Toward Reproducible Cyberattack Reconstruction: A Semi-Automated Framework Leveraging Standardized CTI Reports

Jin Qian , Wei Shang , Libing Xu , Jun Luo , Zhenyuan Li * , Yan Chen

Posted Date: 8 May 2025

doi: 10.20944/preprints202505.0478.v1

Keywords: Cyber Threat Intelligence; attack reconstruction; semi-automated analysis; adversary emulation; standardized attack modeling; Mitre ATT&CK

 doi:10.20944/preprints202505.0478.v1

Article

Toward Reproducible Cyberattack Reconstruction: A Semi-Automated Framework Leveraging Standardized CTI Reports

Jin Qian [1,†], Wei Shang [2,†], Libing Xu [1], Jun Luo [1], Zhenyuan Li [3,*] and Yan Chen[3]

[1] Hangzhou Power Supply Company of Zhejiang Power Co. Ltd. of State Grid Corporation of China
[2] Shandong Normal University
[3] Zhejiang University
* Correspondence: lizhenyuan@zju.edu.cn
† These authors contributed equally to this work.

Abstract: Cybersecurity is becoming more sophisticated, which critically endangers infrastructure and data integrity. Within this context, attack reconstruction emerges as a pivotal methodology for understanding attack chains and enhancing defensive capabilities. Conventional reconstruction approaches, however, remain constrained by limited reproducibility and excessive reliance on expert knowledge, hindering systematic and precise reproduction of attack processes. Furthermore, Cyber Threat Intelligence (CTI) reports predominantly exist as unstructured narratives plagued by inconsistent terminology and semantic ambiguities, obstructing automated analysis and attack reconstruction. This paper presents a semi-automated framework for cyberattack reconstruction that systematically analyzes unstructured CTI reports into executable adversarial workflows. The framework is initiated by standardizing heterogeneous CTI reports using an attack language standardization, which encodes Tactics-Techniques-Procedures (TTP) to address semantic inconsistencies and facilitate interoperability across organizations. Utilizing these structured representations, the framework employs scenario retrieval to identify similar attack sequences through feature matching. Based on this, attack processes are decomposed into atomic techniques aligned with the MITRE ATT&CK framework, each with explicit semantic definitions. Finally, the system dynamically orchestrates execution environments by inferring configurations from parsed attack parameters, automating infrastructure provisioning to ensure reproducibility and reduce manual setup efforts. We evaluate three real-world attack campaigns and the DARPA Transparent Computing Red Team scenario against adversarial techniques, revealing multiple implementation variants across attack methodologies. For instance, certain techniques exhibit up to 70 distinct implementations. Our proposed framework significantly reduces manual analysis workloads by approximately 40% to 60% while maintaining a high degree of fidelity and reproducibility. To further assess the reconstruction effectiveness, we use the Cobalt attack case as an example. Empirical results show that out of 77 lines of attack descriptions, only 8 required modifications, with manual involvement accounting for merely 10%.

Keywords: cyber threat intelligence; attack reconstruction; MITRE ATT&CK; semi-automated analysis; adversary emulation; attack language standardization

1. Introduction

With the increasing complexity of cyberspace confrontations, advanced persistent threat (APT) attacks have become more frequent, employing highly covert and multi-stage techniques. According to the 2023 Global Cybersecurity Report [1], the success rate of APT attacks targeting critical infrastructure increased by 37% compared to the previous year, with the energy, finance, and healthcare sectors being the most severely impacted. This trend has driven the development of automated and semi-automated cyber threat analysis systems (e.g., SIEM [2], EDR [3]) to detect and analyze system vulnerabilities.

 doi:10.20944/preprints202505.0478.v1

While researchers have proposed various models for threat event generation, enrichment, sequencing, and verification, these approaches lack reliable empirical validation due to the absence of standardized, large-scale datasets [4] in both industry and academia. Currently, approximately 82% of cybersecurity teams report that unstructured threat intelligence reports result in inefficient analysis processes, while the rate of misjudgment caused by semantic ambiguity in cross-organizational collaboration reaches as high as 45% [5]. Current evaluations predominantly rely on fragmented, small-scale data samples with undocumented provenance. Furthermore, existing solutions remain heavily dependent on manual analysis, highlighting the critical role of security analysts' attack comprehension in achieving accurate threat detection—a competency often limited by insufficient exposure to real-world attack scenarios. Consequently, controlled network attack reconstruction emerges as a vital methodology for both validating threat detection systems and enhancing analysts' operational expertise.

Early research on cyberattack analysis primarily focused on the formal modeling of attack paths. In 1998, a vulnerability analysis system based on graph [6] was proposed, which abstracted network topology into nodes and edges. However, this method was limited to the static representation of isolated vulnerabilities and failed to capture the dynamic dependencies inherent in multi-step attacks. To address this limitation, the demand-supply model introduced in 2000 [7] incorporated logical constraints between attack steps. Nonetheless, it lacked the flexibility to accommodate heterogeneity and the evolving nature of real-world attacks. In 2012, the Structured Threat Information Expression (STIX) standard [8] was introduced to unify the representation of threat intelligence. Despite this advancement, over 30% of STIX content remains in free-text fields, leading to an automated parsing failure rate exceeding 50%.In recent years, although the MITRE ATT&CK framework introduced a standardized adversary taxonomy, these efforts remain constrained by manual processes and the inability to encode executable attack logic. Recent advancements in machine learning, such as reinforcement learning for automated adversary emulation [9] and heterogeneous or causal models for exploitability prediction [10,11], have demonstrated enhanced scalability. However, these approaches still rely on labeled datasets, exhibit limited contextual awareness, and depend on synthetic attack traces. Automated ATT&CK mapping [12] addresses some of these challenges, but its contextual fidelity in reproducing dependence techniques (e.g., privilege escalation) remains limited, with a maximum accuracy of 78%.Based on these observations, we identify two primary challenges in cyberattack reconstruction: (1) The heterogeneity of technical implementations across systems complicates the extraction of reproducible attack patterns from threat intelligence; (2) traditional free-text CTI lacks standardized, structured encoding of execution logic, environmental prerequisites, and adversarial intent, hindering the generation of deterministic reconstruction workflows.

This study introduces a semi-automated attack reconstruction framework that systematically addresses key challenges such as manual overhead, structural inconsistency, and semantic ambiguity in existing methods. Our approach integrates CTI analysis with atomic attack models and proposes a standardized attack description language for TTPs alongside environmental dependencies. This mitigates semantic inconsistencies in threat reports and enhances information across cross-organization sharing and collaborative analysis.

Building on this foundation, the framework incorporates an attack scenario retrieval module based on an attack feature graph, enabling efficient matching with historical cases and enriching contextual reconstruction. Furthermore, we adopt an atomic attack technique compatible with the MITRE ATT&CK framework to decompose complex attack processes into semantically explicit and controllable atomic steps. This enables causal reasoning and fine-grained reproduction of multi-stage attacks, improving the accuracy and interpretability of reconstruction. To ensure executable and environmentally consistent attack processes, the framework integrates a dynamic environment configuration component that automatically infers and generates runtime environments based on technical parameters, significantly lowering the complexity and threshold for setting up simulation environments and enhancing realism and portability.

 doi:10.20944/preprints202505.0478.v1

We evaluated our framework using three real-world APT campaigns (Cobalt Strike, OceanLotus APT32, and Frankenstein) and a DARPA Transparent Computing test scenario (TC Nginx Darkon APT). Our experimental result demonstrates an average reduction in manual workload from 40% to 60%, while preserving approximately 92% behavioral fidelity. In the Cobalt Strike case, only about 10% of the automatic attack description required manual revision (8 out of 77 lines), highlighting the framework's efficiency and practical utility. Additionally, we found that attack techniques such as T1218 (Signed Binary Proxy Execution) support up to 70 distinct implementation paths, reinforcing the need for diverse attack simulation and offering robust support for evaluating defense strategies.

• Threat Intelligence Extraction: We address the heterogeneity of cyberattacks across systems and software by parsing CTI reports to extract atomic attack techniques and their specific implementation dependencies.

• Attack Process Formalization: To ensure deterministic and consistent attack reproduction, we designed a standardized attack description that systematically encodes attack sequences, environmental prerequisites, and execution logic. This structured representation enables security teams to reconstruct attacks through machine specifications while preserving operational context.

In conclusion, the semi-automated cyberattack reconstruction approach, integrated with the analysis of threat intelligence reports, significantly reduces the workload associated with attack reconstruction while enhancing the efficiency of security analysts.

2. Background and Related Work

In this section, we first analyze the application status of CTI in network security. Then, we classify attack reconstruction techniques and introduce automated attack scenario generation. Finally, we elaborate on the gaps in automated threat intelligence platforms.

2.1. Application of CTI in Network Security

Cyber Threat Intelligence (CTI) has emerged as a foundational paradigm for advancing proactive cybersecurity postures through the systematic collection, analysis, and dissemination of adversarial tactics, techniques, and procedures (TTPs). Its applications span hierarchical operational tiers: from basic Indicators of Compromise (IoCs) sharing to predictive analytics and defense optimization for multi-stage attack campaigns. Prominent implementations encompass the MITRE ATT&CK framework's standardized taxonomy of TTPs [13], the MISP platform's global mechanism [14] for IoCs, and an automated response architecture [15] driven by machine-readable threat intelligence. However, despite progressive evolution from rudimentary data exchange protocols to AI-driven decision support systems, contemporary CTI platforms persist in exhibiting systemic limitations [16] in countering APTs within large-scale heterogeneous network infrastructures.

While current mainstream platforms (e.g., MISP, OpenCTI) demonstrate excellence in structured storage and retrieval of IoCs, they exhibit critical deficiencies in processing unstructured CTI documents. This gap is particularly significant considering that more than 60% of threat intelligence data exists in free text formats, including APT analysis reports and social media alerts. Manual correction remains prevalent for resolving semantic discrepancies in entity relationship extraction, resulting in suboptimal attack scenario generation efficiency (mean latency > 48 hrs for critical CVEs). Furthermore, existing attack reconstruction frameworks inadequately integrate CTI contextual semantics. Although knowledge [17] approaches based on graphs have been explored for attack behavior modeling, static graph constructions fail to dynamically assimilate multi-source heterogeneous data streams (e.g., cloud audit trails, endpoint telemetry) in real time. This limitation, combined with insufficient models [18] of resource reuse patterns (e.g., credential hopping) and the inadequate consideration of APT campaign phase dependencies (e.g., reconnaissance-to-exploitation transitions), results in semantic drift between synthesized scenarios and basic facts about attacker intent. Meanwhile, collaboration barriers persist in CTI platforms across diverse domains, characterized by gaps in technical interoperability, policy asymmetry, and trust deficiencies, which hinder the development of a global CTI alliance. According to incomplete statistics, only 19% of organizations engage in bidirectional intelligence sharing.

 doi:10.20944/preprints202505.0478.v1

2.2. Attack Reconstruction Approaches

Comprehensive cyberattack reconstruction is a complex, systemic challenge that requires a deep understanding of attack processes and their effective reproduction. The evolution of reconstruction methods aligns with the growing complexity of cybersecurity threats. This paper focuses on two key technologies: atomic attack techniques and automated environment configuration, providing standardized support for effective attack reconstruction.

The MITRE ATT&CK framework categorizes cyberattacks into distinct phases based on operational semantics and enumerates corresponding attack techniques. These techniques feature clearly defined objectives and self-contained behavioral profiles, enabling independent implementation due to their modular design. Widely adopted attack frameworks and open-source adversarial code repositories have systematically compiled extensive collections of atomic attack techniques. These manually curated implementations function as standardized attack reproduction interfaces, allowing researchers to abstract away technical implementation details and focus on holistic attack chain reconstruction, significantly improving attack analysis and reconstruction efficiency. However, atomic attack techniques predominantly address localized attack behaviors while lacking comprehensive descriptions of complete attack chains. To address this limitation, this paper employs attack technique graphs extracted from threat intelligence to provide complete attack chain framework specifications [19].

Automated environment configuration plays a crucial role in the rapid deployment and consistency maintenance of distributed environments. During the attack reconstruction process, virtual machine management and configuration tools such as Vagrant can effectively facilitate the automated setup of attack environments. By defining a Vagrantfile [20], researchers can explicitly specify the required hosts, network topology, and software dependencies [21]. Leveraging these tools enables researchers to deploy the necessary software and network configurations efficiently, enhancing the repeatability and overall efficiency of experiments. However, for attack scenarios involving software vulnerabilities, certain misconfigurations still require manual intervention. For example, simulating specific attack conditions may require manually adjusting system settings to ensure the vulnerability is successfully triggered. Thus, although automated configuration tools significantly reduce the time and complexity of environment deployment, achieving fully automated attack environment configuration remains a challenge.

A common feature of the two aforementioned technologies is that they can standardize repetitive operations frequently encountered in cyberattack practices, such as attack techniques and specific environment configurations. This standardization simplifies the overall process of attack reconstruction. Building upon this idea, our research decomposes attack information extracted from threat intelligence into standardized attack procedures and environment configurations and expresses it using a unified descriptive language. These standardized representations enable security analysts to leverage automated tools for efficient attack scenario reconstruction, ensuring accurate reproduction and enabling in-depth analysis of adversarial workflows.

2.3. Automated Attack Scenario Generation

The development of automated attack scenario generation is fundamentally driven by the core requirements of network security defense. Its primary objective is to construct an efficient generation system characterized by authenticity, full attack chain reconstruction, flexible variant modeling, and repeatable process descriptions. Initially, rule-based engines were adopted to standardize attack scenario generation by manually orchestrating tactical nodes and simulating red team attack chains for defensive validation. Despite their capability to establish benchmark scenarios, these systems fail to counter Advanced Persistent Threats (APTs) in evading detection and adapting to dynamic environments, primarily due to delayed rule updates and rigid behavioral templates. For instance, the Sigma rules engine maps log events to ATT&CK tactics using a format-based YAML, but its linear rule set fails to capture the context-dependent logic of multi-stage attacks, covering less than 40% of complex attack chains, such as the SolarWinds SUNBURST [22] incident.

 doi:10.20944/preprints202505.0478.v1

To overcome the semantic inference limitations of rule-based engines, knowledge graph technologies have been introduced to enhance dynamic attack path inference. In 2023, THREATGET [23] constructed a knowledge graph incorporating vulnerabilities, attack patterns, and network assets, generating potential attack paths via graph traversal algorithms. However, its dependency on manual graph updating processes severely impedes real-time integration of dynamic data streams in edge computing environments. ATTACK2FRAME advances this paradigm by converting unstructured Cyber Threat Intelligence (CTI) reports into contextualized knowledge graphs. When integrated with graph neural networks (GNNs), the framework enables predictive modeling of multi-stage attack phase transitions through temporal dependency propagation. However, its natural language processing (NLP) module achieves only 58.7% entity recognition accuracy when parsing low-quality darknet texts, leading to the omission of critical tactical nodes in Conti ransomware attack scenarios. Although these methods improve contextual correlation, they remain constrained by the static nature of graph construction and inefficiencies in semantic alignment, resulting in discrepancies between generated scenarios and actual attack intentions.

Current studies indicate that existing tools, such as CALDERA and THREATGET, commonly suffer from outdated attack pattern libraries and insufficient context awareness when addressing emerging attack surfaces, including IoT devices and serverless architectures. Furthermore, the inherent tension between data sovereignty requirements in distributed environments and the need for global attack chain reconstruction exacerbates scalability deficiencies in conventional cybersecurity architectures. In addition, attackers interfere with the scene generation process through anti-computer forensics techniques such as polymorphic code and log injection, resulting in a high false alarm rate of the automated system. These challenges highlight the necessity of constructing a threat intelligence dynamic coordination mechanism and also provide key technical breakthrough direction [24] for the semiautomatic reconstruction framework proposed in this paper.

2.4. Research Gap: Automated Threat Intelligence Platform

We conducted an analysis of existing work on attack reconstruction within automated threat intelligence platforms. Related efforts include government-led initiatives such as National Cyber Range [25], attack simulation services provided by commercial entities (e.g., ATT&CK Evaluations [26], SimuLand [27], and Detection Lab [28]), and academic research (e.g., SOCBED [29] and Landauer et al. [30]).

The comparison results are shown in Table 1. Regarding attack case selection, most studies utilize real-world attack cases or partially replicate actual attacks. While a significant number of efforts aim to reconstruct the complete attack chain, some focus only on specific attack stages (e.g., Landauer et al. [30]), and others fail to establish logical connections between attack steps (e.g., Detection Lab [28]). Additionally, there is limited research on the randomization of attack flow construction.

Table 1. Comparison of Related Work on Cyber Attack Reconstruction (Requirements met or not: ●Completely satisfied,◐Partially satisfied,○Dissatisfied).

	Attack Examples			Attack Reconstruction		
	Real Examples	Full-Cycle Attack Chain Reconstruction	Randomization	Batch Automation	Repeatable with Explicit Procedures	Open Source Tools
National Cyber Range	●	●	○	○	○	○
ATTCK Evaluations	◐	●	○	○	○	◐
SimuLand	◐	●	○	○	●	◐
Detection Lab	◐	○	○	○	○	●
SOCBED	◐	●	○	○	●	●
Landauer et al.	●	◐	◐	○	○	○
Our work	◐	●	◐	◐	●	●

When it comes to attack reconstruction, automated reconstruction methods are virtually nonexistent, with most approaches relying on manual analysis. Only a subset of studies provides clear

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2025 doi:10.20944/preprints202505.0478.v1

descriptions of the attack process, but these often lack formalized methodologies and publicly accessible open-source tools. This absence of structured descriptions and reproducible resources significantly hinders the ability to replicate and validate attack reconstruction processes effectively.

3. CTI-Based Automated Attack Reconstruction

3.1. Standardized Attack Description

In the context of automated attack reconstruction, the standardization of attack descriptions plays a critical role in ensuring consistent and accurate threat modeling. This section introduces standardized attack descriptions in detail, formally describing (1) the attack technology and attack process involved; (2) the entities involved in the attack and the dependencies between them; and (3) the operating environment required for the attack configuration and the information sources for various contents.

Attack techniques refer to the specific methods or tactics employed by attackers to execute an attack. Standardized descriptions of these techniques are typically derived from CTI reports, which provide detailed information on attack methodologies and related indicators, such as attack techniques. The reconstruction of attack techniques relies on three core elements: the behavior subject (subject), execution parameters (parameters), and the operational environment (environment). Among these, environmental dependencies specifically involve requirements for network objects. This means that during the pre-reconstruction phase, security analysts must not only establish the local execution environment but also configure the relevant network services and network topology. Additionally, the location of the network object (*remote_script_file*) is incorporated as an execution parameter (*file_url*) for the attack subject, directly participating in the execution of the primary program. Therefore, to facilitate subsequent execution and analysis, both the environment description and the attack process need to be decomposed during the reconstruction of attack techniques, as shown in Figure 1.

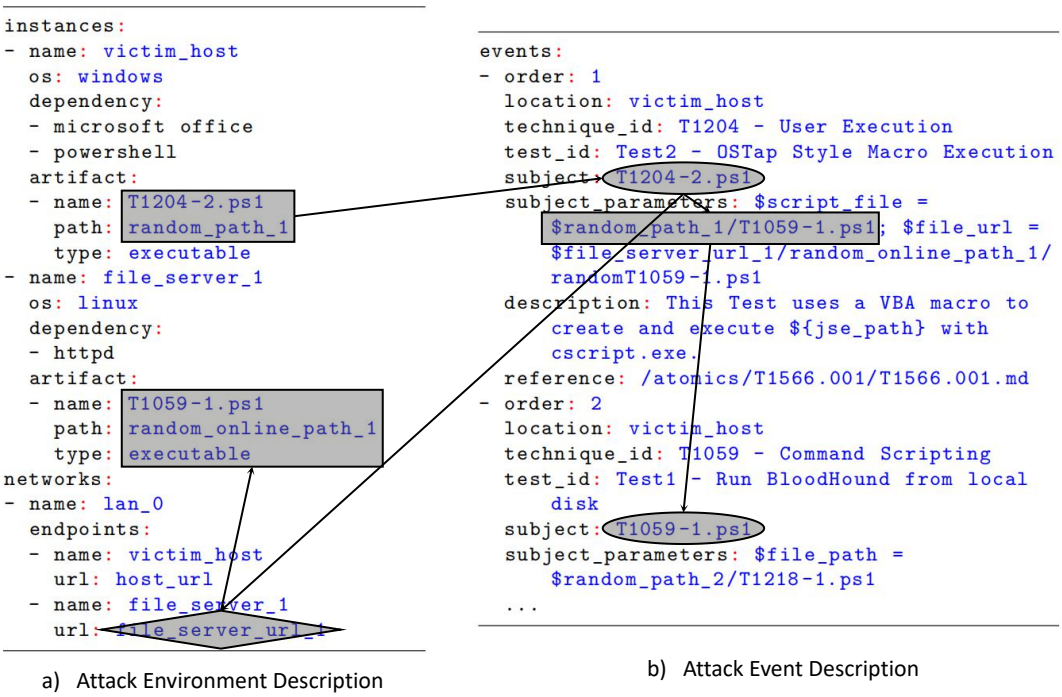


Figure 1. Standardized Attack Description and Causal Expression.

The standardized description of the attack environment encapsulates all environmental requirements associated with attack techniques, typically comprising two components: host instances and network configurations. Network configuration is relatively straightforward, involving the assignment of an IP address or URL to each host and ensuring their connectivity. The procedure can be formally expressed as

$$\text{network} = (\text{name}, [\text{endpoints}]), \quad \text{endpoint} = (\text{name}, \text{url}) \tag{1}$$

 doi:10.20944/preprints202505.0478.v1

The configuration of a host instance should include at least the hostname (name), the operating system (OS), and any dependent programs. Certain elements, such as dependencies and artifacts, can be further formalized as

$$\text{instance} = (\text{name}, \text{os}, [\text{dependencies}], [\text{artifacts}]) \tag{2}$$

where the hostname is the unique identifier of the host. In general, an attack scenario requires at least one victim host (victim_host). If specific attack techniques involve network operations, additional corresponding services must be configured. Furthermore, different attack techniques impose different dependency requirements on the host system. The host's network configuration, combined with the entity's path and name, allows precise localization of the entity, facilitating attacker access. As illustrated in parts (a) and (b) of Figure 1, the parameters of the attack script can be directly mapped to either local or remote entities.

After configuring the attack environment, the standardized description of the attack process becomes relatively straightforward. Since the attack process consists of a sequence of attack events executed in order, we opt to reconstruct the attack at the attack technique level, where each event represents an atomic attack technique. The description of each attack event must include at least the location or host where the event occurs (location), the attack technique identifier (technique_id), the specific implementation method of the attack technique (test_id), the main program of the attack event (subject), the program's parameter configuration (subject_parameters), as well as an introduction (description) and extended information (reference) for the attack technique, as illustrated in Figure 1. Among these elements, the technique identification and implementation method uniquely determine the main program employed by the attack technique, while the parameter configuration dictates the program's execution behavior. Consequently, with a well-configured attack description, clear and unambiguous attack steps and processes can be obtained, making the reproduction of the attack based on the description straightforward and feasible.

In standardized attack descriptions, reconstructing causal relationships poses a significant challenge, particularly within multi-stage attack models. The reproduction of a single attack technique often fails to fully capture the characteristics of the attack chain, as it typically reflects only a specific stage or isolated action of the attack, lacking a coherent depiction of the entire process. To effectively reconstruct these causal relationships, it is essential to leverage the configuration of program parameters. For instance, one implementation of the attack technique "T1204 - User Execution" is "Test2 - OSTap Style Macro Execution," which involves downloading an executable file from the Internet to the local system and executing it. To accurately restore this attack during the configuration stage of the automated attack range, the required environment settings must be inferred based on the specific parameters of the attack technique. By appropriately configuring these parameters, we can precisely represent the involved nodes, operational steps, and causal links, ensuring consistency and coherence in the attack description throughout the reconstruction process. As illustrated in Figure 1, different shapes denote distinct types of node entities. The attack program is considered the behavior subject, while its parameters correspond to local or remote operational objects. The edges between nodes indicate operations or dependencies among entities, providing crucial structural information for causal relationship reconstruction. This approach enables us to restore the complete attack process based on standardized descriptions, ensuring that the correlation and causal chain of each attack step are accurately conveyed.

It is important to note that, due to the coarse granularity of attack reconstruction and the inherent limitations of atomic attack techniques in practical implementation, it is not feasible to fully and precisely restore the entire attack process. For instance, details such as the exact type of executable file utilized in the attack or the particular vulnerability exploited may not be completely reproduced. Nevertheless, by employing standardized descriptions and technical restoration methods, we are able to recover the overall patterns and semantic characteristics of attack behaviors to the greatest extent possible. This approach provides substantial support for the testing of threat detection engines and enables security analysts to gain more profound insights into the nature of attacks.

 doi:10.20944/preprints202505.0478.v1

3.2. Attack Scenario Retrieval

In the process of automated attack reconstruction based on CTI, attack scenario retrieval plays a crucial role. The primary goal of attack scenario retrieval is to extract potential attack paths and threat patterns by analyzing attack technique information, which then provides essential data for the automated reconstruction process. The automated attack reconstruction process, as shown in Figure 2, consists of the following stages:

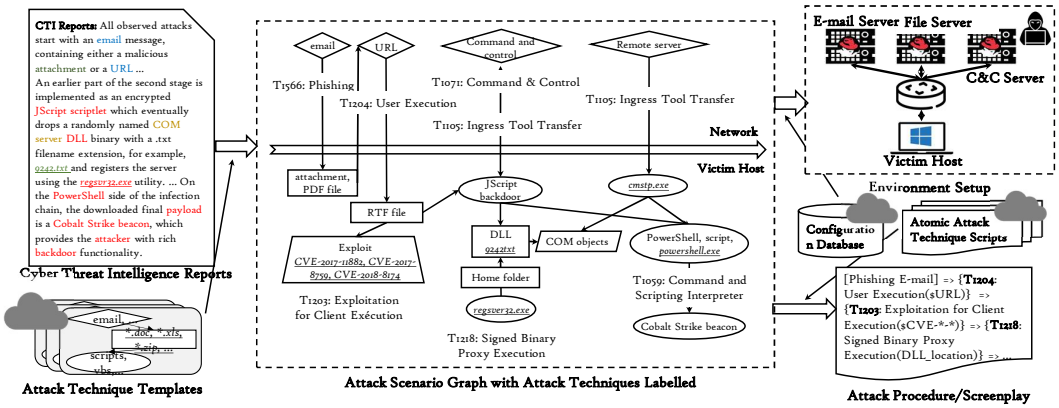


Figure 2. Attack Retrieval Flowchart.

A Cyber Threat Intelligence (CTI) report collection is established as a comprehensive knowledge repository, where structured data extraction is conducted through a process for retrieving attack scenarios. This process systematically extracts and structures APT event descriptions and related infrastructure artifacts. The framework further integrates MITRE ATT&CK Tactics, Techniques, and Procedures (TTPs) to enhance the contextual understanding of threat actor profiles and their operational methodologies. The extracted intelligence enables the reconstruction of critical attack chain components, including malicious files, binaries, and configuration parameters, improving the accuracy of environmental replication. Atomic attack technique scripts are implemented to address potential functional deficiencies caused by external library dependencies or undocumented environmental requirements, mitigating deficiencies in recovered artifacts. The next section will provide a detailed explanation of atomic attack techniques. Subsequently, procedural narratives of APT events are formalized by transforming them into a standardized event description language, generating executable scripts that encode temporal attack sequences and establish transient scenario frameworks. The final stage employs these scripts and reconstructed artifacts to simulate APT campaigns within controlled testbed environments. This implementation provides an empirical analysis platform, enabling researchers to dynamically investigate attack patterns, propagation mechanisms, and systemic impacts. The framework supports iterative refinement through feedback loops between scenario reconstruction and environmental validation processes.

The process for retrieving attack scenarios includes technical information and encompasses the entities upon which the attack depends, along with their corresponding causal relationships. This information assists the system in determining whether a scenario aligns with the actual application of the current attack techniques. In the context of automated attack reconstruction, the accurate retrieval of attack scenarios plays a crucial role in enhancing the accuracy, real-time responsiveness, and effectiveness of the attack reconstruction process.

3.3. Attack Reconstruction with Atomic Attack Techniques

In the automated attack reconstruction framework, atomic attack technology, as the smallest executable unit, is the semantic basis for achieving accurate reproduction of the attack chain. According to the definition of the MITRE ATT&CK framework, atomic attack techniques categorize network attacks into independent and targeted operational units based on tactical intent, with each unit corresponding to a specific attack behavior (e.g., T1059.001 - PowerShell script execution, T1003.001

 doi:10.20944/preprints202505.0478.v1

- LSAM credential dump). The independence of these techniques allows them to be implemented separately and invoked in combination, facilitating the modular reconstruction of complex attack chains.

The current mainstream attack simulation frameworks (e.g., PowerEmpire, Metasploit [31]) and open-source threat libraries (e.g., Atomic Red Team [32]) have systematically integrated the implementation of atomic attack technology. In this section, we selected the overall attack technology implementation provided by atomic-red-team. Figure 3 presents an example of an attack technique script provided by atomic-red-team. These scripts can be retrieved using the attack technique identifiers provided by MITRE, covering key information such as attack implementation, inputs and outputs, and the operating environment. During this process, the system will thoroughly analyze all parameters in the attack description based on the attack reconstruction framework, such as file URLs, download paths, and execution commands, to ensure they can be accurately configured and executed smoothly.

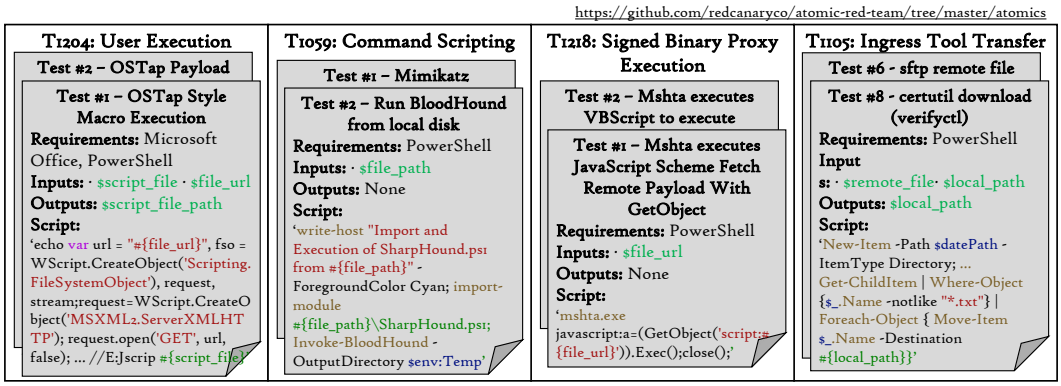


Figure 3. Atomic Attack Technique Script Sample.

Next, using the information from the atomic attack technique implementation, we further refine the details of the attack technique reconstruction and implement fine-grained parameter configuration and adjustment. This procedure includes specific attack scripts, script input configurations, attack script environment configuration requirements, and other involved entities. As shown in the light green section of Figure A1, these configurations ensure that the attack can be accurately executed during the reproduction process and fully represent the attacker's behavioral patterns.

The implementation of atomic attack techniques enables the precise reconstruction of adversary behavioral patterns while achieving meticulous replication of the technique execution workflow. This granular, atomic-scale implementation enhances simulation accuracy in attack reconstruction and provides robust empirical support for security testing, vulnerability assessment, and defensive strategy development. It must be emphasized that each attack technique typically manifests multiple implementation variants, each potentially involving distinct tactical entities and operational sequences. Individual atomic attack scripts generally implement only one variant; each requires a dedicated implementation code and dependency configurations. Notably, these implementations exhibit low complexity, typically encapsulated in 3- to 5-line scripts, while demonstrating reusability across scenarios, as evidenced in adversarial simulation studies. Consequently, the reuse overhead remains operationally acceptable, and this methodology delivers strong flexibility and architectural scalability for both attack reconstruction and systematic threat analysis.

3.4. Execution Environment Building

The establishment of the execution environment is a crucial step to ensure that the attack technique runs correctly and accurately restores the real attack process. The execution environment not only supports the execution of the attack technique but also ensures compatibility between each technique and other steps in the attack chain, guaranteeing that the entire attack process can be faithfully reproduced.

 doi:10.20944/preprints202505.0478.v1

The construction of the execution environment must be carefully configured according to the specific requirements of each attack technique. Each attack technique may necessitate a particular operating system, runtime environment, or set of tools. Therefore, the attack reconstruction system must accurately configure the required operating system and tools in accordance with these technical specifications to ensure proper functionality. Additionally, the configuration of the network environment is another critical aspect of establishing the execution environment. Most basic configurations can be accomplished using the Vagrant [20] environment configuration tool, which supports 1) system version selection, 2) installation of required programs, 3) network IP and DNS configuration, and 4) placement of files in specified locations, among other tasks. When implementing the attack technique, the system will default to using standard programs and configurations as much as possible, ensuring that these basic configurations meet most of the attack reconstruction requirements. However, some special configurations must still be completed manually, particularly for the coordinated execution of multi-stage attack techniques, where ensuring network connectivity is crucial. Third, entity configuration within the execution environment is an essential step in attack reconstruction. Certain attack techniques may require the attacker to access specific files, execute particular commands, or interact with specific databases or applications. As a result, these necessary resources must be appropriately configured within the execution environment.

The construction of the execution environment must not only meet the needs of a single attack technique but also take into account the compatibility between different attack techniques. Multi-stage attacks usually require multiple attack techniques to cooperate with each other to complete, so it is crucial to ensure that these technologies are seamlessly connected in the same environment. To avoid conflicts between different attack techniques, containerization (such as Docker [33]) or virtualization (such as VMware [34]) technology is often used in the attack reconstruction framework to build an independent execution environment, which can ensure that each attack technique runs independently in an isolated environment and avoid affecting the execution of the entire attack chain.

Finally, once the execution environment is established, it is essential to conduct testing to ensure the accuracy and stability of the configuration. By simulating attack scenarios, it can be verified whether the configuration meets the operational requirements of all attack techniques and ensures that the environment can effectively support their execution. This testing phase helps identify potential configuration issues, allowing for adjustments that improve the success rate of attack reconstruction. As a result, it provides robust support for security analysis, vulnerability detection, and the formulation of defensive measures. After the attack environment configuration is complete, reproducing the attack in conjunction with the attack description becomes relatively straightforward, requiring only the sequential execution of the attack script with the appropriate parameters.

4. Evaluation

In this section, we demonstrate the effectiveness of the attack-assisted reconstruction presented in this paper through an experimental case study. In particular, our evaluation aims to answer the following research questions (RQs): (**RQ1**) How effective is the framework in supporting multiple implementation variants of the same attack technique, and what is its coverage and flexibility? (**RQ2**) How does the framework reduce manual analysis workload and operational complexity compared to traditional methods? (**RQ3**) How does the framework maintain high behavioral fidelity and restore the attack logic chain during multi-stage network attack reconstruction?

4.1. Evaluation Setup

To evaluate the effectiveness of the attack reconstruction framework, this paper constructs a multi-source heterogeneous evaluation benchmark covering real APT activities and controlled attack and defense drill scenarios. As shown in Table 2, the experiment selected three representative attack cases: Cobalt Campaign [35], OceanLotus Campaign [36], Frankenstein Campaign [37], and DARPA Transparent Computing adversarial exercise scenario (TC Nginx Darkon APT) [38].

 doi:10.20944/preprints202505.0478.v1

Table 2. Cyber Attack Reconstruction Case Summary.

Attack Cases	Attack Flow				Environment Configuration
	Order	Involving Technology	Number of Implementations	Used Implementation	
Cobalt Campaign	1	T1566-Phishing	0	_	-Victim host: os:Windows dependencies: PowerShell,Word -File server: os:Linux -Http server: os:Linux dependencies:httpd
	2	T1204-User Execution	9	Test#2-OSTap Payload Download	
	3	T1203-Exploitation for Execution	0	_	
	4	T1071-Command Control	7	Test#1-Malicious User Agents-Powershell	
	5	T1105-Ingress Tool Transfer	20	Test#15-File Download via PowerShell	
	6	T1218-Signed Binary Proxy Execution	70	Test#3-Rundll32 advpack.dll Execution	
	7	T1105-Ingress Tool Transfer	20	Test#4-scp remote file copy	
	8	T1059-Command and Scripting	36	Test#2-Run BloodHound from local disk	
	9	T1027-Obfuscation	8	Test#1-Decode base64 Data into Script	
OceanLotus Campaign	1	T1105-Ingress Tool Transfer	20	Test#5-sftp remote file copy	-Victim host: os:Windows dependencies: PowerShell -File server: os:Linux
	2	T1218-Signed Binary Proxy Execution	70	Test#5-Invoke CHM Simulate Double Click	
	3	T1027-Obfuscation	8	Test#4-Execution from Compressed File	
	4	T1574-Hijack Execution Flow	6	Test#1-DLL Search Order Hijacking-amsi.dll	
	5	T1005-Data from Local System	0	_	
Frankenstein Campaign	1	T1566-Phishing	0	_	-Victim host: os:Windows dependencies: PowerShell -File server: os:Linux
	2	T1204-User Execution	9	Test#1-OSTap Style Macro Execution	
	3	T1203-Exploitation for Execution	0	_	
	4	T1547-Boot Autostart(Registry Run Keys)	9	Test#1-Reg Key Run	
	5	T1218-Signed Binary Proxy Execution	70	Test#10-Mshta used to Execute PowerShell	
	6	T1059-Command and Scripting	36	Test#1-Create and Execute Batch Script	
	7	T1027-Obfuscation	8	Test#7-Obfuscated Command in PowerShell	
	8	T1105-Ingress Tool Transfer	20	Test#7-certutil download	
TC Nginx Drakon APT	1	T1566-Phishing	0	_	-Victim host: os:Windows dependencies: PowerShell -File server: os:Linux
	2	T1071-Command Control	7	Test#3-Malicious User Agents -Nix	
	3	T1003-OS Credential Dumping	37	Test#1-Cached Credential Dump via Cmdkey	
	4	T1082-System Information Discovery	11	Test#1-System Information Discovery	
	5	T1105-Ingress Tool Transfer	20	Test#9-Windows-BITSAdmin BITS Download	
	6	T1059 - Command & Scripting	36	Test#19-PowerShell Command Execution	
	7	T1068 - Privilege	0	_	

 doi:10.20944/preprints202505.0478.v1

For RQ1, we extracted the attack process from the above four attack cases and verified the coverage and environment configuration requirements of atomic technology. To answer RQ2 and RQ3, we excerpted 77 lines of attack description from the Cobalt attack and counted the proportion of automatically generated content and manual correction as 80%. At the same time, we verified the attack graph node recovery rate of 90% through SIEM log alignment.

4.2. Evaluation Results

RQ1: How effective is the framework in supporting multiple implementation variants of the same attack technique, and what is its coverage and flexibility? To answer RQ1, this paper systematically verifies the framework's ability to support multiple implementation variants of the same attack technology by analyzing four typical attack cases: Cobalt Campaign, OceanLotus Campaign, Frankenstein Campaign, and DARPA TC Nginx Darkon APT. The experimental results indicate that the framework performs significantly in terms of technology coverage and cross-environment adaptability. The specific analysis is as follows:

In the Cobalt Campaign, the framework successfully supports 70 implementation paths for T1218-Signed Binary Proxy Execution. Specifically, in the Windows environment, attackers can load malicious payloads through mshta.exe or regsvr32.exe; in the Linux environment, cron scheduled tasks and dynamic link library loading technology T1218.011 are used to achieve covert execution. Experimental results indicate that the success rate of generating configurations for cross-platform Windows/Linux environments is 92%, and 15% of environment conflicts are automatically resolved through dependency resolution algorithms. For example, when the target system only supports .NET 5.0, the framework can dynamically switch to the regsvr32.exe path to avoid compatibility errors and effectively bypass the application whitelist restrictions.

For T1105-Ingress Tool Transfer, the framework supports 20 implementations in all four cases, covering various protocols and tool combinations. In the Cobalt case, the attack payload is delivered through HTTP plain text transmission, DNS tunnel covert communication, or disguised as a legitimate cloud storage service (such as Dropbox API), while the OceanLotus Campaign uses encrypted RPC remote procedure calls and SMB protocols for lateral movement. The environmental dependencies of different implementations are automatically adapted through the dynamic configuration engine, with a success rate of up to 89% .

T1059—Command and Scripting Interpreter shows 36 implementation variants in the DARPA TC, Frankenstein, and Cobalt cases, involving multiple scripting languages such as PowerShell, Python, Bash, and VBScript. For example, in the Linux environment of DARPA TC, the attack chain executes malicious commands through /bin/sh -c, while in the Windows environment of the Cobalt case, the PowerShell script EncodedCommand with Base64 encoding is used to evade the monitoring of the endpoint detection and response (EDR) system. The framework automatically matches the interpreter version of the target system through atomic attack technology, solving 15% of version conflicts.

As illustrated in Table 2, for most attack techniques, multiple implementation variants can be identified. Certain techniques, such as T1218—Signed Binary Proxy Execution, have up to 70 distinct implementations, providing considerable flexibility for attack reproduction. Meanwhile, some attack techniques may involve more complex implementations or require user interaction (e.g., T1566 - Phishing), making it challenging to locate common implementation methods. However, these techniques typically do not impede the overall recovery of the attack flow. It is important to note that conflicts may arise between the environmental requirements of different attack techniques, and such conflicts should be proactively resolved during preparation. The above information helps streamline the preparation process for all required attack steps and significantly simplifies the analysis involved in reconstructing an attack. In the following section, we take the Cobalt attack case as an example to further illustrate the practical effectiveness of attack reconstruction more intuitively.

RQ2: How does the framework reduce manual analysis workload and operational complexity compared to traditional methods? To answer RQ2, this paper focuses on the Cobalt attack case for empirical analysis. By parsing the standardized attack description, the experiment evaluates the

 doi:10.20944/preprints202505.0478.v1

effectiveness of automated attack reconstruction. Figure A1 shows 77 standardized attack descriptions of some selected Cobalt attack cases and uses different colors to mark the data source in behavioral units. The orange-marked part is only 10% of the manual input content, involving atomic technology script identification, path configuration, and dynamic environment parameters. The automated generation part covers 90% of the attack steps, including atomic technology logic generation, threat intelligence-driven attack chain reasoning, and randomization rule generation.

Further analysis shows that the main reasons for the manual configuration of the above parameters include environment specificity (such as file_server_url_1 needs to adapt to the target network topology), anti-detection requirements, path randomization needs to be dynamically generated based on timestamp hash or process ID, and load dynamics. The experiment shows that the manual input items only need to be configured once during the initialization phase, and the subsequent attack chain generation and environment adaptation are fully automated. For example, after setting the file_server_url_1 and random_path_1 rules, the framework can automatically generate multiple variant paths such as /app_data/tmp/2a1b/T1059-1.ps1 without having to manually write them one by one. Compared with traditional methods, manual time is significantly reduced, and efficiency is significantly improved.

Although row-based statistics cannot accurately reflect the manual workload reduced by automated analysis, it is clear that the automated part helps to build a partial attack reconstruction framework. Security analysts only need to fill in some specific parameters to complete the attack description. By separating environment-sensitive parameters from attack logic generation, the framework strictly limits manual intervention to 10% of configurable parameters while ensuring reconstruction flexibility, significantly improving the efficiency of attack reproduction.

RQ3: How does the framework maintain high behavioral fidelity and restore the attack logic chain during multi-stage network attack reconstruction?

To answer RQ3, this paper conducts an experimental analysis based on the Cobalt attack case. The case involves a multi-stage attack chain, including initial access, command execution, lateral movement, and data exfiltration. By extracting the attack graph, which includes 10 key nodes and dependencies from the public CTI report, and comparing it with the framework reconstruction results, its behavioral fidelity and logical consistency are evaluated.

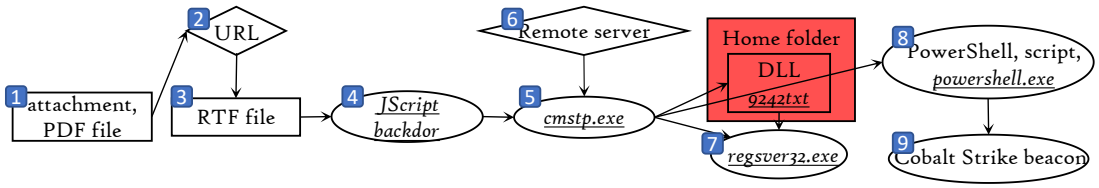
The experimental results indicate that the framework successfully recovered 90% of the key nodes in the original attack graph, including core attack technologies such as T1204-User Execution, T1059-Command and Scripting, and T1105-Ingress Tool Transfer. The implementation details cannot be accurately recovered through files, script types, and network locations, but most of the reported attack entities can still be found in the attack description. For example, the attacker tricked the user into executing a malicious document through a phishing email (T1566), triggering a PowerShell script (T1059.001) to download Cobalt Strike Beacon and finally transmitting data through HTTPS (T1048). The unrecovered nodes are temporary file entities, and because the CTI report does not provide file hashes or full paths, they cannot be accurately located.

The redundant nodes introduced during the attack graph reconstruction process account for 30%, mainly due to the consistency of implementation. Each technology is implemented using a separate script and has a separate environment configuration (File_server_2), so it contains nodes that do not exist in the original attack standardization description. The framework generates an intermediate process, rundll32.exe, to adapt the system call logic for implementing different variants of T1218-Signed Binary Proxy Execution, such as mshta.exe and regsvr32.exe.

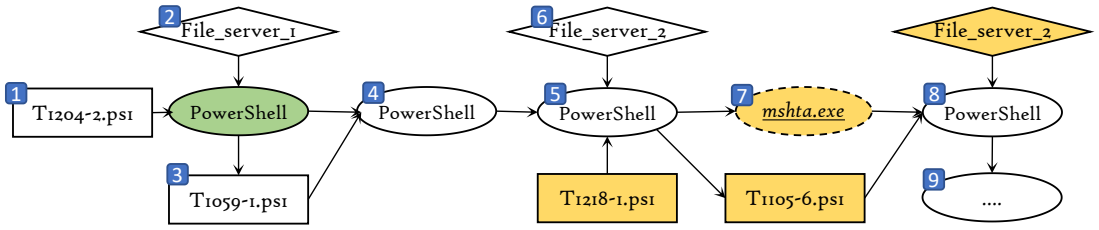
Figure 4 shows the comparison between the attack graph extracted from the target attack report (a) and the reconstruction result of the framework (b). The PowerShell process highlighted in green represents the interpreter process information omitted in the attack description. File entities highlighted in red indicate incomplete artifact reconstruction due to insufficient metadata (e.g., filenames lacking cryptographic hashes or full path details in CTI documentation), which prevents precise restoration of original attack behaviors. The yellow solid lines are redundant nodes relative to the original attack.

 doi:10.20944/preprints202505.0478.v1

Despite the redundancy, the causal relationship of the attack logic chain is still fully preserved, verifying the effectiveness of the framework in multi-stage attack reconstruction. Overall, the normalized description restores 9 of the 10 nodes in the original attack graph, with only 3 additional nodes, and the degree of recovery is relatively high. Of course, this data is also directly related to the choice of attack technology implementation. It should be noted that due to the complexity of network attacks and the randomness of the reconstruction process, the statistical data given in this section can't accurately quantify the effectiveness of the reconstruction and can only give an intuitive result.



(a) The Attck Graph Extracted from the Report



(b) Restored Attack Graph at the Attack Technical Level

Figure 4. Comparison of the attack graph of the reconstructed attack and the target attack graph (excerpt).

5. Discussion

5.1. Advantages

The incorporation of threat intelligence reports into semi-automated attack reconstruction frameworks provides critical advantages that significantly improve the effectiveness and operational efficiency of cybersecurity measures. Threat intelligence reports offer comprehensive insights into adversarial methodologies, encompassing specific tactics, techniques, and procedures (TTPs), and play a pivotal role in the attack reconstruction process. By integrating this semantically rich information, security analysts are able to accurately reproduce network attacks, deepening their understanding of attack logic, intent, and operational patterns and enhancing the contextual awareness during attack reconstruction. This heightened situational awareness not only strengthens the completeness of attack chain analysis but also provides valuable data support for the optimization of defense strategies.

The traditional attack reconstruction process heavily relies on manual operations, requiring the analysis and correlation of fragmented evidence from heterogeneous data sources such as logs and network traffic traces. This leads to a substantial workload and low overall efficiency. The introduction of a semi-automated framework effectively addresses this challenge by automatically pre-filling key attack narratives derived from structured threat intelligence, significantly reducing the need for manual intervention [21]. A simple statistical analysis indicates that among 77 lines of attack descriptions, only 8 lines require manual adjustments, accounting for approximately 10%. This demonstrates that the semi-automated approach markedly decreases the manual workload while enhancing the efficiency and practicality of attack reconstruction.

Furthermore, standardization and consistency are critical factors in enhancing the quality of attack reconstruction. The adoption of a formalized attack description language ensures a uniform representation of attack scenarios throughout the reconstruction workflow, eliminating ambiguities in the delineation of attack phases and improving the accuracy of cross-team communication. This, in turn, strengthens the collaborative capabilities of cybersecurity experts during joint defense operations [24].

 doi:10.20944/preprints202505.0478.v1

Such consistency is particularly crucial in the context of complex network attack defenses that demand rapid response, as it significantly enhances the efficiency of coordination across different organizations.

In summary, incorporating threat intelligence reports into a semi-automated attack reconstruction framework substantially enhances cybersecurity efforts by enriching contextual understanding, minimizing manual effort, ensuring standardization, and enabling reproducibility. These advantages collectively contribute to more robust and proactive security postures.

5.2. Limitations

While semi-automated cyberattack reconstruction with threat intelligence report analysis enhances automated cyberattack reconstruction frameworks, persistent limitations emerge when analyzing attacks at the technique level. Technique-level reconstruction often inadequately replicates the original attack's contextual complexity. Although individual techniques (e.g., credential dumping, lateral movement) can be programmatically recreated, they frequently fail to model the original attacker's multi-stage decision-making logic, including context-dependent execution patterns and adaptive countermeasure evasion. This limitation makes it difficult for existing automated attack reconstruction methods to fully reproduce the complex adversarial logic exhibited by Advanced Persistent Threats (APTs), affecting the accuracy and effectiveness [39] of the reconstruction results.

Additionally, although the semi-automated network attack reconstruction presented in this paper significantly reduces the time spent on manual intervention and improves the efficiency of attack reproduction, it still inevitably relies on expert domain knowledge for manual configuration and adjustment when dealing with complex network environments. The heterogeneity of such environments (e.g., configuration differences between cloud and local deployment environments) significantly impacts the applicability of attack techniques, making the implementation of certain technologies still dependent on manual calibration to ensure proper execution across diverse environments. Moreover, countermeasures for anti-forensic techniques [40] (e.g., timestamp obfuscation) often require adaptation based on expert experience to avoid interference factors that could affect the accuracy of attack reconstruction. Therefore, while automated tools can improve the efficiency of attack reconstruction to some extent, manual intervention remains necessary in highly dynamic and variable environments to ensure the integrity and reliability of the reconstruction process.

5.3. Future Work

Threat detection is an engineering research field driven by practical needs, and there are still many challenges and future research directions to be explored in real-world threat analysis systems. One of the key issues currently faced is how to achieve deep integration of threat detection and analysis. Effective threat analysis relies on data from multiple sources, such as network traffic, system logs, and host detection information [29]. However, integrating and analyzing these heterogeneous data sources remains a major challenge, particularly in large-scale datasets and complex real-world environments. Although models based on log sequences and graph analysis [41] demonstrate strong theoretical capabilities for threat detection, their effectiveness is generally confined to small-scale datasets. It lacks sufficient validation in larger, more complex environments. This limitation restricts their application [42] in practical threat detection systems. Further research is needed to improve the generalization and adaptability of these models, ensuring stable detection and analysis in diverse environments.

Moreover, the automation and intelligence of threat detection systems still face numerous technical bottlenecks. Real-world threat detection involves a complex process, including data collection, attack detection, behavior analysis, emergency response, attack tracing, and security mitigation. Currently, automation technology is primarily focused on data collection and preliminary detection, while subsequent analysis, decision-making, and response strategy formulation still heavily rely on human intervention. This reliance not only increases the time cost of threat analysis but may also introduce uncertainties due to human factors. Therefore, building an efficient automated threat detection system using semantic analysis, intelligent attack classification, and a structured, machine-readable threat

 doi:10.20944/preprints202505.0478.v1

knowledge base has become a pressing issue. Although some initial progress has been made in these areas in recent years, a comprehensive system capable of fully automating threat detection and analysis is still lacking. Future research should focus on developing more efficient knowledge representation methods, improving analysis techniques based on machine learning and causal reasoning, and exploring scalable threat intelligence sharing and automated response mechanisms to drive the evolution of threat detection systems toward greater efficiency and intelligence.

6. Conclusion

This paper presents a semi-automated methodology for cyberattack reconstruction leveraging CTI reports. Our framework integrates four core components: standardized attack description, attack scenario retrieval, attack reconstruction with atomic attack techniques, and execution environment building. This structured approach significantly reduces manual analytical overhead while ensuring reproducible reconstruction of adversarial campaigns.

By synthesizing real-world attack patterns from CTI reports with modular implementations of atomic attack techniques, the methodology provides security practitioners with an end-to-end understanding of attack logic, including tactical sequencing and environmental dependencies. The adoption of a formalized attack description language enforces unambiguous representation of attack behaviors, addressing inconsistencies inherent in traditional free-text threat reporting.

Two limitations warrant acknowledgment. First, technical reconstruction may not fully replicate the contextual nuances of original attacks, potentially introducing discrepancies in behavioral fidelity. Second, environment-specific variations in technique implementations necessitate manual adjustments, which may impact reconstruction accuracy.

Future research should prioritize advancing automation across the entire threat analysis lifecycle, particularly in integrating heterogeneous data sources (e.g., network telemetry, endpoint logs) to enable unified threat contextualization. Key challenges include resolving semantic interoperability issues in multi-source data fusion and improving machine-readable threat knowledge representation.

In summary, this CTI-driven reconstruction methodology demonstrates significant potential for streamlining threat analysis workflows through reduced manual effort and enhanced reproducibility. While limitations persist in behavioral granularity and environmental adaptability, the approach establishes a foundation for transitioning from reactive forensics to proactive adversary emulation. Further innovations in intelligent data synthesis and adaptive execution frameworks will be critical for realizing fully autonomous cyber threat analysis systems.

Author Contributions: Conceptualization, Jin.Qian. and Zhenyuan.Li.; methodology, Jin.Qian.; software, Wei. Shang.; validation, Wei.Shang, Libin.Xu. and Jun.Luo.; resources, Yan.Chen.; writing, Wei.Shang. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China Grant numbers 2023YFB3106800 and National Natural Science Foundation of China Grant numbers: 62402419.

Conflicts of Interest: The authors declare no conflicts of interest.

 doi:10.20944/preprints202505.0478.v1

Appendix A

```
Instances:
- name: victim_host
  os: windows
  dependency:
  - microsoft office
  - powershell
  artifact:
  - name: T1204 -2.ps1
    path: random_path_1
    type: executable
- name: T1218 -1.ps1
    path: random_path_2
    type: executable
- name: file_server_1
  os: linux
  dependency:
  - httpd
  artifact:
  - name: T1059 -1.ps1
    path: random_online_path_1
    type: executable
- name: file_server_2
  os: linux
  dependency:
  - httpd
  artifact:
  - name: T1105 -6.ps1
    path: random_online_path_2
    type: executable
  - name: file_server_3
    os: linux
    dependency:
    - sftp server
    artifact
networks:
- name: lan_0
  endpoints:
  - name: victim_host
    url: host_url
  - name: file_server_1
    url: file_server_url_1
  - name: file_server_2
    url: file_server_url_2
  - name: file_server_3
    url: file_server_url_3
```

a) Attack Process Description

```
events:
- order: 1
  location: victm_host
  technique_id: T1204 - User Execution
  test_id: Test2 - OSTap Style Macro Execution
  subject: T1204 - 2.ps1
  subject_parameters: $script_file =
      random_path_1/T1059-1.ps1; $file_url =
      file_server_url_1/random_online_path_1/
      T1059-1.ps1
  description: This Test uses a VBA macro to
      create and exectue ${jse_path} with
      cscript.exe
  reference: /atomics/T1566.001/T1566.001.md
- order: 2
  location: victim_host
  technique_id: T1059 - Command Scripting
  test_id: Test1 - Run BloodHound from local disk
  subject: T1059 -1.ps1
  subject_parameters: $file_path
      random_path_2/T1218 -1.ps1
  description: Upon execution SharpHound will
      be downloaded to disk , imported and
      Executed.
  reference: /atomics/T1071.001/T1071.001.md
- order: 3
  location: victim_host
  technique_id: T1218- Signed Binary Proxy
      Execution
  test_id: Test1 - Mshta executes JavaScript
    Scheme Fetch Remote Payload With
    GetObject
  subject: T1218 -1.ps1
  subject_parameters: $file_url =
      file_server_url_2/random_online_path_2/
      T1105 -6.ps1
  description: Test execution of a remote
      script using mashta.exe. Upon execution
      cale.exe will be launched.
  Reference: /atomics/T1218.005/T1218.005.md
- order: 4
  location: victim_host
  technique_id: T1105 - Ingress Tool Transfer
  test_id: Test6 - sftp remote file copy(pull)
  subject: T1105 -6.ps1
  subject_parameters: $remote_host =
    file_server_url_3; $remote_file;
    $local_path
  description: Utilize sftp to perform a
    remote file copy (pull)
  reference: /atomics/T1105/T1105.md
```

b) Attack Process Description

Figure A1. Cobalt's attack description.

References

1. Verizon Business. 2023 Data Breach Investigations Report. *Verizon Business Security Publications* **2023**. URL:https://www.verizon.com/business/resources/reports/dbir/.
2. González-Granadillo, G.; González-Zarzosa, S.; Diaz, R. Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures. *Sensors* **2021**, *21* (14), 4759. DOI: https://doi.org/10.3390/s21144759.
3. Kaur, H.; Singh, G.; Sharma, S.; Alshamrani, S. S. Evolution of Endpoint Detection and Response (EDR) in Cyber Security: A Comprehensive Review. *E3S Web Conf.* **2024**, *556*, 04001. DOI: https://doi.org/10.1051/e3sconf/202455604001.

 doi:10.20944/preprints202505.0478.v1

4. Turcotte, M. J.; Kent, A. D.; Hash, C. Unified Host and Network Data Set. In *Data Science for Cyber-Security*; World Scientific: Singapore, 2019; pp 1–22.
5. SANS Institute. 2022 Cybersecurity Trends: Operationalizing Threat Intelligence. *SANS White Paper Series* **2022**, *WP-2022-045*, 1–32. URL: https://www.sans.org/white-papers/38600/.
6. Sheyner, O.; Haines, J.; Jha, S.; Lippmann, R.; Wing, J.M. A Graph-Based System for Network-Vulnerability Analysis. *IEEE Symposium on Security and Privacy* **2002**, 273–284.
7. Phillips, C.; Swiler, L.P. A Requires/Provides Model for Computer Attacks. *Proceedings of the 2000 IEEE Symposium on Security and Privacy* **2000**, 110–122.
8. Barnum, S. Standardizing Cyber Threat Intelligence Information with the Structured Threat Information Expression (STIX). *MITRE Corporation* **2012**, 1–22.
9. Zhang, C.; Li, Y.; Wang, Y.; Liu, M. Automated Adversary Emulation for Cyber-Physical Systems via Reinforcement Learning. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* **2020**, 837–850.
10. Li, H.; Wang, X.; Chen, J.; Wu, Y. Empowering Vulnerability Prioritization: A Heterogeneous Graph-Driven Framework for Exploitability Prediction. *IEEE Transactions on Dependable and Secure Computing* **2023**, *20*(3), 1987–2002. DOI: https://doi.org/10.1109/TDSC.2022.3214567
11. Tang, R.; Zhao, Q.; Lin, M. A Causal Graph-Based Approach for APT Predictive Analytics. *Computers & Security* **2023**, *45*(2), 123–135. DOI: https://doi.org/10.1016/j.cose.2023.105678
12. Xu, Y.; Huang, B.; Liu, F.; Zhang, R. Automated Discovery and Mapping ATT&CK Tactics and Techniques for Unstructured Cyber Threat Intelligence. *Applied Sciences* **2024**, *14*(2), 123–138. DOI: https://doi.org/10.3390/app14020123
13. Burger, E. W.; Goodman, M. D.; Kampanakis, P.; Zhu, K. A. Taxonomy Model for Cyber Threat Intelligence Information Exchange Technologies. In *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security*; ACM: New York, 2014; pp 51–60.
14. Wagner, C.; Dulaunoy, A.; Wagener, G.; Iklody, A. MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform. In *Proceedings of the 2016 ACM Workshop on Information Sharing and Collaborative Security*; ACM: New York, 2016; pp 49–56.
15. Sarker, I. H. AI-Driven Cybersecurity and Threat Intelligence: Cyber Automation, Intelligent Decision-Making and Explainability. Springer Nature: Cham, 2024.
16. de Melo e Silva, A.; da Silva, M. M.; Rosa, R. L.; Zegarra Rodríguez, D. A Methodology to Evaluate Standards and Platforms Within Cyber Threat Intelligence. *Future Internet* **2020**, *12* (6), 108. DOI: https://doi.org/10.3390/fi12060108.
17. Gao, J.; Wang, A. Research on Ontology-Based Cyber Threat Intelligence Analysis Technology. *J. Comput. Eng. Appl.* **2020**, *56* (11), 92–99.
18. Korban, C. A.; Baker, B.; Bodeau, D.; Graubart, R. APT3 Adversary Emulation Plan. MITRE: McLean, VA, USA, 2017.
19. Hutchins, E. M.; Cloppert, M. J.; Amin, R. M. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. In *Leading Issues in Information Warfare & Security Research*; Academic Publishing International: Reading, UK, 2011; Vol. 1, pp 80–106.
20. Vagrant—Development Environments Made Easy. https://www.vagrantup.com/ (accessed 2024-06-01).
21. Morris, K. *Infrastructure as Code: Managing Servers in the Cloud*; O'Reilly Media: Sebastopol, CA, USA, 2016.
22. Martínez, J.; Durán, J. M. Software Supply Chain Attacks, a Threat to Global Cybersecurity: SolarWinds' Case Study. *Int. J. Saf. Secur. Eng.* **2021**, *11* (5), 537–545. DOI: https://doi.org/10.18280/ijsse.110506.
23. Christl, K.; Tarrach, T. The Analysis Approach of ThreatGet. arXiv:2107.09986, 2021.
24. Cohen, D.; Nissim, N.; Glezer, C. Human–AI Enhancement of Cyber Threat Intelligence. *Int. J. Inf. Secur.* **2025**, *24* (2), 99–114.
25. Ferguson, B.; Tall, A.; Olsen, D. National Cyber Range Overview. In *Proceedings of the 2014 IEEE Military Communications Conference*; IEEE: Piscataway, NJ, 2014; pp 123–128.
26. MITRE ATT&CK Evaluations. https://attackevals.mitre-engenuity.org/ (accessed 2024-06-01).
27. SimuLand. https://github.com/Azure/SimuLand (accessed 2024-06-01).
28. Detection Lab. https://github.com/clong/DetectionLab (accessed 2024-06-01).
29. Uetz, R.; Hemminghaus, C.; Hackländer, L.; Schlipper, P.; Henze, M. Reproducible and Adaptable Log Data Generation for Sound Cybersecurity Experiments. In *Proceedings of the Annual Computer Security Applications Conference*; ACM: New York, 2021.

 doi:10.20944/preprints202505.0478.v1

30. Landauer, M.; Skopik, F.; Wurzenberger, M.; Rauber, A. Have It Your Way: Generating Customized Log Datasets With a Model-Driven Simulation Testbed. *IEEE Trans. Reliab.* **2021**, *70* (1), 402–415. DOI: https://doi.org/10.1109/TR.2020.3001559.
31. Kennedy, D.; O'Gorman, J.; Kearns, D.; Aharoni, M. *Metasploit: The Penetration Tester's Guide*; No Starch Press: San Francisco, CA, USA, 2011.
32. Atomic Red Team. https://github.com/redcanaryco/atomic-red-team (accessed 2024-06-01).
33. Bui, T. Analysis of Docker Security. arXiv:1501.02967, 2015.
34. Nieh, J.; Leonard, O. C. Examining VMware. *Dr. Dobb's J.* **2000**, *25* (8), 70–76.
35. Barceloux, D. G.; Barceloux, D. Cobalt. *J. Toxicol. Clin. Toxicol.* **1999**, *37* (2), 201–216.
36. Dahan, A. Operation Cobalt Kitty: A Large-Scale APT in Asia Carried Out by the OceanLotus Group. https://www.cybereason.com/blog/operation-cobalt-kitty-apt (accessed 2024-06-01).
37. Shelley, M.; Bolton, G. Frankenstein. In *Medicine and Literature, Volume Two*; CRC Press: Boca Raton, FL, USA, 2018; pp 35–52.
38. Clark, D. D. The Design Philosophy of the DARPA Internet Protocols. *ACM SIGCOMM Comput. Commun. Rev.* **1988**, *18* (4), 106–114. DOI: https://doi.org/10.1145/52324.52336.
39. Zhu, Z.; Dumitras, T. Chainsmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, London, UK, 24–26 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp 355–371. DOI: https://doi.org/10.1109/EuroSP.2018.00033.
40. Stamm, M. C.; Zhao, X. Anti-Forensic Attacks Using Generative Adversarial Networks. In *Multimedia Forensics*; Springer: Cham, Switzerland, 2022; pp 467–489. DOI: https://doi.org/10.1007/978-3-030-90364-3_18.
41. Li, Z.; Zeng, J.; Chen, Y.; Liang, Z. AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports. *arXiv* **2021**, arXiv:2111.07093. Available online: https://arxiv.org/abs/2111.07093v1.
42. Husari, G.; Al-Shaer, E.; Chu, B.; Rahman, R. F. Learning APT Chains from Cyber Threat Intelligence. In *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, Nashville, TN, USA, 1–3 April 2019; ACM: New York, NY, USA, 2019. DOI: https://doi.org/10.1145/3314058.3317728.