Attack Trace Dataset Generation Considering Organizational System Characteristics

1st Masahito Kumazaki
Kyoto University
Kyoto, Japan
kumazaki@inet.media.kyoto-u.ac.jp

2nd Daisuke Kotani
Kyoto University
Kyoto, Japan
kotani@media.kyoto-u.ac.jp

3rd Yasuo Okabe
Kyoto University
Kyoto, Japan
Okabe@media.kyoto-u.ac.jp

***Abstract*—In the context of cyberattack response, leveraging damage states that can potentially be reported earlier than data requiring specialized analysis may lead to reduced response times and mitigated damage. However, while datasets such as those for attack sequences exist, there is a lack of datasets that correlate attack methods with their respective damage states. This paper proposes a dataset generation tool for attack traces, including mappings between attack methods and damage states. The proposed tool operates within a virtual environment that replicates organizational settings, automatically executing attacks and collecting information. This approach addresses challenges such as variations in data caused by different environments and the burden of manual data collection.**

***Index Terms*—dataset generation, attack trace**

I. Introduction

Cyberattack damage targeting organizations, such as corporations, is a serious issue, necessitating effective damage mitigation measures. Responding to such cyberattacks involves a series of activities, including preemptive measures such as multi-layered defenses with firewalls, attack detection and analysis, containment to prevent damage escalation, malware removal and system recovery, and post-attack countermeasure evaluation [1]. However, because these attacks often involve thorough preliminary reconnaissance and use techniques tailored to the target organization, achieving complete intrusion prevention is challenging, highlighting the limitations of relying solely on preemptive defenses. Consequently, there is a need not only for intrusion prevention but also for rapid attack detection and response to contain the scope of compromise and reduce overall damage. In the stages of post-detection analysis and response, organizations gather attack traces through investigations of logs and security device alerts. Based on these traces, they identify the attack methods used, assess the progression stage of the attack, and take containment actions as necessary. Although there exist multiple technologies and research initiatives that support analysis and response efforts [2] [3], these tools are often designed with the assumption that users have a certain level of security expertise. Consequently, the time required for trace analysis depends significantly on the skill level of the responder, creating a risk of delayed response in organizations that lack specialized response teams such as CSIRTs.

To address the above-mentioned challenges, an incident response approach that relies less on the responder's skill level and instead leverages information more accessible to non-experts is a potential solution. This paper focuses on damage states caused by cyberattacks as one such approach. Although comprehensive investigation is required to fully assess all damage, it is reasonable to assume that some degree of damage —such as alert notifications or system malfunctions—will be identifiable as soon as the attack is detected. Additionally, significant damages that disrupt normal business operations are likely to be recognized early, even by personnel without specialized expertise, such as a CSIRT. By narrowing the investigative focus based on observed damage states, it may be possible to reduce the impact of cyberattacks. However, to evaluate and validate technologies supporting this type of incident response [4], each organization must identify the specific damage states associated with various attack methods. Although datasets detailing attack sequences exist [5], there is a lack of datasets linking these sequences with corresponding damage states. Furthermore, information that includes sensitive damage states often depends significantly on the applications and device configurations unique to each organization. As a result, each organization must prepare data tailored to its specific environment, a process that incurs significant costs.

This paper proposes a dataset generation tool for attack traces, which includes mappings between attack methods and corresponding damage states. The proposed tool operates within a virtual environment that replicates each organization's specific device and network configurations, allowing it to account for variations in damage states across different environments. By automating the selection and execution of attack methods, as well as post-attack data collection, the tool reduces the burden of manually designing attack scenarios during data collection. The information on attack techniques, required tools, and execution commands is sourced from red team attack tools. The tool is designed to use essential attack-related information, such as target IP addresses, usernames, and passwords, which the user provides in a prepared file. Based on this gathered information, the tool selects the appropriate attack for the target, then collects various logs post-attack and records the file differences from before and after the attack as indicators of damage. Furthermore, we developed a prototype of the proposed tool and conducted functional verification in a small-scale environment. While the results

confirmed successful automation of attack execution and data collection, they also highlighted areas needing improvement, particularly in the methods for capturing file differences as damage indicators and in mapping damage states for attack methods that do not produce logs. Additionally, we analyzed elements that can serve as early indicators of damage and examined supplementary information that should be collected.

II. Related Works

A. Red-Team Tools

In this paper, we utilize security evaluation tools capable of reproducing multi-stage attack sequences and operable across multiple platforms as our attack tools. Examples of such tools include Metasploit, provided by Rapid7 [6], and Atomic Red Team, developed by Red Canary [7].

These attack tools are designed to operate based on attack methods categorized within the MITRE ATT&CK framework [8], which provides a structured taxonomy for understanding and simulating cyberattack tactics and techniques [9]. MITRE ATT&CK is a knowledge base, published by the U.S. nonprofit organization MITRE, that catalogs adversarial Tactics and Techniques used in cyberattacks. In ATT&CK, the stages of an attack sequence are segmented into units called "Tactics," each representing a specific attack objective. By combining various Tactics, the framework allows for the replication of complex attack sequences. Examples of Tactics include "Initial Access," which aims at gaining entry to a network, and "Credential Access," which focuses on credential theft. Furthermore, within each Tactic, ATT&CK organizes the attack methods employed into units called "Techniques."

The attack tools referenced in this paper are assumed to enable detailed simulated attacks by recreating attack sequences through Tactic combinations and selecting specific attack methods, or Techniques, within each Tactic. Each of these tools has distinct features. For instance, CALDERA [10] provides a user-friendly GUI that allows customization, execution, and result verification of attacks. In this study, we select Atomic Red Team as our tool due to its suitability for automation and its accessibility, offering plaintext descriptions of each Technique ' s execution tool and command.

B. CyberAttack Dataset

Existing datasets for validating cyberattacks include those developed for the evaluation of Intrusion Detection Systems (IDS) [11] [12] [13] and for malware detection [14] [15]. However, there is limited research focusing on obtaining comprehensive attack data that is not dependent on specific devices [5] [16].

Takahashi et al. proposed APTGen, an approach for generating datasets targeting advanced persistent threats (APT) [5]. APTGen generates attack sequences based on MITRE ATT&CK by analyzing existing incident reports and replicates these sequences by selecting appropriate attack tools based on execution environment logs. Subsequently, simulated attacks are conducted in the test environment using the generated sequence information, and logs corresponding to each sequence are collected to create a dataset. Compared to previous studies on the automated generation of attack sequences [17], APTGen emphasizes the enumeration of feasible attack sequences and the selection of attack tools for execution within the environment, facilitating the generation of datasets that include these sequences and their associated logs. However, the data associated with attack sequences are limited to logs, without inclusion of environment-dependent damage information.

Tsuda et al. introduced STARDUST, a platform designed to collect data from targeted attacks [16]. STARDUST constructs a parallel network that mimics the organization ' s real network and, upon intrusion, redirects attackers to the parallel network, enabling the collection of data produced by attack sequences on the organizational network. However, when creating datasets using STARDUST, it is necessary for the organization to devise its own attack scenarios. Additionally, both APTGen and STARDUST require users to conduct the attacks and gather information, resulting in considerable cost in dataset creation.

C. Attack Detection and Response Techniques

Information that serves as indicators of cyberattacks, such as suspicious communications or excessive access requests to confidential data, is utilized not only for preemptive measures like blacklist-based defenses but also for post-intrusion detection and response efforts. Kanemoto et al. proposed a method that uses attack traces extracted from emulation results of attack code to assess the success of attacks, enabling the identification of relevant alerts within the large volume generated by IDS during attack detection [2]. Additionally, technologies supporting the sharing of attack trace information between organizations, such as the Malware Information Sharing Platform (MISP), allow for the storage, sharing, and correlation analysis of threat intelligence and attack traces [3].

However, effectively leveraging these detection and response technologies requires prior investigation of attack traces present within one ' s own organization, which may be challenging depending on the skill level of the responder. To address such challenges, research has been conducted with the aim of assisting early response even in cases where responders have limited expertise [18], as well as studies that attempt to identify attack methods based on partial information, such as damage indicators, when detailed attack traces are unavailable [4]. Nonetheless, the task of correlating attack traces with damage states remains the responsibility of the individual organization.

III. Attack Trace Data Set Generation Tool

A. Outline

One type of information considered indicative of damage is file differences before and after an attack. Depending on the attack method, file system changes may occur on the target device, such as malware placement during the initial infiltration or lateral movement stages, or file output of reconnaissance results during privilege escalation and internal reconnaissance stages. Additionally, even for attack methods that aim to leave

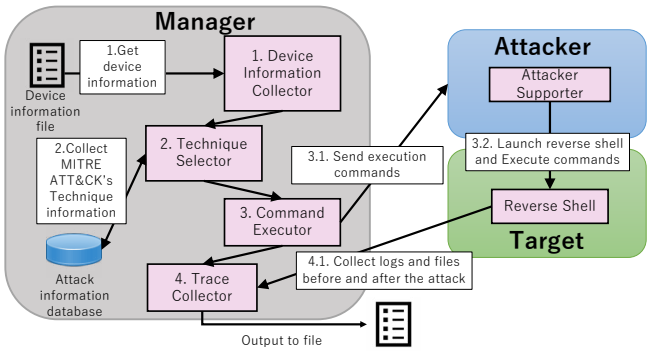

Fig. 1: Structure and Workflow of the Generation Tool

TABLE I: Device Information of Attacker and Target

Tag	Description
IP	IPv4 address of the device
Platform	Platform of the device
Role	Role of the device (e.g., personal terminal, file server)
Executors	Applications avalable on the device
Route	IPv4 address of the router required for each connection
Related	IPv4 address of the related device (e.g., management server)

no residual file differences at the end of an attack, temporary files or other artifacts may be created at intermediate stages. Given that file differences can occur at various stages of attack methods, and that certain methods, like malware placement on shared servers, produce observable damage that multiple individuals can detect, this paper proposes an attack trace generation tool that links attack methods with corresponding damage states represented by file differences.

An overview of the attack trace dataset generation tool is shown in Figure 1. It is assumed that users will prepare device information files containing details about each device, as outlined in Table I. The attack information database uses Atomic Red Team [7]. The proposed tool comprises a Manager, which stores collected information, including device and attack data; an Attacker, which executes commands on targets for attack and information collection; and a Target, the designated attack target. The Attacker, Target, and a router to replicate the organizational network are all created in a virtual environment, with each virtual machine ' s initial state saved using virtualization software's snapshot function. Each snapshot is configured so that upon reboot, a support module launches on the Attacker and a reverse shell on the Target.

After collecting device information from the device information file, the proposed tool selects an appropriate attack method based on conditions such as the target platform and available attack tools. Subsequently, the Attacker, Target, and associated devices are initialized based on IP address information. After restoring the Attacker to its initial snapshot state and rebooting it, the support module registers attack

TABLE II: Output File Format

Tag		Description
Tech_ID		Technique ID from MITRE ATT&CK
Phase		Tactic from MITRE ATT&CK
Platform		Platform of the Target
Target_IP		IPv4 address of the Target
Route		IPv4 address of the router required for each connection
Commands and Results	Commands	Executed commands
	Logs	Collected logs
	Creates	Files generated after the attack
	Changes	Files modified after the attack
	Deletes	Files deleted after the attack

commands to the Attacker. Restoring the Target to its initial state and rebooting it enables command execution from the Attacker via the Target's reverse shell. Before executing the attack commands, the Attacker initiates a command to capture the initial state of the Target's files and sends this initial file state to the Manager for storage. Following attack execution, the Attacker takes a snapshot and transmits the Target's post-attack files and logs to the Manager, then restores the post-attack snapshot to retrieve additional files. Finally, the Manager extracts file differences from the Target ' s files before and after the attack and outputs them in a file along with information on the selected attack method, attacking devices, logs, and file differences. The format of the output file is shown in Table II.

B. Atomic Red Team

The Atomic Red Team [7], used in this paper as an attack information database, is an attack tool that enables simulated attacks based on the MITRE ATT&CK framework. It consists of an index that organizes Tactics and Techniques by platform, as well as files detailing each attack method. This tool allows for attack execution through a Ruby API and provides plaintext access to detailed information on each Technique, including the following attributes.

- Technique ID from MITRE ATT&CK
- Overview of Technique
- Prerequisites for the Attack (e.g., Required Applications)
- Executor
- Command for Executing the Attack
- Command for Concealinf Attack Activity
- Requirement for Elevated Privileges

In the proposed tool, the platform of the Target, obtained from the device information, is used to retrieve a set of attack methods from the index. Next, based on the applications available on the Target, the tool searches for attack methods that meet the required conditions and retrieves relevant information such as the Technique ID, execution application, and execution and concealment commands.

C. Scope of Attacks and Attack Traces Envisioned by the Proposed Tool

In actual attacks, concealment activities are often conducted after a successful intrusion to hide traces of the attack. Additionally, cases may arise where the attack fails or is interrupted

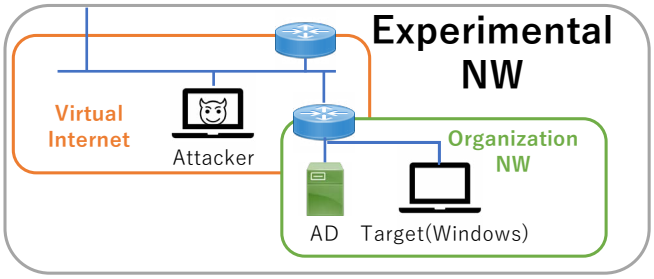

Fig. 2: Prototype Environment for the Functional Verification

TABLE III: Device Information of Attacker and Target

Device	Tag	Description
Attacker	IP	192.168.56.1
	Platform	Ubuntu 22.04
	Role	Personal terminal
	Executor	bash
	Route	192.168.56.254
	Related	None
Target	IP	192.168.30.2
	Platform	Windows11
	Role	Personal terminal
	Executor	command_prompt
	Route	192.168.56.254, 192.168.30.254
	Related	192.168.30.100

mid-execution, resulting in a different damage state than if the attack had completed. The proposed tool anticipates such scenarios by collecting additional damage states for each attack method in cases where the execution command is interrupted midway or when a concealment command is executed after the execution command. Regarding the environment for data collection with the proposed tool, fully replicating actual device and network configurations is challenging from both personnel and equipment cost perspectives. Therefore, the proposed tool addresses this issue by creating a small-scale environment consisting only of the Attacker executing the attack, the Manager as the target, associated devices, and the necessary routers for connectivity. However, this limited environment may miss certain attack traces. Ideally, attack traces encompass various indicators, such as suspicious communication patterns or requests for specific files, command execution histories, and standard output content. In the proposed tool, file differences and logs serve as the main indicators of damage. Nonetheless, some security devices may be configured to log specific events, such as multiple terminals accessing a particular file within a short timeframe. Under such configurations, when testing lateral movement in the proposed tool, logs may not be generated, although they would be expected during an actual lateral movement scenario. In this way, the proposed tool does not account for attack traces based on aggregated statistics from multiple devices; instead, it focuses on traces related to the Target and the devices directly responsible for managing and monitoring it.

IV. Implementation of the Prototype

We implemented a prototype to verify how the proposed tool operates and to evaluate the scope of data collection. We developed each module of the tool in Python and created the virtual environment using VirtualBox[1]. Figure 2 shows the small-scale environment where we conducted the functional verification.

A. Functional Verification with the Prototype

We conducted functional verification of the implemented prototype within the environment shown in Figure 2. Table III presents the device information for the Attacker and Target, and Table IV details the attack methods used. Both attack methods involve saving internal information from the terminal to a temporary file, allowing damage states to be verified through file differences.

In the prototype, after collecting the device information and attack method details, we launched the virtual machines for the Attacker and Target, executing the attack through a reverse shell activated at the Target's virtual machine startup. To capture file differences, we used robocopy to obtain the Target's files before and after the attack. After executing the attack, we used wevtutil to collect application, system, and security logs from the device.

B. Results of Functional Verification

As a result of the functional verification, we confirmed that components related to automation, such as the booting of the Attacker device and the execution of commands on the Target, operated without issues. However, in the data collection phase, the file generated during the execution of Attack Method 1 was successfully identified, whereas the text file generated during the execution of Attack Method 2 could not be retrieved. Figure 3 shows the files generated post-attack execution. To investigate whether the issue stemmed from the execution command, we used robocopy on the Target after the attack to manually obtain the file differences. As the generated text file was successfully identified in this case, we determined that the issue was not with the execution command itself but rather a limitation in the robocopy specifications. Based on these findings, alternative methods for obtaining file differences should be considered. Currently, we are exploring approaches such as comparing snapshots using the Volume Shadow Copy Service (VSS) or transferring files via FTP.

Additionally, all collected logs were empty, which we attribute to the lack of log generation on the terminal. As demonstrated, depending on how logging is configured within an organization, collecting logs from terminals may not provide sufficient information to correlate attack methods with damage states. Furthermore, during reconnaissance phases like directory structure exploration, certain attack methods may avoid generating actions that leave traces in logs. To establish correlations between such attack methods and their damage states, it is necessary to consider whether additional information sources available during attack execution should be collected by the tool.

[1]https://www.virtualbox.org/

TABLE IV: Technique Used for Functional Verification

ID		T1119
Tech_name		Automated Collection
Method 1	Platform	Windows
	Executor	command_prompt
	Command	mkdir %temp%\T1119_command_prompt_collection >nul 2>&1
		dir c: /b /s .docx |findstr /e .docx
		for /R c: %f in (*.docx) do copy %f %temp%\T1119_command_prompt_collection
Method 2	Platform	Windows
	Executor	command_prompt
	Command	sc query type=service >%TEMP%\T1119_1.txt
		doskey /history >%TEMP%\T1119_2.txt
		wmic process list >%TEMP%\T1119_3.txt
		tree C:\AtomicRedTeam\atomics >%TEMP%\T1119_4.txt


```
{
    "Tech_ID": "T1119",
    "Phase": "collection",
    "Platform": "windows",
    "Target_IP": "192.168.30.2",
    "Commands_and_Results": [
        {
            "Commands": [
                "mkdir %temp%\\T1119_command_prompt_collection >nul
                2>&1",
                "dir c: /b /s .docx | findstr /e .docx",
                "for /R c: %f in (*.docx) do copy %f
                %temp%\\T1119_command_prompt_collection"
            ],
            "creates": [
                " T1119_command_prompt_collection"
            ],
            "deletes": [],
            "Logs": {
                "Application": "",
                "Security": "",
                "System": ""
            },
            "changes": []
        },
        {
            "Commands": [
                "sc query type=service > %TEMP%\\T1119_1.txt",
                "doskey /history > %TEMP%\\T1119_2.txt",
                "wmic process list > %TEMP%\\T1119_3.txt",
                "tree C:\\AtomicRedTeam\\atomics > %TEMP%\\T1119_4.
                txt"
            ],
            "creates": [],
            "deletes": [],
            "Logs": {
                "Application": "",
                "Security": "",
                "System": ""
            },
            "changes": []
        }
    ]
}
```


Fig. 3: File Differences and Log Outputs after Execution

C. Future Works

From the results obtained, a key challenge moving forward is determining the types of information to be treated as damage states. Currently, file differences on the Target before and after an attack are used as the primary damage indicator. However, other potential early-detectable damages include disruptions to business continuity, such as device failures or network disconnections, as well as alerts from monitoring devices that may notify incidents before detailed investigations. To accommodate these damage types, we need to explore collecting additional information, such as changes in network connectivity and memory dump files after an attack.

V. Conclusion

In this paper, we proposed a dataset generation tool for attack traces, linking attack methods with damage states, to support the use of early-detectable damage states in cyber-attack responses even in environments with limited security expertise. The proposed tool replicates organizational device and network configurations in a virtual environment to account for variations in damage states across different settings. By automating attack selection, execution, and post-attack data collection, the tool reduces the burden of dataset creation.

Additionally, we developed a prototype of the attack trace dataset generation tool and conducted functional verification. While the results confirmed automation of attack execution and data collection, we identified cases where file differences could not be obtained due to limitations with certain execution commands. Therefore, it is necessary to explore improvements to the current method or investigate alternative approaches for obtaining file differences. Furthermore, to support other damage states that may be identified prior to detailed investigations, we plan to consider expanding data collection to include additional information, such as network connectivity changes and memory dumps.

Acknowledgment

This work was supported by JST SPRING, Grant Number JPMJSP2110.

References

[1] P. Cichonski, T. Millar, T. Grance, and K. Scarfone,"Computer security incident handling guide, NIST Special Publication, 800.61, pp. 1-147, 2012.

[2] Y. Kanemoto, K. Aoki, J. Miyoshi, H. Shimada and H. Takakura, "Detecting Successful Attacks against Web Application based-on Attack Code Emulation (In Japanese)," Journal of Information Processing, Vol. 60, Num. 3, pp. 945-955, 2019.

[3] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, pp. 49-56, 2016.

[4] M. Kumazaki, H. Hasegawa, Y. Yamaguchi, H. Shimada, and H. Takakura, “Cyber Attack Stage Tracing System Based on Attack Scenario Comparison,” the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022), pp. 587-594, 2022.
[5] Y. Takahashi, S. Shima, R. Tanabe, and K. Yoshioka, “APTGen: an approach towards generating practical dataset labelled with targeted attack sequences,” In Proceedings of the 13th USENIX Conference on Cyber Security Experimentation and Test ,p. 8, 2020.
[6] Rapid7, Metasploit, https://www.metasploit.com/, 2024/11/11 Access.
[7] Red Canary, Atomic Red Team, https://github.com/redcanaryco/atomic-red-team, 2024/11/11 Access.
[8] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, “Mitre att&ck: Design and Philosophy, Technical report,” 2018. https://www.mitre.org/publications/technical-papers/mitre-attack-design-and-philosophy
[9] P. Zilberman, R. Puzis, S. Bruskin, S. Shwarz, and Y. Elovici, “Sok: A survey of open-source threat emulators,” arXiv preprint arXiv:2003.01518, 2020.
[10] MITRE, CALDERA, https://caldera.mitre.org/, 2024/11/11 Access.
[11] MIT Litcoln Laboratory, MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation, https://archive.ll.mit.edu/ideval/data/index.html, 2024/11/11 Access.
[12] KDD Cup 1999 Data, https://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/kddcup99.html, 2024/11/11 Access.
[13] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, “Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation,” Proceedings of the first workshop on building analysis datasets and gathering experience returns for security, pp. 29-36, 2011.
[14] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, “Microsoft malware classification challenge,” arXiv preprint arXiv:1802.10135, 2018.
[15] H. S. Anderson, P. Roth. “EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models,” arXiv preprint arXiv:1804.04637, 2018.
[16] Y. Tsuda, T. Tomine, N. Kanaya, D. Makita, H. Ushimaru, M. Jingu, et al.“STARDUST: Large-scale Infrastructure for Luring Cyber Adversaries (in Japanese),” Computer Security Symposium 2017, Vol 2017, No. 2, 2017.
[17] G. Falco, A. Viswanathan, C. Caldera, and H. Shrobe, “A master attack methodology for an AI-based automated attack planner for smart cities,” IEEE Access, 6, pp. 48360-48373, 2018.
[18] M. Kumazaki, H. Hasegawa, Y. Yamaguchi, H. Shimada, and H. Takakura, “Incident Response Support System for Multi-Located Network by Correlation Analysis of Individual Events,” The 4th International Conference on Information Science and Systems, pp. 1-6, 2021.