collected CTIs were subjected to the pre-processing described in Section 2.3.3 and then made into a domain corpus for language model training. The dataset consists of about 3,000,000 lines totaling about 320MB.
- *Dataset for NER.* We randomly picked up the collected CTIs and prepared 100 CTIs annotated with named entities. This dataset consists of 13,479 sentences, 193,027 words, and 4,562 named entities in total.
- *Dataset for extracting IOC relations.* We randomly picked up the collected CTIs and prepared 100 CTIs that include at least one IOC. This dataset contains 2,371 IOCs.
- *Dataset of structured CTI.* Among the 78,868 CTIs mentioned above, 55,266 CTIs are evaluated by the CTI classifier in this section. The number of unique IOCs associated with named entities by CyNER is 297,101, and it consists 55,884 hashes, 198,430 URLs, and 42,786 IP addresses. Note that URLs in the Alexa Top 10,000 and private IP addresses defined in RFCs are excluded because they are highly likely to be false positives.

The following evaluations were conducted using the above data-sets. All evaluations were performed on a commodity PC with an Intel Xeon E5-2698 v4 (2.2 GHz, 20 cores) running Ubuntu 18.04 OS with 256 GB RAM and 4 Tesla V100 (VRAM 128 GB).

Table 2.3 NER accuracy of each model.

Method	Model	Precision	Recall	F-measure
Dictionary [13, 14, 17, 22, 23]	-	0.74	0.56	0.65
CRF [18, 65]	-	0.68	0.68	0.68
BERT [25]	bert-base-uncased	0.81	0.70	0.75
	bert-large-uncased	0.78	0.73	0.75
	roberta-base	0.78	0.73	0.76
	roberta-large	0.85	0.74	0.78
	albert-base	0.84	0.73	0.77
	albert-large	0.84	0.70	0.77
CyNER (BERT fine-tuned by domain corpus)	bert-base-uncased	0.81	0.76	0.78
	bert-large-uncased	0.78	0.76	0.77
	roberta-base	0.81	0.79	0.80
	roberta-large	0.80	0.80	0.80
	albert-base	0.81	0.74	0.78
	albert-large	0.80	0.74	0.78

2.4.3 Result

Evaluation 1: Named Entity Recognition Accuracy.

In this evaluation, 70 articles (70%) were used for training and 30 for verification. The training data were further divided into a 70% training set and a 30% validation set for training. For the models, we used BERT and its later variants, RoBERT and ALBERT, which are representative of the pre-training models available in huggingface. For each model, we used large, which has more parameters and higher accuracy, and base, which is lighter. In addition, in order to compare machine learning-based models with dictionary-based methods, dictionary-based NER was used as a baseline. Specifically, we registered named entities in the training data into a dictionary and extracted named entities from the validation data using the dictionary. As another baseline, we used the NER model with CRF, which is a well-known conventional method. We trained each model for 200 epochs. The final accuracy of both models is shown in Table 2.3, which includes the values of the data for validation.

First, we can see that the accuracy of all machine learning-based models was higher than that of the baseline dictionary. In order to determine the effect of pre-training with the domain corpus, we evaluated the accuracy with and without the domain corpus for each model except the baseline. The results without the domain corpus are listed in the *BERT* row, and the results with the domain corpus are listed in the *CyNER* row. From the experimental results, we can see that the F-measure improved in all the models, with a maximum

Table 2.4 Recognition result of each named entity.

Item	Precision	Recall	F-measure	No. of named entities
attack_pattern_name	0.92	0.92	0.92	63
campaign_name	1.00	1.00	1.00	1
grouping_name	0.92	0.67	0.78	278
identity_name	0.86	0.82	0.84	327
malware_label	0.86	0.96	0.91	373
malware_name	0.76	0.52	0.62	174
tool_name	0.54	0.71	0.61	31
vulnerability_name	0.85	0.82	0.84	102
Mean/Total	0.80	0.79	0.80	1,349

improvement of about 2.4 points in *bert-base-uncase* (from 0.7523 to 0.7760). In addition, roberta-large had the highest accuracy among all models with an F-measure of 0.8012. This result confirms that pre-training with domain corpora can improve the accuracy of NER in the field of cyber security. Note that the following evaluations and analyses are conducted using CyNER with roberta-large.

Next, the accuracy for each named entity is shown in Table 2.4 and the confusion matrix is shown in Fig. 2.3. From Table 2.4, we can see that the extraction accuracy of *malware_name* and *tool_name* tended to be slightly lower than that of other named entities. This was probably due to the fact that the naming conventions for malware names and tool names are free, and thus there were many variations. In addition, a relatively high number of new words that did not exist in the training data were also included. In Fig. 2.3, we can see that many of these items were incorrectly classified as O (items that are not unique expressions).

Evaluation 2: Relation Extraction Accuracy.

In this evaluation, we selected PositionRank [86], TopicRank [87], and MultipartiteRank [85] as the key phrase extraction methods and measured the percentage of correct answers when performing the relation extraction described in Section 2.3.7 using each method. In this case, only nouns, proper nouns, and adjectives were used as candidates for key phrases. We then compared the list of extracted key phrases with the list of unique expressions, and extracted those that matched as related words. The percentage of correct answers in this evaluation is shown in Table 2.6. MultipartiteRank had the highest percentage of correct

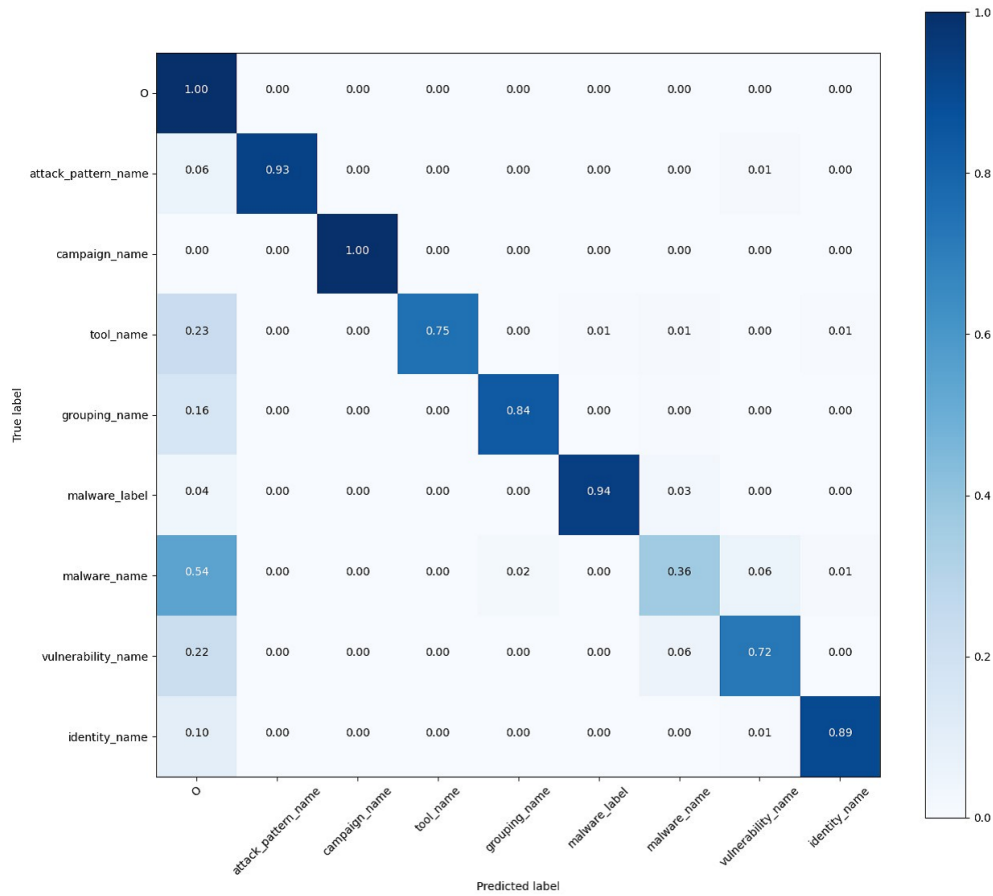


Figure 2.3 Confusion matrix of each named entity.

answers, at 81.65%. The accuracy of PositionRank was almost the same, at 77.22%. Both methods favored words close to the beginning of the sentence, which suggests that key phrases related to IOCs in CTI have a high co-occurrence with words close to the beginning of the sentence. These results demonstrate that BERT is suitable for CTI classification, RoBERTa for NER, and MultipartiteRank for relation extraction.

Evaluation 3: Processing Time.

Table 2.5 shows the time required from CTI preprocessing to STIX generation. The listed values are the average processing times for the 51,039 sentences out of 78,868 that were determined to be CTI using the classifier constructed in Evaluation 1. The average length of the sentences was 968 lines.

Table 2.5 Processing time of proposed method per CTI.

Process	Time (sec.)
Pre-processing	0.42
CTI Classification	1.12
Named Entity Recognition	0.87
Relation Extraction	1.92
STIX Generation	0.41
Total	4.74

As shown in the table, the total processing time was about 4.74 seconds per article. In order to handle 60,000 CTIs per month [64], it is necessary to handle about 2,000 CTIs per day. If each article takes about 4.74 seconds, the total processing time is $2,000 \times 4.74 = 9,480$ seconds (about 2.63 hours).

As a preliminary step to the above process, there is also the processing time for fine-tuning RoBERTa used for the CTI classification and NER. We measured the processing time for each of these two processes over 200 epochs and found that the CTI classification took about 2,100 seconds (roughly 10.5 seconds per epoch) and NER took about 3,400 seconds (roughly 17.4 seconds per epoch). Since these fine-tuning processes are expected to be performed in batches on a daily or weekly basis, we assume from these results that they can be used in actual work.

To summarize, fine-tuning on a daily or weekly basis took about 5,500 seconds (roughly 1.5 hours) in total, converting STIX took about 4.74 seconds per article on average, and it took about 2.63 hours for an expected daily volume of 2,000 articles, all of which is within the range of practical use.

2.5 Analysis

2.5.1 Overview

In this section, we use CyNER to structure and analyze CTI. In this way, we verify the possibility of using CyNER for CTI-based security operations. Specifically, we conducted the following three analyses.

Table 2.6 Accuracy of relation extraction.

Method	Accuracy (%)
PositionRank	77.22
TopicRank	69.59
MultipartiteRank	81.65

Table 2.7 IOC coverage of each platform.

IOC type	Method	Total	Existed (rate)	Did not exist (rate)
SHA256	CyNER	1,000	1,000 (100%)	0 (0.0%)
	VT	1,000	906 (90.6%)	94 (9.4%)
	OTX	1,000	25 (2.5%)	975 (97.5%)
IPv4	CyNER	1,000	1,000 (100%)	0 (0.0%)
	VT	1,000	998 (99.8%)	2 (0.2%)
	OTX	1,000	195 (19.5%)	805 (80.5%)

- 1. IOC coverage.** CyNER extracts named entities from unstructured CTI and associates them with IOCs. We compare and evaluate whether the coverage of IOCs and information associated with IOCs extracted by CyNER is as good as that of the de facto service. Specifically, we compare VirusTotal, a service for evaluating IOCs, and AlienVault OTX, a platform for sharing structured CTIs.
- 2. Time-series information.** By using the proposed method, we can handle the time-series information of CTI and IOCs from the past to the present in a unified manner. Therefore, we analyze CTI and IOC from the viewpoint of time series and examine the possibility of using them.
- 3. Information source relation.** CyNER can structure CTI data from multiple sources in a unified STIX format and handle them in a unified manner. We examine the relationship between the IOCs and the information sources, and examine the possibility of using them.

2.5.2 IOC Coverage

In this evaluation, we checked whether or not the IOCs extracted by CyNER were included in VirusTotal and OTX, and compared their coverage. We randomly selected IOCs that were associated with one or more malware. The properties of the IOCs are the hash value of the malware (SHA256) and the communication destination (IPv4 address). In addition, we selected 1,000 hash values and 1,000 communication destinations, and used them for comparison.

First, Table 2.7 shows the results of the coverage evaluation. In this evaluation, we compared the coverage of VirusTotal and OTX based on 1,000 SHA256 and 1,000 IPv4 addresses each which are associated with the malware families extracted by CyNER. The coverage of OTX was 2.5% for SHA256 and 19.5% for IPv4, which is relatively low, probably due to the fact that OTX relies heavily on manual and expert registration. In contrast, the coverage of VirusTotal was 90.6% for SHA256 and 99.8% for IPv4, which included most of the IOCs extracted by CyNER. Since the number of contributors to VirusTotal is larger than that of OTX, it is assumed that most of the IOCs listed in the public CTI, which is the source of information for CyNER, have already been submitted. However, among the IOCs extracted by CyNER, there were some that were not included in either service, so it is useful to be able to structure such IOCs automatically and with contextual information by linking them to malware families.

Next, we compared the results in terms of the amount of information. First, although OTX lacks some coverage (as described above), it can be tagged manually and often contains the same amount of information or more than CyNER. In addition, VirusTotal can scan the target with dozens of AV products and URL scanners. The results of these scans are shown in Fig. 2.4.

Most of the files (SHA256) were detected by more than 30 AV products, which means that the results can be used to estimate with high accuracy whether the target is malware or not. In contrast, for the communication destination (IPv4), almost all of them were detected by fewer than ten engines, and it is not easy to estimate whether the target is malicious or not using only these results. This may be indirectly due to the fact that it is not easy to determine whether the target is malicious or not by simple scanning due to cloaking and the use of non-well-known ports. We also verified whether malware families can be estimated based on the scan results of VirusTotal using AVCLASS [88]. Of the 906 samples included in

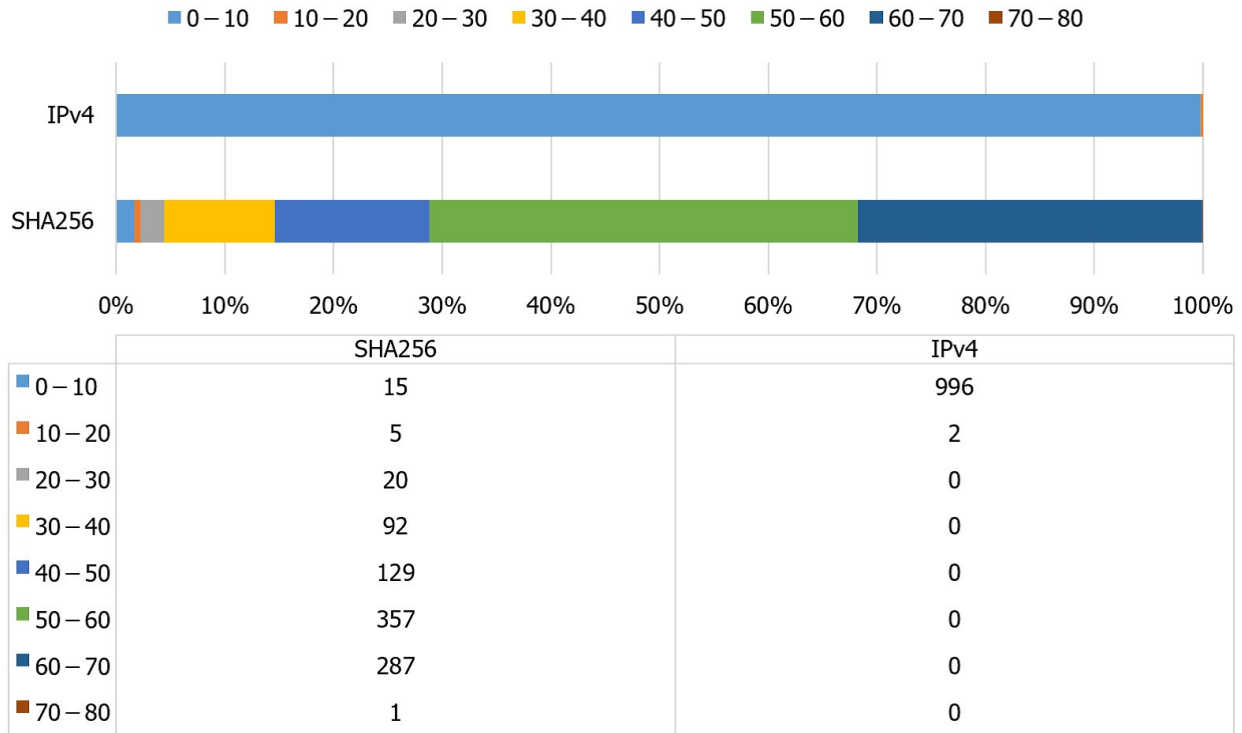


Figure 2.4 Number of AV detections for IOCs included in VirusTotal.

VirusTotal, we were able to estimate the malware family for about 40% as well as CyNER, but not for the remaining 60%. This can be attributed to the lack of information associated with the malware family, since some specimens and AV engines were detected with generic names such as *Generic.Trojan*. In addition, although samples that download malware in the latter stage were related to a specific malware family in a series of attacks, they were detected only as “Downloader” in isolation, and thus, similarly, no information related to the malware family could be obtained. On the other hand, CyNER can link IOCs to malware families in CTI, so it is highly possible to determine whether a malware is malicious or not regardless of whether it is SHA256 or IPv4. In addition, since CyNER links IOCs to attacks mentioned in CTI without depending on the nature of the sample, it is possible to link even Downloader to malware families that were dropped in the later stages.

2.5.3 Time-series

In this section, we analyze IOC from the view point of time-series. For each IOC, the date when it was first reported by CTI and the date when it was last reported were recorded, and

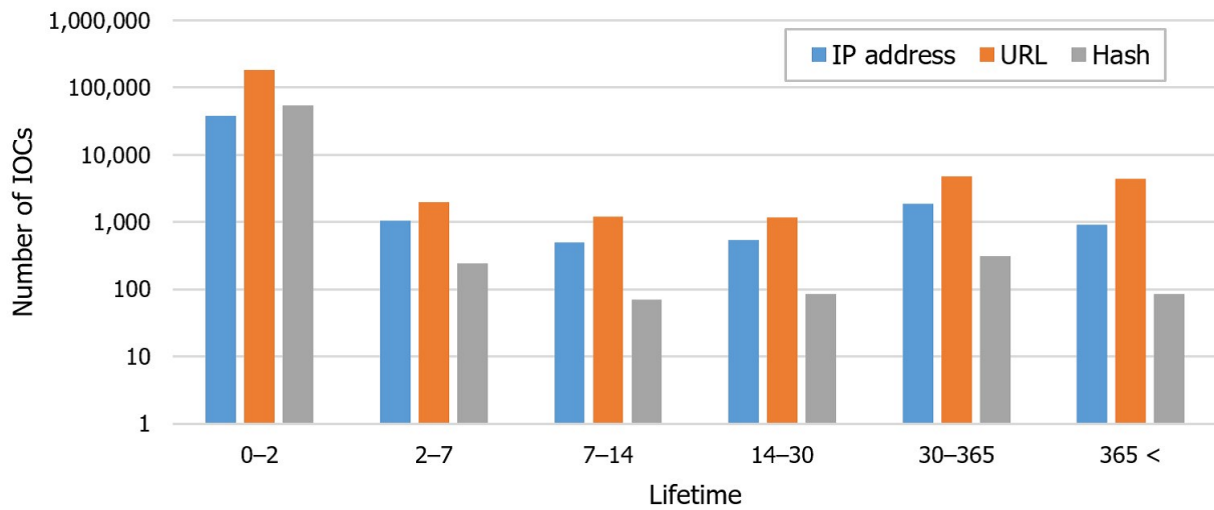


Figure 2.5 Lifetime for each type of IOC.

the difference in the number of days between them was defined as the observation period. The observation period was defined as the difference in the number of days between the two dates. Figure 2.5 shows the results of the survey divided by the type of IOC (IP address, URL, and hash), and the observation period plotted by its length. As we can see, most of the observation periods are within the range of 0 to 2 days for all the IOC types. In particular, 55,085 hash values, or more than 98% of all IOCs, fall within this range. This is probably due to the fact that it is relatively easy to detect malware based on hash values. Moreover, because hash values can be changed by variants, malware with the same hash tends not to be used for a long time. In addition, many of the URLs and IP addresses have a short observation period, suggesting that they are used and discarded after each attack.

On the other hand, although not the majority, 19,010 cases had an observation period of more than three days, and some of them were reported for a long period of time, especially for URLs and IP addresses. For example, 59[.]188[.]0[.]197 was an IP address that had been observed for a relatively long period of time (792 days). This IP address was reported as the C2 server in the spear phishing attack of the *Temper Panda* group in 2014. The IP address was later reported to have been used in an attack by the same group in 2015, suggesting that it is one of the attack infrastructures that the group has been continuously exploiting. The same IP address was also reported to have been used in an attack by the *APT16* group in 2015, suggesting that the attack infrastructure may be shared by multiple attack groups. Thus, it is possible that we can automatically extract more dangerous IOCs by extracting

IOCs that have been observed for a long time.

2.5.4 Information Source Relation

In this section, we describe the relationship between IOCs and information sources. As mentioned earlier, we used 35 sites as the information sources, and for each IOC, we identified which information source it was included in, and summarized the number of information sources that included the IOC. The results are shown in Fig. 2.6.

In this evaluation, 289,141 IOCs, or more than 97% of the total, were included in only one source. This result means that there are many IOCs that are included only in a specific source, and suggests that it is desirable to increase the number of sources in order to improve the comprehensiveness, which is one of the evaluation items in the previous section. On the other hand, there were 7,970 IOCs that were included in more than one source. For example, the hash value of *WannaCry*¹ was included in four sources, and the hash value of *BadRabbit*² was included in five sources. Thus, IOCs that are included in multiple sources are likely to be more threatening.

In this way, it is possible to automatically extract IOCs with a higher threat level by calculating the number of sources that contain IOCs and extracting the ones with the largest number.

2.6 Discussion

2.6.1 Practicality

Due to resource constraints, there is a real need to add only those threats that are of a higher level to the block list. For such a requirement, CyNER can be used to select those that are associated with a specific threat, or those that have been reported over a long period of time or across multiple sources. In addition, CyNER can be used to present long-term IOCs in chronological order, or to present IOCs that have been reported across multiple sources, which is expected to improve the efficiency of operations for the aforementioned requirements. In addition, since the IOCs are structured as STIX, which is a common

¹f8812f1deb8001f3b7672b6fc85640ecb123bc2304b563728e6235ccbe782d85

²8ebc97e05c8e1073bda2efb6f4d00ad7e789260afa2c276f0c72740b838a0a93

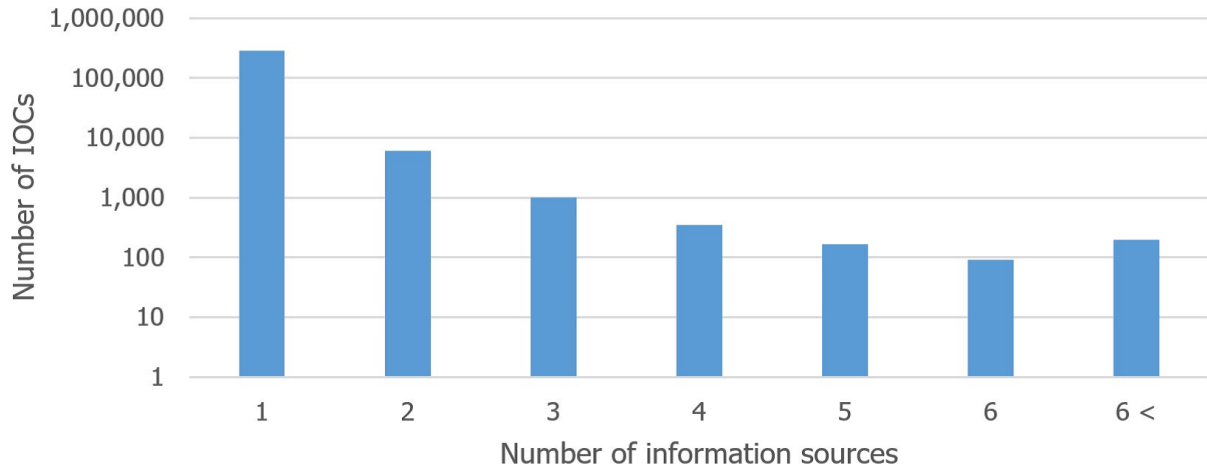


Figure 2.6 Number of sources including the IOC.

format, it is expected that the utilization by existing libraries (e.g., visualization by STIX Visualizer [89]) and the automatic linkage with the security appliance can be utilized.

On the other hand, there are some points that need to be considered in practical use. Although we used the F-measure uniformly in this accuracy evaluation, the accuracy that we consider important differs depending on the task. For example, for manual incident response, coverage is important even if false positives are tolerated. In contrast, when creating a block list, true positives are important because it is undesirable to over-detect normal communication. It is therefore necessary to choose which indicators are important depending on the task. In addition, it has been suggested that CTI potentially contains false positives [27] so it may be desirable to introduce a separate filter for sensitive applications against false positives.

The processing time was measured by processing each article in series, but in reality, it can be further accelerated by parallelizing each article or pipelining each process. In fine-tuning, we trained for a fixed 200 epochs, but we can complete the process in a shorter time by terminating the training using Early-stopping [90] and other methods. We also used a domain corpus to improve the accuracy of NER, but this requires pre-training, which in turn increases the processing time. RoBERTa-LARGE, which was mainly used in this study, took about 17 hours to train three epochs. In some cases, it is necessary to consider using a lighter model.

2.6.2 Limitation

False Positives and False Negatives. If the URL or IP address is defanged using an unknown method, there is a possibility that it cannot be refanged and cannot be extracted using regular expressions. However, in most cases, the defanging method is standardized for each CTI site, so we assume it is possible to deal with the problem by establishing a refang rule for each site.

In CyNER, all IOCs included in a CTI are handled flatly, and all IOCs are linked to a word that represents the CTI. In other words, the proposed method is likely to be incompatible with CTIs containing multiple topics in a single CTI, such as weekly reports.

Information Sources. In this dissertation, we focused on blogs and official announcements as CTI sources, but CTIs are published in other forms as well. One major CTI source is SNS, e.g., Twitter. Therefore, a lot of research has focused on collecting CTI from SNS: [91] gathers the vendor's patch release information from SNS, and [92] gathers threats or vulnerability information from SNS. In SNS analysis research, there are a number of unique challenges, e.g., texts are shorter than common articles, so extracting information is difficult [91,92], or fake information is potentially included, and verification is necessary [93]. Of course, intelligence in SNS has its advantages, primarily in that it is more prompt than in blogs; thus, in the future, we plan to extend CyNER for the importation of other sources for intelligence promptness and coverage. In addition, although 35 sources were used in this study, the information obtained in this experiment is not necessarily coverage, since other sources may exist.

2.6.3 Research Ethics

When collecting CTIs for evaluation in this dissertation, a certain interval was set for each access when information was obtained from the same site. In addition, as described in the design section, we checked for updates to the articles, and if there were none, we did not attempt further access. These measures reduce the unnecessary load on the CTI distribution site.

2.7 Conclusion

In this dissertation, we proposed CyNER, a method to automatically convert CTIs written in natural language into STIX, with the aim of improving the efficiency of analysis. CyNER extracts named entities and relations between named entities from CTI and then automatically structures them into STIX 2.1 format. Key phrases are extracted in units of CTI and then associated with noncontextual IOCs. This enables the extraction of relations that have no relation in the neighborhood, which is not possible in previous RE methods.

We extracted 297,101 IOCs from 55,266 CTIs of 35 information sources using CyNER, and conducted a crossover analysis. The results showed that CyNER can extract IOCs that are not included in the existing reputation services. We also found that 19,274 IOCs are continuously reported and that some IOCs are exploited across multiple attack groups for more than a year. From the above results, it is expected that CyNER will contribute to the efficiency of CTI analysis. Future work will include improving the accuracy of each task and evaluating CyNER on larger datasets.

Chapter 3

Tracing Diffusion of Classified Information on KVM

3.1 Introduction

With increase in the use of personal computers, occasion to process classified information using a computer has also increased. Associated with this, the leakage of classified information to an outside computer has become a serious problem. According to a personal information leakage analysis [94], a leakage often occurs as a result of inadvertence and mismanagement, which accounts for approximately 57% of all known leakage cases. To prevent information leakage, it is important for the user to grasp the situation surrounding the classified information. In addition, cyber-attacks aiming at the theft of classified information have become increasingly sophisticated. Therefore, it is difficult to completely prevent such attacks, and it has become important to reduce the amount of damage incurred to users by detecting the transfer of classified information outside their computer [95].

To trace the status of classified information in a computer, and manage the resources that contain such information, an OS-based function for tracing the diffusion of classified information [96] (particularly, an OS-based tracing function) has been proposed. This function manages any process that has the potential to diffuse classified information. In addition, the OS-based tracing function visualizes the diffusion using a directed graph [97], and traces the diffusion of the classified information on multiple computers [98]. Such functions are efficient for grasping the use situation and preventing leakage of classified information.

On the other hand, an OS-based tracing function is executed within the OS, and therefore, has the potential of being detected and disabled by an adversary or a malicious user. If the OS-based tracing function is disabled, the victim cannot detect the information leakage and there is a risk of increased damage. In addition, the function cannot be introduced in a closed-source OS such as Windows, because its implementation requires a modification of the OS's source code. Further, when the kernel version is updated, the function must be adapted to the newly updated kernel.

Similar to an OS-based tracing function, a large number of methods for protecting sensitive files have been proposed [36, 39, 43, 45]. However, because they are implemented within the OS, there is a problem in that the operational environment is limited, and they can be detected and disabled, similar to an OS-based tracing function. To resolve the above mentioned problems, methods for protecting sensitive files from outside the OS have been proposed [40, 41]. These methods demonstrate the effectiveness of an outside OS implementation of a security system. However, it is difficult to identify the cause of an information leakage because these methods are aimed solely at information leakage prevention and do not trace the diffusion of classified information.

Based on the above mentioned observations, we designed a function for tracing the diffusion of classified information in a guest OS using a VMM specifically, a VMM-based tracing function. This VMM-based tracing function provides the guest OS with functions that are equivalent to an OS-based tracing function, without the need to modify the source code of the guest OS. It is expected that attacks specifically targeting this function will be difficult to achieve because a VMM is more robust than an OS.

A preliminary description of the VMM-based tracing function has already been presented in my graduation thesis; it focused on introducing the basic design. This dissertation describes the detailed design and implementation of the function for file operation, child process creation, and IPC using a kernel-based virtual machine (KVM) [99]. The VMM-based tracing function hooks system calls that possibly cause information leakage. Moreover, the function traces the status of the classified information in the guest OS from outside it. We have implemented a prototype of the VMM-based tracing function for a Linux guest. This dissertation also describes an evaluation including traceability, amount of the modified source code, and performance of the VMM-based tracing function.

3.2 OS-Based Function for Tracing Diffusion of Classified Information

3.2.1 About This Section

The OS-based tracing function has been proposed in [96] by Tabata et al. The VMM-based tracing function, which is my proposed function in this dissertation, is based on this OS-based tracing function. Thus, this section describes the OS-based tracing function before the description of the proposed VMM-based tracing function.

3.2.2 Classified Information Diffusion Path

The OS-based tracing function manages any files or processes that have the potential to diffuse classified information. Classified information can be diffused through any process that involves opening a classified file, reading its contents, communicating with another process, or writing such content to another files. Therefore, the diffusion of classified information is caused by the following operations.

- (1) File operation
- (2) Child process creation
- (3) Inter-process communication

The OS-based tracing function traces the diffusion of classified information by monitoring the system calls related to such operations.

3.2.3 Purpose

As described in Section 3.1, the leakage of classified information often occurs as a result of inadvertent handling and mismanagement (e.g., the improper transmission of an e-mail). This is attributed to the user's inability to grasp the location of classified information. In addition, it is difficult to completely prevent the leakage of such information. Therefore, when an information leakage occurs, it is important to detect the event and grasp its cause. Based on the above background, the tracing function aims at achieving the following.

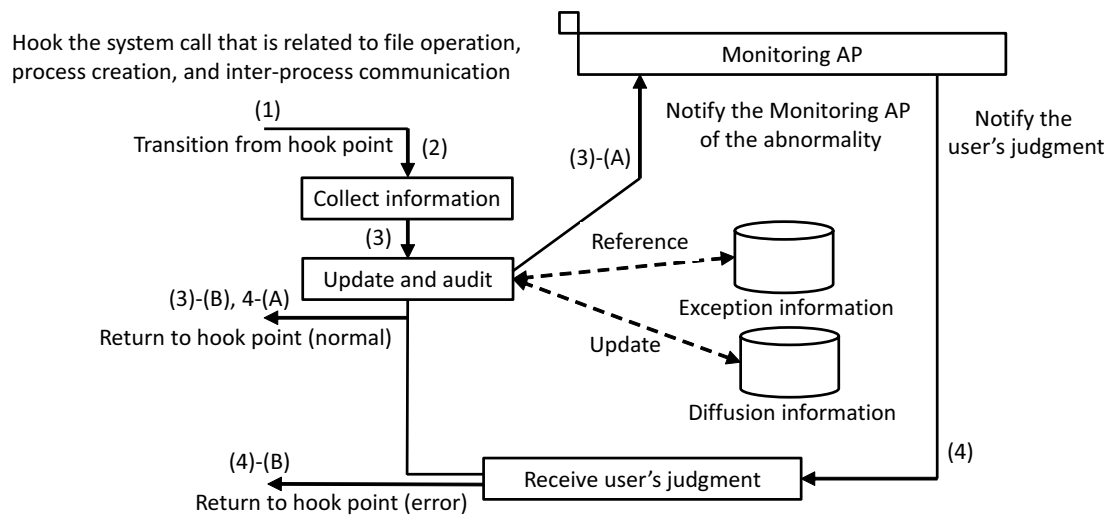


Figure 3.1 Overview of the OS-based tracing function.

Purpose 1 Tracing the diffusion of classified information inside the computer.

Purpose 2 Detecting the leakage of classified information to outside the computer and recording its cause.

3.2.4 Overview of the OS-based Tracing Function

An OS-based tracing function [96], an overview which is shown in **Fig. 3.1** have been proposed. This OS-based tracing function traces the diffusion of classified information as follows:

- (1) The OS-based tracing function hooks system calls that are related to the diffusion of classified information.
- (2) The OS-based tracing function collects information for tracing the diffusion of classified information such as files handled by a system call or the transmission-destination process.
- (3) The OS-based tracing function updates the diffusion information using the information collected during (2), and audits the potential leakage of classified information.

- (A) If the audit discovers the possibility of a leakage of classified information, the OS-based tracing function notifies such leakage to the monitoring application program (AP).
 - (B) If the audit does not detect any possibility of such leakage, the OS-based tracing function returns the control to the system call.
- (4) After receiving the results of the user's judgment based on the monitoring AP, the OS-based tracing function controls the system call accordingly as follows:
- (A) If the user's judgment is affirmative, the system call processing is continued.
 - (B) If the user's judgment is negative, the system call processing is terminated as an error.

In addition, the OS-based tracing function excludes files and processes that are unrelated to the diffusion of classified information. These files and processes are registered with the exception information.

3.2.5 Problems with the OS-Based Tracing Function

The OS-based tracing function meets the purpose described in Section 3.2.3. However, this function has certain problems as follows:

Problem 1 The OS's source code must be modified before introduction.

In order to introduce the OS-based tracing function, it is necessary to modify the OS's source code. Therefore, the OS-based tracing function cannot be implemented in a closed-source OS such as Windows. Furthermore, when the kernel version of the OS is updated, the OS-based tracing function must modify the source code again after the OS is updated.

Problem 2 There is a risk of an attack invalidating the tracing function

The OS-based tracing function is implemented in the OS. Therefore, an adversary or a malicious user can invalidate the function by attacking the OS. If the function is invalidated, it becomes difficult to prevent information from being leaked and grasp the location of the classified information.

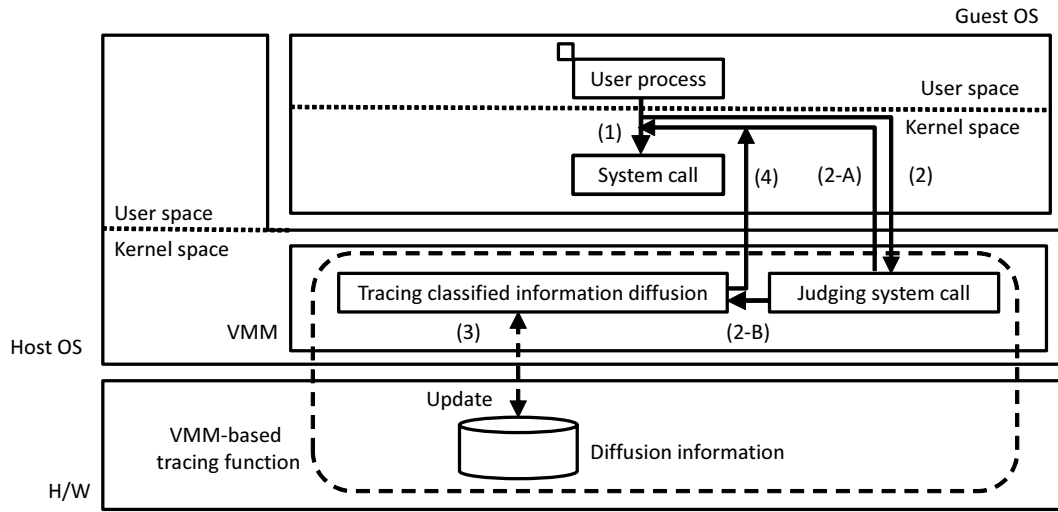


Figure 3.2 Overview of the VMM-based tracing function.

Further, as described in Section 3.1, there are similar problems in the existing methods for protecting sensitive files. In this dissertation, we propose a method that resolves both problems.

3.3 VMM-Based Function for Tracing Diffusion of Classified Information

3.3.1 Requirements

To resolve the problems detailed above in Section 3.2.5, the following are required:

Requirement 1 The OS's source code must not be modified.

One solution to Problem 1 is to avoid the modification of the OS's source code. This ensures that the function can be introduced in a closed-source OS such as Windows.

Requirement 2 The function should be isolated from the OS.

Isolating the function from the OS is a solution to Problem 2. Such a solution makes it difficult for an adversary or a malicious user to attack the function directly.

3.3.2 Overview of the VMM-Based Tracing Function

The VMM-based tracing function is functionally equivalent to the OS-based tracing function. In particular, the VMM-based tracing function manages any files or processes that have the potential to diffuse classified information. Moreover, the VMM-based tracing function traces the status of the classified information in a computer, and manages the resources that contain the classified information by monitoring the three operations described in Section 3.2.2. The user can always grasp the location of their classified information using the list of classified information stored in the VMM. Furthermore, when a diffusion of classified information is detected, the VMM-based tracing function records the pathname of the destination file, inode number, name of the command causing diffusion, and the process ID (PID). Therefore, the user can detect the information leakage using the above information and suppress the damage even if a leakage has occurred.

Figure 3.2 shows an overview of the VMM-based tracing function. The VMM-based tracing function traces the diffusion of classified information as follows:

- (1) A user program in the guest OS requests a system call.
- (2) The VMM-based tracing function hooks the system call in the guest OS with the VMM. After identifying the hooked system call, the following processing is conducted.
 - (A) When a hooked system call is unrelated to the diffusion of classified information, control is returned to the guest OS and the system call processing is continued.
 - (B) When a hooked system call is related to the diffusion of classified information, the VMM-based tracing function collects the information needed to trace the diffusion.
- (3) The VMM-based tracing function updates the diffusion information using the information collected in (2-B) if the classified information is diffused.
- (4) Control is returned to the guest OS and the system call processing is continued.

Given these steps, the VMM-based tracing function provides the guest OS with functions equivalent to an OS-based tracing function.

3.3.3 Tasks

To implement the VMM-based tracing function, the following tasks are required:

Task 1 Collecting the system call information with the VMM.

The classified information is diffused by the system call. Therefore, it is necessary to hook the system call. Further, the VMM-based tracing function collects the system call's information owing to the judgment of whether the system call is related to the classified information diffusion.

Task 2 Collecting the OS information with the VMM.

The VMM-based tracing function manages any file or process that has the potential to diffuse classified information. Therefore, it is necessary to collect the information from the OS, such as the processes that are running, their transmission destination, and the files handled by the processes that are running.

In Sections 3.3.4 and 3.3.5, we describe the procedure by which the above tasks are accomplished. Further, this procedure is tailored for a 64-bit version of Linux in which the system call is executed by SYSCALL/SYSRET.

3.3.4 Collecting System Call Information with Virtual Machine Monitor

Hooking a System Call Entry

The VMM-based tracing function hooks the system call entry (viz., SYSCALL). By hooking SYSCALL, we can detect system call requests. In order to hook SYSCALL, the VMM-based tracing function sets the breakpoint-address register to SYSCALL's address. A breakpoint-address register specifies the breakpoint address and a debug exception is generated when a memory access is made to the breakpoint address. Thus, a debug exception occurs upon executing SYSCALL. Therefore, the VMM-based tracing function can hook SYSCALL with the VMM by detecting debug exceptions in the guest OS.

Hooking a System Call Exit

Each system call returns information concerning the success or failure of the system call, and the details of the file handled by the system call as a return value. It is necessary to collect the details about the file that is handled by the running process so that the VMM-based tracing function can trace the diffusion of the classified information. Thus, the VMM-based tracing function also hooks the system call exit (viz., SYSRET). By hooking SYSRET, it is possible to obtain the system call's return value. In order to hook SYSRET, the VMM-based tracing function sets the breakpoint-address register to SYSRET's address as well as hooking SYSCALL. Consequently, by detecting debug exception in the guest OS, the VMM-based tracing function can hook SYSRET with the VMM.

Collecting Information

It is necessary to judge whether the hooked system call is related to the diffusion of the classified information. To identify the system call, the VMM-based tracing function uses a system call number. In addition, it is necessary for the VMM-based tracing function to identify the transmission-destination file or process. A system call takes the file or process information to an argument. By obtaining the system call's argument, it is consequently possible to identify the transmission-destination file or process. Furthermore, as already described, the VMM-based tracing function obtains the system call's return value and utilizes the return value for identifying the transmission-destination file or process.

3.3.5 Collecting OS Information with Virtual Machine Monitor

The VMM-based tracing function traces the diffusion of classified information using information from the OS, such as file information and process information. Then, the semantic gap [100] must be bridged so that the VMM-based tracing function can obtain the OS information with the VMM. The semantic gap is the gap between the guest OS as it is viewed from the outside and the view of it from the inside. To bridge the semantic gap, the VMM-based tracing function constructs a semantic view by retrieving information about the guest OS beforehand.

3.3.6 Advantages

Implementing the VMM-based tracing function has three advantages as follows:

Advantage 1 The tracing function can be introduced in various environments.

Unlike the OS-based tracing function, the VMM-based tracing function can be implemented without modifying the target OS's source code. This makes it possible to introduce the tracing function in a closed-source OS such as Windows.

Advantage 2 The attacks aimed at the tracing function are made more difficult.

The VMM-based tracing function is implemented in the VMM. From this, an adversary's or a malicious user's attacks aimed at the tracing function are made more difficult, because the VMM is isolated from the guest OS.

Advantage 3 The tracing function will continue to be available even if the kernel version is updated.

The OS-based tracing function fully depends on target OS's kernel version, because this function is implemented by modifying each system call in the target OS's kernel. On the other hand, the VMM-based tracing function is only depends on following two items.

(1) System call specifications

The VMM-based tracing function identifies the invoked system call using system call number and obtains the system call's argument for tracing the diffusion of classified information. Therefore, the VMM-based tracing function depends on the system call specifications.

(2) Some data structures

The VMM-based tracing function obtains some OS information (e.g., process information) by guest OS's data structure. Thus, the VMM-based tracing function also depends on some data structures.

Therefore, the tracing function will continue to be available even if the kernel version is updated, provided that the system call specifications and the data structure remain unchanged.

3.4 Implementation

3.4.1 Environment

This section describes the implementation of the VMM-based tracing function using a KVM as the VMM and 64-bit Linux 3.6.10 as the guest OS. The VMM-based tracing function detects requests for system calls by hooking SYSCALL, and it obtains return values by hooking SYSRET. Therefore, the system call in the guest OS is executed by SYSCALL/SYSRET. Further, the guest OS is fully virtualized using Intel Virtualization Technology (VT).

3.4.2 File Operation

The VMM-based tracing function hooks the `open()`, `read()`, `write()`, and `close()` system calls that are related to file operations. Further, to trace the diffusion of classified information by file operations, the VMM-based tracing function collects the following information:

- (1) Current-process identifier
- (2) Identifier of the file that is handled by the system call.

It is necessary for the VMM-based tracing function to collect the current-process identifier in order to judge whether the process requesting the system call is a management target when the VMM-based tracing function hooks each system call. To identify the current process, the VMM-based tracing function uses the PID. The VMM-based tracing function obtains the PID when the function hooks the SYSCALL. Moreover, it is necessary for the VMM-based tracing function to identify the file that is handled by the system call when the VMM-based tracing function judges whether the file that is read is a management target, and to register the written file with the diffusion information. To identify the file that is handled by the system call, the VMM-based tracing function uses the inode number. The VMM-based tracing function obtains the inode number by following the data structure from the process-control block to the file structure. Then, the VMM-based tracing function identifies the inode number by using the file descriptor. The file descriptor is obtained with the system call's return value in cases where `open()` is hooked. Likewise, the file descriptor is obtained by the system call's argument in cases where `read()`, `write()`, and `close()` are hooked.

3.4.3 Child Process Creation

The VMM-based tracing function hooks the `clone()` system call, which is related to child process creation. Moreover, in order to trace the diffusion of classified information by child process creation, the VMM-based tracing function collects the following information:

- (1) System call's product identifier
- (2) Parent-process identifier
- (3) Child-process identifier

The `clone()` system call creates not only a new process but also a new thread. The threads in the same process share resources such as information related to the opened files. Therefore, thread creation does not diffuse the classified information outside the process. On the other hand, child process creation diffuses resources from the parent process to the child process. Thus, it is necessary to judge whether the `clone()` creates a process or a thread. If the thread is created, the `CLONE_THREAD` flag, which is the argument of `clone()`, is set. Consequently, by auditing the argument of `clone()`, we can judge whether the product of `clone()` is a process or thread. The `CLONE_THREAD` flag is obtained from the `clone()` argument when the VMM-based tracing function hooks the `clone()` system call entry.

Moreover, when the parent process is a management target, there is a risk that the classified information will be diffused to the child process. Therefore, to judge whether the parent process is a management target, it is necessary to collect the parent-process identifier. To do so, the VMM-based tracing function uses the parent process's PID. The parent process' PID is obtained from the process-control block when the VMM-based tracing function hooks the `clone()` system call entry.

Furthermore, the VMM-based tracing function registers the child process with the diffusion information when the VMM-based tracing function judges that the classified information is diffused to the child process. Thus, the child-process identifier must be obtained. To identify the child process, the VMM-based tracing function uses the child process' PID. When `clone()` creates a new process, the return value is the child process's thread ID (TID) and the TID is identical to its PID. Thus, the child process' PID is obtained from the return value of `clone()` when the VMM-based tracing function hooks the `clone()` system call exit.

3.4.4 Inter-process Communication

Target Tracing

Section 3.4.2 and 3.4.3 described a method for tracing the diffusion of classified information using (1) file operation and (2) child process creation, respectively. Next, we describe a method for tracing the diffusion of classified information through (3) IPC using the VMM-based tracing function.

IPC is implemented through various methods, for example, a pipe, FIFO, message queue, shared memory, or socket. Among these, IPC using a socket is applied not only for local IPC, but also for remote IPC. Remote IPC has the possibility of classified information leaking outside the user's computer. Therefore, tracing remote IPC has a higher level of importance.

In this dissertation, we describe a method for tracing IPC using a socket.

Tasks for Tracing Inter-process Communication

To detect the diffusion and leakage of classified information through IPC using the VMM-based tracing function, the following tasks are required:

Implementation Task 1 Detecting the diffusion and leakage of classified information through socket communication.

Implementation Task 2 Obtaining the necessary information for tracing a socket communication.

To trace a socket communication using a VMM, it is necessary to hook the system call related to the socket communication, and use it to detect the diffusion and leakage of classified information. Furthermore, tracing the socket communication with the VMM is complex and difficult when using a method similar to trace a file operation or child process creation because the socket intermediates the communication and is used for communication with an outside computer. It is therefore necessary to manage the socket, which is used as an IPC communication path, and detect the propagation of classified information.

To accomplish **Implementation Task 1**, the VMM-based tracing function hooks the system calls, which causes the diffusion and leakage of classified information through a socket communication. For **Implementation Task 2**, we describe the solutions to each socket communication, i.e., local IPC and remote IPC.

Table 3.1 System call utilized for socket communication. The relevant parts are marked with “✓.”

Category	Name of system call	Possibility of diffusion	Possibility of leakage	Necessity for tracing
Setup	socket			
	bind			
Server	listen			
	accept			
Client	connect			
Output	sendto	✓	✓	✓
	sendmsg			
	write			
	sendfile			
Input	recvfrom	✓		✓
	recvmsg			
	read			
Termination	shutdown			✓
	close			

Detecting the Diffusion and Leakage of Classified Information through a Socket Communication

Table 3.1 shows the system calls used for a socket communication along with its name, category, possibility of classified information diffusion and information leakage, and tracing necessity. Table 3.1 shows a case for Linux 3.6.10, which differs depending on the kernel version.

The diffusion and leakage of classified information do not occur during the socket creation or the establishment of a connection because the data cannot be exchanged. Therefore, from socket creation to establishment of connection, the VMM-based tracing function does not trace the processed system calls.

The diffusion of classified information occurs when the data communication by an *Output/Input* system call is actually conducted. Thus, the VMM-based tracing function traces *Output/Input* system calls. In addition, an actual information leakage occurs through the transmission of data outside the computer using an *Output* system call. Therefore, the VMM-based tracing function audits the potential for leaking classified information when it hooks to an *Output* system call.

Furthermore, the terminated socket is excluded from the managed socket list because it

remains unused.

Obtaining the Necessary Information for Tracing a Socket Communication

Local IPC

In local IPC through a socket communication, the VMM-based tracing function hooks an *Output/Input* system call and identifies the socket of the communication medium. At that time, the VMM-based tracing function traces the diffusion of classified information through IPC by managing whether the socket can obtain this information.

Remote IPC

During remote IPC using socket communication, the VMM-based tracing function hooks the *Output* system call and detects an information leakage. Then, to identify the process and computer at the destination of the communication, the VMM-based tracing function obtains its IP address and port number.

Flow of Tracing Socket Communication

Solving these tasks enables the VMM-based tracing function to trace the diffusion of classified information and detect the leakage of such information through a socket communication. The flow of the socket communication tracing is as follows:

- (1) The VMM-based tracing function hooks the system call related to the *Output* in the guest OS from the VMM.
- (2) When a process issuing the *Output* system call is managed, the following processing is conducted.
 - (A) In the case of local IPC, the socket is appended to the managed socket list.
 - (B) In the case of remote IPC, the possibility of an information leakage is determined, and the VMM-based tracing function notifies the user as such by producing a log.
- (3) Control is returned to the guest OS and the *Output* system call process is continued.
- (4) The VMM-based tracing function hooks the system call related to the *Input* in the guest OS from the VMM.

Table 3.2 Evaluation environment.

CPU		Intel Core i5-3470, 3.2 GHz
OS	Guest	Fedora 18 (Linux 3.6.10, 64bit)
	Host	Fedora 18 (Linux 3.6.10, 64bit)
Memory	Guest	1,024 MB
	Host	4,096 MB
VMM		KVM-kmod-3.6

- (5) When the socket utilized for IPC is managed, the process issuing the *Input* system call is appended to the managed process list.
- (6) Control is returned to the guest OS and the *Input* system call process is continued.

3.5 Evaluation

3.5.1 Experimental Setup

We evaluated the following three items.

- (1) Traceability
- (2) Lines of code (LOC)
- (3) Overhead

First, we audit whether the VMM-based tracing function can trace the diffusion of classified information and can detect the leakage of such information by conducting the assumed scenario in the guest OS. In addition, to evaluate the cost for implementation, we compared the amount of LOC of the OS-based tracing function and the VMM-based tracing function. Further, we measured the overhead incurred by the VMM-based tracing function. We then measured the performance of the system call, microbenchmark, and application (AP). **Table 3.2** shows the evaluation environment. We evaluated the VMM-based tracing function with Core i5-3470 (3.2 GHz, 4-cores) and 4,096 MB of memory. The guest OS is allocated one virtual CPU and 1,024 MB of memory. Additionally, the EPT is disabled.

```
write()
sensitive data is diffused to "usb/dst.txt" (inode number: 158) by "vim" (pid: 1380)
-----trace file list-----
no.: inode number, file path
  0:      524493, fujii/secret.txt
  1:      158, usb/dst.txt
-----
```

Figure 3.3 Log generated by the VMM-based tracing function in (Assumed Scenario 1).

3.5.2 Traceability

Evaluation Methods of Traceability

To evaluate the traceability of the VMM-based tracing function, we performed the following scenario.

Assumed Scenario 1 Export to external device

After editing a text file using the text editor, write out the edited data onto a USB memory. The same processing is performed for an unmanaged file.

Assumed Scenario 2 Copy of the directory unit

Prepare a directory that has 10 files, i.e., 5 classified files and 5 unclassified files, and copy this directory to another directory.

Assumed Scenario 3 Send the classified information through local IPC using a UNIX domain socket.

Conduct local IPC through the UNIX domain socket from the managed client to an unmanaged server.

Assumed Scenario 4 Send the classified information outside the computer.

Send the managed file outside the computer using a `scp` command.

Assumed Scenario 5 Send an e-mail with an attached file which include classified information.

Send the e-mail with managed file using a `mail` command.

Using the above scenarios, we verify whether the VMM-based tracing function can trace the diffusion of the classified information.

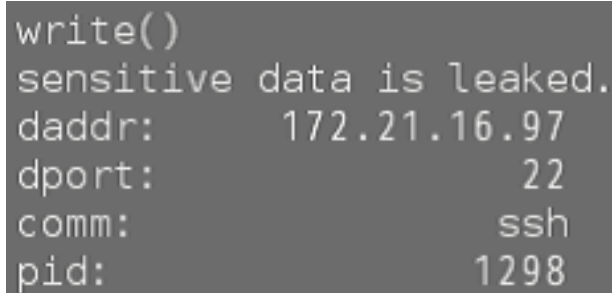
Experimental Result of Traceability

Figure 3.3 shows the log generated by the VMM-based tracing function when **Assumed Scenario 1** is executed. As described in Section 3.3.2, when the diffusion of classified information is detected, the VMM-based tracing function records the pathname of the destination file, inode number, command name that is the cause of diffusion, and PID. In **Assumed Scenario 1**, the classified file is `fujii/secret.txt` (inode number: 524493), and the file written in the USB memory is `usb/dst.txt` (inode number: 158). From the message shown in Fig. 3.3, we can infer that the classified information is diffused to `usb/dst.txt` by `vim`, which is the text editor. It also confirms that `usb/dst.txt` and 158, which is the inode number of `usb/dst.txt`, are recorded in the trace file list. In contrast, the classified information is not diffused by the operation of the unclassified file. Then, we performed the same processing for an unmanaged file. In the above experiment, we observed that the information of the process executed on the unclassified file is not recorded.

However, when **Assumed Scenario 2** is executed by `cp -r src_dir dst_dir`, the 10 new files are all judged to be the classified file. Although five files are judged to be the classified files and other five files are judged to be unclassified file, the false positive occurs. This false positive occurred because the process collectively executes `cp`, which is judged to be the classified process at the time point of reading data from a classified file and the files written out from this process are all judged to be classified files. On the other hand, when **Assumed Scenario 2** is executed by `find src_dir | xargs -iX cp X dst_dir`, a misdetection did not occur. This is because each copy operation is executed by one process and the above problem is avoided by combining `find`, `xargs`, and `cp`. In this scenario, there is a possibility that misdetection may occur. However, a false negative does not occur.

During local IPC, the classified information is diffused to a socket when the managed process sends it there. Next, the classified information is also diffused to the other process by receiving it from the managed socket. When **Assumed scenario 3** is executed, the VMM-based tracing function detects the diffusion of classified information to the socket and appends it to the managed socket list when it hooks a `sendto()`. In addition, the VMM-based tracing function detects the diffusion of classified information from the socket to process and appends it to the managed process list when it hooks a `recvfrom()`.

During remote IPC, the classified information is leaked when the managed process sends it outside the computer. When **Assumed scenario 4** executed by a `scp` command, the



```
write()
sensitive data is leaked.
daddr: 172.21.16.97
dport: 22
comm: ssh
pid: 1298
```

Figure 3.4 Log generated by the VMM-based tracing function in (Assumed Scenario 4).

VMM-based tracing function detects the information leakage when it hooks a `write()` system call that sends the classified information outside the computer. **Figure 3.4** shows the log generated by the VMM-based tracing function when it detects an information leakage. As shown in Fig. 3.4, the VMM-based tracing function records the destination IP addresses (`daddr`), port numbers (`dport`), command names (`comm`), and PIDs (`pid`). The user can identify the cause of the information leakage from such information. Correspondingly, the VMM-based tracing function detects the information leakage and records the equivalent information illustrated in Fig. 3.4 when **Assumed Scenario 5** is executed.

According to the above results, we can say that the VMM-based tracing function traces the diffusion of classified information and detects the leakage of such information accurately. Further, the VMM-based tracing function causes no false negative. Even if the function detects an information leakage excessively, it is important that no information leakage occurs.

3.5.3 Lines of Code

Evaluation Methods of Lines of Code

We count the logical LOC and the number of files modified for implementing the tracing function. The logical LOC is the number of coding lines excluding only line made by a symbol, whitespace, and comment. To count the logical LOC, we use LocMetrics [101]. Then, the counting target is the logical LOC-related file operation and process creation.

Further, the scale of the source code is a great variation in each implementation environment owing to the fact that the OS-based tracing function is implemented within the OS and the VMM-based tracing function is implemented within the VMM. Thus, we institute a

Table 3.3 Comparison of logical LOC and the number of files modified for the tracing function.

	Logical LOC			Number of files		
	Total	Added	Rate (%)	Total	Added/Modified	Rate (%)
OS-based tracing function	47,222	763	1.61	101	14	13.9
VMM-based tracing function	35,555	894	2.51	49	10	20.4

counting target to directories that have the stored files modified for implementing the tracing function. In particular, the kernel/, fs/, and init/ directory under Linux OS, and the x86/ directories under the KVM are treated as a counting target.

Comparative Result of Lines of Code

The result of counting logical LOC and the number of files modified for implement the tracing function is presented in **Table 3.3**. The amount of logical LOC of the VMM-based tracing function is 131 more lines than that of the OS-based tracing function. This is attributed to the function that collects the information from the OS, such as file information and process information. The difference in the rate of logical LOC is 0.90% and it can be said that this is extremely small.

As already said in Section 3.5.3, the modified files for implementing the OS-based tracing function are scattered in multiple directories. This is due to the fact that the OS-based tracing function is implemented by modifying each system call that is related to the diffusion of classified information. In contrast, the VMM-based tracing function traces the diffusion of the classified information by hooking the entry point of the system call unitary. Thus, the modified files for implementing the VMM-based tracing function are localized in a single directory. Further, the total number of files modified for implementing the VMM-based tracing function is 10, and it is within the 70% as compared to that of the OS-based tracing function.

In summary, the VMM-based tracing function can be implemented only by slight addition and localizing the range of modification as compared to the OS-based tracing function.

3.5.4 Overheads

Evaluation Method of Overheads

The evaluation items are listed below.

(1) System Call

To measure the overhead generated through the introduction of the VMM-based tracing function, we measured the performance of the guest OS both before and after introduction. Moreover, for comparison, we measured the performance when the process was both managed and unmanaged. In this evaluation, we measured the overhead of `write()`, `read()`, `close()`, `clone()`, `sendto()` and `recvfrom()`, which is related to the diffusion of classified information. On the other hand, the VMM-based tracing function hooks all system calls even if a system call is unrelated to the diffusion of classified information. We then measured the performance of `getpid()` to clearly determine the impact of the performance of the system calls unrelated to the diffusion of classified information.

(2) Microbenchmark

We use LMBench [102] version 3 as a microbenchmark. To evaluate the influences on the performance of the basic functions of an OS, we measured the latency of it. We then measured the performance of the guest OS both before and after introducing the VMM-based tracing function as well as evaluation of system call overhead.

(3) Application

Performance degradation with the VMM-based tracing function will occur in each VM-Exit caused by the system call invocation. To evaluate the impact of these overheads on the application programs, we measure the performance of building `bzImage` that issues a large number of `read()` and `write()` system calls.

We also measure the average response time of web server by using ApacheBench, version 2.3. We use the web server for evaluation of application since the VMM-based tracing function monitors a socket communication and a socket communication occurred frequently in the processing of web server. This evaluation uses an Apache 2.4.6 as the web server. This web server runs on the guest OS of the VMM-based tracing function and provides web pages with 1 Gbps network. To measure the average

Table 3.4 Overhead of system calls incurred by the VMM-based tracing function (μs).

system call	Untraced	Traced (Relative performance)		Overhead	
		Operation of unmanaged target	Operation of managed target	Operation of unmanaged target	Operation of managed target
write (file)	0.60	2.76 (459.06%)	3.17 (626.16%)	2.16	2.57
read (file)	0.30	2.25 (740.52%)	2.26 (744.44%)	1.95	1.96
close (file)	0.28	2.32 (814.58%)	2.40 (842.69%)	2.03	2.12
clone	101.60	108.84 (107.13%)	113.95 (112.16%)	7.24	12.35
sendto	12.92	36.68 (283.83%)	47.10 (364.48%)	23.76	34.18
recvfrom	12.42	57.19 (460.36%)	62.00 (499.10%)	44.77	49.58
getpid	0.015	0.016 (106.86%)	-	0.0011	-

response time of web server, the client machine sends an HTTP request 1,000 times to the web server by using ApacheBench and calculates its average. The ApacheBench runs on the client machine described in Section 3.5.1. In addition, for comparison, we measure the average response time before and after introduction of the VMM-based tracing function. In measurement after introduction, we measure the average response time both when the index.html is managed and unmanaged.

System Call

Table 3.4 shows the overhead of the system calls incurred by the VMM-based tracing function. In Table 3.4, *Untraced* shows the measurement prior to the introduction of the VMM-based tracing function and *Traced* shows the measurement after its introduction. *Operation of managed target* and *Operation of unmanaged target* in *Traced* show the measurements conducted while operating a managed/unmanaged file, process, or socket. In addition, *Overhead* is calculated using the following formula: (*measurement in each environment* – *measurement before the introduction of the function*).

The overhead of write(), read(), close(), clone(), sendto() and recvfrom() during the operation of an unmanaged target is 2.16, 1.95, 2.03, 7.24, 23.76, and 44.77 μs , respectively, with relative performance level of 459.06%, 740.52%, 814.58%, 107.13%, 283.83%, and 460.36%, whereas during the operation of a managed target is 2.57, 1.96, 2.12, 12.35, 34.18, and 49.58 μs , respectively, with relative performance level of 626.16%, 744.44%, 842.69%, 112.16%, 364.48%, and 499.10%. These are relatively large value.

When the VMM-based tracing function determines that the hooked system call is related to the diffusion of classified information, it hooks the `SYSRET` and obtains the system call's return value. Subsequently, it stores system call's arguments and returns the control to the guest OS. Moreover, the VMM-based tracing function obtains the OS information (e.g., inode number) by using the system call's arguments and return values. It is suspected that this additional processing is the cause of the large overheads. The overhead during the operation of a managed target is greater than that of an unmanaged target because the managed file/process/socket list is scanned when judging whether the file/process/socket is managed. It seems that the overhead incurred after the introduction of the VMM-based tracing function will be increased with the number of managed files, processes, and sockets.

On the other hand, the overhead of `getpid()`, which is unrelated to the diffusion of classified information, is $0.0011\ \mu\text{s}$, with a relative performance level of 106.86%, which is relatively small value. In the case of system calls unrelated to the diffusion of classified information, the performance impact is slight because the VMM-based tracing function only judges whether the hooked system call is related to the diffusion of classified information.

To summarize, owing to the additional processing for tracing information, the overhead of the system calls that are related to the diffusion of classified information is large as compared to that of the other system calls.

Microbenchmark

Table 3.5 shows the latency of the basic functions of an OS measured by using LMbench. By comparing the measurement result before introduction of the VMM-based tracing function and after introduction, we find that the overhead of *fork proc*, *exec proc*, and *sh proc* is 121.87–125.47%. These values are relatively small.

Next, let us consider *null call*, *null I/O*, *stat*, *open clos*, *sig inst*, and *sig hndl*. By comparing the measurement result before introduction of the VMM-based tracing function and after introduction, we find that these items' performance is influenced by 260.56–4400.00% and this is very large. This is because the measurement values in the *Untraced* are small and the overhead ratio in the *Traced* is relatively large. The actual measurement values are subsided to about 1–2 μs in terms of the impact on most items. However, the overhead of *open clos* is $5.35\ \mu\text{s}$, and this is large as compared to that of the other items. This is attributed to the fact that the `open()` and `close()` system calls are related to the diffusion of

Table 3.5 LMBench results (μ s).

Item	Untraced	Traced Relative performance	Overhead
null call	0.04	1.76 (4400.00%)	1.72
null I/O	0.09	2.33 (2588.89%)	2.24
stat	0.56	1.73 (308.93%)	1.17
open clos	1.18	6.53 (553.39%)	5.35
sig inst	0.12	1.15 (958.33%)	1.03
sig hndl	0.71	1.85 (260.56%)	1.14
fork proc	459	561 (122.22%)	102
exec proc	1253	1527 (121.87%)	274
sh proc	3549	4453 (125.47%)	904
slct TCP	2.08	6.82 (327.88%)	4.74
AF UNIX	7.24	39.57 (546.55%)	32.33
UDP	13.33	63.23 (474.34%)	49.90
TCP	16.77	60.50 (360.76%)	43.73
TCP conn	80.33	133.00 (165.57%)	52.67

classified information and are heavily traced by the VMM-based tracing function.

The overhead of `slct TCP`, `AF UNIX`, `UDP`, and `TCP` used to conduct the transmission/reception processing is within the range of 4 to 50 μ s, which is relative performance level of 327.88 to 546.55%. It seems that such overhead is caused by the process used to determine whether the process/socket is managed as by the evaluation described in Section 3.5.4. In addition, the overhead of `TCP conn` is 52.67 μ s, which is the largest value in items related to the network. It is thought that the cause of this overhead is the large number of VM-Exits incurred by issuing a system call between the socket creation and a connection establishment. On the other hand, the relative performance is 165.57%, and the ratio of increased overhead is relatively small compared to the other determined value. This is attributed to the system calls issued between the socket creation and a connection establishment being unrelated to the diffusion of classified information.

In summary, the latency of the basic functions of an OS influenced by the VMM-based tracing function is relatively small. However, the performance of processing with system calls that are related to the diffusion of classified information declines.

Table 3.6 Time and overhead for building bzImage. (s)

	Untraced	Traced (Relative performance)				Overhead	
		Managed file: 0		Managed file: 10		Managed file: 0	Managed file: 10
Real time	579.159	660.566	(114.056%)	675.056	(116.558%)	81.407	95.897
User time	473.940	489.827	(103.352%)	498.350	(105.150%)	15.887	24.410
System time	85.853	114.133	(132.940%)	131.380	(153.029%)	28.280	45.527

Application

Building BzImage

Table 3.6 shows the time and overhead for building bzImage. *Managed file: 0* and *Managed file: 10* in Table 3.6 show the measurements in the case of not registering the management file and of registering the 10 management files. *Overhead* are calculated by following formula: (*measurement in each environment – measurement before the introduction of the function*).

As shown in Table 3.6, the overhead of the VMM-based tracing function is 95.897 s. It is thought that the reasons for this overhead is the fact that building bzImage includes a large number of `read()` and `write()` system calls that have a large overhead, as shown in Table 3.4. Moreover, the overhead ratio of the *System time* is larger than that of the *User time*. The VMM-based tracing function hooks the `SYSRET` of the guest OS for collecting the OS information (e.g., inode number). Due to the `SYSRET` processing on the kernel land, the overhead of the *System time* is large. This also means that the overhead increases depending on the number of system call invocations.

It can be seen that the processing time in the case of registering the 10 management files is larger by about 15 s than that without the management files, as shown in Table 3.6. The VMM-based tracing function audits whether the system call treats the classified file for each system call invocation. To achieve the above audit, the VMM-based tracing function scans the managed file list. This scan causes the lengthening of the processing time. Further, the VMM-based tracing function appends all the files that have the potential to be classified files to the managed file list. Therefore, the false positive will generated. If the managed file list is bloated by the false positive, the processing time will increase. Thus, reducing the false positive is the research task for performance improvement in the future.

On the other hand, the relative performance of the VMM-based tracing function is 116.558%.

Table 3.7 Average response time and overhead of Web server. (ms)

File size	Untraced	Traced (Relative performance)		Overhead	
		Operation of unmanaged target	Operation of managed target	Operation of unmanaged target	Operation of managed target
1 KB	1.732	1.862 (107.506%)	1.868 (107.852%)	0.130	0.136
10 KB	1.783	1.839 (103.141%)	1.853 (103.926%)	0.056	0.070
100 KB	5.916	6.183 (104.513%)	6.371 (107.691%)	0.267	0.455
1,000 KB	48.479	50.965 (105.128%)	51.991 (107.244%)	2.486	3.512

This is very small value as compared to the overhead of system call (maximum 499.10%) and that of microbenchmark (maximum 546.55%).

Web Server

Table 3.7 shows the average response time and overhead of Web server. As shown in Table 3.7, the relative performance of *Operation of unmanaged target* is within the range of 103.141 to 107.506% and that of *Operation of managed target* is within the range of 103.926 to 107.852%. In other words, the performance degradation with the VMM-based tracing function is less than about 10%. This is very small value as compared to the overhead of system call and that of microbenchmark as well as evaluation of building bzImage. Although the overhead percentage of system call and microbenchmark is relatively large, that of total processing time of application is small. Moreover, the overhead in case the *Operation of managed target* is larger than that of *unmanaged target*. This is attributed to additional processing for managed file and process as well as evaluation of system call.

Furthermore, the overhead per HTTP request is within the range of 0.130 to 2.486 μ s in *Operation of unmanaged target* and is within the range of 0.136 to 3.512 μ s in *Operation of managed target*. These overheads may include the overhead of system call and microbenchmark measured in earlier Sections. Although the overhead percentage of system call and microbenchmark is relatively large, that of total processing time of application is small.

In conclusion, overhead generated by introduction of the VMM-based tracing function is relatively large from the viewpoint of system call or microbenchmark level, however, it is relatively small from the viewpoint of application level.

3.6 Conclusion

In this dissertation, we described the design and implementation of the VMM-based tracing function for file operation, child process creation, and IPC using the KVM. The VMM-based tracing function traces the diffusion of classified information and detects the leakage of such information from outside the OS. To trace the diffusion of classified information and detect the leakage of such information, hooking the system call to the guest OS and collecting the required information are necessary. Moreover, by obtaining the system call's return value, the tracing accuracy can be improved. The VMM-based tracing function is implemented without modifying the OS's source code. Therefore, we expect that the VMM-based tracing function can be introduced in various environments. Moreover, it is difficult to attack the function directly, owing to the isolation of the VMM from the OS. Furthermore, even if the kernel version is updated, the VMM-based tracing function will continue to be available, provided that the system call specifications and the data structure remain unchanged. In summary, the VMM-based tracing function resolves the problems of the existing method including the OS-based tracing function and can trace the diffusion of classified information.

We implemented and evaluated the prototype of the VMM-based tracing function. Then, we verified the traceability of the VMM-based tracing function. Moreover, we demonstrated that the VMM-based tracing function can be implemented only by slight addition and localizing the range of modification as compared to the OS-based tracing function. During a performance evaluation, the overhead of the system calls related to the diffusion of classified information was within the range of 1.95 to 49.58 μ s, with a relative performance level of 107.13% to 842.69%, which are relatively large values. On the other hand, the overhead of `getpid()`, which is unrelated to the diffusion of classified information, was 0.0011 μ s, which is a relative performance level of 106.86%, a small value compared to those obtained for system calls related to the diffusion of classified information. In addition, the performance degradation of building `bzImage` is 16.558% and that of web server is less than about 10%. According to this result, we can say that the overhead generated by introduction of the VMM-based tracing function is relatively small from the viewpoint of application level.

IPC is implemented through various methods, for example, a pipe, FIFO, message queue, shared memory, or socket. This dissertation focuses on tracing the socket communication because of its high priority. However, it is also important to trace the other IPCs. Similar to the socket communication, these are conducted through a communication medium. Therefore,

these are also traceable by managing whether their communication medium has classified information based on the proposed method.

In future works, we will implement the proposed VMM-based tracing function for IPC, excluding a socket communication and implement the function for preventing the leakage of classified information. In addition, we will reduce the false positive, reduce the overhead incurred by the VMM-based tracing function, and evaluate its performance.

Chapter 4

Survey and Analysis on ATT&CK Mapping Function of Online Sandbox for Understanding and Efficient Using

4.1 Introduction

Malware plays an important role in cyber attacks, and a large amount of new malware is being discovered every day [103]. To respond to such a large amount of malware, dynamic analysis, which automatically analyzes malware, has become the de facto standard. In addition, online services with dynamic analysis functions have become widespread as online sandboxes, and these are widely use because these do not require construction of an on-premise analysis environment and can be used through a Web interface. One support for analysis is a function to map the malware behavior to each element of the MITRE ATT&CK[®] techniques [104] (hereinafter referred to as “technique”).

The technique represents the attack function of the malware, and by referring to the mapping result, we can grasp the outline of the function of the malware. This function is particularly useful for malware analysts, because it enables identifying the characteristic functions of the malware even when analyzing it manually as well as automating the analysis. Because of its usefulness, the function for mapping malware activities onto techniques has been adopted in online sandboxes. For example, since around 2018, mapping functions have been implemented in JoeSandbox [105] and Hybrid Analysis [106], which have been widely

used for a long time. The same feature has been implemented in Hatching Triage [107], an online sandbox released somewhat later on. Furthermore, the technique mapping function has been introduced into some commercial sandboxes [108, 109], and is expected to become a defacto standard for sandbox functions in the future.

General guidelines for mapping to techniques are given [110]. Detection methods are described in the “Detection” section of each technique. On the other hand, there are many techniques that do not provide specific detection rules or detection thresholds, so the mapping function to techniques in the online sandbox is implementation-dependent. Therefore, the actual situation of the mapping function of ATT&CK in various sandboxes needs to be understood to carry out security operations. However, to the best of our knowledge, no quantitative survey has been conducted on the actual status of this function and the existence of differences among online sandboxes.

Therefore, in this dissertation, we surveyed the online sandboxes with the ATT&CK mapping function. We quantified the differences among the online sandboxes and the differences with other methods such as static analysis and manual reporting. By doing so, we clarified the analysis capability of the current technique mapping function of online sandboxes and its limitations, in order to improve the usability. On the basis of the results of the survey, we also derived best practices for using the technique mapping function.

The contributions of this study are as follows:

- We obtained 26,078 analysis reports and 328,702 technique mapping results from multiple online sandboxes and performed the first quantitative research and analysis on them.
- We analyzed the differences in mapping tendencies of techniques among online sandboxes and discovered that the mapping consistency for the same sample was low, and those for 117 out of 153 techniques were significantly different.
- We compared the mapping results for malware with those for benign files and discovered that 32 techniques had no significant differences in their mapping tendencies. Because these techniques tend to be mapped to benign files, determining if their behavior is truly malicious or not is a high priority.
- For technique mapping, we compared the results with those of static analysis-based methods and manual reports, and discovered that there were differences in the extrac-

tion characteristics of these methods. Specifically, we quantitatively revealed that an online sandbox is not good at extracting tactical techniques outside its context, such as *Reconnaissance* and *Resource Development*. However, we showed that *Initial Access*, which appears to be outside the context of the sandbox, can be partially extracted. Furthermore, we quantitatively revealed that the extractions of techniques that have a specific and mechanically defined detection method are significantly better than those of other methods.

- Based on the survey and analysis conducted during the study, we derived the best practices, such as it is recommended to compare the mapping results with the analysis results of multiple online sandboxes and extraction methods as much as possible, substitute using mapping results for each task for which they are to be used, accounting for the possibility of false positives. We also discussed the effective usage of analysis report.

4.2 Background and Research Questions

4.2.1 Online Sandbox

A sandbox is a dynamic analysis environment in which malware is executed and its behavior is observed. As mentioned earlier, the currently existing amount of malware is enormous and many efforts have been made to improve efficiency through automatic dynamic analysis using sandboxes. For example, dynamic analysis is used to automate the generation of reports [111], the creation of malware detection rules [59,60], and the identification of malware variants by clustering [112]. The results from dynamic analysis in sandboxes are used by analysts for analyzing malware [113].

Online services with dynamic analysis functions are widely used as online sandboxes because they do not require the construction of an on-premise analysis environment and can be used through a Web interface. In addition to conventional commercial sandboxes and the open source cuckoo sandbox [114], online sandboxes such as JoeSandbox [105] and any.run [115] are shown as sandboxes used by analysts [113].

4.2.2 MITRE ATT&CK

MITRE ATT&CK [104], which stands for Adversarial Tactics, Techniques, and Common Knowledge, is a knowledge base/framework that organizes and systematizes cyber attack tactics and techniques by attack lifecycle. ATT&CK is composed of tactics, which represent the goals to be achieved by an attack, and techniques, which are the attack techniques used to achieve the goals. The use of ATT&CK has attracted much attention in recent years because of its potential for various applications, since it enables cyber attacks to be described in a common language. For example, it can be used to simplify the understanding of the overall picture of cyber attacks, to standardize and improve the comprehensiveness of attack methods and detection/countermeasure techniques, and to facilitate information exchange through a common language. Moreover, clarifying attack methods (TTPs: Tactics, Techniques, and Procedures) is an important objective in malware analysis [113], and a survey revealed that analysts use MITRE ATT&CK to organize TTPs [113,116]. Thus, the use of ATT&CK is expected to improve the efficiency of malware analysis.

4.2.3 Problems

As mentioned in Section 4.2.2, while ATT&CK has been utilized in many online sandboxes, there are still many implementation-dependent aspects of associating malware behavior with ATT&CK techniques. For example, *T1071 (Application Layer Protocol)* provides a detection method to analyze network data for uncommon data flows (e.g., a client sending significantly more data than it receives from a server). However, it is difficult to uniquely define *uncommon*; thus, whether the communication is *common* or *uncommon* depends on the threshold to be set and its implementation.

There are also some techniques which are difficult to detect in the online sandbox layer. For example, *T1195 (Supply Chain Compromise)* means that the initial intrusion was caused by a supply chain attack, but it is difficult to detect because it occurs outside the context of the online sandbox analysis.

However, these ATT&CK techniques are difficult to detect because they occur outside the context of the analysis in the online sandboxes. Because the results of the analysis are affected by these features and have the potential to negatively impact the destination of the analysis results, the actual state of the mapping function to the technique in various online

sandboxes needs to be understood to carry out security operations.

4.2.4 Research Questions

On the basis of the aforementioned issues, four RQs (Research Questions) were designed and a survey was conducted.

- **RQ1: Are there differences in ATT&CK mapping capabilities between on-line sandboxes?**

As mentioned in Section 4.2.3, the mapping function of techniques among online sandboxes have some differences. By quantitatively testing this hypothesis, we aim to understand the actual situation of this function.

- **RQ2: Are there techniques that are easy or difficult to extract in online sandboxes?**

Because the technique mapping function in the online sandbox requires mechanical mapping and there are out-of-context attacks, some techniques can be extracted and others cannot. Therefore, we examine this item in order to improve the usability of the technique mapping function in the online sandbox.

- **RQ3: Are there techniques that tend to be mapped to benign files?**

Some techniques, such as the aforementioned *T1071*, require a threshold to determine whether an observed potential attack is truly an attack. Depending on the rule settings, and not only the threshold, it is possible to map ATT&CK techniques even if the behavior is benign. Such incorrect mapping may induce false positives and have negative effects on the analysis results. Thus, it is examined whether any techniques tend to be mapped to benign files, and if this is the case, we try to determine which techniques are likely to be mapped to benign files and those that are not.

- **RQ4: Are there differences in characteristic between other technique detection methods?**

As mentioned in Section 4.2.1 and 4.2.2, technique mapping is effective in security operations and is not just utilized in online sandboxes. For example, there are examples of mapping functions that use static analysis or manual mapping on the basis of various observation results which are published as threat reports. Each of these

mapping methods has its own potential strengths and weaknesses, and there may be differences among them. By understanding these differences and the strengths and weaknesses of each method, we hope to obtain suggestions on which method should be used depending on the situation and analysis target.

4.3 Methodology

4.3.1 Design of Survey

First, to solve RQs 1–3, we collected malware analysis reports from online sandboxes and obtained the mapping results to the ATT&CK technique. To solve RQ4, we also collected static analysis-based analysis results, manually generated threat reports for comparison, and extracted the mapping results to the ATT&CK technique. We then compared the results with those mapped automatically by an online sandbox.

4.3.2 Survey Subjects

In this study, the following online sandbox services with the capability of mapping to technique were selected for the survey.

- JoeSandbox [105]
- Hybrid Analysis [106]
- Hatching Triage [107]

We also selected three threat information sites to collect human written reports related to RQ4.

- MANDIANT [117]
- Cisco Talos [118]
- Trend Micro [119]

Table 4.1 Data overview.

Information source	Number of reports	Number of techniques	
		Unique	Total
JoeSandbox	13,184	143	284,975
Hybrid Analysis	1,012	104	13,351
Hatching Triage	11,882	38	30,376
Total of online sandboxes	26,078	167	328,702
Static analysis (VirusTotal+capa)	3,918	64	19,291
Manual report	50	180	697

Table 4.2 Similarity of MITRE ATT&CK Technique mapping results between sandboxes by formula (1).

Combination			Average	Mean	Max.	min.	Number of reports
JoeSandbox	Hybrid Analysis	Hatching Triage					
✓	✓		0.146	0.143	0.350	0.024	1,125
✓		✓	0.080	0.071	0.500	0.023	11,882
	✓	✓	0.144	0.125	0.500	0.029	1,012
✓	✓	✓	0.042	0.035	0.154	0.019	1,012

These sites were selected as the target of this study because they provide the results of mapping to techniques in tabular form, etc., regarding threat information.

Additionally, we utilized capa [120] (v3.0.2) to obtain the results of static analysis-based analysis. Capa is a tool that takes the binary to be analyzed as the input and outputs the results of static analysis. The output includes the mapping result to technique, and we used this mapping result to compare with the mapping result of other methods.

Note that Intezer Analyze [121], which is a kind of online sandbox, has a mapping function to technique, but the documentation states that it uses capa. Therefore, although Intezer Analyze is an online sandbox, we judged that its technique mapping function is based on static analysis and excluded it from the verification in RQ1 to RQ3.

4.3.3 Dataset

In processing the online sandbox reports, we mainly collected those from JoeSandbox. Specifically, we collected 20,435 analysis reports of malware analyzed during the period of

Table 4.3 Analysis environment for each sandbox.

Analysis Environment	Online sandbox		
	JoeSandbox	Hybrid Analysis	Hatching Triage
Windows 7 (32-bit)	0	400	0
Windows 7 (64-bit)	86	612	3
Windows 10 (64-bit)	926	0	1,007
Windows 11 (64-bit)	0	0	2
Total	1,012	1,012	1,012

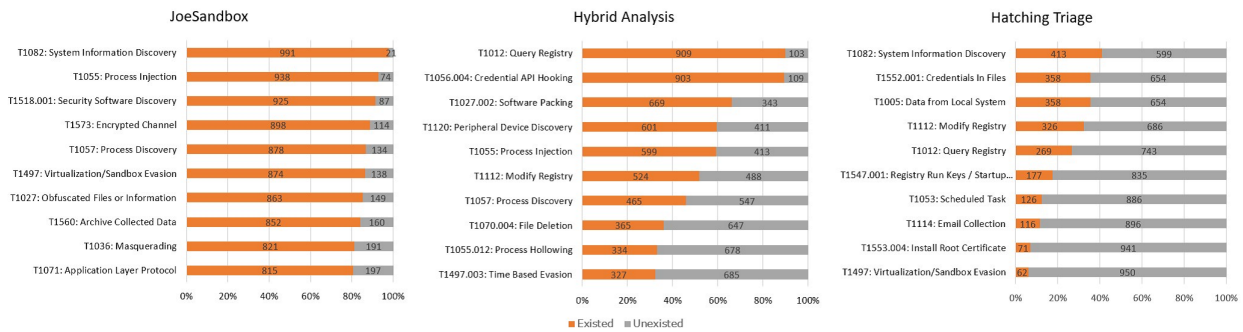


Figure 4.1 Top 10 MITRE ATT&CK Technique for each sandbox.

September 24, 2021 to October 23, 2021. From these reports, we extracted 13,184 malware analysis results, i.e., reports that analyzed files instead of URLs and were judged to be “malicious”, and selected these as the target of our investigation. After that, we obtained the analysis results for the same samples from Hybrid Analysis and Hatching Triage on the basis of the hash values of the 13,184 samples extracted from JoeSandbox. However, not all the analysis reports for all the samples existed in each online sandbox, and only 1,012 out of 13,184 reports existed in Hybrid Analysis and 11,882 in Hatching Triage. The total number of reports was 26,078, and the number of analysis results of the same sample in all sandboxes was 1,012. After that, techniques were extracted from each report to form a dataset. Specifically, JoeSandbox and Hatching Triage extracted techniques by analyzing the structure of the reports, and Hybrid Analysis used techniques provided in csv format.

We selected 50 cases from threat information sites that contained mapping results to the ATT&CK technique and manually extracted the list of techniques summarized at the end of sentences, etc., to form a dataset.

Furthermore, the static analysis-based results were obtained by retrieving actual samples

Table 4.4 Number of observations and presence of significant differences among sandboxes for each MITRE ATT&CK Technique (top 10 observations for each sandbox).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist		
T1082	System Information Discovery	991	21	207	805	413	599	2.29E-285	✓
T1055	Process Injection	938	74	598	414	0	1,012	0	✓
T1518.001	Security Software Discovery	925	87	53	959	3	1,009	0	✓
T1573	Encrypted Channel	898	114	223	789	0	1,012	0	✓
T1057	Process Discovery	878	134	465	547	0	1,012	0	✓
T1497	Virtualization/Sandbox Evasion	874	138	241	771	62	950	0	✓
T1027	Obfuscated Files or Information	863	149	7	1,005	0	1,012	0	✓
T1560	Archive Collected Data	852	160	3	1,009	0	1,012	0	✓
T1036	Masquerading	821	191	89	923	0	1,012	0	✓
T1071	Application Layer Protocol	815	197	0	1,012	0	1,012	0	✓
T1012	Query Registry	289	723	909	103	269	743	2.94E-228	✓
T1056.004	Credential API Hooking	57	955	902	110	0	1,012	0	✓
T1027.002	Software Packing	769	243	669	343	0	1,012	1.18E-301	✓
T1120	Peripheral Device Discovery	9	1,003	601	411	38	974	1.95E-285	✓
T1112	Modify Registry	39	973	524	488	326	686	5.78E-124	✓
T1070.004	File Deletion	133	879	365	647	9	1,003	1.81E-101	✓
T1055.012	Process Hollowing	0	1,012	333	679	0	1,012	3.66E-163	✓
T1497.003	Time Based Evasion	0	1,012	326	686	0	1,012	2.45E-159	✓
T1552.001	Credentials In Files	58	954	2	1,010	358	654	3.48E-133	✓
T1005	Data from Local System	453	559	84	928	358	654	1.66E-76	✓
T1547.001	Registry Run Keys / Startup Folder	208	804	162	850	177	835	0.025182647	✓
T1053	Scheduled Task/Job	183	829	115	897	126	886	1.74E-05	✓
T1114	Email Collection	322	690	122	890	116	896	6.13E-40	✓
T1553.004	Install Root Certificate	2	1,010	0	1,012	71	941	1.29E-30	✓

from VirusTotal on the basis of the hash values of 13,184 malware samples obtained from JoeSandbox and analyzing each sample with capa. However, only 11,973 samples actually existed in VirusTotal and could be obtained. Because capa supports only some file formats such as PE and ELF formats, and because obfuscated specimens are excluded from the analysis, static analysis was successful and techniques were extracted as datasets for 3,918 samples. These data are summarized in the Table 4.1.

Here, MITRE ATT&CK is basically updated every six months, and the names of the techniques may change or be consolidated. To reduce the impact of these version differences on the analysis, we used the datasheet [122], which summarizes the correspondence of each technique with its predecessors, to assign names to the MITRE ATT&CK Technique v9. For example, the technique ID and its name are updated from *T1045 (Software Packing)* to *T1027.002 (Obfuscated Files or Information: Software Packing)*. The reason for the

unification to v9 is that as of December 2021, the relevant datasheet is compatible with v9.

4.4 Results

4.4.1 Overview of Survey

In this section, we analyze the mapping results to ATT&CK collected from each online sandbox to derive the actual situation and best practices for its use.

First, we compare the mapping results of each sandbox to the same sample and resolve RQ1. Second, RQ2 is solved by measuring the coverage of all mapping results collected for all techniques. We also solve RQ3 by comparing the results of technique mappings to benign files with those to malware, and deriving the technique that tends to be mapped to both. Finally, we collect static analysis-based analysis results and manually written threat reports, and compare the ATT&CK mapping results performed by each of them with the results automatically mapped by the online sandbox to solve RQ4.

To solve the RQs, we used a statistical test method. The Yates' chi-square test was used as the test method because there were a few items with a small number of occurrences in all the test targets. The significance level was set at 0.05.

4.4.2 RQ1: Are There Differences in ATT&CK Mapping Capabilities between Online Sandboxes?

To answer this RQ, we utilized the reports that existed for the same sample in each sandbox. To measure the degree of consistency of the techniques in each sandbox, the set similarity of the techniques of each sample was calculated using a formula inspired by the Jaccard coefficient in (1) below.

$$Sim(S_1, S_2, \dots, S_n) = \frac{|S_1 \cap S_2 \dots \cap S_n|}{|S_1 \cup S_2 \dots \cup S_n|} \quad (4.1)$$

The calculation results are shown in the Table 4.2. The analysis environment for each analysis sandbox is shown in the Table 4.3. Each environment includes a web browser, PDF viewer, Office software, etc. The mean values of the Jaccard coefficients were 0.146, 0.080, and 0.144 between the two sandboxes, and 0.042 between the three sandboxes, indicating

a low degree of consistency. The top 10 techniques with the highest number among 1,012 cases in common for all sandboxes are shown in the Fig. 4.1. Although all results are mapped to the same samples, the top 10 techniques and their percentages are all different. For example, *T1082 (System Information Discovery)* in JoeSandbox is mapped to 991 out of 1,012 specimens, which is almost all samples, while Hatching Triage is mapped to 413 samples, although these are in the same position. It can be confirmed that Hybrid Analysis is not even in the top 10.

A crosstabulation table was created for each technique, and a chi-square test was conducted to verify whether there was a significant difference between sandboxes for the 153 techniques detected in any of the sandboxes. As a result, we found that 36 techniques were not significantly different from each other (i.e., similar in all sandboxes), while 117 techniques were significantly different from each other. The results of the test for all 153 techniques are shown in the Table B.1 in Appendix B.1. Table 4.4 shows the number of observations in each sandbox, the p-value of the chi-square test, and the presence or absence of a significant difference when the significance level is set to 0.05 for each of the 1,012 samples in all sandboxes. The table shows that there is a significant difference in the number of observations among the top 10 techniques in each sandbox. This indicates that there are differences in the ATT&CK mapping functions of the sandboxes surveyed in this study, and that there are techniques that are suitable for extraction.

In the above comparison, the v8 and earlier techniques were renamed as the v9 techniques as described in Section 5.4.2. Table 4.5 shows the v8 and earlier techniques used in each sandbox extracted during this naming process. First, in the JoeSandbox, all techniques except *T1064 (Scripting)* were v9 as far as we could confirm. Although *T1064* is deprecated, it is still available on the ATT&CK page as of December 2021, which means that JoeSandbox's technique mapping function is highly maintainable. On the other hand, there are 21 and 15 obsolete techniques remaining in Hybrid Analysis and Hatching Triage, respectively. These are not necessarily undesirable because they are useful in terms of consistency with the mapping results before the revision in the same sandbox. However, if the mapping results are to be compared with those of other sandboxes or other methods, or if the mapping results are to be used in reports, etc., it is assumed that adverse effects due to the difference in versions may occur, and therefore, it is necessary to perform name matching, etc.

In conclusion, the ATT&CK mapping function can be said to differ among the online

Table 4.5 Usage of the deprecated MITRE ATT&CK Technique per sandbox.

#	Deprecated TID	Deprecated technique	Updated TID	Updated technique	JoeSandbox	Hybrid Analysis	Hatching Triage
1	T1215	Kernel Modules and Extensions	T1547.006	Kernel Modules and Extensions		✓	
2	T1179	Hooking	T1056.004	Credential API Hooking		✓	
3	T1168	Local Job Scheduling	T1053	Scheduled Task/Job		✓	
4	T1158	Hidden Files and Directories	T1564.001	Hidden Files and Directories			✓
5	T1130	Install Root Certificate	T1553.004	Install Root Certificate			✓
6	T1116	Code Signing	T1553.002	Code Signing		✓	
7	T1107	File Deletion	T1070.004	File Deletion		✓	✓
8	T1094	Custom Command and Control Protocol	T1095	NonApplication Layer Protocol		✓	
9	T1089	Disabling Security Tools	T1562.001	Disable or Modify Tools		✓	✓
10	T1088	Bypass User Account Control	T1548.002	Bypass User Access Control		✓	✓
11	T1086	PowerShell	T1059.001	PowerShell		✓	
12	T1085	Rundll32	T1218.011	Rundll32		✓	
13	T1081	Credentials in Files	T1552.001	Credentials In Files			✓
14	T1076	Remote Desktop Protocol	T1021.001	Remote Desktop Protocol		✓	✓
15	T1067	Bootkit	T1542.003	Bootkit			✓
16	T1065	Uncommonly Used Port	T1571	NonStandard Port		✓	
17	T1064	Scripting	N/A	N/A	✓	✓	✓
18	T1063	Security Software Discovery	T1518.001	Security Software Discovery		✓	✓
19	T1060	Registry Run Keys/Startup Folder	T1547.001	Registry Run Keys/Startup Folder		✓	✓
20	T1050	New Service	T1543.003	Windows Service		✓	✓
21	T1045	Software Packing	T1027.002	Software Packing		✓	
22	T1044	File System Permissions Weakness	T1574.010	Services File Permissions Weakness		✓	
23	T1043	Commonly Used Port	N/A	N/A		✓	
24	T1042	Change Default File Association	T1546.001	Change Default File Association			✓
25	T1035	Service Execution	T1569.002	Service Execution		✓	
26	T1031	Modify Existing Service	T1543.003	Windows Service			✓
27	T1004	Winlogon Helper DLL	T1547.004	Winlogon Helper DLL			✓
28	T1002	Data Compressed	T1560	Archive Collected Data		✓	
Total					1	21	15

sandboxes.

4.4.3 RQ2: Are There Techniques that are Easy or Difficult to extract in Online Sandboxes?

To answer this RQ, we utilized 26,078 reports from all sandboxes. First, we extracted the techniques from all the reports and performed a chi-square test to confirm that there was a significant difference between the extracted techniques. Then we calculated the number of techniques that existed in more than one case and those that did not. Figure 4.2 shows a visualization of the techniques that existed in more than one case using ATT&CK Navigator [123] only at the granularity of techniques (not including sub-techniques). Among the total of 568 techniques, only 175 (29.40%) were found to exist, while the remaining 70.60% did not. Particularly noteworthy were *Reconnaissance* and *Resource Development*, which are the preliminary stages of an attack, both of which had zero cases. These are techniques applied

before the malware is executed and it was confirmed that it is difficult to extract techniques with the online sandbox function that extracts techniques from the analysis log after the malware is basically executed.

Table 4.6 shows the values aggregated for each tactic. *Excluding Reconnaissance* and *Resource Development*, the coverage rates for *Exfiltration* (11.76%) and *Impact* (23.08%) are low.

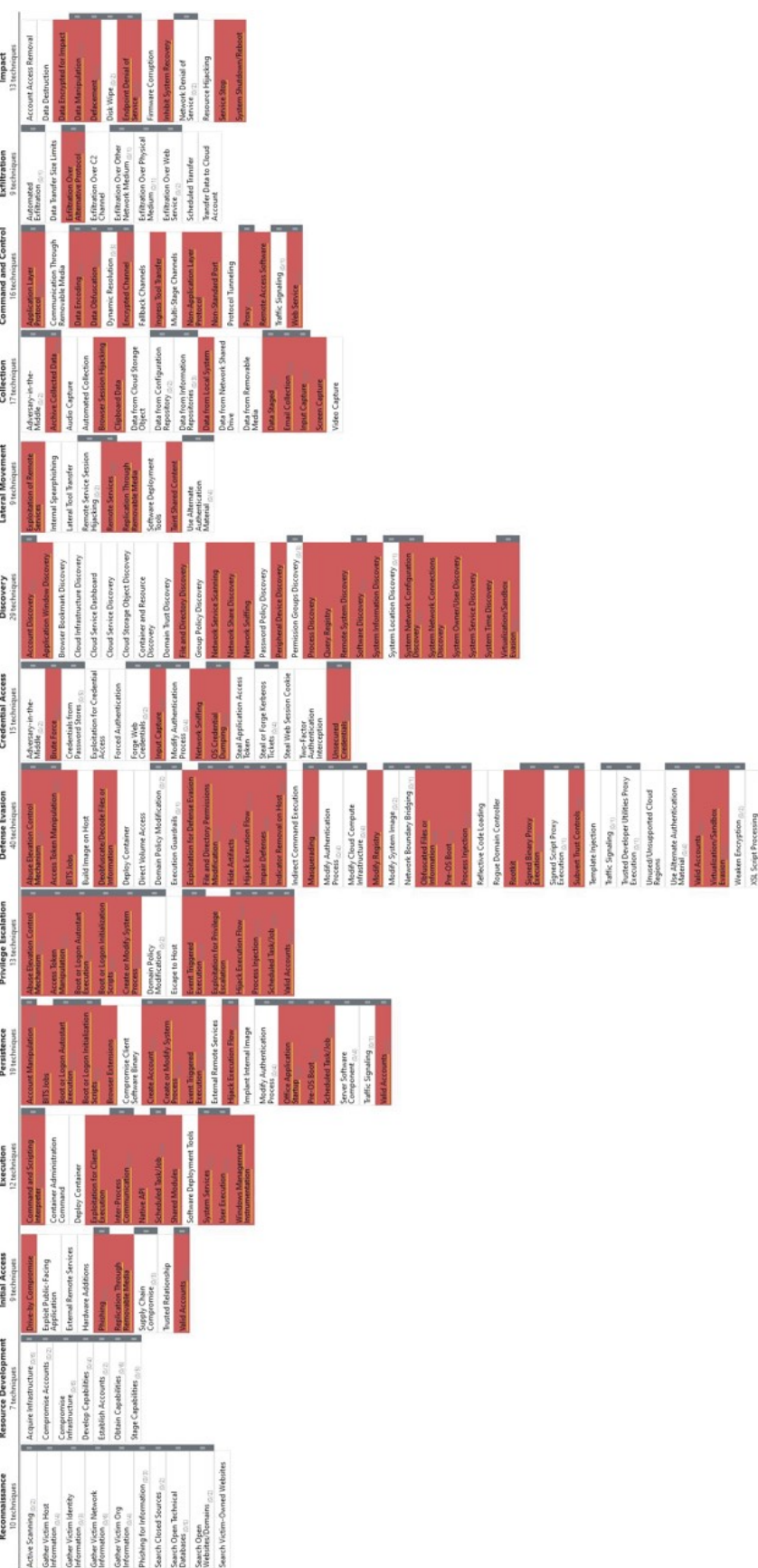


Figure 4.2 More than one MITRE ATT&CK Technique was found in the sandbox analysis results.

Table 4.6 Number and percentage of each MITRE ATT&CK Tactic present.

Tactic	Number of existing techniques	Total number of techniques	ratio (%)
Reconnaissance	0	41	0.00
Resource Development	0	32	0.00
Initial Access	4	15	26.67
Execution	15	44	34.09
Persistence	28	83	33.73
Privilege Escalation	25	69	36.23
Defense Evasion	42	121	34.71
Discovery	18	35	51.43
Lateral Movement	7	25	28.00
Collection	7	27	25.93
Command and Control	13	33	39.39
Exfiltration	2	17	11.76
Impact	6	26	23.08
Total	167	568	29.40

This may be partly because these techniques are related to data removal and system destruction, which are outside the context of online sandboxes and include a relatively high level of abstraction. Note that although *Initial Access* appears to be undetectable because it is intuitively outside the context of the online sandbox, it was partially detected (4/15). We confirmed that *Initial Access* was associated with, for example, a PDF file sample. For *Drive-by Compromise* among *Initial Access*, the URL included in the PDF file was the starting point of *Drive-by Compromise*, and there were several cases wherein the infection started from this point. The online sandbox identifies it by finding iframes.

From these results, we can confirm that in current online sandboxes, there are differences in the extraction tendencies for each technique and tactic. This suggests that some techniques are relatively easy to extract, and those that are currently extractable account for most of them. Furthermore, it infers that some techniques are potentially difficult to extract.

4.4.4 RQ3: Are There Techniques that Tend to be Mapped to Benign Files?

As mentioned in Section 5.4.2, the reports obtained from JoeSandbox include non-malicious files. Therefore, for this RQ, we utilized the reports obtained from JoeSandbox for benign

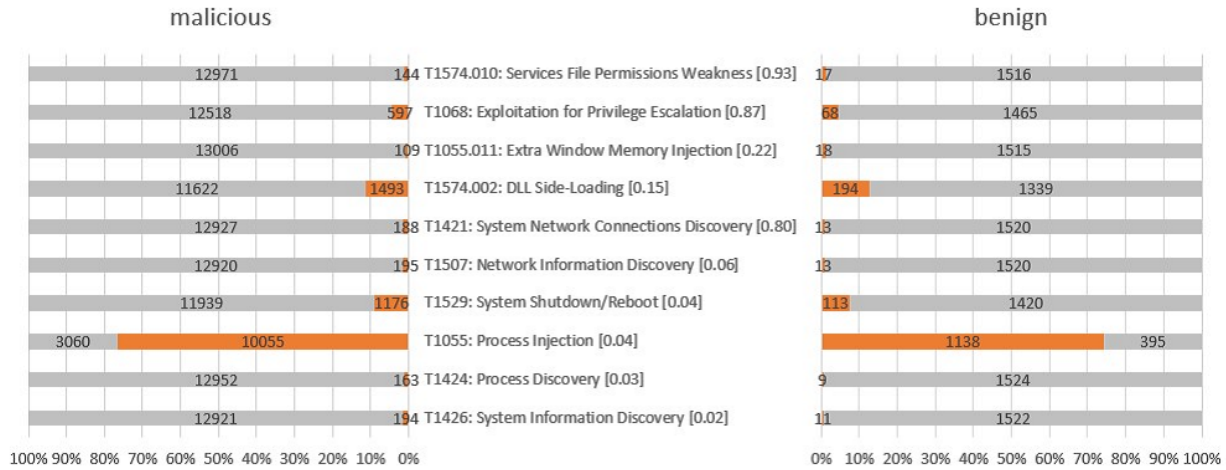


Figure 4.3 MITRE ATT&CK Technique for the top 10 p-values

files and for malware. Specifically, we compared 1,533 reports labelled as “clean” with 13,184 reports on malware. For each technique, we tested whether there was a significant difference between benign files and malware, and extracted them without a significant difference.

As a result, it was discovered that 32 techniques were not significantly different. The butterfly chart of the techniques with high p-values is shown in Fig. 4.3. For design reasons, techniques with less than 100 occurrences are omitted from the figure, and the values in square brackets denote the p-values. Figure 4.3 infers that all the techniques are present in a similar percentage for both benign files and malware, and it should be verified whether these techniques are truly related to malicious activity. The butterfly charts of the techniques with low p-values are shown in Fig. 4.4, wherein it is indicated that these techniques have high true positives. The number of observations and test results for all the techniques are shown in Table B.2 presented in Appendix B.1.

Techniques such as *T1027.002 Software Packing*, *T1018 Remote Service Discovery*, and *T1003 OS Credential Discovery*, which can be expressed by the binary values of “executed” or “not executed” and are not easily found in benign files, tend to have high true positives. On the other hand, behaviors such as *T1447 Delete Device Data* and *T1426 Process Injection*, which are easily performed even in benign files and can be benign or malicious depending on the context, are difficult to definitively distinguish by means of rules and tend to cause false positives.

In summary, some techniques are prone to be assigned not only to malwares but also to

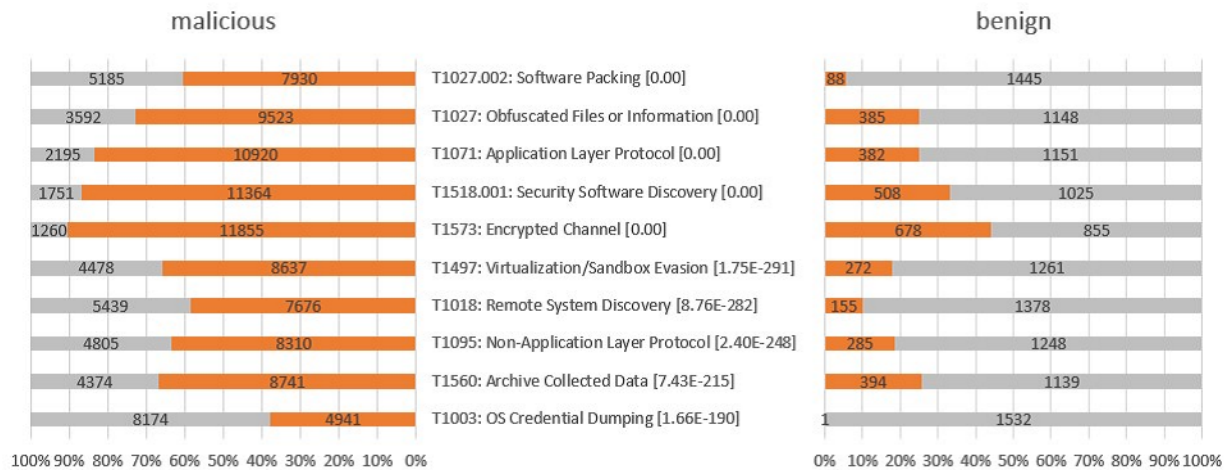


Figure 4.4 MITRE ATT&CK Technique for the lower 10 p-values

Table 4.7 Extraction trend of MITRE ATT&CK Technique by each method.

Combination			Number	Ratio (%)
Online sandbox	Static analysis	Manual report		
✓			25	11.91
✓	✓		2	0.95
✓		✓	54	25.71
	✓		3	1.43
	✓	✓	2	0.95
		✓	86	40.95
✓	✓	✓	38	18.10
Total			210	100.00

benign files.



4.4.5 RQ4: Are There Differences in Characteristic between Other Technique Detection Methods?

To answer this RQ, we utilized 26,078 reports from all online sandboxes, 50 manual reports, and 3,918 static analysis results extracted by capa. In all of the reports, we counted the number of techniques that were found only in each method and the techniques that were found in multiple methods. The results of this analysis are shown in Fig. 4.5 and Table 4.7.

The number of techniques confirmed by all the methods was 38, which is only 18.10% of the total techniques confirmed. On the other hand, some techniques were confirmed only by specific methods. Techniques of 54.29% in total were confirmed; 3 (1.43%) by static analysis only, 25 (11.91%) by online sandbox and 86 (40.95%) by manual report. First, it can be seen that the manual report covers techniques that are difficult to extract with the online sandbox and static analysis, focusing on the techniques of Reconnaissance and Resource Development. Furthermore, *T1040 (Network Sniffing)*, *T1091 (Replication Through Removable Media)*, *T1137 (Office Application Startup)*, and *T1197 (BITS Jobs)* etc. were confirmed only in the online sandbox. The common features of these techniques are that the detection methods are specifically described in the “Detection” section of each technique, such as executing a specific API, executing a specific command, modifying a specific registry, etc., and that these can be detected mechanically. These behaviors are likely to be manifested by actually executing the malware, and it is inferred that they are detected in online sandboxes. Although these features are difficult to detect by static analysis, these can potentially be detected manually. However, we believe that this result was obtained because it is more likely to be observed in the online sandbox which can be executed mechanically and the number of observations can be scaled.

To verify the RQ4 quantitatively, a chi-square test was conducted on the techniques confirmed by multiple methods, between two methods for those confirmed by two methods, and between all combinations of methods ($_3C_2=3$ methods) for those confirmed by three methods, to verify the significant difference between methods for each technique. As a result, out of 193 combinations tested, 141 combinations had significant differences. Of these, a selection of techniques including those with significant differences is shown in Table 4.8.

Table 4.8 Technique observed in multiple methods and presence/absence of significant differences between methods (excerpt).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		Combination	p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist			
T1497	Virtualization/Sandbox Evasion	9,577	16,501	2	3,916	4	46	(all) sandbox+static	0	✓
T1497	Virtualization/Sandbox Evasion	9,577	16,501	2	3,916	4	46	(all) sandbox+report	5.99E-06	✓
T1497	Virtualization/Sandbox Evasion	9,577	16,501	2	3,916	4	46	(all) static+report	4.36E-36	✓
T1027.002	Software Packing	8,649	17,429	4	3,914	2	48	(all) sandbox+static	0	✓
T1027.002	Software Packing	8,649	17,429	4	3,914	2	48	(all) sandbox+report	0.000175584	✓
T1027.002	Software Packing	8,649	17,429	4	3,914	2	48	(all) static+report	1.82E-07	✓
T1027	Obfuscated Files or Information	9,530	16,548	1,412	2,506	15	35	(all) sandbox+static	0.551849477	-
T1027	Obfuscated Files or Information	9,530	16,548	1,412	2,506	15	35	(all) sandbox+report	0	✓
T1027	Obfuscated Files or Information	9,530	16,548	1,412	2,506	15	35	(all) static+report	0.46179638	-
T1518.001	Security Software Discovery	11,428	14,650	3	3,915	2	48	(all) sandbox+static	0	✓
T1518.001	Security Software Discovery	11,428	14,650	3	3,915	2	48	(all) sandbox+report	1.07E-07	✓
T1518.001	Security Software Discovery	11,428	14,650	3	3,915	2	48	(all) static+report	8.17E-09	✓
T1057	Process Discovery	9,569	16,509	99	3,819	7	43	(all) sandbox+static	0	✓
T1057	Process Discovery	9,569	16,509	99	3,819	7	43	(all) sandbox+report	8.45E-16	✓
T1057	Process Discovery	9,569	16,509	99	3,819	7	43	(all) static+report	5.16E-06	✓
T1082	System Information Discovery	15,879	10,199	2,416	1,502	11	39	(all) sandbox+static	0.363771896	-
T1082	System Information Discovery	15,879	10,199	2,416	1,502	11	39	(all) sandbox+report	3.48E-300	✓
T1082	System Information Discovery	15,879	10,199	2,416	1,502	11	39	(all) static+report	2.51E-08	✓
T1569.002	Service Execution	858	25,220	125	3,793	5	45	(all) sandbox+static	0.78040016	-
T1569.002	Service Execution	858	25,220	125	3,793	5	45	(all) sandbox+report	0	✓
T1569.002	Service Execution	858	25,220	125	3,793	5	45	(all) static+report	0.022133283	✓
T1083	File and Directory Discovery	6,818	19,260	1,748	2,170	12	38	(all) sandbox+static	1.11E-125	✓
T1083	File and Directory Discovery	6,818	19,260	1,748	2,170	12	38	(all) sandbox+report	0	✓
T1083	File and Directory Discovery	6,818	19,260	1,748	2,170	12	38	(all) static+report	0.005565762	✓
T1012	Query Registry	7,460	18,618	724	3,194	4	46	(all) sandbox+static	4.45E-40	✓
T1012	Query Registry	7,460	18,618	724	3,194	4	46	(all) sandbox+report	0	✓
T1012	Query Registry	7,460	18,618	724	3,194	4	46	(all) static+report	0.085716776	-
T1033	System Owner/User Discovery	2,845	23,233	201	3,717	5	45	(all) sandbox+static	8.13E-29	✓
T1033	System Owner/User Discovery	2,845	23,233	201	3,717	5	45	(all) sandbox+report	4.35E-260	✓
T1033	System Owner/User Discovery	2,845	23,233	201	3,717	5	45	(all) static+report	0.221871132	-
T1115	Clipboard Data	1,955	24,123	238	3,680	1	49	(all) sandbox+static	0.001601174	✓
T1115	Clipboard Data	1,955	24,123	238	3,680	1	49	(all) sandbox+report	0	✓
T1115	Clipboard Data	1,955	24,123	238	3,680	1	49	(all) static+report	0.365872328	-
T1059	Command and Scripting Interpreter	3,122	22,956	1,801	2,117	11	39	(all) sandbox+static	0	✓
T1059	Command and Scripting Interpreter	3,122	22,956	1,801	2,117	11	39	(all) sandbox+report	0	✓
T1059	Command and Scripting Interpreter	3,122	22,956	1,801	2,117	11	39	(all) static+report	0.001203964	✓
T1113	Screen Capture	664	25,414	403	3,515	3	47	(all) sandbox+static	7.12E-131	✓
T1113	Screen Capture	664	25,414	403	3,515	3	47	(all) sandbox+report	0	✓
T1113	Screen Capture	664	25,414	403	3,515	3	47	(all) static+report	0.447946249	-
T1222	File and Directory Permissions Modification	628	25,450	237	3,681	1	49	(all) sandbox+static	1.17E-36	✓
T1222	File and Directory Permissions Modification	628	25,450	237	3,681	1	49	(all) sandbox+report	0	✓
T1222	File and Directory Permissions Modification	628	25,450	237	3,681	1	49	(all) static+report	0.368942641	-
T1129	Shared Modules	920	25,158	3,392	526	1	49	(all) sandbox+static	0	✓
T1129	Shared Modules	920	25,158	3,392	526	1	49	(all) sandbox+report	0	✓
T1129	Shared Modules	920	25,158	3,392	526	1	49	(all) static+report	1.84E-62	✓
T1564.003	Hidden Window	26	26,052	516	3,402	0	50	sandbox+static	0	✓
T1135	Network Share Discovery	21	26,057	21	3,897	3	47	(all) sandbox+static	6.00E-12	✓
T1135	Network Share Discovery	21	26,057	21	3,897	3	47	(all) sandbox+report	0	✓
T1135	Network Share Discovery	21	26,057	21	3,897	3	47	(all) static+report	5.49E-05	✓
T1489	Service Stop	22	26,056	25	3,893	7	43	(all) sandbox+static	1.81E-15	✓
T1489	Service Stop	22	26,056	25	3,893	7	43	(all) sandbox+report	0	✓
T1489	Service Stop	22	26,056	25	3,893	7	43	(all) static+report	2.97E-22	✓
T1402	Broadcast Receivers	1	26,077	0	3,918	5	45	sandbox+report	0	✓
T1566.001	Spearphishing Attachment	3	26,075	0	3,918	4	46	sandbox+report	8.54E-200	✓
T1560.002	Archive via Library	3	26,075	9	3,909	1	49	(all) sandbox+static	2.85E-09	✓
T1560.002	Archive via Library	3	26,075	9	3,909	1	49	(all) sandbox+report	0	✓
T1560.002	Archive via Library	3	26,075	9	3,909	1	49	(all) static+report	0.288413195	-
T1056.001	Keylogging	4	26,074	532	3,386	1	49	(all) sandbox+static	0	✓
T1056.001	Keylogging	4	26,074	532	3,386	1	49	(all) sandbox+report	0	✓
T1056.001	Keylogging	4	26,074	532	3,386	1	49	(all) static+report	0.029476232	✓

For example, although *T1566.001 (Spearphishing Attachment)* was found in both the online sandbox and the manual report, it is basically outside the context of the online sandbox, so intuitively it is easier to detect in the manual report. In fact, it was found in a small number of cases (3 out of 26,075) in the online sandbox, while it was found in 4 out of 46 cases in the manual report. The results of both tests are “significantly different”, indicating that the detection is significant in the manual reports, as assumed.

Therefore, it can be said that the tendency to extract techniques differs depending on the extraction method. The details of the test results can be found in Table B.3 in Appendix B.1.

4.5 Discussion

4.5.1 Best Practice

As shown in RQ1, there are differences in the ATT&CK mapping function among online sandboxes. RQ4 shows that differences can also occur depending on the extraction method. Therefore, it is recommended to compare the analysis and mapping results of multiple online sandboxes and extraction methods as much as possible and use these in a way so that these complement each other.

Moreover, as described in RQ2–4, some techniques are difficult to extract mechanically via the online sandbox and conversely, some techniques are prone to be false positives. Particularly, as shown in RQ3, some ATT&CK techniques tend to be mapped to benign files. These ATT&CK techniques are defined as techniques used in attacks and should not be mapped to the behavior of benign files. As a side effect of the emphasis on coverage, the mapping of ATT&CK techniques with benign files can result in false positives and should be handled cautiously. By understanding the characteristics of each technique, those that are prone to false positives can be more effectively used, for example, by manually confirming their authenticity, even if they are automatically mapped. It would also be effective to change the way the technique mapping function is used based on the task to be performed. For example, if a researcher wants to comprehend the bigger picture of an attack, completely discarding false positives may have negative effects such as making it difficult to understand the flow of the attack. In such a cases, false positives can be allowed to some extent, and such techniques can be presented with a message stating that the technique has a high number

of false positives, or the log of the technique mapping can be presented as well, and the final judgment can be left to the analyst. In contrast, for a task that requires true positives such as creating detection rules along with mapping results, techniques with high false positives can be rejected.

However, collecting several reports for a single sample is not always desirable from the viewpoint of efficiency. As mentioned in Section 4.2, there are differences in the ATT&CK mapping function; hence, it is considered that efficient analysis can be achieved by collecting at least two reports, manually verifying the authenticity of only those techniques that can be easily mapped to benign files, focusing only on the more important techniques [124] among the extracted ones, and so on.

As shown in the section on RQ1, there are cases wherein the mapping is done on an older version of the technique. This may be because the mapping was done before technique revision, or the mapping function does not support the latest techniques. However, it is crucial to identify whether the data are mapped to the latest version of the technique and read the data accordingly.

4.5.2 Limitation

This study has some limitations. First, the reports collected in this study are primarily those analyzed by JoeSandbox from September 24 to October 23, 2021 and do not include all malware analysis results. Next, there is evasive malware that detects the analysis environment and then avoids malicious behavior. Therefore, even if the samples were identical, these do not always behave maliciously in all sandboxes. Even if these exhibit malicious behavior it is not always identical. In fact, as presented in the Table 4.3, different versions of the OS were used among the sandboxes in some cases and this possibly affected the analysis results. However, it was confirmed that in several cases, the samples common to all sandboxes were judged as “malicious” or assigned a high maliciousness level by the judgment mechanism of each sandbox. If evasive malware is mostly found in a particular sandbox, the number of “malicious” samples in that sandbox should be high, whereas the number of “benign” samples in another sandbox should be high. Therefore, it is unlikely that the ATT&CK mapping function would have been different in one sandbox, but not in another owing to detection of the analysis environment or other accidental factors. However, it is possible that there are some samples that behave maliciously in all sandboxes but change

their behavior significantly to confuse the analyst. A limitation of this study is that the presence of such samples was not considered.

In the RQ3 survey, we found that *Exfiltration* and *Impact*, which are the latter stages of malware behavior, were less common. There is malware that bypasses the sandbox and malware that finishes its attack when the C2 server is closed. One reason for this may be that the more advanced the tactics are, the more difficult it is for the malware to perform the technique that corresponds to the tactics. This is a factor that depends only on the detection evasion function of malware, not on the ease of extracting the technique and may appear as noise in this study. Additionally, the collection of benign files is difficult except for JoeSandbox, and as a result, the verification of RQ3 is limited to the JoeSandbox results only.

Manual reports may also contain larger sample errors, since the absolute number of such reports is smaller than that of the online sandbox analysis reports. There are reports that there are omissions in the technique mentioned in the report [125], which may also have an impact. In addition, the granularity of the targets of online sandboxes and static analysis is different from that of publicly available manually written reports, as most of them target entire attack campaigns or threats, while online sandboxes and static analysis target a single malware sample. This difference in the granularity of the target may have affected the results of the survey described in this dissertation.

Because the number of online sandboxes that we covered in this study was three, the results described in this dissertation may not fully include the nature of online sandboxes as a whole. For example, SandPrint [54], which investigated the fingerprinting potential of online sandboxes, covered 20 services. One reason for the small number of surveyed services is that not all sandboxes are equipped with the technique mapping function, which is the subject of this dissertation's survey.

In this dissertation, we have tried to keep the number of survey targets as large as possible in order to control each limitation.

4.5.3 Research Ethics

In this study, when collecting analysis reports of malware, a certain interval was set for each access when information was obtained from the same site. By applying this measure, the load on each service was reduced, and the survey was conducted.

4.6 Conclusion

In this study we investigate the function for mapping malware analysis results to the relevant ATT&CK techniques in three online sandboxes.

Analysis of survey results reveals that the mapping characteristics differ among the sandboxes. We also compared the results with those of static analysis-based techniques and manually written reports, and showed that there were differences in the mapping tendencies among the techniques. Specifically, we quantitatively revealed that the online sandbox is not good at extracting tactical techniques outside the context of the sandbox. On the other hand, the online sandbox is significantly better than other methods at extracting techniques where the detection method is specific and mechanically defined.

We can therefore infer that malware analysis can be performed more efficiently and reliably by being aware of these factors when using the online sandbox. For example, best practices may include it is desirable to compare the mapping results with the analysis results of multiple online sandboxes and extraction methods as much as possible, and to use them in a way that complements each other, or to use the mapping results in different ways for different tasks, considering the possibility of false positives.

Future work includes expanding the scope of the survey and investigating more efficient ways to use the technique mapping function on the basis of the survey results.

Chapter 5

Automatic URL Signature Construction and Impact Assessment

5.1 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 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 dissertation, 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.

5.2 Background

5.2.1 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 example, *Emotet* uses HTTP Communication to upload files on infected terminals to an external server [126]. The malware used by the attack group *Lazarus* is known to attempt to download attack modules and receive attack commands using http communication [127]. It has also been shown that *xxmm* can receive attack commands from outside using HTTP(s) communication and upload files on infected terminals [128]. 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 SOC tasks is to create and apply signatures to block such communications. Below is the flow of this work (Figure 5.1).

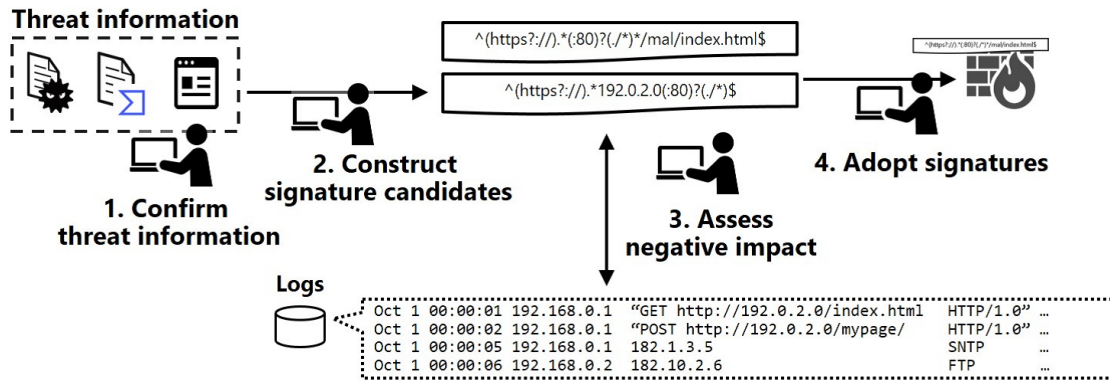


Figure 5.1 Example of operation flow for developing signatures and impact assessment.

- (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.

5.2.2 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.

5.3 Automatic signature generation and impact assessment system

5.3.1 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.

5.3.2 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. 5.2. First, the threat information to make the signature is acquired. The malware analysis result obtained from VT (VirusTotal) is used 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.

5.3.3 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 op-

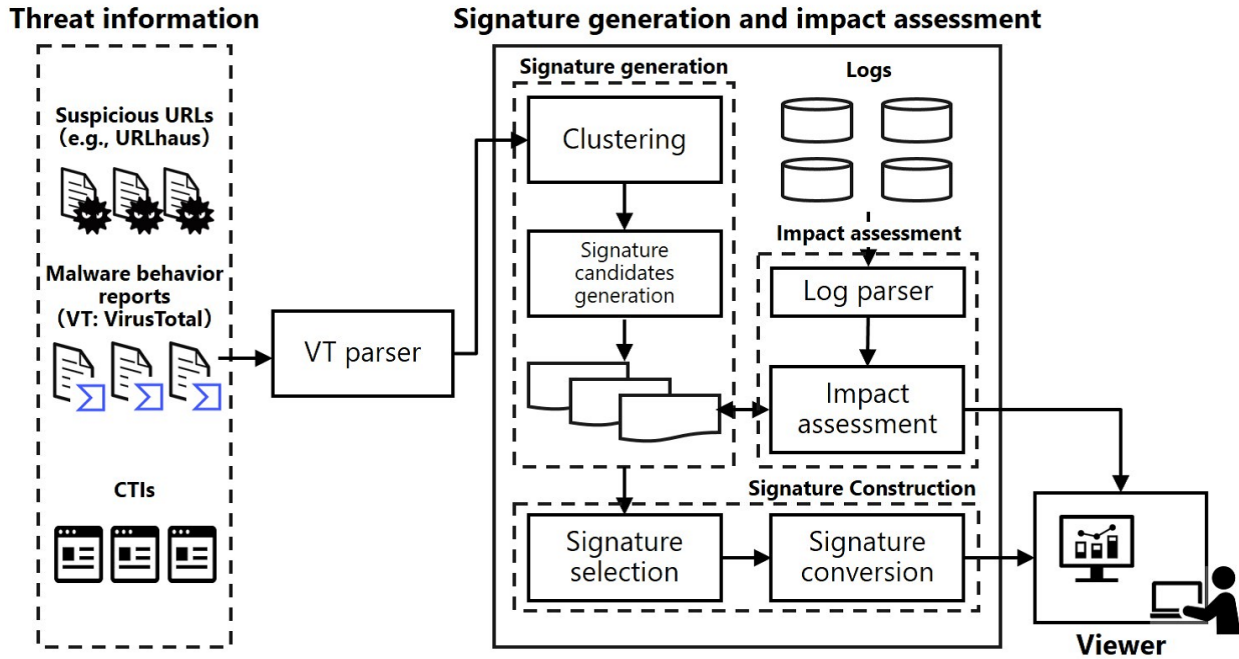


Figure 5.2 Overview of SIGMA.

erationally feasible range. Therefore, in our method, as shown in Fig. 5.3, 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 5.1, following previous studies [58]. 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.

The second step, hierarchical clustering, is performed on each of the previously created clusters. Hierarchical clustering is performed on the URLs in each cluster, using the edit distance between URL strings. In this process, inspired by [129], 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

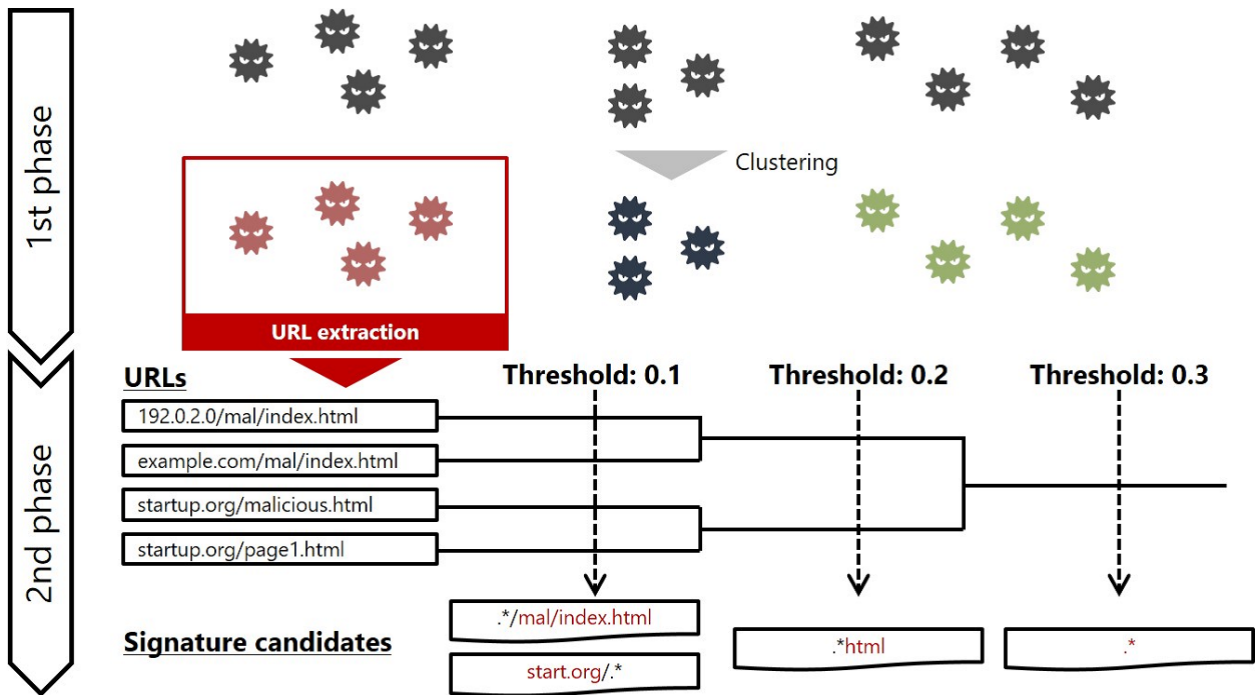


Figure 5.3 Overview of signature generation.

avoidance by reducing numbers, base64, etc. in the signatures to regular expressions using the method in [130] (e.g., detecting numbers and converting them into expressions such as [0–9]).

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 ranking (Alexa Rank, etc.) as normal. The communication to be removed by this

Table 5.1 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

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 [131,132]. 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.

5.3.4 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. 5.4). In this study, we used the access log of the forward proxy *squid* [133] 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

5.3.5 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

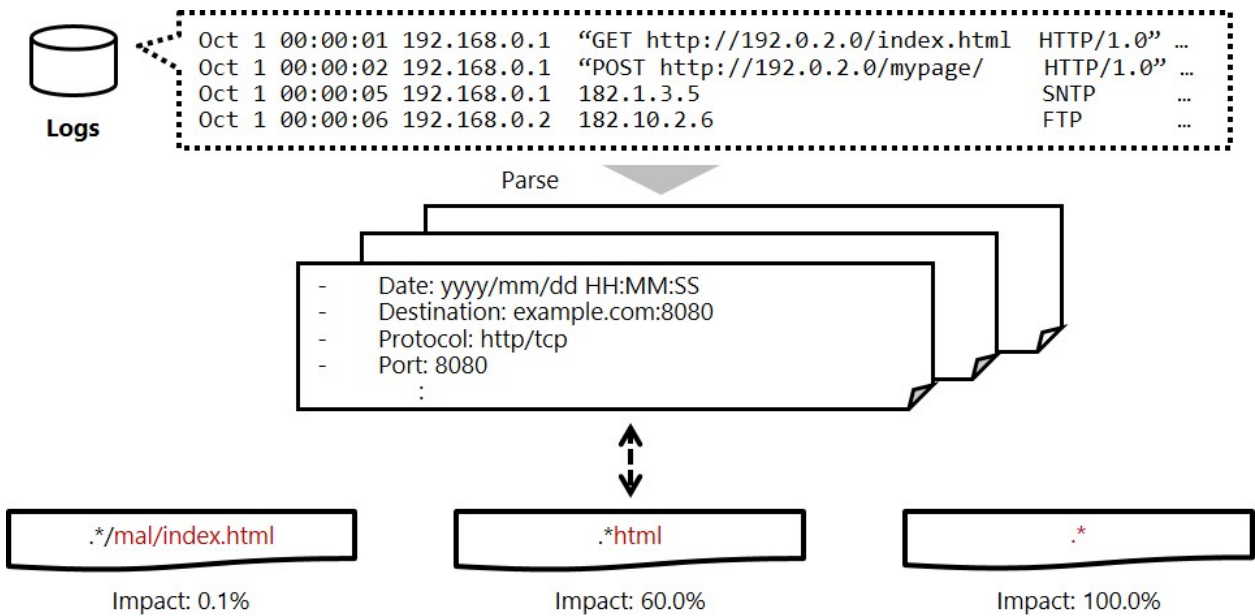


Figure 5.4 Overview of impact assessment.

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. 5.5). In this study, we assumed that the forward proxy blocks the communication; thus, SIGMA converts signature candidate into the access control format of *squid*.

5.3.6 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. 5.6. The 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

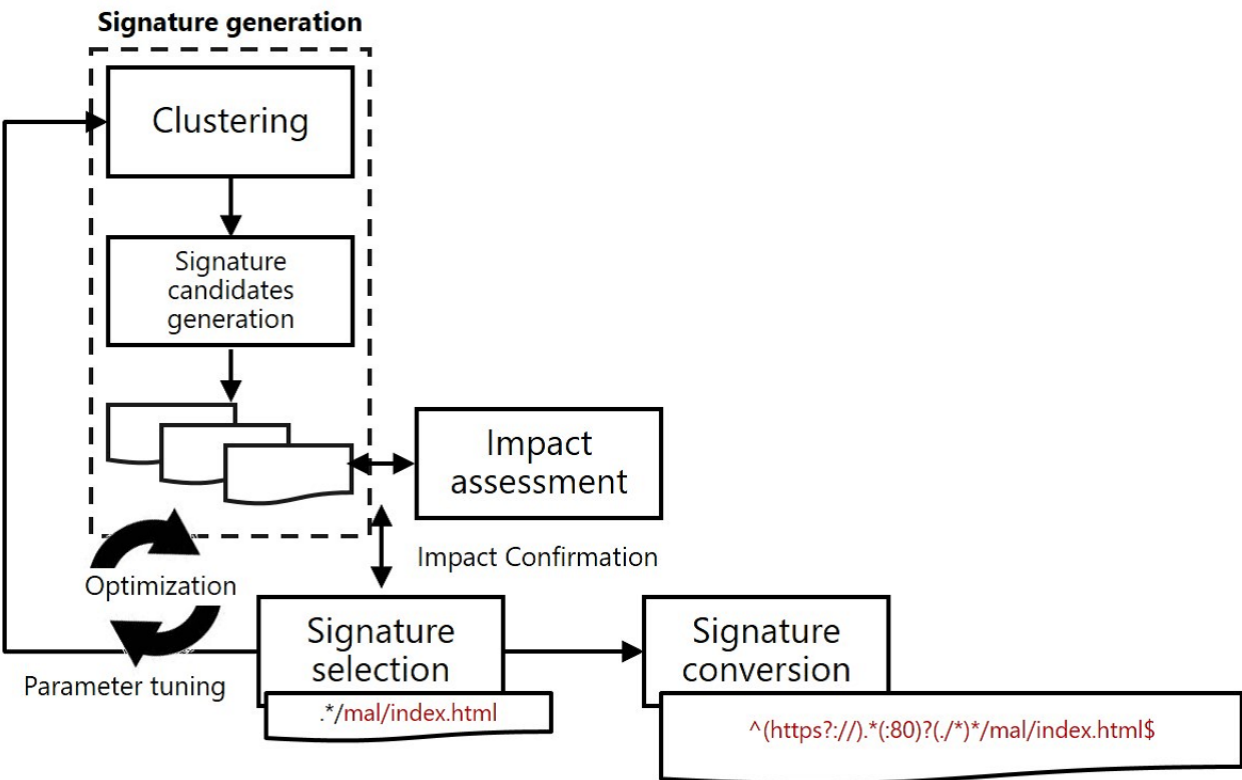


Figure 5.5 Overview of signature construction.

Table 5.2 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
<i>IcedID</i>	0	1		0	2	0	0	0	0	0	0	0	0	0	0	27	3	0	0

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.

List of signatures

#	Created	Signature	Impact	Status
1	2021-03-03	<code>^https?://192.0.2.1:65432/.*\$</code>	4 (0.5%)	not reviewed
2	2021-03-03	<code>^https?://example.com/.*\$</code>	100 (20.2%)	rejected
3	2021-03-02	<code>^https?://192.0.2.123:1111.html.exe\$</code>	11 (0.2%)	accepted

Signature details

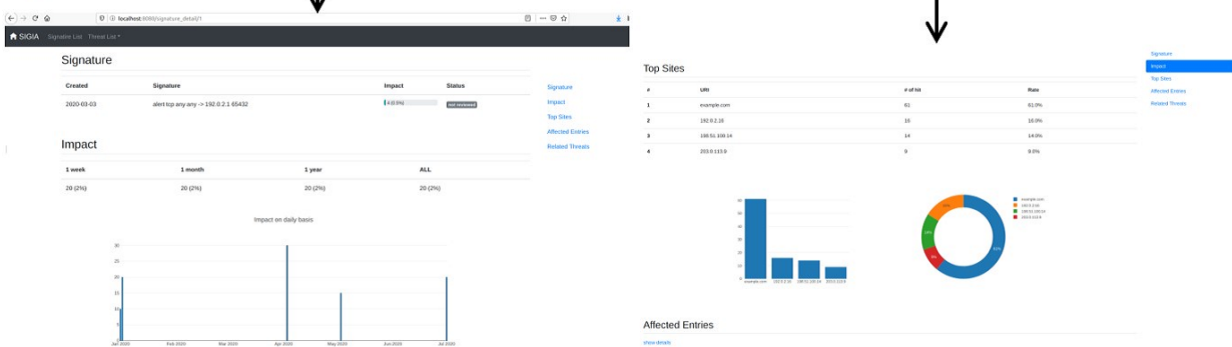


Figure 5.6 Overview of signature viewer.

5.4 Evaluation

5.4.1 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.

Table 5.3 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
<i>IcedID</i>	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	5	1	0	0

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.

5.4.2 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 [62])
 - 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.

5.4.3 Evaluation Results

Evaluation 1: Impact Assessment of Benign Communication Removal.

This evaluation is based on *IcedID* samples. 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

Table 5.4 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

1,107 samples from November 2020, when the *IcedID* samples were prevalent, were selected for verification. Tables 5.2 and 5.3 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 *IcedIDs*, and the remainder were *IcedID* samples from the past. In contrast, in the allow list method, the majority of *IcedIDs* (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 *IcedIDs*, 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.

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 over-detection rate and business impact were the same in this experiment. The results are shown in Table 5.4.

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.

Evaluation 3: Processing Time.

The evaluation environment is shown in Fig. 5.7. 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 [134] with *Ubuntu* 22.04 LTS OS [135], 2 virtual cores, and 8GB memory. *INetSim* 1.3.2 [136] 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 5.5 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 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.

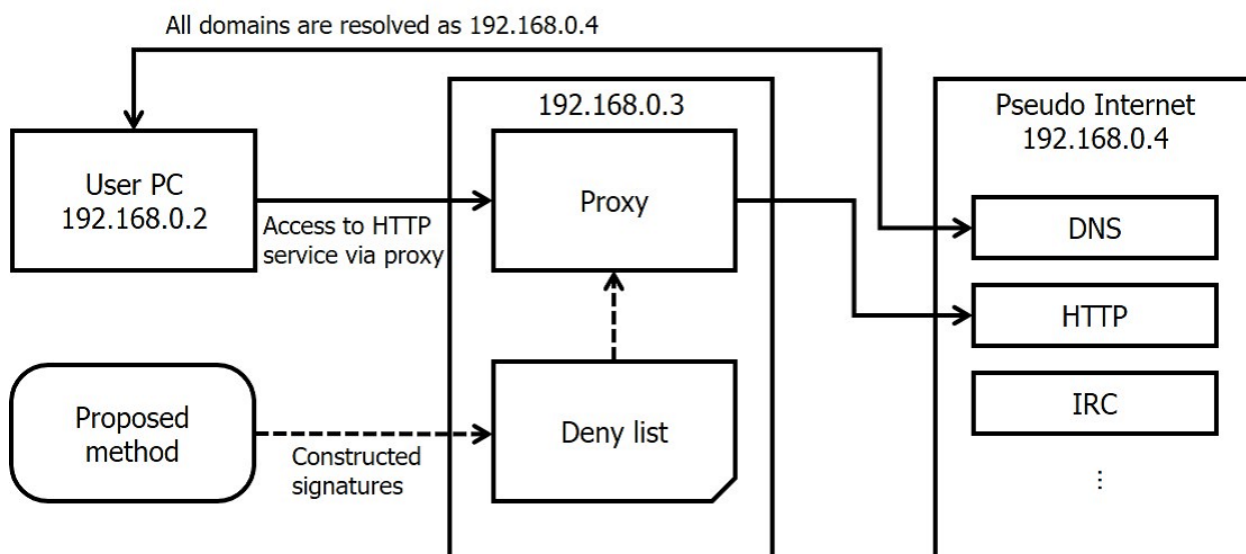


Figure 5.7 Overview of experimental environment.

Table 5.5 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

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.

5.5 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/CSIRT operations. As discussed in Section 5.2.2, 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 automatically detect unknown URLs, and in this respect, it is superior to the deny-list method, which can only detect known 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.

5.6 Conclusion

In this dissertation, 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.

Chapter 6

Conclusions

6.1 Concluding Remarks

In this dissertation, we have introduced and presented the evaluation of methods that improve the effectiveness and efficiency of security operations.

Chapter 1 describes the background and problems of this study. The importance of countermeasures against cyberattacks has been increasing each year due to the growth in number and sophistication of cyberattacks. Under these circumstances, many organizations have established organizations such as the SOC and CSIRT, which detect and respond to cyberattacks. In this chapter, the phases of security operations are systematically organized, and the problems in each phase are presented. Specifically, these problems are the costs of manually handling CTI and creating signatures for detecting malicious communications, the difficulties of detecting and identifying malicious hosts and tracking and detecting leakages of sensitive information, and the lack of understanding regarding the support mechanism for dynamic malware analysis. The research strategies for each problem are also described. We conducted research based on these strategies; the details of this research are presented in chapters 2 to 5.

Chapter 2 describes a method for structuring CTI and a threat-analysis method utilizing structured CTI. With the increase in the number and sophistication of cyberattacks, the importance of collecting and analyzing CTI during non-emergencies to keep up with the latest threat-related information and respond efficiently by utilizing this collected information in case of emergencies is increasing. On the other hand, most CTI is written in natural

language, making analysis time-consuming and expensive. Additionally, various organizations publish information separately, making cross-sectional analysis difficult. To solve these problems, we propose CyNER, which supports analysis by automatically structuring CTI in the STIX format, a common format for CTI. CyNER extracts named entities in the context of CTI and converts them to the STIX format by extracting relationships between the named entities and IOCs. This is expected to improve the efficiency of analysis and realize cross-sectional analysis. In the evaluation, we showed that the model trained on a corpus of the cybersecurity domain could improve the F value of NER by up to 2.6 points. We also cross-sectionally analyzed CTI and showed that CyNER could extract IOCs not included in existing reputation sites, that more than 97% of IOCs were included in only a single source, and that CyNER could automatically extract IOCs that had been exploited over time and across multiple attack groups. This indicated the potential of CyNER in contributing to the efficiency of CTI analysis.

Chapter 3 describes a method for tracking the diffusion of classified information on a guest OS using a VMM and detecting information leakage outside the OS. The leakage of information has increased in recent years; to address the problem, a function for tracing the diffusion of classified information within an OS has been proposed. However, this function has the following two shortcomings. First, to introduce the function, the source code of the OS needs to be modified. Second, there is a risk that the function will be disabled when the OS is attacked. Thus, we designed a function for tracing the diffusion of classified information in a guest OS by using a VMM. By using a VMM, the proposed function can be introduced in various environments without modifying the source code of the OS. It was also believed that attacks specifically targeting the proposed function would be difficult to achieve because the VMM was isolated from the OS. This dissertation describes the design and implementation of the proposed function for file operations, child process creations, and IPC in the guest OS through the KVM, a type of VMM; demonstrates the traceability of diffusing classified information by file operations, child process creations, and IPC; and evaluates the logical lines of code required to implement the proposed function and deal with the performance overheads. The evaluation showed that the implementation and usability of the proposed function were realistic.

Chapter 4 describes the results of the investigation of the mapping function of the ATT&CK technique, which is one of the analytical support functions in the online sandbox. Dynamic

analysis, which automatically analyzes malware, has become the de facto method for dealing with large amounts of malware. Regarding analytical support functions, there is a function that maps malware behavior to each element of the MITRE ATT&CK technique; this function has been adopted by many online sandboxes, and it contributes to the efficiency of analysis. This function depends on the implementation of mapping rules, which may affect the analysis results. Therefore, we conducted a survey on online sandboxes that had a function for mapping techniques, clarifying the actual status of the mapping function, such as the differences in mapping among sandboxes and those between mapping trends and manual mapping. We also derived the best practices for their use.

Chapter 5 describes a method for automatically creating signatures to detect communications toward suspicious destinations using the results of malware analysis. Since attacker hosts play a significant role in cyberattacks, communications to malicious hosts in cyberattacks are important. On the other hand, while these signatures should block malicious communications, they should not block benign communications in normal business operations. Therefore, generating such signatures requires a high degree of business understanding and is highly personalized. Additionally, the generated signatures need to be tested for their impact on benign communications in actual operation, which is also expensive. To solve these problems, we propose a system that automatically generates signatures that block malicious communications without interfering with benign ones and automatically performs the impact assessment of these signatures. The proposed system is expected to reduce the human resources needed for signature generation and the cost of impact assessment and assist in deciding whether a signature is applicable or not. This dissertation describes the design and implementation of the system and its evaluation using a prototype. The evaluation showed that the system could reduce the cost of each task by automatically creating signatures and evaluating their impact. The evaluation also showed that the generated signatures could block communications to malicious hosts and that the overhead incurred by the proposed system was within the practical range.

6.2 Future Directions

The future directions of the current study are listed below.

- (1) Generalization of classified information tracking and leakage detection methods

In this dissertation, we have implemented and evaluated a method for tracing and detecting the leakage of confidential information using a KVM, Intel CPU, and the 64-bit Linux 3.6.10 as the OS. Various CPUs and OSs are emerging in modern computing environments in academic and enterprise networks. In other words, it is assumed that practical security operations will require monitoring environments other than those discussed in this dissertation. The implementation described in this dissertation is based on Intel VT, a function of Intel CPUs, and process structures and system calls of a specific version of Linux. Thus, the proposed method cannot support the wide variety of environments described above and needs to be generalized so that it can monitor environments with a mixture of various CPUs and OSs.

(2) Consideration for improving the efficiency of other security operations

This dissertation systematically organizes security operations and proposes and evaluates methods to mitigate the problems in each phase. This dissertation does not necessarily cover all security operations, which are quite diverse. Therefore, improvements for the efficiency of operations not covered in this dissertation need to be considered.

(3) Evaluation through actual security operations

The evaluation of each method in this dissertation is limited to system evaluations and not practical security operations. Therefore, the practical applicability of each method needs to be evaluated through use in actual security operations. It is also desirable to receive feedback from operators on the usability and other aspects of the methods to further improve them.

The number of cyberattacks and the damage caused by them are increasing every year and are expected to increase further in the future. Therefore, security operations against such attacks will be required more than ever before; currently, the number of operators is limited. Thus, strengthening the defense capability of operators and reducing their workload through systemization, automation, and providing support through methods like those described in this dissertation is required. Generally, there is always a gap between research and practical implementations; this gap needs to be particularly reduced in the field of security operations. Accordingly, we will continue to conduct practical research in this field.

Attackers are increasingly working together; groups of such attackers are called threat actors [137–139]. Regarding defense against such entities, each organization is defending

itself within its own organization or against its own customers that are considered malicious. There is an asymmetry between the attacker and defender, with the attacker having an advantage in that the defender has no choice but to follow up on the attack; these asymmetries are beginning to appear in organizations. Although there is a certain amount of sensitive information in each organization that cannot be shared, we suggest realizing a collective defense by sharing information to the greatest extent possible—like it is mentioned in the observation results described in Chapter 3—to improve the overall capability of organizations as defenders.

Acknowledgments

First, I would like to express my sincere gratitude to my supervisor, Professor Toshihiro Yamauchi, for providing me with this precious opportunity to conduct a study as a PhD student in his laboratory. Without his guidance and persistent help, the preparation of this dissertation would not have been possible. I am deeply grateful to Professor Norikazu Takahashi and Associate Professor Yoshinari Nomura for their constant support and would also like to thank them for their great advice and feedback for revising the dissertation.

I especially would like to express my deepest appreciation to Professor Hideo Taniguchi and Associate Professor Masaya Sato for their continued support, extensive inputs and encouragement. I would also like to offer my special thanks to Assistant Professor Hiroki Kuzuno for the considerable support and encouragement.

I greatly appreciate Mr. Tetsuro Kito, Dr. Tomohiro Shigemoto, Dr. Nobutaka Kawaguchi, Mr. Rei Yamagishi, Professor Masato Terada, Mr. Takayuki Sato, Mr. Sho Aoki, Mr. Tomoya Suzuki, and Mr. Shoya Kojima of Hitachi, Ltd. for the support and useful advice they have provided me with ever since I joined Hitachi, Ltd. I am very grateful to Dr. Yu Tsuda for his valuable cooperation, insightful comments, and continuous encouragement in my experiments.

My heartfelt appreciation goes to the past and present members of Yamauchi Laboratory, Taniguchi Laboratory, and Nomura Laboratory for their kind help in both my research and life.

I would also like to express my gratitude to Hitachi, Ltd. for the financial support.

Finally, I would like to thank my family members for their daily support during all these years of work.

References

- [1] MIC: ICT Comprehensive Cybersecurity Measures 2022 (in Japanese). https://www.soumu.go.jp/main_content/000830903.pdf.
- [2] CYBERNETICA: Cybersecurity Domain Analysis. https://cyber.ee/uploads/Cyber_Sec_Domain_Analysis_2_2f75b4512f.pdf.
- [3] OWASP: OWASP Security Operations Center (SOC) Framework Project. [https://wiki.owasp.org/index.php/OWASP_Security_Operations_Center_\(SOC\)_Framework_Project](https://wiki.owasp.org/index.php/OWASP_Security_Operations_Center_(SOC)_Framework_Project).
- [4] Schinagl, S., Schoon, K. and Paans, R.: A Framework for Designing a Security Operations Centre (SOC), *2015 48th Hawaii International Conference on System Sciences*, pp. 2253–2262 (2015).
- [5] Radu, S. G.: Comparative Analysis of Security Operations Centre Architectures; Proposals and Architectural Considerations for Frameworks and Operating Models, *Innovative Security Solutions for Information Technology and Communications*, Springer International Publishing, pp. 248–260 (2016).
- [6] Schinagl, S., Schoon, K. and Paans, R.: A Framework for Designing a Security Operations Centre (SOC), *2015 48th Hawaii International Conference on System Sciences*, pp. 2253–2262 (2015).
- [7] Trellix: What Is a Security Operations Center (SOC)? <https://www.trellix.com/en-us/security-awareness/operations/what-is-soc.html>.
- [8] TechTarget: Building an effective security operations center framework. <https://www.techtarget.com/searchsecurity/tip/Building-an-effective-security-operations-center-framework>.

- [9] Blazic, B. J.: The cybersecurity labour shortage in Europe: Moving to a new concept for education and training, *Technology in Society*, Vol. 67, pp. 1–13 (2021).
- [10] Kokulu, F. B., Soneji, A., Bao, T., Shoshitaishvili, Y., Zhao, Z., Doupé, A. and Ahn, G.-J.: Matched and Mismatched SOCs: A Qualitative Study on Security Operations Center Issues, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pp. 1955–1970 (2019).
- [11] ITR Market View: Endpoint / Harmless / Web Isolation / CASB / CSPM / SOAR Market 2021 (in Japanese). <https://www.itr.co.jp/report/marketview/M21001100.html>.
- [12] Verizon: 2022 Data Breach Investigations Report. <https://www.verizon.com/business/resources/reports/dbir/>.
- [13] Mavroeidis, V. and Bromander, S.: Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence, *2017 European Intelligence and Security Informatics Conference*, EISIC '17, pp. 91–98 (2017).
- [14] Obrst, L., Chase, P. and Markeloff, R.: Developing an Ontology of the Cyber Security Domain, *STIDS*, pp. 49–56 (2012).
- [15] Jones, C. L., Bridges, R. A., Huffer, K. M. T. and Goodall, J. R.: Towards a Relation Extraction Framework for Cyber-Security Concepts, *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, CISR '15 (2015).
- [16] Joshi, A., Lal, R., Finin, T. and Joshi, A.: Extracting Cybersecurity Related Linked Data from Text, *2013 IEEE Seventh International Conference on Semantic Computing*, ICSC '13, pp. 252–259 (2013).
- [17] Liao, X., Yuan, K., Wang, X., Li, Z., Xing, L. and Beyah, R.: Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 755–766 (2016).

- [18] Mulwad, V., Li, W., Joshi, A., Finin, T. and Viswanathan, K.: Extracting Information about Security Vulnerabilities from Web Text, *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, Vol. 3, pp. 257–260 (2011).
- [19] Ramnani, R. R., Shivaram, K., Sengupta, S. and M., A. K.: Semi-Automated Information Extraction from Unstructured Threat Advisories, *Proceedings of the 10th Innovations in Software Engineering Conference*, ISEC '17, pp. 181–187 (2017).
- [20] Zhu, Z. and Dumitras, T.: FeatureSmith: Automatically Engineering Features for Malware Detection by Mining the Security Literature, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 767–778 (2016).
- [21] Feng, X., Liao, X., Wang, X., Wang, H., Li, Q., Yang, K., Zhu, H. and Sun, L.: Understanding and Securing Device Vulnerabilities through Automated Bug Report Analysis, *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC '19, pp. 887–903 (2019).
- [22] Husari, G., Al-Shaer, E., Ahmed, M., Chu, B. and Niu, X.: TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources, *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC '17, pp. 103–115 (2017).
- [23] Zhu, Z. and Dumitras, T.: ChainSmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports, *2018 IEEE European Symposium on Security and Privacy*, EuroS&P '18, pp. 458–472 (2018).
- [24] Milajerdi, S. M., Eshete, B., Gjomemo, R. and Venkatakrishnan, V.: POIROT: Aligning Attack Behavior with Kernel Audit Records for Cyber Threat Hunting, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pp. 1795–1812 (2019).
- [25] Satvat, K., Gjomemo, R. and Venkatakrishnan, V. N.: EXTRACTOR: Extracting Attack Behavior from Threat Reports, *CoRR*, Vol. abs/2104.08618 (2021).

- [26] Metcalf, L. and Spring, J. M.: Blacklist Ecosystem Analysis: Spanning Jan 2012 to Jun 2014, *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*, WISCS '15, pp. 13–22 (2015).
- [27] Li, V. G., Dunn, M., Pearce, P., McCoy, D., Voelker, G. M. and Savage, S.: Reading the Tea leaves: A Comparative Analysis of Threat Intelligence, *28th USENIX Security Symposium*, SEC '19, pp. 851–867 (2019).
- [28] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P. and Sheth, A. N.: TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones, *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementatio*, OSDI '10, pp. 1–6 (2010).
- [29] Zhu, D. Y., Jung, J., Song, D., Kohno, T. and Wetherall, D.: TaintEraser: Protecting Sensitive Data Leaks Using Application-level Taint Tracking, *SIGOPS Oper. Syst. Rev.*, Vol. 45, No. 1, pp. 142–154 (2011).
- [30] Zavou, A., Portokalidis, G. and Keromytis, A. D.: Taint-exchange: A Generic System for Cross-process and Cross-host Taint Tracking, *Proceedings of the 6th International Conference on Advances in Information and Computer Security*, IWSEC '11, pp. 113–128 (2011).
- [31] Portokalidis, G., Slowinska, A. and Bos, H.: Argos: An Emulator for Fingerprinting Zero-day Attacks for Advertised Honeypots with Automatic Signature Generation, *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, EuroSys '06, pp. 15–27 (2006).
- [32] Bellard, F.: QEMU, a Fast and Portable Dynamic Translator, *Proceedings of the Annual Conference on USENIX Annual Technical Conferenc*, ATEC '05 (2005).
- [33] Ho, A., Fetterman, M., Clark, C., Warfield, A. and Hand, S.: Practical Taint-based Protection Using Demand Emulation, *SIGOPS Oper. Syst. Rev.*, Vol. 40, No. 4, pp. 29–41 (2006).
- [34] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the Art of Virtualization, *Proceedings of the*

- Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pp. 164–177 (2003).
- [35] Tam, K., Khan, S. J., Fattori, A. and Cavallaro, L.: CopperDroid: Automatic Reconstruction of Android Malware Behaviors, *22nd Annual Network and Distributed System Security Symposium*, NDSS '15 (2015).
- [36] Nadkarni, A. and Enck, W.: Preventing accidental data disclosure in modern operating systems, *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, CCS '13, pp. 1029–1042 (2013).
- [37] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P. and Wang, X. S.: AppIntent: analyzing sensitive data transmission in android for privacy leakage detection, *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, CSS '13, pp. 1043–1054 (2013).
- [38] Michael, I., G., Deokhwan, K., Jeff, P., Limei, G., Nguyen, N. and Martin, R.: Information Flow Analysis of Android Applications in DroidSafe, *22nd Annual Network and Distributed System Security Symposium*, NDSS '15 (2015).
- [39] Yumerefendi, A. R., Mickle, B. and Cox, L. P.: TightLip: Keeping Applications from Spilling the Bean, *Proceedings of the 4th USENIX Conference on Networked Systems Design and Implementation*, NSDI '07, pp. 159–172 (2007).
- [40] Wang, J., Yu, M., Li, B., Qi, Z. and Guan, H.: Hypervisor-based Protection of Sensitive Files in a Compromised System, *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pp. 1765–1770 (2012).
- [41] Zhao, X., Borders, K. and Prakash, A.: Towards protecting sensitive files in a compromised system, *Third IEEE International Security in Storage Workshop*, SISW '05, pp. 21–28 (2005).
- [42] Borders, K., Zhao, X. and Prakash, A.: Securing Sensitive Content in a View-only File System, *Proceedings of the ACM Workshop on Digital Rights Management*, DRM '06, pp. 27–36 (2006).

- [43] Wurster, G. and van Oorschot, P. C.: A Control Point for Reducing Root Abuse of File-system Privilege, *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pp. 224–236 (2010).
- [44] Michael, M., An-I, Andy, W. and Peter, R.: Cashtags: Protecting the Input and Display of Sensitive Data, *24th USENIX Security Symposium*, SEC '15, pp. 961–976 (2015).
- [45] Chen, J., Chen, H., Bauman, E., Lin, Z., Zang, B. and Guan, H.: You Shouldn't Collect My Secrets: Thwarting Sensitive Keystroke Leakage in Mobile IME Apps, *24th USENIX Security Symposium*, SEC '15, pp. 657–690 (2015).
- [46] Sakamoto, S., Okuda, K., Nakatsuka, R. and Yamauchi, T.: DroidTrack: Tracking and Visualizing Information Diffusion for Preventing Information Leakage on Android, *Journal of Internet Services and Information Security*, Vol. 4, pp. 55–69 (2014).
- [47] Isohara, T., Takemori, K., Miyake, Y., Qu, N. and Perrig, A.: LSM-Based Secure System Monitoring Using Kernel Protection Schemes, *2010 International Conference on Availability, Reliability and Security*, ARES '10, pp. 591–596 (2010).
- [48] Sato, M. and Yamauchi, T.: VMM-Based Log-Tampering and Loss Detection Scheme, *Journal of Internet Technology*, Vol. 13, No. 4, pp. 655–666 (2012).
- [49] Sato, M. and Yamauchi, T.: Secure and Fast Log Transfer Mechanism for Virtual Machine, *Journal of Information Processing*, Vol. 22, No. 4, pp. 597–608 (2014).
- [50] Takada, T. and Koike, H.: NIGELOG: Protecting Logging Information by Hiding Multiple Backups in Directories, *Proceedings. Tenth International Workshop on Database and Expert Systems Applications*, DEXA '99, pp. 874–878 (1999).
- [51] Joo, J. W., Park, J. H., Suk, S. K. and Lee, D. G.: LISS: Log Data Integrity Support Scheme for Reliable Log Analysis of OSP, *The Journal of Convergence*, Vol. 5, No. 2, pp. 1–5 (2014).
- [52] Al-Shaer, R., Spring, J. M. and Christou, E.: Eliana Christou: Learning the Associations of MITRE ATT & CK Adversarial Techniques, *2020 IEEE Conference on Communications and Network Security*, CNS '20, pp. 1–9 (2020).

- [53] Hemberg, E., Kelly, J., Shlapentokh-Rothman, M., Reinstadler, B. M., Xu, K., Rutar, N. and O'Reilly, U.: Linking Threat Tactics, Techniques, and Patterns with Defensive Weaknesses, Vulnerabilities and Affected Platform Configurations for Cyber Hunting, *arXiv* (2020).
- [54] Yokoyama, A., Ishii, K., Tanabe, R., Papa, Y., Yoshioka, K., Matsumoto, T., Kasama, T., Inoue, D., Brengel, M., Backes, M. and Rossow, C.: SandPrint: Fingerprinting Malware Sandboxes to Provide Intelligence for Sandbox Evasion, *The 19th International Symposium on Research in Attacks, Intrusions and Defenses, RAID '16*, pp. 165–187 (2016).
- [55] Bulazel, A. and Yener, B.: A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion: PC, Mobile, and Web, *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium, ROOTS '17*, pp. 1–21 (2017).
- [56] Nappa, A., Papadopoulos, P., Varvello, M., Gomez, D. A., Tapiador, J. and Lanzi, A.: PoW-How: An Enduring Timing Side-Channel to Evade Online Malware Sandboxes, *European Symposium on Research in Computer Security, ESORICS '21*, pp. 86–109 (2021).
- [57] Mirsky, Y., Doitshman, T., Elovici, Y. and Shabtai, A.: Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection, *25th Annual Network and Distributed System Security Symposium, NDSS '18* (2018).
- [58] Perdisci, R., Lee, W. and Feamster, N.: Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces, *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI'10* (2010).
- [59] Paleari, R., Martignoni, L., Passerini, E., Davidson, D., Fredrikson, M., Giffin, J. and Jha, S.: Automatic Generation of Remediation Procedures for Malware Infections, *19th USENIX Security Symposium, SEC '10* (2010).
- [60] Kirat, D. and Vigna, G.: MalGene: Automatic Extraction of Malware Analysis Evasion Signature, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15*, pp. 769–780 (2015).

- [61] Feng, Y., Bastani, O., Martins, R., Dillig, I. and Anand, S.: Automated Synthesis of Semantic Malware Signatures using Maximum Satisfiability, *24th Annual Network and Distributed System Security Symposium*, NDSS '17 (2017).
- [62] Kurogome, Y., Otsuki, Y., Kawakoya, Y., Iwamura, M., Hayashi, S., Mori, T. and Sen, K.: EIGER: Automated IOC Generation for Accurate and Interpretable Endpoint Malware Detection, *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, pp. 687–701 (2019).
- [63] McNeil, N., Bridges, R. A., Iannacone, M. D., Czejdo, B., Perez, N. and Goodall, J. R.: PACE: Pattern Accurate Computationally Efficient Bootstrapping for Timely Discovery of Cyber-security Concepts, *2013 12th International Conference on Machine Learning and Applications*, ICMLA '17, Vol. 2, pp. 60–65 (2013).
- [64] IBM: IBM Watson to Tackle Cybercrime (2016). <https://newsroom.ibm.com/2016-05-10-IBM-Watson-to-Tackle-Cybercrime?lnk=hmhm>.
- [65] Wåreus, E. and Hell, M.: Automated CPE Labeling of CVE Summaries with Machine Learning, *Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '20, pp. 3–22 (2020).
- [66] OASIS: Introduction to STIX (2021). <https://oasis-open.github.io/cti-documentation/stix/intro.html>.
- [67] Mandiant: OpenIOC (2013). https://github.com/mandiant/OpenIOC_1.1.
- [68] MISP project: MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing (2021). <https://www.misp-project.org/>.
- [69] Facebook: Facebook ThreatExchange Overview (2021). <https://developers.facebook.com/programs/threatexchange/>.
- [70] DoD: Defense Industrial Base Cybersecurity Information Sharing Program (2021). <https://dibnet.dod.mil/portal/intranet/>.
- [71] CISA: Automated Indicator Sharing (AIS) (2021). <https://www.us-cert.gov/ais>.
- [72] AlienVault: Open Threat Intelligence (2021). <https://otx.alienvault.com/>.

- [73] OpenCTI-Platform: OpenCTI (2021). <https://github.com/OpenCTI-Platform/opencti>.
- [74] Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space, *1st International Conference on Learning Representations, Workshop Track Proceedings*, ICLR '13 (2013).
- [75] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, NAACL-HLT '19, pp. 4171–4186 (2019).
- [76] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V.: RoBERTa: A Robustly Optimized BERT Pretraining Approach, *CoRR*, Vol. abs/1907.11692 (2019).
- [77] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R.: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, *International Conference of Learning Representations*, CISR '20 (2020).
- [78] Gupta, P., Rajaram, S., Schutze, H. and Runkler, T.: Neural Relation Extraction within and across Sentence Boundaries, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 01, pp. 6513–6520 (2019).
- [79] Min, B., Shi, S., Grishman, R. and Lin, C.-Y.: Ensemble Semantics for Large-scale Unsupervised Relation Extraction, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP '12, pp. 1027–1037 (2012).
- [80] Peng, N., Poon, H., Quirk, C., Toutanova, K. and Yih, W.-t.: Cross-Sentence N-ary Relation Extraction with Graph LSTMs, *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 101–115 (2017).
- [81] Takanobu, R., Zhang, T., Liu, J. and Huang, M.: A Hierarchical Framework for Relation Extraction with Reinforcement Learning, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 01, pp. 7072–7079 (2019).

- [82] Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N. and Androutsopoulos, I.: LEGAL-BERT: The Muppets straight out of Law School, *Findings of the Association for Computational Linguistics*, EMNLP '20, pp. 2898–2904 (2020).
- [83] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P. J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, *Journal of Machine Learning Research*, Vol. 21, No. 140, pp. 1–67 (2020).
- [84] The Hugging Face Team: Huggingface: Transformers (2020). <https://huggingface.co/transformers/>.
- [85] Boudin, F.: Unsupervised Keyphrase Extraction with Multipartite Graphs, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 667–672 (2018).
- [86] Florescu, C. and Caragea, C.: PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL '17, pp. 1105–1115 (2017).
- [87] Bougouin, A., Boudin, F. and Daille, B.: TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction, *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, IJCNLP '13, pp. 543–551 (2013).
- [88] Sebastián, S. and Caballero, J.: AVclass2: Massive Malware Tag Extraction from AV Labels, *Annual Computer Security Applications Conference*, ACSAC '20, pp. 42–53 (2020).
- [89] OASIS: STIX Visualizer (2021). <https://oasis-open.github.io/cti-stix-visualization/>.
- [90] Caruana, R., Lawrence, S. and Giles, L.: Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping, *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS '00, pp. 381–387 (2000).

- [91] Syed, R.: Analyzing Software Vendors' Patch Release Behavior in the Age of Social Media, *Proceedings of the International Conference on Information Systems - Transforming Society with Digital Innovation*, ICIS '17 (2017).
- [92] Mittal, S., Das, P. K., Mulwad, V., Joshi, A. and Finin, T.: CyberTwitter: Using Twitter to generate alerts for cybersecurity threats and vulnerabilities, *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '16, pp. 860–867 (2016).
- [93] Yang, S., Shu, K., Wang, S., Gu, R., Wu, F. and Liu, H.: Unsupervised Fake News Detection on Social Media: A Generative Approach, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 01, pp. 5644–5651 (2019).
- [94] Japan Network Security Association: 2008 Information Security Incident. http://www.jnsa.org/result/incident/data/2008incident_survey_e_v1.0.pdf.
- [95] theguardian: Antivirus software is dead, says security expert at Symantec. <http://www.theguardian.com/technology/2014/may/06/antivirus-software-fails-catch-attacks-security-expert-symantec>.
- [96] Tabata, T., Hakomori, S., Ohashi, K., Uemura, S., Yokoyama, K. and Taniguchi, H.: Tracing Classified Information Diffusion for Protecting Information Leakage, *IPSI Journal*, Vol. 50, No. 9, pp. 2088–2102 (2009).
- [97] Nomura, Y., Hakomori, S., Yokoyama, K. and Taniguchi, H.: Proceedings of 4th Int. Conf. on Computing, Communications and Control Technologies, *Tracing the Diffusion of Classified Information Triggered by File Open System Cal*, CCCT '06, pp. 312–317 (2006).
- [98] Otsubo, N., Uemura, S., Yamauchi, T. and Taniguchi, H.: Design and Evaluation of a Diffusion Tracing Function for Classified Information Among Multiple Computers, *7th FTRA International Conference on Multimedia and Ubiquitous Engineering*, Vol. 240, pp. 235–242 (2013).
- [99] KVM: Main Page — KVM (2016). https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792.

- [100] Chen, P. and Noble, B.: When virtual is better than real [operating system relocation to virtual machines], *Proceedings Eighth Workshop on Hot Topics in Operating Systems*, HotOS '01, pp. 133–138 (2001).
- [101] LocMetrics: . <http://www.locmetrics.com/>.
- [102] McVoy, L. and Staelin, C.: lmbench: Portable Tools for Performance Analysis, *USENIX 1996 Annual Technical Conference*, USENIX ATC '96 (1996).
- [103] AV-TEST: Malware Statistics & Trends Report. <https://www.av-test.org/en/statistics/malware/>.
- [104] MITRE: ATT&CK. <https://attack.mitre.org/>.
- [105] JoeSandbox: Automated Malware Analysis - Joe Sandbox Cloud Basic. <https://www.joesandbox.com/>.
- [106] Hybrid Analysis: Free Automated Malware Analysis Service - powered by Falcon Sandbox. <https://www.hybrid-analysis.com/>.
- [107] Hatching Triage: Hatching Triage — Sandbox for High-Volume Automated Malware Analysis. <https://tria.ge/>.
- [108] McAfee: McAfee Advanced Threat Defense Leverages MITRE ATT&CK Framework. <https://www.mcafee.com/enterprise/en-us/assets/solution-briefs/sb-atd-leverages-mitre.pdf>.
- [109] Trend Micro: TREND MICRO VISION ONE. <https://cdw-prod.adobecqms.net/content/dam/cdw/on-domain-ca/brand/trend-micro/trendmicro-vision-one-solution-brief-aoda-v2.pdf>.
- [110] CISA: Best Practices for MITRE ATT&CK Mapping. <https://www.cisa.gov/uscert/sites/default/files/publications/Best%20Practices%20for%20MITRE%20ATTCK%20Mapping.pdf>.
- [111] Sun, B., Fujino, A., Mori, T., Ban, T., Takahashi, T. and Inoue, D.: Automatically Generating Malware Analysis Reports Using Sandbox Logs, *IEICE Transactions on Information and Systems*, Vol. E101.D, No. 11, pp. 2622–2632 (2018).

- [112] Rieck, K., Trinius, P., Willems, C. and Holz, T.: Automatic analysis of malware behavior using machine learning, *Journal of Computer Security*, Vol. 19, No. 4, pp. 639–668 (2011).
- [113] Yong Wong, M., Landen, M., Antonakakis, M., Blough, D. M., Redmiles, E. M. and Ahamad, M.: An Inside Look into the Practice of Malware Analysis, *The 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, pp. 3053–3069 (2021).
- [114] Stichting Cuckoo Foundation: Cuckoo Sandbox - Automated Malware Analysis. <https://cuckoosandbox.org/>.
- [115] any.run: ANY.RUN - Interactive Online Malware Sandbox. <https://any.run>.
- [116] Yamagishi, R., Fujii, S. and Sato, T.: Clarification of Malware Dynamic Analysis Tasks by User Investigation, *Computer Security Symposium 2021*, CSS '21, pp. 112–119 (2021). (in Japanese).
- [117] Mandiant: Cyber Security & Threat Intelligence Resources. [https://www.mandiant.com/resources?f\[0\]=layout:article_report](https://www.mandiant.com/resources?f[0]=layout:article_report).
- [118] Cisco Talos: Cisco Talos Intelligence Group - Comprehensive Threat Intelligence. <https://talosintelligence.com/>.
- [119] Trend Micro: Research, News, and Perspectives. https://www.trendmicro.com/en_us/research.html.
- [120] Mandiant: GitHub - mandiant/capa: The FLARE team's open-source tool to identify capabilities in executable files. <https://github.com/mandiant/capa>.
- [121] Intezer Analyze: Intezer Analyze - All-In-One Malware Analysis Platform. <https://analyze.intezer.com/>.
- [122] MITRE: subtechniques-csv.zip. <https://attack.mitre.org/docs/subtechniques/subtechniques-csv.zip>.
- [123] MITRE: MITRE ATT&CK Navigator. <https://mitre-attack.github.io/attack-navigator/>.

- [124] MITRE: Sightings Ecosystem: A Data-driven Analysis of ATT&CK in the Wild. <https://web.mitre-engenuity.org/hubfs/Center/%20for/%20Threat/%20Informed/%20Defense/CTID-Sightings-Ecosystem-Report.pdf>.
- [125] Takahashi, Y., Shima, S., Tanabe, R. and Yoshioka, K.: APTGen: An Approach towards Generating Practical Dataset Labelled with Targeted Attack Sequences, *13th USENIX Workshop on Cyber Security Experimentation and Test*, CSET '20 (2020).
- [126] Unit42: Case Study: Emotet Thread Hijacking, an Email Attack Technique. <https://unit42.paloaltonetworks.com/emotet-thread-hijacking/>.
- [127] JPCERT/CC: Malware Used by Lazarus after Network Intrusion. <https://blogs.jpcert.or.jp/en/2020/08/Lazarus-malware.html>.
- [128] Nakatsuru, Y.: Understanding Command and Control - An Anatomy of xxmm Communication - (2019). https://jsac.jpcert.or.jp/archive/2019/pdf/JSAC2019_8_nakatsuru_en.pdf.
- [129] Serita, S., Fujii, Y., Kakuta, T., Michiori, Y., Ohtori, T., Kishiro, T. and Terada, M.: Automatic Generation of URL Regular Expression for Detecting Malicious Traffic, *Computer Security Symposium 2014*, CSS '14, pp. 242–249 (2014). (in Japanese).
- [130] Nelms, T., Perdisci, R. and Ahamad, M.: ExecScent: Mining for New C&C Domains in Live Networks with Adaptive Control Protocol Templates, *22nd USENIX Security Symposium*, SEC '13, pp. 589–604 (2013).
- [131] JUNIPER NETWORKS: COVID-19 and FMLA Campaigns used to install 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>.
- [132] 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/>.
- [133] Squid: Optimising Web Delivery (2022). <http://www.squid-cache.org/>.
- [134] VMware: ESXi (2022). <https://www.vmware.com/products/esxi-and-esx.html>.

-
- [135] Canonical Ltd.: Enterprise Open Source and Linux — Ubuntu (2022). <https://ubuntu.com/>.
 - [136] INetSim: Internet Services Simulation Suite (2022). <https://www.inetsim.org/>.
 - [137] MITRE: Groups. <https://attack.mitre.org/groups/>.
 - [138] APTMAP: Threat Actor Map. <https://aptmap.netlify.app/>.
 - [139] ETDA: Threat Group Cards: A Threat Actor Encyclopedia. <https://apt.etda.org.th/cgi-bin/listgroups.cgi>.

Appendix A

Information Extraction from Unstructured Text of CTI Sources with Noncontextual IOCs

A.1 Source of CTI

Table A.1 shows 35 sources of CTIs used by CyNER. We're very grateful to all of the CTI publishers.

A.2 Refang Rules

Table A.2 shows all the refang rules implemented in CyNER.

Table A.1 Source websites of CTIs.

#	Publisher	URL
1	Avast Blog	https://blog.avast.com/
2	Certego	http://www.certego.net/en/news/
3	Checkpoint	https://blog.checkpoint.com/
4	Cisco Talos	https://blog.talosintelligence.com/
5	Cofense	https://cofense.com/blog/
6	Crowdstrike	https://www.crowdstrike.com/blog/category/threat-intel-research/
7	Cylance	https://threatvector.cylance.com
8	Dancho Danchev's Blog	https://ddanchev.blogspot.com/
9	Dynamo	https://blog.dynamoo.com/
10	FireEye Blogs, Threat Research	https://www.fireeye.com/blog/threat-research.html
11	Fox-it	https://blog.fox-it.com/
12	Hexacorn	http://www.hexacorn.com/blog/
13	ICS-CERT, advisories	https://ics-cert.us-cert.gov/advisories
14	ICS-CERT, alerts	https://ics-cert.us-cert.gov/alerts
15	InQuest Blog	http://blog.inquest.net/blog/
16	Kaspersky lab, securelist	https://securelist.com/
17	krebs on security	https://krebsonsecurity.com/
18	malware-traffic-analysis	https://www.malware-traffic-analysis.net/
19	Malwarebytes Labs, Threat Analysis	https://blog.malwarebytes.com/category/threat-analysis/
20	MalwareMustDie	http://blog.malwaremustdie.org/
21	McAfee Threat Center	http://www.mcafee.com/us/threat_center/
22	Naked Security	https://nakedsecurity.sophos.com/
23	360 Netlab Blog	http://blog.netlab.360.com/
24	paloalto cybersecurity	https://researchcenter.paloaltonetworks.com/cybersecurity-2/
25	Sucuri	https://blog.sucuri.net/
26	Symantec	https://symantec.com/blogs/threat-intelligence
27	TaoSecurity	https://taosecurity.blogspot.com/
28	The Hacker News	https://thehackernews.com/
29	Threatpost	https://threatpost.com/blog/
30	TrendLabs Security Intelligence Blog	https://blog.trendmicro.com/trendlabs-security-intelligence/
31	US-CERT, alerts	https://www.us-cert.gov/ncas/alerts
32	Webroot	https://www.webroot.com/blog/
33	WeLiveSecurity	https://www.welivesecurity.com/
34	Zscaler blogs	https://www.zscaler.com/blogs/research
35	The DFIR Report	https://thedfirreport.com/

Table A.2 Refang and defang rules.

Category	Before refanging	After refanging
URL	"hccp", "hxxp", "hXXp", "xxxx", "[http]"	"http"
URL	"hxxps", "xxxxx", "[https]"	"https"
URL	"http :/", "http/", "http:/"	"http:/"
URL	"https :/", "https/", "https:/"	"https:/"
URL	":/"	":/"
URL	"\"	"/"
URL	"[www]", "(www)"	"www"
IPv4/URL	"(.)", "[.]", "[dot]", "(dot)", "[punkt]", "(punkt)", "DOT", "DOT "	."
IPv4/URL	".com"	".com"
IPv6/URL	"[:]"	":"

Appendix B

Survey and Analysis on ATT&CK Mapping Function of Online Sandbox for Understanding and Efficient Using

B.1 Detailed information on the validation of the ATT&CK Technique mapping function

This section shows detail of statistical tests of each RQ.

First, Table B.1 shows the results for all techniques for the number of techniques observed and the presence of significant differences in each sandbox as described in RQ1. As in Table 4.4, the number of observations in each sandbox, the p-value of the chi-square test, and the presence of significant differences at a significance level of 0.05 are shown for each technique for the 1,012 samples in all sandboxes.

Next, Table B.2 shows the significant difference between malware and benign files for each technique described in RQ3. This table shows the number of observations, the p-value of the chi-square test, and the presence or absence of a significant difference when the significance level is set to 0.05 for each technique for the 13,115 malware and 1,531 benign files that existed in JoeSandbox.

Table B.3 shows the techniques observed in the multiple methods described in RQ4 and whether there are significant differences between methods. The table shows the number of observations per method, the p-value of the chi-square test, and the presence of significant

differences when the significance level is set to 0.05 for the techniques observed in the online sandbox, static analysis, and reports.

Table B.1 Number of observations and presence of significant differences among sandboxes for each MITRE ATT&CK Technique (RQ1) (1/3).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist		
T1055	Process Injection	938	74	598	414	0	1,012	0	✓
T1497	Virtualization/Sandbox Evasion	874	138	241	771	62	950	0	✓
T1027.002	Software Packing	769	243	669	343	0	1,012	1.18E-301	✓
T1027	Obfuscated Files or Information	863	149	7	1,005	0	1,012	0	✓
T1518.001	Security Software Discovery	925	87	53	959	3	1,009	0	✓
T1057	Process Discovery	878	134	465	547	0	1,012	0	✓
T1082	System Information Discovery	991	21	207	805	413	599	2.29E-285	✓
T1560	Archive Collected Data	852	160	3	1,009	0	1,012	0	✓
T1573	Encrypted Channel	898	114	223	789	0	1,012	0	✓
T1071	Application Layer Protocol	815	197	0	1,012	0	1,012	0	✓
T1036	Masquerading	821	191	89	923	0	1,012	0	✓
T1095	Non-Application Layer Protocol	744	268	3	1,009	0	1,012	0	✓
T1105	Ingress Tool Transfer	540	472	47	965	0	1,012	6.30E-247	✓
T1078	Valid Accounts	30	982	0	1,012	0	1,012	6.94E-14	✓
T1106	Native API	293	719	4	1,008	0	1,012	5.33E-138	✓
T1203	Exploitation for Client Execution	65	947	34	978	32	980	0.000276521	✓
T1569.002	Service Execution	47	965	5	1,007	0	1,012	1.03E-17	✓
T1574.002	DLL Side-Loading	110	902	0	1,012	0	1,012	2.70E-50	✓
T1546.011	Application Shimming	124	888	0	1,012	0	1,012	7.15E-57	✓
T1543.003	Windows Service	69	943	15	997	23	989	1.91E-11	✓
T1068	Exploitation for Privilege Escalation	27	985	0	1,012	0	1,012	1.48E-12	✓
T1134	Access Token Manipulation	172	840	0	1,012	0	1,012	6.54E-80	✓
T1140	Deobfuscate/Decode Files or Information	579	433	5	1,007	0	1,012	9.15E-307	✓
T1070.006	Timestomp	180	832	0	1,012	0	1,012	7.95E-84	✓
T1218.010	Regsvr32	3	1,009	5	1,007	0	1,012	0.092432672	-
T1218.011	Rundll32	48	964	7	1,005	0	1,012	6.02E-17	✓
T1056	Input Capture	379	633	2	1,010	0	1,012	5.66E-187	✓
T1124	System Time Discovery	271	741	11	1,001	0	1,012	1.48E-120	✓
T1120	Peripheral Device Discovery	9	1,003	601	411	38	974	1.95E-285	✓
T1083	File and Directory Discovery	581	431	18	994	0	1,012	1.80E-296	✓
T1012	Query Registry	289	723	909	103	269	743	2.94E-228	✓
T1070.004	File Deletion	133	879	365	647	9	1,003	1.81E-101	✓
T1087	Account Discovery	227	785	0	1,012	0	1,012	2.81E-107	✓
T1033	System Owner/User Discovery	227	785	16	996	0	1,012	2.83E-94	✓
T1018	Remote System Discovery	691	321	14	998	14	998	0	✓
T1115	Clipboard Data	196	816	3	1,009	0	1,012	4.29E-89	✓
T1529	System Shutdown/Reboot	103	909	0	1,012	0	1,012	4.97E-47	✓
T1070	Indicator Removal on Host	3	1,009	1	1,011	0	1,012	0.173373213	-
T1003	OS Credential Dumping	478	534	52	960	0	1,012	1.48E-205	✓
T1571	Non-Standard Port	367	645	256	756	0	1,012	6.16E-94	✓
T1059	Command and Scripting Interpreter	222	790	4	1,008	13	999	9.41E-91	✓
T1547.001	Registry Run Keys / Startup Folder	208	804	162	850	177	835	0.025182647	✓
T1574.010	Services File Permissions Weakness	5	1,007	3	1,009	0	1,012	0.092432672	-
T1010	Application Window Discovery	501	511	148	864	0	1,012	6.83E-170	✓
T1016	System Network Configuration Discovery	118	894	59	953	0	1,012	6.17E-28	✓
T1113	Screen Capture	34	978	6	1,006	0	1,012	1.35E-11	✓
T1486	Data Encrypted for Impact	19	993	19	993	0	1,012	6.64E-05	✓
T1053	Scheduled Task/Job	183	829	115	897	126	886	1.74E-05	✓
T1562.001	Disable or Modify Tools	695	317	11	1,001	13	999	0	✓
T1112	Modify Registry	39	973	524	488	326	686	5.78E-124	✓
T1005	Data from Local System	453	559	84	928	358	654	1.66E-76	✓
T1114	Email Collection	322	690	122	890	116	896	6.13E-40	✓
T1047	Windows Management Instrumentation	422	590	42	970	0	1,012	7.58E-180	✓
T1222	File and Directory Permissions Modification	64	948	0	1,012	3	1,009	1.16E-26	✓
T1564.001	Hidden Files and Directories	133	879	4	1,008	5	1,007	1.04E-53	✓
T1552.002	Credentials in Registry	232	780	0	1,012	0	1,012	8.08E-110	✓
T1037.005	Startup Items	33	979	0	1,012	0	1,012	3.24E-15	✓
T1189	Drive-by Compromise	1	1,011	0	1,012	0	1,012	0.367758249	-
T1102	Web Service	22	990	0	1,012	32	980	2.61E-07	✓
T1014	Rootkit	39	973	0	1,012	0	1,012	6.95E-18	✓

Table B.1 Number of observations and presence of significant differences among sandboxes for each MITRE ATT&CK Technique (RQ1) (2/3).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist		
T1056.004	Credential API Hooking	57	955	902	110	0	1,012	0	✓
T1059.001	PowerShell	22	990	94	918	0	1,012	5.90E-29	✓
T1197	BITS Jobs	1	1,011	0	1,012	1	1,011	0.606330781	-
T1552.001	Credentials In Files	58	954	2	1,010	358	654	3.48E-133	✓
T1007	System Service Discovery	33	979	0	1,012	0	1,012	3.24E-15	✓
T1219	Remote Access Software	54	958	0	1,012	0	1,012	1.33E-24	✓
T1406	Obfuscated Files or Information	6	1,006	0	1,012	0	1,012	0.002449476	✓
T1523	Evade Analysis Environment	1	1,011	0	1,012	0	1,012	0.367758249	-
T1412	Capture SMS Messages	3	1,009	1	1,011	0	1,012	0.173373213	-
T1426	System Information Discovery	3	1,009	0	1,012	0	1,012	0.049639551	✓
T1449	Exploit SS7 to Redirect Phone Calls/SMS	4	1,008	0	1,012	0	1,012	0.018219241	✓
T1448	Carrier Billing Fraud	4	1,008	0	1,012	0	1,012	0.018219241	✓
T1418	Application Discovery	5	1,007	1	1,011	0	1,012	0.029988818	✓
T1409	Access Stored Application Data	1	1,011	0	1,012	0	1,012	0.367758249	-
T1421	System Network Connections Discovery	3	1,009	1	1,011	0	1,012	0.173373213	-
T1422	System Network Configuration Discovery	2	1,010	0	1,012	0	1,012	0.135156976	-
T1430	Location Tracking	5	1,007	1	1,011	0	1,012	0.029988818	✓
T1424	Process Discovery	3	1,009	0	1,012	0	1,012	0.049639551	✓
T1432	Access Contact List	2	1,010	0	1,012	0	1,012	0.135156976	-
T1433	Access Call Log	1	1,011	0	1,012	0	1,012	0.367758249	-
T1507	Network Information Discovery	5	1,007	0	1,012	0	1,012	0.0066826	✓
T1439	Eavesdrop on Insecure Network Communication	2	1,010	0	1,012	0	1,012	0.135156976	-
T1472	Generate Fraudulent Advertising Revenue	1	1,011	0	1,012	0	1,012	0.367758249	-
T1447	Delete Device Data	4	1,008	0	1,012	0	1,012	0.018219241	✓
T1129	Shared Modules	90	922	0	1,012	0	1,012	5.24E-41	✓
T1136	Create Account	16	996	0	1,012	0	1,012	1.03E-07	✓
T1564.002	Hidden Users	12	1,000	0	1,012	0	1,012	5.86E-06	✓
T1049	System Network Connections Discovery	5	1,007	0	1,012	0	1,012	0.0066826	✓
T1499	Endpoint Denial of Service	11	1,001	0	1,012	0	1,012	1.60E-05	✓
T1566.002	Spearphishing Link	5	1,007	1	1,011	0	1,012	0.029988818	✓
T1429	Capture Audio	2	1,010	0	1,012	0	1,012	0.135156976	-
T1080	Taint Shared Content	7	1,005	0	1,012	0	1,012	0.000897249	✓
T1055.011	Extra Window Memory Injection	8	1,004	31	981	0	1,012	1.72E-09	✓
T1547.008	LSASS Driver	5	1,007	0	1,012	0	1,012	0.0066826	✓
T1021.001	Remote Desktop Protocol	1	1,011	230	782	1	1,011	5.13E-107	✓
T1574.001	DLL Search Order Hijacking	1	1,011	0	1,012	0	1,012	0.367758249	-
T1490	Inhibit System Recovery	3	1,009	6	1,006	9	1,003	0.221142869	-
T1185	Man in the Browser	18	994	0	1,012	0	1,012	1.37E-08	✓
T1048	Exfiltration Over Alternative Protocol	5	1,007	0	1,012	0	1,012	0.0066826	✓
T1091	Replication Through Removable Media	9	1,003	0	1,012	3	1,009	0.005139326	✓
T1090.003	Multi-hop Proxy	8	1,004	0	1,012	0	1,012	0.000328447	✓
T1090	Proxy	24	988	2	1,010	10	1,002	2.87E-05	✓
T1564.003	Hidden Window	0	1,012	3	1,009	0	1,012	0.049639551	✓
T1542.003	Bootkit	8	1,004	9	1,003	8	1,004	0.960470399	-
T1053.001	At (Linux)	2	1,010	0	1,012	0	1,012	0.135156976	-
T1547.006	Kernel Modules and Extensions	1	1,011	30	982	0	1,012	4.70E-13	✓
T1553.004	Install Root Certificate	2	1,010	0	1,012	71	941	1.29E-30	✓
T1001	Data Obfuscation	5	1,007	0	1,012	0	1,012	0.0066826	✓
T1562.004	Disable or Modify System Firewall	0	1,012	7	1,005	0	1,012	0.000897249	✓
T1548.002	Bypass User Access Control	4	1,008	1	1,011	2	1,010	0.367030256	-
T1491	Defacement	10	1,002	2	1,010	9	1,003	0.065011494	-
T1564.004	NTFS File Attributes	1	1,011	163	849	0	1,012	1.20E-74	✓
T1135	Network Share Discovery	3	1,009	1	1,011	0	1,012	0.173373213	-
T1553.002	Code Signing	0	1,012	45	967	0	1,012	1.45E-20	✓
T1046	Network Service Scanning	0	1,012	1	1,011	0	1,012	0.367758249	-
T1176	Browser Extensions	1	1,011	0	1,012	0	1,012	0.367758249	-
T1218.005	Mshta	0	1,012	7	1,005	0	1,012	0.000897249	✓
T1413	Access Sensitive Data in Device Logs	3	1,009	0	1,012	0	1,012	0.049639551	✓
T1547.004	Winlogon Helper DLL	0	1,012	1	1,011	7	1,005	0.004565621	✓
T1546.001	Change Default File Association	0	1,012	0	1,012	2	1,010	0.135156976	-
T1098	Account Manipulation	0	1,012	0	1,012	1	1,011	0.367758249	-

Table B.1 Number of observations and presence of significant differences among sandboxes for each MITRE ATT&CK Technique (RQ1) (3/3).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist		
T1489	Service Stop	0	1,012	17	995	2	1,010	1.10E-06	✓
T1055.012	Process Hollowing	0	1,012	333	679	0	1,012	3.66E-163	✓
T1055.003	Thread Execution Hijacking	0	1,012	39	973	0	1,012	6.95E-18	✓
T1497.003	Time Based Evasion	0	1,012	326	686	0	1,012	2.45E-159	✓
T1071.001	Web Protocols	0	1,012	161	851	0	1,012	1.46E-74	✓
T1204	User Execution	0	1,012	41	971	0	1,012	8.92E-19	✓
T1137	Office Application Startup	0	1,012	35	977	0	1,012	4.19E-16	✓
T1074.001	Local Data Staging	0	1,012	83	929	0	1,012	8.72E-38	✓
T1053.005	Scheduled Task	0	1,012	115	897	0	1,012	1.23E-52	✓
T1059.003	Windows Command Shell	0	1,012	103	909	0	1,012	4.97E-47	✓
T1036.005	Match Legitimate Name or Location	0	1,012	12	1,000	0	1,012	5.86E-06	✓
T1565	Data Manipulation	0	1,012	50	962	0	1,012	8.35E-23	✓
T1218.007	Msiexec	0	1,012	7	1,005	0	1,012	0.000897249	✓
T1132.001	Standard Encoding	0	1,012	28	984	0	1,012	5.33E-13	✓
T1402	Broadcast Receivers	0	1,012	1	1,011	0	1,012	0.367758249	-
T1420	File and Directory Discovery	0	1,012	1	1,011	0	1,012	0.367758249	-
T1582	SMS Control	0	1,012	1	1,011	0	1,012	0.367758249	-
T1204.002	Malicious File	0	1,012	15	997	0	1,012	2.84E-07	✓
T1070.001	Clear Windows Event Logs	0	1,012	20	992	0	1,012	1.81E-09	✓
T1559.001	Component Object Model	0	1,012	16	996	0	1,012	1.03E-07	✓
T1218	Signed Binary Proxy Execution	0	1,012	1	1,011	0	1,012	0.367758249	-
T1059.005	Visual Basic	0	1,012	14	998	0	1,012	7.79E-07	✓
T1114.001	Local Email Collection	0	1,012	3	1,009	0	1,012	0.049639551	✓
T1222.001	Windows File and Directory Permissions Modification	0	1,012	3	1,009	0	1,012	0.049639551	✓
T1546.007	Netsh Helper DLL	0	1,012	2	1,010	0	1,012	0.135156976	-
T1566.001	Spearphishing Attachment	0	1,012	1	1,011	0	1,012	0.367758249	-
T1560.002	Archive via Library	0	1,012	1	1,011	0	1,012	0.367758249	-
T1056.001	Keylogging	0	1,012	4	1,008	0	1,012	0.018219241	✓
T1048.003	Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol	0	1,012	1	1,011	0	1,012	0.367758249	-
T1059.007	JavaScript	0	1,012	1	1,011	0	1,012	0.367758249	-
T1055.001	Dynamic-link Library Injection	0	1,012	1	1,011	0	1,012	0.367758249	-

Table B.2 Presence of significant differences between malware and benign files for each technique (RQ3) (1/2).

TID	Technique	Malicious		Clean		p-value	Statistical significance
		exist	unexist	exist	unexist		
T1573	Encrypted Channel	11,855	1,260	678	855	0	✓
T1518.001	Security Software Discovery	11,364	1,751	508	1,025	0	✓
T1071	Application Layer Protocol	10,920	2,195	382	1,151	0	✓
T1082	System Information Discovery	10,352	2,763	919	614	2.26E-62	✓
T1055	Process Injection	10,055	3,060	1,138	395	0.036382082	-
T1027	Obfuscated Files or Information	9,523	3,592	385	1,148	0.00E+00	✓
T1057	Process Discovery	9,081	4,034	529	1,004	2.80E-161	✓
T1036	Masquerading	8,760	4,355	948	585	0.000116312	✓
T1560	Archive Collected Data	8,741	4,374	394	1,139	7.43E-215	✓
T1497	Virtualization/Sandbox Evasion	8,637	4,478	272	1,261	1.75E-291	✓
T1095	Non-Application Layer Protocol	8,310	4,805	285	1,248	2.40E-248	✓
T1027.002	Software Packing	7,930	5,185	88	1,445	0	✓
T1018	Remote System Discovery	7,676	5,439	155	1,378	8.76E-283	✓
T1083	File and Directory Discovery	6,796	6,319	958	575	2.90E-15	✓
T1562.001	Disable or Modify Tools	6,406	6,709	187	1,346	1.17E-163	✓
T1105	Ingress Tool Transfer	6,352	6,763	347	1,186	8.29E-82	✓
T1140	Deobfuscate/Decode Files or Information	6,332	6,783	203	1,330	5.11E-150	✓
T1003	OS Credential Dumping	4,941	8,174	1	1,532	1.66E-190	✓
T1571	Non-Standard Port	4,940	8,175	27	1,506	2.21E-173	✓
T1010	Application Window Discovery	4,719	8,396	123	1,410	3.57E-107	✓
T1005	Data from Local System	4,171	8,944	8	1,525	6.19E-145	✓
T1106	Native API	4,152	8,963	232	1,301	1.37E-40	✓
T1124	System Time Discovery	4,085	9,030	268	1,265	2.23E-28	✓
T1056	Input Capture	3,996	9,119	129	1,404	1.67E-73	✓
T1047	Windows Management Instrumentation	3,491	9,624	17	1,516	2.36E-108	✓
T1059	Command and Scripting Interpreter	3,015	10,100	243	1,290	2.51E-10	✓
T1012	Query Registry	2,958	10,157	229	1,304	1.00E-11	✓
T1033	System Owner/User Discovery	2,828	10,287	82	1,451	5.30E-51	✓
T1114	Email Collection	2,761	10,354	8	1,525	9.05E-84	✓
T1087	Account Discovery	2,730	10,385	55	1,478	3.03E-59	✓
T1070.006	Timestamp	2,183	10,932	46	1,487	9.40E-45	✓
T1070.004	File Deletion	2,138	10,977	77	1,456	3.03E-31	✓
T1134	Access Token Manipulation	1,984	11,131	133	1,400	1.38E-11	✓
T1115	Clipboard Data	1,950	11,165	63	1,470	8.49E-31	✓
T1203	Exploitation for Client Execution	1,823	11,292	80	1,453	1.63E-21	✓
T1547.001	Registry Run Keys / Startup Folder	1,823	11,292	84	1,449	2.69E-20	✓
T1552.002	Credentials in Registry	1,741	11,374	0	1,533	6.97E-52	✓
T1546.011	Application Shimming	1,735	11,380	83	1,450	2.32E-18	✓
T1574.002	DLL Side-Loading	1,493	11,622	194	1,339	0.151912084	-
T1053	Scheduled Task/Job	1,400	11,715	12	1,521	3.72E-35	✓
T1564.001	Hidden Files and Directories	1,384	11,731	10	1,523	1.34E-35	✓
T1016	System Network Configuration Discovery	1,232	11,883	0	1,533	8.40E-36	✓
T1529	System Shutdown/Reboot	1,176	11,939	113	1,420	0.041438039	-
T1218.011	Rundll32	1,169	11,946	72	1,461	2.67E-08	✓
T1543.003	Windows Service	930	12,185	73	1,460	0.000770065	✓
T1129	Shared Modules	920	12,195	0	1,533	1.63E-26	✓
T1569.002	Service Execution	845	12,270	27	1,506	3.50E-13	✓
T1113	Screen Capture	658	12,457	19	1,514	4.06E-11	✓
T1068	Exploitation for Privilege Escalation	597	12,518	68	1,465	0.886976911	-
T1552.001	Credentials in Files	564	12,551	0	1,533	2.21E-16	✓
T1112	Modify Registry	557	12,558	9	1,524	3.28E-12	✓
T1078	Valid Accounts	529	12,586	11	1,522	1.13E-10	✓
T1219	Remote Access Software	526	12,589	0	1,533	2.51E-15	✓
T1056.004	Credential API Hooking	521	12,594	0	1,533	3.45E-15	✓
T1222	File and Directory Permissions Modification	470	12,645	5	1,528	1.62E-11	✓
T1014	Rootkit	399	12,716	0	1,533	7.85E-12	✓
T1037.005	Startup Items	323	12,792	2	1,531	7.70E-09	✓
T1007	System Service Discovery	320	12,795	4	1,529	6.76E-08	✓
T1120	Peripheral Device Discovery	305	12,810	90	1,443	1.01E-15	✓
T1059.001	PowerShell	289	12,826	0	1,533	7.77E-09	✓
T1070	Indicator Removal on Host	286	12,829	8	1,525	1.82E-05	✓

Table B.2 Presence of significant differences between malware and benign files for each technique (RQ3) (2/2).

TID	Technique	Malicious		Clean		p-value	Statistical significance
		exist	unexist	exist	unexist		
T1486	Data Encrypted for Impact	271	12,844	5	1,528	3.44E-06	✓
T1102	Web Service	252	12,863	0	1,533	7.84E-08	✓
T1218.010	Regsvr32	241	12,874	18	1,515	0.077975619	-
T1091	Replication Through Removable Media	225	12,890	86	1,447	3.59E-23	✓
T1406	Obfuscated Files or Information	201	12,914	11	1,522	0.015720353	-
T1507	Network Information Discovery	195	12,920	13	1,520	0.059249023	-
T1426	System Information Discovery	194	12,921	11	1,522	0.022177882	-
T1421	System Network Connections Discovery	188	12,927	13	1,520	0.080383406	-
T1447	Delete Device Data	184	12,931	10	1,523	0.020631478	-
T1424	Process Discovery	163	12,952	9	1,524	0.033169614	-
T1185	Man in the Browser	150	12,965	1	1,532	0.000132267	✓
T1418	Application Discovery	147	12,968	2	1,531	0.000427987	✓
T1574.010	Services File Permissions Weakness	144	12,971	17	1,516	0.927883606	-
T1136	Create Account	140	12,975	2	1,531	0.000660962	✓
T1564.002	Hidden Users	116	12,999	0	1,533	0.000393093	✓
T1055.011	Extra Window Memory Injection	109	13,006	18	1,515	0.220443697	-
T1499	Endpoint Denial of Service	101	13,014	0	1,533	0.001020654	✓
T1422	System Network Configuration Discovery	97	13,018	2	1,531	0.009605471	✓
T1090	Proxy	97	13,018	0	1,533	0.001317978	✓
T1523	Evade Analysis Environment	95	13,020	0	1,533	0.00149803	✓
T1080	Taint Shared Content	81	13,034	0	1,533	0.003689418	✓
T1548.002	Bypass User Access Control	77	13,038	0	1,533	0.004782079	✓
T1547.008	LSASS Driver	75	13,040	0	1,533	0.005446388	✓
T1429	Capture Audio	71	13,044	3	1,530	0.10609741	-
T1491	Defacement	69	13,046	0	1,533	0.008059561	✓
T1048	Exfiltration Over Alternative Protocol	61	13,054	3	1,530	0.190616084	-
T1001	Data Obfuscation	48	13,067	0	1,533	0.032644518	-
T1542.003	Bootkit	48	13,067	1	1,532	0.089877812	-
T1049	System Network Connections Discovery	46	13,069	1	1,532	0.102735537	-
T1564.004	NTFS File Attributes	43	13,072	0	1,533	0.045959464	-
T1090.003	Multi-hop Proxy	41	13,074	0	1,533	0.052771664	-
T1566.002	Spearphishing Link	39	13,076	84	1,449	6.37E-97	✓
T1490	Inhibit System Recovery	29	13,086	0	1,533	0.123719937	-
T1021.001	Remote Desktop Protocol	28	13,087	0	1,533	0.133131673	-
T1189	Drive-by Compromise	26	13,089	45	1,488	4.70E-47	✓
T1553.004	Install Root Certificate	25	13,090	1	1,532	0.433632044	-
T1564.003	Hidden Window	23	13,092	0	1,533	0.19357285	-
T1547.006	Kernel Modules and Extensions	22	13,093	0	1,533	0.20900378	-
T1135	Network Share Discovery	20	13,095	0	1,533	0.24206902	-
T1562.004	Disable or Modify System Firewall	16	13,099	0	1,533	0.337188376	-
T1574.001	DLL Search Order Hijacking	15	13,100	33	1,500	1.65E-38	✓
T1433	Access Call Log	14	13,101	0	1,533	0.399175466	-
T1564	Hide Artifacts	11	13,104	0	1,533	0.521084905	-
T1543.002	Systemd Service	10	13,105	0	1,533	0.572197451	-
T1176	Browser Extensions	7	13,108	1	1,532	0.69681483	-
T1110	Brute Force	6	13,109	0	1,533	0.864492365	-
T1546.012	Image File Execution Options Injection	5	13,110	0	1,533	0.972864077	-
T1046	Network Service Scanning	5	13,110	0	1,533	0.972864077	-
T1197	BITS Jobs	3	13,112	0	1,533	0.72565624	-
T1546.006	LC_LOAD_DYLIB Addition	3	13,112	0	1,533	0.72565624	-
T1543.001	Launch Agent	3	13,112	0	1,533	0.72565624	-
T1547.011	Plist Modification	3	13,112	5	1,528	2.32E-05	✓
T1040	Network Sniffing	3	13,112	0	1,533	0.72565624	-
T1211	Exploitation for Defense Evasion	2	13,113	0	1,533	0.501883284	-
T1056.002	GUI Input Capture	2	13,113	0	1,533	0.501883284	-
T1532	Data Encrypted	2	13,113	0	1,533	0.501883284	-
T1218.005	Mshta	2	13,113	0	1,533	0.501883284	-
T1553.002	Code Signing	2	13,113	1	1,532	0.72565624	-
T1132	Data Encoding	1	13,114	2	1,531	0.025273731	-
T1573.002	Asymmetric Cryptography	1	13,114	0	1,533	0.196511029	-
T1210	Exploitation of Remote Services	0	13,115	2	1,531	0.00286684	✓

Table B.3 Technique observed in multiple methods and presence/absence of significant differences between methods (RQ4) (1/4).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		Combination	p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist			
T1055	Process Injection	10,690	15,388	0	3,918	16	34	sandbox+report	0.251052825	-
T1497	Virtualization/Sandbox Evasion	9,577	16,501	2	3,916	4	46	(all) sandbox+static	0	✓
T1497	Virtualization/Sandbox Evasion	9,577	16,501	2	3,916	4	46	(all) sandbox+report	5.99E-06	✓
T1497	Virtualization/Sandbox Evasion	9,577	16,501	2	3,916	4	46	(all) static+report	4.36E-36	✓
T1027.002	Software Packing	8,649	17,429	4	3,914	2	48	(all) sandbox+static	0	✓
T1027.002	Software Packing	8,649	17,429	4	3,914	2	48	(all) sandbox+report	0.000175584	✓
T1027.002	Software Packing	8,649	17,429	4	3,914	2	48	(all) static+report	1.82E-07	✓
T1027	Obfuscated Files or Information	9,530	16,548	1,412	2,506	15	35	(all) sandbox+static	0.551849477	-
T1027	Obfuscated Files or Information	9,530	16,548	1,412	2,506	15	35	(all) sandbox+report	0	✓
T1027	Obfuscated Files or Information	9,530	16,548	1,412	2,506	15	35	(all) static+report	0.46179638	-
T1518.001	Security Software Discovery	11,428	14,650	3	3,915	2	48	(all) sandbox+static	0	✓
T1518.001	Security Software Discovery	11,428	14,650	3	3,915	2	48	(all) sandbox+report	1.07E-07	✓
T1518.001	Security Software Discovery	11,428	14,650	3	3,915	2	48	(all) static+report	8.17E-09	✓
T1057	Process Discovery	9569	16,509	99	3,819	7	43	(all) sandbox+static	0	✓
T1057	Process Discovery	9569	16,509	99	3,819	7	43	(all) sandbox+report	8.45E-16	✓
T1057	Process Discovery	9569	16,509	99	3,819	7	43	(all) static+report	5.16E-06	✓
T1082	System Information Discovery	15,879	10,199	2,416	1,502	11	39	(all) sandbox+static	0.363771896	-
T1082	System Information Discovery	15,879	10,199	2,416	1,502	11	39	(all) sandbox+report	3.48E-300	✓
T1082	System Information Discovery	15,879	10,199	2,416	1,502	11	39	(all) static+report	2.51E-08	✓
T1560	Archive Collected Data	8,750	17,328	0	3,918	3	47	sandbox+report	7.07E-05	✓
T1573	Encrypted Channel	12,093	13,985	0	3,918	1	49	sandbox+report	8.02E-10	✓
T1071	Application Layer Protocol	10,920	15,158	0	3,918	10	40	sandbox+report	0.002797683	✓
T1036	Masquerading	8,851	17,227	0	3,918	5	45	sandbox+report	0.000618542	✓
T1105	Ingress Tool Transfer	6,401	19,677	0	3,918	15	35	sandbox+report	0.464919838	-
T1078	Valid Accounts	529	25,549	0	3,918	11	39	sandbox+report	4.54E-21	✓
T1106	Native API	4,156	21,922	0	3,918	2	48	sandbox+report	0.034712164	✓
T1203	Exploitation for Client Execution	2,415	23,663	0	3,918	2	48	sandbox+report	0.299105373	-
T1569.002	Service Execution	858	25,220	125	3,793	5	45	(all) sandbox+static	0.78040016	-
T1569.002	Service Execution	858	25,220	125	3,793	5	45	(all) sandbox+report	0	✓
T1569.002	Service Execution	858	25,220	125	3,793	5	45	(all) static+report	0.022133283	✓
T1574.002	DLL Side-Loading	1,493	24,585	0	3,918	1	49	sandbox+report	0.407363774	-
T1543.003	Windows Service	1,338	24,740	42	3,876	2	48	(all) sandbox+static	1.93E-29	✓
T1543.003	Windows Service	1,338	24,740	42	3,876	2	48	(all) sandbox+report	1.56E-67	✓
T1543.003	Windows Service	1,338	24,740	42	3,876	2	48	(all) static+report	0.198753535	-
T1068	Exploitation for Privilege Escalation	597	25,481	0	3,918	2	48	sandbox+report	0.737961337	-
T1134	Access Token Manipulation	1,985	24,093	144	3,774	0	50	sandbox+static	4.94E-19	✓
T1140	Deobfuscate/Decode Files or Information	6,337	19,741	122	3,796	11	39	(all) sandbox+static	1.59E-198	✓
T1140	Deobfuscate/Decode Files or Information	6,337	19,741	122	3,796	11	39	(all) sandbox+report	5.10E-51	✓
T1140	Deobfuscate/Decode Files or Information	6,337	19,741	122	3,796	11	39	(all) static+report	3.00E-12	✓
T1070.006	Timestamp	2,183	23,895	17	3,901	2	48	(all) sandbox+static	2.21E-70	✓
T1070.006	Timestamp	2,183	23,895	17	3,901	2	48	(all) sandbox+report	8.05E-07	✓
T1070.006	Timestamp	2,183	23,895	17	3,901	2	48	(all) static+report	0.009351916	✓
T1056	Input Capture	3,999	22,079	0	3,918	5	45	sandbox+report	0.395487368	-
T1124	System Time Discovery	4,099	21,979	0	3,918	2	48	sandbox+report	0.037422522	✓
T1120	Peripheral Device Discovery	1,687	24,391	0	3,918	2	48	sandbox+report	0.673395827	-
T1083	File and Directory Discovery	6,818	19,260	1,748	2,170	12	38	(all) sandbox+static	1.11E-125	✓
T1083	File and Directory Discovery	6,818	19,260	1,748	2,170	12	38	(all) sandbox+report	0	✓
T1083	File and Directory Discovery	6,818	19,260	1,748	2,170	12	38	(all) static+report	0.005565762	✓
T1012	Query Registry	7,460	18,618	724	3,194	4	46	(all) sandbox+static	4.45E-40	✓
T1012	Query Registry	7,460	18,618	724	3,194	4	46	(all) sandbox+report	0	✓
T1012	Query Registry	7,460	18,618	724	3,194	4	46	(all) static+report	0.085716776	-
T1070.004	File Deletion	2,550	23,528	1	3,917	7	43	(all) sandbox+static	2.81E-92	✓
T1070.004	File Deletion	2,550	23,528	1	3,917	7	43	(all) sandbox+report	0.155138889	-
T1070.004	File Deletion	2,550	23,528	1	3,917	7	43	(all) static+report	1.20E-91	✓
T1087	Account Discovery	2,730	23,348	135	3,783	9	41	(all) sandbox+static	5.06E-44	✓
T1087	Account Discovery	2,730	23,348	135	3,783	9	41	(all) sandbox+report	4.16E-172	✓
T1087	Account Discovery	2,730	23,348	135	3,783	9	41	(all) static+report	3.62E-07	✓

Table B.3 Technique observed in multiple methods and presence/absence of significant differences between methods (RQ4) (2/4).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		Combination	p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist			
T1033	System Owner/User Discovery	2,845	23,233	201	3,717	5	45	(all) sandbox+static	8.13E-29	✓
T1033	System Owner/User Discovery	2,845	23,233	201	3,717	5	45	(all) sandbox+report	4.35E-260	✓
T1033	System Owner/User Discovery	2,845	23,233	201	3,717	5	45	(all) static+report	0.221871132	-
T1018	Remote System Discovery	7,846	18,232	0	3,918	5	45	sandbox+report	0.003275006	✓
T1115	Clipboard Data	1,955	24,123	238	3,680	1	49	(all) sandbox+static	0.001601174	✓
T1115	Clipboard Data	1,955	24,123	238	3,680	1	49	(all) sandbox+report	0	✓
T1115	Clipboard Data	1,955	24,123	238	3,680	1	49	(all) static+report	0.365872328	-
T1529	System Shutdown/Reboot	1,176	24,902	41	3,877	2	48	(all) sandbox+static	1.96E-24	✓
T1529	System Shutdown/Reboot	1,176	24,902	41	3,877	2	48	(all) sandbox+report	4.45E-75	✓
T1529	System Shutdown/Reboot	1,176	24,902	41	3,877	2	48	(all) static+report	0.187797059	-
T1070	Indicator Removal on Host	290	25,788	0	3,918	6	44	sandbox+report	4.14E-11	✓
T1003	OS Credential Dumping	4,994	21,084	0	3,918	10	40	sandbox+report	0.978207255	-
T1571	Non-Standard Port	5,205	20,873	0	3,918	3	47	sandbox+report	0.02194729	✓
T1059	Command and Scripting Interpreter	3,122	22,956	1,801	2,117	11	39	(all) sandbox+static	0	✓
T1059	Command and Scripting Interpreter	3,122	22,956	1,801	2,117	11	39	(all) sandbox+report	0	✓
T1059	Command and Scripting Interpreter	3,122	22,956	1,801	2,117	11	39	(all) static+report	0.001203964	✓
T1547.001	Registry Run Keys / Startup Folder	4,302	21,776	104	3,814	5	45	(all) sandbox+static	4.84E-115	✓
T1547.001	Registry Run Keys / Startup Folder	4,302	21,776	104	3,814	5	45	(all) sandbox+report	1.25E-66	✓
T1547.001	Registry Run Keys / Startup Folder	4,302	21,776	104	3,814	5	45	(all) static+report	0.006481233	✓
T1010	Application Window Discovery	4,897	21,181	1,096	2,822	0	50	sandbox+static	6.03E-41	✓
T1016	System Network Configuration Discovery	1,483	24,595	89	3,829	3	47	(all) sandbox+static	5.29E-19	✓
T1016	System Network Configuration Discovery	1,483	24,595	89	3,829	3	47	(all) sandbox+report	4.44E-186	✓
T1016	System Network Configuration Discovery	1,483	24,595	89	3,829	3	47	(all) static+report	0.204823814	-
T1113	Screen Capture	664	25,414	403	3,515	3	47	(all) sandbox+static	7.12E-131	✓
T1113	Screen Capture	664	25,414	403	3,515	3	47	(all) sandbox+report	0	✓
T1113	Screen Capture	664	25,414	403	3,515	3	47	(all) static+report	0.447946249	-
T1486	Data Encrypted for Impact	290	25,788	0	3,918	11	39	sandbox+report	1.41E-39	✓
T1053	Scheduled Task/Job	2,787	23,291	0	3,918	12	38	sandbox+report	0.004923383	✓
T1562.001	Disable or Modify Tools	6,784	19,294	0	3,918	9	41	sandbox+report	0.258742312	-
T1112	Modify Registry	4,886	21,192	195	3,723	6	44	(all) sandbox+static	1.82E-101	✓
T1112	Modify Registry	4,886	21,192	195	3,723	6	44	(all) sandbox+report	9.55E-132	✓
T1112	Modify Registry	4,886	21,192	195	3,723	6	44	(all) static+report	0.054137315	-
T1005	Data from Local System	8,036	18,042	0	3,918	3	47	sandbox+report	0.000267483	✓
T1114	Email Collection	3,995	22,083	0	3,918	2	48	sandbox+report	0.042887734	✓
T1047	Windows Management Instrumentation	3,542	22,536	8	3,910	3	47	(all) sandbox+static	8.44E-129	✓
T1047	Windows Management Instrumentation	3,542	22,536	8	3,910	3	47	(all) sandbox+report	0.991008892	-
T1047	Windows Management Instrumentation	3,542	22,536	8	3,910	3	47	(all) static+report	1.64E-10	✓
T1222	File and Directory Permissions Modification	628	25,450	237	3,681	1	49	(all) sandbox+static	1.17E-36	✓
T1222	File and Directory Permissions Modification	628	25,450	237	3,681	1	49	(all) sandbox+report	0	✓
T1222	File and Directory Permissions Modification	628	25,450	237	3,681	1	49	(all) static+report	0.368942641	-
T1189	Drive-by Compromise	26	26,052	0	3,918	2	48	sandbox+report	3.90E-10	✓
T1102	Web Service	588	25,490	0	3,918	2	48	sandbox+report	0.72375299	-
T1014	Rootkit	399	25,679	0	3,918	1	49	sandbox+report	0.759558725	-
T1059.001	PowerShell	386	25,692	0	3,918	14	36	sandbox+report	8.40E-49	✓
T1007	System Service Discovery	320	25,758	26	3,892	2	48	(all) sandbox+static	0.002703009	✓
T1007	System Service Discovery	320	25,758	26	3,892	2	48	(all) sandbox+report	9.37E-138	✓
T1007	System Service Discovery	320	25,758	26	3,892	2	48	(all) static+report	0.051117737	-
T1219	Remote Access Software	526	25,552	0	3,918	6	44	sandbox+report	7.05E-06	✓
T1406	Obfuscated Files or Information	201	25,877	0	3,918	3	47	sandbox+report	0.00069151	✓
T1523	Evade Analysis Environment	95	25,983	0	3,918	1	49	sandbox+report	0.459299844	-
T1426	System Information Discovery	194	25,884	0	3,918	1	49	sandbox+report	0.834750965	-
T1448	Carrier Billing Fraud	140	25,938	0	39,18	1	49	sandbox+report	0.656507257	-
T1418	Application Discovery	148	25,930	0	3,918	1	49	sandbox+report	0.686269057	-
T1409	Access Stored Application Data	68	26,010	0	3,918	1	49	sandbox+report	0.310144153	-
T1422	System Network Configuration Discovery	97	25,981	0	3,918	1	49	sandbox+report	0.469326426	-
T1430	Location Tracking	172	25,906	0	3,918	1	49	sandbox+report	0.768093651	-
T1507	Network Information Discovery	195	25,883	0	3,918	1	49	sandbox+report	0.837615936	-
T1472	Generate Fraudulent Advertising Revenue	100	25,978	0	3,918	3	47	sandbox+report	1.97E-07	✓

Table B.3 Technique observed in multiple methods and presence/absence of significant differences between methods (RQ4) (3/4).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		Combination	p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist			
T1129	Shared Modules	920	25,158	3,392	526	1	49	(all) sandbox+static	0	✓
T1129	Shared Modules	920	25,158	3,392	526	1	49	(all) sandbox+report	0	✓
T1129	Shared Modules	920	25,158	3,392	526	1	49	(all) static+report	1.84E-62	✓
T1136	Create Account	140	25,938	1	3,917	1	49	(all) sandbox+static	2.26E-05	✓
T1136	Create Account	140	25,938	1	3,917	1	49	(all) sandbox+report	0.656507257	-
T1136	Create Account	140	25,938	1	3,917	1	49	(all) static+report	0.002607017	✓
T1049	System Network Connections Discovery	78	26,000	0	3,918	2	48	sandbox+report	0.000558551	✓
T1499	Endpoint Denial of Service	101	25,977	6	3,912	1	49	(all) sandbox+static	0.031667893	✓
T1499	Endpoint Denial of Service	101	25,977	6	3,912	1	49	(all) sandbox+report	7.16E-29	✓
T1499	Endpoint Denial of Service	101	25,977	6	3,912	1	49	(all) static+report	0.162536612	-
T1566.002	Spearphishing Link	42	26,036	0	3,918	4	46	sandbox+report	1.03E-30	✓
T1513	Screen Capture	26	26,052	0	3,918	1	49	sandbox+report	0.048237909	✓
T1080	Taint Shared Content	81	25,997	0	3,918	1	49	sandbox+report	0.385245941	-
T1021.001	Remote Desktop Protocol	309	25,769	0	3,918	8	42	sandbox+report	4.96E-19	✓
T1490	Inhibit System Recovery	61	26,017	1	3,917	6	44	(all) sandbox+static	0.0127993	✓
T1490	Inhibit System Recovery	61	26,017	1	3,917	6	44	(all) sandbox+report	0.22796386	-
T1490	Inhibit System Recovery	61	26,017	1	3,917	6	44	(all) static+report	3.07E-75	✓
T1185	Man in the Browser	150	25,928	0	3,918	2	48	sandbox+report	0.024410287	✓
T1048	Exfiltration Over Alternative Protocol	61	26,017	0	3,918	4	46	sandbox+report	8.62E-22	✓
T1090.003	Multi-hop Proxy	41	26,037	0	3,918	1	49	sandbox+report	0.138137537	-
T1090	Proxy	116	25,962	0	3,918	5	45	sandbox+report	5.61E-19	✓
T1564.003	Hidden Window	26	26,052	516	3,402	0	50	sandbox+static	0	✓
T1132	Data Encoding	2	26,076	0	3,918	2	48	sandbox+report	2.28E-65	✓
T1553.004	Install Root Certificate	807	25,271	0	3,918	1	49	sandbox+report	0.969842387	-
T1001	Data Obfuscation	48	26,030	0	3,918	3	47	sandbox+report	1.31E-14	✓
T1564	Hide Artifacts	11	26,067	6	3,912	0	50	sandbox+static	0.018224449	✓
T1562.004	Disable or Modify System Firewall	25	26,053	3	3,915	2	48	(all) sandbox+static	0.929678359	-
T1562.004	Disable or Modify System Firewall	25	26,053	3	3,915	2	48	(all) sandbox+report	1.11E-25	✓
T1562.004	Disable or Modify System Firewall	25	26,053	3	3,915	2	48	(all) static+report	8.17E-09	✓
T1548.002	Bypass User Access Control	122	25,956	0	3,918	1	49	sandbox+report	0.584212563	-
T1564.004	NTFS File Attributes	210	25,868	0	3,918	1	49	sandbox+report	0.879043574	-
T1546.012	Image File Execution Options Injection	5	26,073	0	3,918	1	49	sandbox+report	5.02E-06	✓
T1135	Network Share Discovery	21	26,057	21	3,897	3	47	(all) sandbox+static	6.00E-12	✓
T1135	Network Share Discovery	21	26,057	21	3,897	3	47	(all) sandbox+report	0	✓
T1135	Network Share Discovery	21	26,057	21	3,897	3	47	(all) static+report	5.49E-05	✓
T1553.002	Code Signing	58	26,020	0	3,918	3	47	sandbox+report	2.74E-12	✓
T1546.004	.bash_profile and .bashrc	3	26,075	1	3,917	0	50	sandbox+static	0.973401992	-
T1110	Brute Force	6	26,072	0	3,918	1	49	sandbox+report	2.57E-05	✓
T1046	Network Service Scanning	7	26,071	0	3,918	3	47	sandbox+report	4.44E-72	✓
T1532	Data Encrypted	2	26,076	0	3,918	1	49	sandbox+report	6.58E-11	✓
T1218.005	Mshta	9	26,069	0	3,918	2	48	sandbox+report	1.87E-24	✓
T1573.002	Asymmetric Cryptography	1	26,077	0	3,918	3	47	sandbox+report	7.32E-179	✓
T1547.004	Winlogon Helper DLL	79	25,999	6	3,912	0	50	sandbox+static	0.137936971	-
T1098	Account Manipulation	29	26,049	1	3,917	3	47	(all) sandbox+static	0.189859721	-
T1098	Account Manipulation	29	26,049	1	3,917	3	47	(all) sandbox+report	0.057723068	-
T1098	Account Manipulation	29	26,049	1	3,917	3	47	(all) static+report	4.46E-28	✓
T1489	Service Stop	22	26,056	25	3,893	7	43	(all) sandbox+static	1.81E-15	✓
T1489	Service Stop	22	26,056	25	3,893	7	43	(all) sandbox+report	0	✓
T1489	Service Stop	22	26,056	25	3,893	7	43	(all) static+report	2.97E-22	✓
T1055.012	Process Hollowing	349	25,729	18	3,900	0	50	sandbox+static	4.48E-06	✓
T1055.003	Thread Execution Hijacking	41	26,037	24	3,894	0	50	sandbox+static	3.19E-08	✓
T1071.001	Web Protocols	173	25,905	0	3,918	7	43	sandbox+report	5.99E-26	✓
T1204	User Execution	48	26,030	0	3,918	7	43	sandbox+report	7.90E-87	✓
T1053.005	Scheduled Task	120	25,958	9	3,909	3	47	(all) sandbox+static	0.054301071	-
T1053.005	Scheduled Task	120	25,958	9	3,909	3	47	(all) sandbox+report	1.59E-55	✓
T1053.005	Scheduled Task	120	25,958	9	3,909	3	47	(all) static+report	1.14E-09	✓
T1059.003	Windows Command Shell	108	25,970	16	3,902	5	45	(all) sandbox+static	0.935416564	-
T1059.003	Windows Command Shell	108	25,970	16	3,902	5	45	(all) sandbox+report	3.97E-177	✓
T1059.003	Windows Command Shell	108	25,970	16	3,902	5	45	(all) static+report	9.74E-17	✓

Table B.3 Technique observed in multiple methods and presence/absence of significant differences between methods (RQ4) (4/4).

TID	Technique	JoeSandbox		Hybrid Analysis		Hatching Triage		Combination	p-value	Statistical significance
		exist	unexist	exist	unexist	exist	unexist			
T1036.005	Match Legitimate Name or Location	13	26,065	0	3,918	1	49	sandbox+report	0.003796366	✓
T1565	Data Manipulation	51	26,027	0	3,918	1	49	sandbox+report	0.203352928	-
T1132.001	Standard Encoding	31	26,047	0	3,918	6	44	sandbox+report	7.77E-93	✓
T1402	Broadcast Receivers	1	26,077	0	3,918	5	45	sandbox+report	0	✓
T1420	File and Directory Discovery	1	26,077	0	3,918	1	49	sandbox+report	9.89E-16	✓
T1204.002	Malicious File	18	26,060	0	3,918	4	46	sandbox+report	6.76E-64	✓
T1070.001	Clear Windows Event Logs	23	26,055	0	3,918	3	47	sandbox+report	3.79E-28	✓
T1218	Signed Binary Proxy Execution	1	26,077	0	3,918	2	48	sandbox+report	9.54E-87	✓
T1059.005	Visual Basic	14	26,064	0	3,918	3	47	sandbox+report	1.05E-42	✓
T1566.001	Spearphishing Attachment	3	26,075	0	3,918	4	46	sandbox+report	8.54E-200	✓
T1560.002	Archive via Library	3	26,075	9	3,909	1	49	(all) sandbox+static	2.85E-09	✓
T1560.002	Archive via Library	3	26,075	9	3,909	1	49	(all) sandbox+report	0	✓
T1560.002	Archive via Library	3	26,075	9	3,909	1	49	(all) static+report	0.288413195	-
T1056.001	Keylogging	4	26,074	532	3,386	1	49	(all) sandbox+static	0	✓
T1056.001	Keylogging	4	26,074	532	3,386	1	49	(all) sandbox+report	0	✓
T1056.001	Keylogging	4	26,074	532	3,386	1	49	(all) static+report	0.029476232	✓
T1048.003	Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol	1	26,077	0	3,918	1	49	sandbox+report	9.89E-16	✓
T1059.007	JavaScript	1	26,077	0	3,918	2	48	sandbox+report	9.54E-87	✓
T1055.001	Dynamic-link Library Injection	1	26,077	4	3,914	0	50	sandbox+static	0.000157776	✓