國立臺灣大學管理學院資訊管理學系

碩士論文

Department of Information Management
College of Management
National Taiwan University
Master's Thesis

基於大型語言模型之網路威脅情報攻擊生命週期建構

LLM-Based Construction of Attack Life Cycles from CTI Reports

吳和謙

He-Chien Wu

指導教授：孫雅麗 博士

Advisor: Yeali Sun, Ph.D.

中華民國 115 年 2 月

February, 2026

doi:10.6342/NTU202600249

致謝

謹此致謝，感謝在本論文完成過程中，曾給予本人指導、協助與支持的師長與同儕。

其中特別感謝我的指導教授孫雅麗老師，在研究所求學的這段期間，老師不僅在資訊安全相關領域與研究方法上給予我紮實且深入的指導，也在研究態度與處事方式上給予我諸多啟發，使我獲益良多。老師始終展現的耐心與熱忱，亦是我由衷敬佩並努力效仿的典範。在研究與論文撰寫的過程中，老師多次引導我釐清方向，於迷惘之際指引前路，使我得以順利完成本論文。

另感謝本論文之口試委員陳孟彰老師、蕭舜文老師、陳俊良老師與黃意婷老師，撥冗審閱本文並提供寶貴意見與建議，使本論文得以更加完善。

也感謝每周於中研院的會議中，與我們共同討論的楊名全老師、翁國維學長及郭英仁學長，透過多元觀點的交流與建議，協助釐清研究方向，使研究得以順利推進

感謝實驗室的瑋翔學長、惟恩和伃孜，在我感到困惑與迷惘之時，能夠彼此扶持、共同前進，於研究與生活中給予我莫大的支持與鼓勵。

最後，謹向我的家人與朋友致上最深的感謝。感謝你們在我專注於論文研究期間，始終給予理解、支持、鼓勵與陪伴，成為我持續前行的重要力量。

doi:10.6342/NTU202600249

摘要

攻擊行動（attack campaigns）通常分散記錄於多份網路威脅情報（Cyber Threat Intelligence, CTI）報告中，而每份報告僅提供對敵對行為的部分且片段化描述。此外，CTI 敘事往往使用異質且高度依賴語境的語言，其表述方式與 MITRE ATT&CK 架構中對戰術（techniques）的標準化描述存在顯著差異，難以直接對應兩者的語意。上述的資訊片段化與語意不一致問題，對於自動化建構完整連貫的攻擊生命週期（attack life cycle）造成了重大阻礙。

本研究提出一套自動化分析框架，利用大型語言模型（Large Language Models, LLMs），透過以攻擊事件（incident）為核心的推理、結構化事件提取（event extraction）、戰術對應以及多來源整合，從分散的 CTI 證據中建構完整的攻擊生命週期。不同於直接將句子映射至技術標籤的作法，本研究透過引入基於 LLM 的批註過濾機制，將惡意行為識別與戰術對應加以解耦。該過濾機制運用語境推理能力，識別中心攻擊事件範圍內之具體攻擊行為。提取出的事件會被轉換為結構化表示，並經由結合封閉（closed -world）與開放（open-world）的標註流程，對應至 MITRE ATT&CK 戰術，隨後再透過以證據為基礎的驗證步驟進行確認。

在取得經 TTP 標註的事件後，系統進一步推論事件之間的時間順序與因果關係，以建構具多階段特性的攻擊生命週期。為克服跨情資來源所造成的描述片段化問題，本研究亦提出一個 LLM 輔助的合成模組，能夠整合多份 CTI 報告各自建構之攻擊生命週期。透過個案研究顯示，所建構之攻擊行動在事件涵蓋範圍與因果關係上，與既有的生命週期模型具有一致性，驗證了從異質 CTI 敘事中自動化建構攻擊行動的可行性。本研究所提出的框架推進了自動化 CTI 分析的發展，使其由單純的戰術識別邁向對攻擊行動流程的整體性建構。

關鍵詞：多威脅情報報告融合、資安攻擊生命週期重建、網路威脅情報惡意活動辯識與擷取、資安攻擊 TTP 標記、人工智慧輔助資安

doi:10.6342/NTU202600249

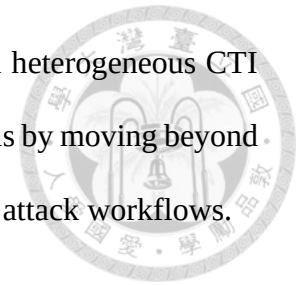
Abstract

Attack campaigns are typically documented across multiple Cyber Threat Intelligence (CTI) reports, each providing only partial and fragmented descriptions of adversarial activities. Furthermore, CTI narratives often employ heterogeneous, context-dependent language that differs substantially from the canonical representations of techniques in the MITRE ATT&CK framework, making direct semantic alignment unreliable. This fragmentation and semantic mismatch hinder automated construction of coherent attack campaign life cycles.

This work proposes an automated framework leveraging Large Language Models (LLMs) to construct comprehensive attack campaign life cycles by constructing campaign progression from fragmented CTI evidence through incident-scoped reasoning, structured event extraction, technique alignment, and multi-source synthesis. Rather than directly mapping sentences to techniques, we decouple malicious activity identification from technique-level alignment by introducing an LLM-based annotation filter that identifies concrete adversarial actions attributable to the focal incident using contextual reasoning. Extracted events are transformed into structured representations and subsequently aligned with MITRE ATT&CK techniques through combined closed-world and open-world labeling stages, followed by evidence-grounded validation.

Using TTP-labeled events, our system infers temporal ordering and causal dependencies to construct multi-phase campaign life cycles. To overcome fragmentation across intelligence sources, we further introduce an LLM-assisted synthesis module that integrates life cycles derived from multiple reports into a unified campaign representation. A case study demonstrates that constructed campaigns exhibit event coverage and causal relationships consistent with established life cycle models,

doi:10.6342/NTU202600249



illustrating the feasibility of automated campaign construction from heterogeneous CTI narratives. The proposed framework advances automated CTI analysis by moving beyond technique identification toward coherent construction of operational attack workflows.

Keywords: CTI Report Synthesis, Attack Campaign Life Cycle Construction, Malicious Event Identification in CTI, Attack Campaign TTP Labeling, AI-Assisted Cybersecurity

doi:10.6342/NTU202600249

Table of Contents

doi:10.6342/NTU202600249

List of Figures

doi:10.6342/NTU202600249

doi:10.6342/NTU202600249

List of Tables

doi:10.6342/NTU202600249

Chapter 1. Introduction

An attack campaign, as defined in the MITRE ATT&CK framework [1], refers to a collection of intrusion activities conducted over time against common targets and operational objectives. Understanding the life cycle of such campaigns is critical for threat hunting, incident response, and strategic defense planning. In practice, however, information describing an attack campaign rarely appears in a single comprehensive source. Instead, evidence is distributed across multiple Cyber Threat Intelligence (CTI) reports authored by different organizations, each capturing only partial and complementary observations. Analysts must therefore manually correlate fragmented intelligence to construct the overall attack workflow, a process that is time-consuming, inconsistent, and difficult to scale.

A fundamental challenge arises from the narrative nature of CTI reporting. Reports are written primarily for human analysts and employ heterogeneous, context-dependent language that often differs significantly from the abstract and canonical technique descriptions provided by structured frameworks such as MITRE ATT&CK [1]. Malicious behaviors may be described implicitly, distributed across multiple sentences, or expressed through operational details that share little lexical overlap with ATT&CK technique definitions. Consequently, approaches relying on direct semantic similarity or keyword matching frequently fail to capture attacker intent, especially in incident-scoped analysis where contextual reasoning is essential.

A further complication is that CTI reports typically mix incident-specific observations with background analysis, historical campaigns, defensive recommendations, and attacker capability descriptions. Sentences referencing malicious techniques therefore do not always correspond to actions actually performed in the focal

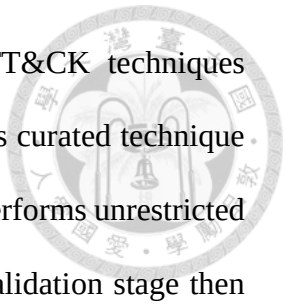
doi:10.6342/NTU202600249

incident. Without careful filtering, automated extraction systems risk incorporating irrelevant or cross-campaign activities, resulting in inaccurate campaign constructions. Reliable construction thus requires identifying malicious activities that are both concrete and incident-attributable, followed by structured reasoning to infer temporal ordering and causal dependencies among them.

Recent advances in Large Language Models (LLMs) have demonstrated strong capabilities in contextual reasoning, information extraction, and multi-document synthesis across diverse domains [2], [3], [4], [5], [6]. These developments create new opportunities for automating CTI analysis beyond traditional similarity-based or rule-driven approaches. However, directly mapping CTI narratives to ATT&CK techniques remains challenging due to implicit descriptions, contextual dependencies, and inconsistencies across reports. Moreover, existing work typically focuses on sentence-level technique classification or entity extraction, and rarely addresses construction of coherent attack campaign life cycles or synthesis across multiple intelligence sources.

This work addresses these challenges through an LLM-driven framework that constructs attack campaign life cycles from fragmented CTI reporting using incident-scoped reasoning, structured event extraction, technique alignment, and multi-source synthesis. Rather than directly mapping sentences to ATT&CK techniques, we decouple malicious activity identification from technique-level alignment. Our system first performs incident-scoped malicious activity extraction using an LLM-based annotation filter that evaluates sentences together with local context to determine whether they describe concrete adversarial actions attributable to the focal incident. Validated events are transformed into structured representations capturing action semantics and supporting evidence, forming an intermediate abstraction that enables reliable downstream processing.

doi:10.6342/NTU202600249



Extracted events are subsequently aligned with MITRE ATT&CK techniques through a two-stage labeling process. A closed-world stage leverages curated technique lists commonly provided in CTI reports, while an open-world stage performs unrestricted alignment across the ATT&CK knowledge base. An independent validation stage then resolves competing candidates using evidence-grounded reasoning, selecting mappings supported explicitly by textual evidence or proposing alternatives when necessary. Using the resulting TTP-labeled events, our system infers temporal ordering and causal dependencies among activities to construct coherent multi-phase campaign life cycles.

Because individual CTI reports typically provide only partial campaign perspectives, we further introduce an LLM-assisted synthesis module that integrates life cycles derived from multiple reports into a unified campaign representation. Through a demonstration case study, we illustrate how extracted events and inferred causal relationships align with established life cycle models, demonstrating the feasibility of automated campaign construction from heterogeneous CTI narratives.

Compared with prior work, our approach differs in several important ways. Existing efforts commonly perform sentence-level technique classification or rely on surface-level semantic similarity between CTI statements and ATT&CK descriptions [6], [7]. Such methods often misclassify descriptive or historical content as incident activity and struggle when malicious behavior is expressed implicitly or across multiple sentences. In contrast, our framework introduces incident-scoped filtering to ensure extracted activities correspond to actions actually performed in the incident, separates event extraction from technique alignment to improve robustness, constructs campaign life cycles through temporal and causal reasoning, and synthesizes life cycle information across multiple CTI sources to address intelligence fragmentation. These capabilities move beyond technique identification toward automated construction of operational attack workflows.

doi:10.6342/NTU202600249

The contributions of this work are summarized as follows:

1. **Incident-scoped malicious activity identification.**

   We introduce an LLM-based annotation filtering mechanism that identifies concrete adversarial activities grounded in incident context while excluding descriptive, speculative, or cross-campaign statements.

2. **Robust ATT&CK technique alignment framework.**

   We propose a multi-stage labeling and validation strategy combining closed-world, open-world, and evidence-grounded reasoning to improve reliability in mapping extracted activities to MITRE ATT&CK techniques.

3. **Automated life cycle construction.**

   We develop a method for inferring temporal ordering and causal dependencies among malicious activities, enabling systematic construction of multi-phase attack campaign life cycles.

4. **Multi-source campaign synthesis.**

   We demonstrate that campaign life cycles derived from separate CTI reports can be synthesized into unified representations, overcoming fragmentation across intelligence sources.

5. **Practical life cycle coverage demonstration.**

   Through case study analysis, we illustrate how constructed campaigns exhibit event coverage and causal relationships consistent with established life cycle models.

Prior research has explored automated CTI information extraction, ATT&CK technique classification, and attack knowledge graph construction [3], [6], [7], [8]. However, existing approaches often rely on surface-level similarity matching or focus on entity extraction rather than incident-scoped activity construction. Multi-document

doi:10.6342/NTU202600249

intelligence synthesis and life cycle reasoning remain underexplored. The following section reviews related work in CTI extraction, ATT&CK-based analysis, and automated attack modeling, highlighting how our approach extends prior efforts.

The remainder of this thesis is organized as follows. Chapter 2 reviews related work. Chapter 3 presents the overall framework and system architecture, describes incident-scoped malicious activity extraction, details technique labeling and validation procedures, and explains life cycle construction through temporal and causal reasoning. Chapter 4 introduces multi-source life cycle synthesis. Chapter 5 presents evaluation results and analysis. Finally, Chapter 6 concludes the work and discusses limitations and future directions.

doi:10.6342/NTU202600249

Chapter 2. Background and Related Work

This chapter situates the proposed research within existing work on cyber threat intelligence (CTI), adversary behavior modeling, and large language model (LLM)–assisted security analysis. It introduces the foundational concepts and frameworks referenced throughout the thesis and reviews prior research relevant to malicious activity extraction, technique labeling, and attack campaign construction. The chapter concludes by identifying gaps in existing approaches that motivate the design choices of this work.

2.1 Cyber Threat Intelligence (CTI)

Cyber Threat Intelligence (CTI) refers to curated knowledge about adversaries, their capabilities, infrastructure, and operational behaviors, collected and analyzed to support cyber defense. CTI is commonly disseminated through analyst-authored reports published by security vendors, research organizations, and incident response teams. These reports document observed attack campaigns, malware families, and intrusion techniques, often with the goal of informing detection, mitigation, and strategic decision-making.

CTI reports are primarily narrative in nature. Rather than presenting structured execution traces, they describe attacker behavior through prose, blending technical observations with contextual analysis, hypotheses, and defensive guidance. As a result, CTI narratives frequently interleave concrete attacker actions with background information, historical context, and inferred intent. While this narrative style is well suited for human analysts, it poses challenges for automated analysis, particularly when attempting to construct campaign-level workflows.

Another defining characteristic of CTI is incompleteness. Individual reports rarely capture an entire attack campaign end to end. Visibility is constrained by sensor coverage, reporting scope, and publication timing. Consequently, different reports describing the

doi:10.6342/NTU202600249

same campaign often emphasize different phases or techniques, leading to fragmented and partially overlapping views of adversary activity. These properties make CTI a rich but noisy data source, requiring interpretation, synthesis, and reasoning beyond surface-level text extraction.

2.2 MITRE ATT&CK® Framework

The MITRE ATT&CK® framework is a widely adopted knowledge base for representing adversary behavior in terms of tactics, techniques, and procedures (TTPs) [1]. ATT&CK organizes observed attacker actions into a matrix of tactics, which represent high-level adversary goals, and techniques, which describe concrete methods used to achieve those goals. Sub-techniques further refine technique definitions to capture specific implementation patterns.

ATT&CK serves as a common vocabulary for threat intelligence sharing, detection engineering, and adversary emulation. Security vendors frequently reference ATT&CK techniques in CTI reports to contextualize observed behaviors and align them with a standardized taxonomy. However, ATT&CK techniques are abstract behavioral categories rather than direct descriptions of textual evidence. Mapping narrative CTI statements to ATT&CK techniques therefore requires interpretation and judgment.

Importantly, ATT&CK does not aim to represent full attack workflows or temporal sequences. While techniques are grouped by tactic, the framework does not encode ordering, causality, or campaign-specific structure. As a result, ATT&CK alone is insufficient for constructing attack life cycles from CTI reports. Instead, it provides a technique-level labeling layer that must be combined with additional reasoning to infer temporal progression and inter-event relationships.

doi:10.6342/NTU202600249

2.3 Event Extraction

Event Extraction (EE) aims to identify and structure task-relevant actions from unstructured text. In traditional information extraction pipelines, Named Entity Recognition (NER) and Relation Extraction (RE) are typically performed prior to EE (e.g., CASIE [8]). In our setting, however, EE constitutes the core analytical component, as it captures the dynamic, action-centric, and semantically rich relationships that characterize adversarial behavior in CTI reports.

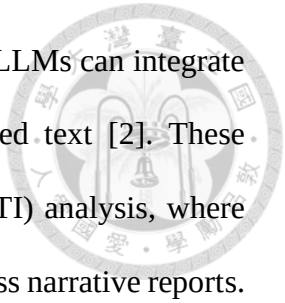
To operationalize this task, we initially attempted to define multiple *who–did–what–to–whom–using–what* trigger patterns for each MITRE ATT&CK technique and sub-technique, and constructed a knowledge base to support a TTP Event Identification and Classification Agent (TEICA). This agent was designed to identify and classify CTI statements by aligning them with MITRE ATT&CK techniques based on predefined trigger sets. However, this approach yielded limited performance. The primary reason is that the semantic correspondence between CTI-derived malicious activity statements and MITRE ATT&CK technique descriptions is frequently weak, underspecified, or implicit. This mismatch makes it difficult for trigger-based or direct semantic similarity methods to reliably capture the underlying intent or objective of malicious activities.

In recent years, this task has increasingly been approached using large language models, which enable flexible, schema-driven extraction without requiring task-specific architectures.

2.4 Large Language Models (LLMs) for CTI Analysis

Recent advances in Large Language Models (LLMs) have enabled generative approaches to information extraction, in which structured outputs are produced through conditional text generation rather than task-specific classifiers [5]. By leveraging

doi:10.6342/NTU202600249



contextual reasoning strategies such as chain-of-thought prompting, LLMs can integrate long-range context and handle implicit relationships in unstructured text [2]. These capabilities make LLMs attractive for Cyber Threat Intelligence (CTI) analysis, where malicious activities are often described implicitly and distributed across narrative reports.

Recent studies have applied LLMs to CTI-related tasks including report summarization, indicator extraction, and ATT&CK technique identification [3]. However, both the generative information extraction literature and CTI-specific evaluations highlight important limitations. Generative models may hallucinate unsupported entities or actions, overgeneralize from background knowledge, or fail to adhere to extraction schemas when constraints are weak [4], [5]. In CTI analysis, such errors are particularly problematic because extracted activities must be incident-attributable and traceable to explicit textual evidence.

Systematic evaluations of automated TTP extraction further show that many existing approaches rely on surface-level semantic similarity or sentence-level classification, leading to misclassification of background or historical content as incident activity and poor handling of implicit descriptions [6]. As a result, recent work increasingly advocates positioning LLMs as bounded reasoning components within structured pipelines, where their outputs are constrained, validated, and integrated with explicit analytical logic rather than treated as authoritative [3], [4], [6].

The framework proposed in this work follows this perspective by employing LLMs in narrowly scoped roles that leverage contextual reasoning while preserving controllability and evidence grounding. Incident-scoped activity identification and technique alignment are assisted by LLM reasoning, while attack campaign life cycle construction is governed by structured representations and explicit reasoning procedures.

doi:10.6342/NTU202600249

Chapter 3. Attack Life Cycle Construction

In this chapter, we will describe the construction of a coherent attack life cycle from an unstructured Cyber Threat Intelligence (CTI) report by transforming fragmented textual descriptions into structured, ordered, and interpretable representations of adversary behavior.

The process begins with CTI report preprocessing, in which raw reports (e.g., PDF) are segmented into sentences while preserving local context. These sentence groups serve as the input to an incident-scoped malicious activity identification stage, where an annotation filter identifies text describing concrete adversary actions and excludes background information, mitigation guidance, and speculative analysis. This stage leverages a large language model, ChatGPT [9], as a bounded reasoning component to evaluate each sentence group in context and determine incident attribution.

Identified malicious activity statements are then passed to the LLM-based TTP labeling and validation stage. In this stage, another independent LLM, ChatGPT, assists in assigning one or more MITRE ATT&CK technique labels to each events, followed by a validation step to ensure semantic consistency between the activity description and the assigned techniques.

The resulting set of validated, TTP-labeled events is used as input to the attack life cycle construction module. ChatGPT is further employed as a reasoning component to assist with event ordering, causal relationship inference, and attack phase mapping, operating over the validated event set under structured prompts and task-specific constraints. These reasoning outputs are used to derive a coherent sequence of malicious activities connected by inferred causal relationships. Events are subsequently mapped to high-level attack phases, yielding a structured attack campaign life cycle that reflects both

doi:10.6342/NTU202600249

temporal progression and strategic intent.

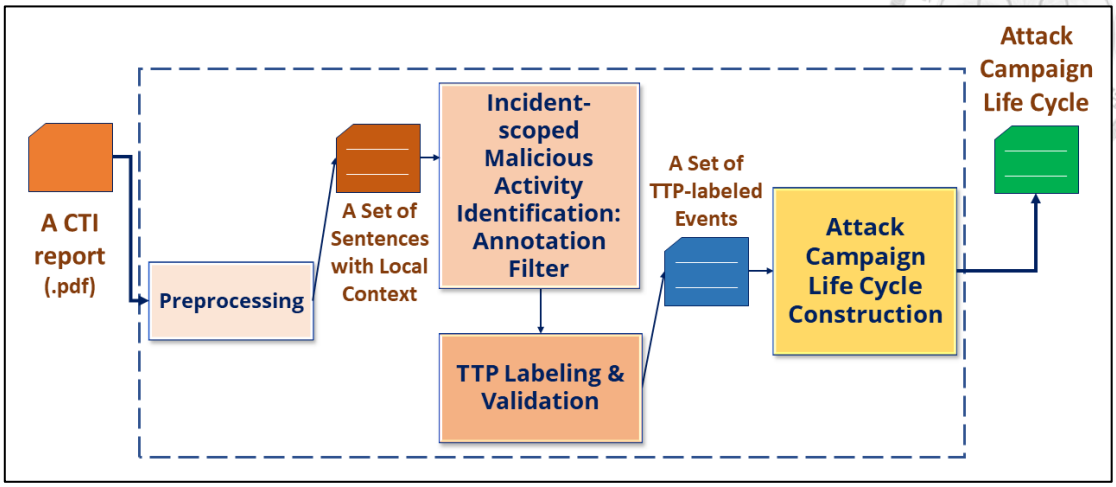


Figure 3.1: System Architecture

Figure 3.1: System Architecture illustrates the complete pipeline for constructing an attack life cycle from a single CTI report. The system operates in a staged manner, transforming unstructured threat intelligence into a structured, phase-organized representation of adversary behavior.

3.1 CTI Preprocessing

Prior to malicious activity identification, CTI reports are preprocessed to produce analyzable text segments. Non-content elements—such as titles, section headers, tables, figures, captions, and appendices—are removed, retaining only the narrative text contained within paragraphs. The remaining text is then segmented into individual sentences, each of which is assigned a unique identifier. Based on our observations, tables and figures do not contain descriptions of malicious activity but instead provide ancillary information; therefore, their removal does not affect the completeness of the constructed attack life cycle.

To preserve local context, each sentence is embedded within a context window consisting of the three preceding and three following sentences, when available. Each

doi:10.6342/NTU202600249

segment is represented as a structured object containing a target sentence and its surrounding context. The context window is used solely to disambiguate incident scope and clarify whether the target sentence describes an executed action, rather than background information or general capability.

This segmentation strategy provides sufficient contextual information for interpretation while maintaining sentence-level granularity, enabling consistent downstream reasoning and annotation.

3.2 Incident-scoped Malicious Activity Identification

Challenges in Interpreting CTI Narratives

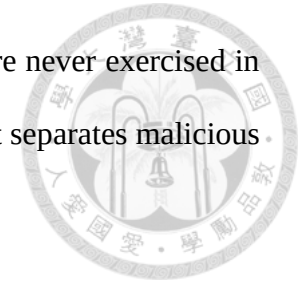
CTI reports describe adversarial behavior using heterogeneous, narrative-driven language that often differs substantially from the abstract and canonical descriptions used in structured frameworks such as MITRE ATT&CK. Attack activities may be described implicitly, spread across multiple sentences, or embedded within broader analytical commentary. As a result, the semantic correspondence between CTI-derived activity statements and ATT&CK technique descriptions is frequently weak, underspecified, or indirect.

This mismatch presents a significant challenge for automated analysis. Approaches that rely on surface-level semantic similarity or keyword overlap between CTI text and ATT&CK descriptions often fail to capture the underlying intent or objective of adversary actions. The problem is exacerbated in incident-scoped analysis, where context plays a critical role in determining whether a sentence describes an action performed during the focal campaign, a historical behavior, a general capability, or an analyst inference.

Furthermore, CTI reports commonly interleave descriptions of attacker actions with background information, comparative analysis, detection guidance, and defensive recommendations. Without explicit incident gating, automated systems risk conflating

doi:10.6342/NTU202600249

behaviors from different campaigns or labeling capabilities that were never exercised in the incident being analyzed. These challenges motivate a design that separates malicious activity identification from technique-level classification.



Design Rationale

A key design decision in this work is to decouple malicious activity identification from MITRE ATT&CK technique mapping. Rather than attempting to directly match CTI sentences to ATT&CK techniques based on semantic similarity, the system first identifies and structures incident-scoped malicious activities.

By focusing first on what the attacker actually did, independent of how that behavior should be categorized, the system avoids premature or unsupported technique assignments. The output of this stage is a set of structured malicious activity events that capture action semantics in a form more suitable for reliable alignment with standardized adversary behavior models.

This design reflects the reality of CTI reporting, where descriptions of attacker behavior are often operational and narrative in nature, and only indirectly map to abstract technique definitions.

Annotation Filter

To implement incident-scoped activity identification, we introduce an LLM-based annotation filter. The filter operates on segmented CTI text and reasons over each target sentence together with its local context to determine whether it describes a concrete malicious activity performed as part of the focal incident. A target sentence is labeled as a malicious activity only if it satisfies all of the following criteria:

1. **Concrete Action**

   The sentence describes a specific action or behavior that was performed or attempted by the adversary.

doi:10.6342/NTU202600249

2. **Malicious Intent**

   The action exhibits adversarial intent, such as exploitation, code execution, persistence, command-and-control communication, credential access, data collection, or evasion.

3. **Incident Attribution**

   The action is directly attributable to the focal incident described in the report, rather than to historical activity, another campaign, or general actor capability.

Sentences that are descriptive, analytical, speculative, capability-based, historical, defensive, or related to other incidents are explicitly excluded, even if they reference malicious techniques.

Event Representation

For sentences identified as malicious activities, the annotation filter produces a structured event representation consisting of:

- a unique event identifier inherited from the preprocessing stage,
- a concise summary capturing the action semantics, and
- a brief justification explaining why the sentence was labeled as a malicious activity in the context of the incident.

These structured events serve as the input to downstream technique labeling and attack life cycle construction. By abstracting narrative CTI text into incident-scoped malicious activity events, this representation facilitates more reliable alignment with standardized adversary behavior frameworks in later stages.

3.3 TTP Labeling

A core requirement for constructing attack life cycles from CTI reports is to represent extracted malicious activity events in a standardized behavioral vocabulary. This work adopts MITRE ATT&CK techniques as the technique-level representation of

doi:10.6342/NTU202600249

adversary behavior. Given a sequence of malicious activity events—each represented by concise activity statements—the TTP labeling task assigns each event either (i) the most appropriate ATT&CK technique(s) or (ii) no technique when the available textual evidence is insufficient.

TTP labeling from CTI text is challenging for three primary reasons. First, although CTI reports often include a list of ATT&CK techniques associated with a campaign, the linkage between individual activity descriptions and specific techniques is rarely explicit. Second, unconstrained technique assignment risks over-labeling, introducing techniques that are not directly supported by the report evidence.

To address these challenges, the TTP labeling process in this work comprises three settings:

A. **Baseline Labeling** (closed-world): the model selects from the CTI-listed techniques.

B. **Open-World Labeling** (unconstrained): the model selects from the full ATT&CK technique space.

C. **Validation** (evidence-based adjudication): a separate LLM evaluates the labels produced in **A** and **B** using strict, explicit-meaning-only reasoning and may accept, reject, or refine them.

A. Baseline Labeling (Closed-World)

In the Baseline (closed-world) labeling setting, the language model is provided with a sequence of malicious activity events together with a restricted set of MITRE ATT&CK techniques explicitly listed in the CTI report. For each event, the model determines whether any technique from this predefined list is supported by the activity description. If no listed technique matches the event, the model is permitted to leave the event unlabeled.

doi:10.6342/NTU202600249

This setting evaluates how effectively an LLM can perform event-to-technique mapping when constrained to the same candidate technique set available to a human analyst reading the report. By restricting the output space to CTI-listed techniques, this approach serves as a baseline for understanding the limitations of direct, report-bounded labeling without introducing additional techniques beyond those already referenced by the report authors.

B. Open-World Labeling

In the open-world labeling setting, the language model is tasked with labeling each malicious activity event using the full MITRE ATT&CK technique space, without restricting predictions to techniques listed in the source report. The model selects the technique that best matches the described behavior and provides a brief rationale for its decision.

This setting reflects an analyst-style labeling scenario in which technique selection is driven by semantic interpretation rather than report-level constraints. While open-world labeling enables the identification of relevant techniques that may be omitted from the CTI report's technique list, it also introduces greater variability and the risk of assigning techniques that are not explicitly supported by the textual evidence.

C. Validation

The technique labels produced under the baseline and open-world settings may differ. To address these discrepancies, we introduce an LLM-based validation stage that evaluates and adjudicates the candidate labels generated by both approaches. This stage employs an independent language model to assess the techniques assigned in the closed-world and open-world labeling processes. Rather than enforcing a binary selection between the two, the validator applies strict explicit-meaning-only reasoning, grounding its decisions solely in information that is directly stated in the malicious activity

doi:10.6342/NTU202600249

description.

For each event, the validator may produce one of the following outcomes:

1. accept the Baseline (closed-world) technique,
2. accept the open-world technique,
3. accept both techniques when each is explicitly supported by the text,
4. reject both techniques and leave the event unlabeled, or
5. when neither candidate is supported but the text clearly describes a different ATT&CK-defined behavior, suggest an alternative technique that better matches the explicit evidence.

This design allows the validation stage to function not only as a conflict resolver, but also as a conservative evidence filter that prevents forced or speculative mappings while preserving valid multi-technique behaviors. These validated TTP-labeled events serve as the foundation for subsequent attack life cycle construction and synthesis, as described in later sections.

3.4 Attack Life Cycle Construction

The goal of attack life cycle construction is to transform a set of independently extracted and validated malicious activity events into a coherent, campaign-level representation of adversary behavior. While individual events capture localized actions described in Cyber Threat Intelligence (CTI) reports, analysts are ultimately interested in understanding how these actions relate to one another temporally and strategically, and how they collectively form an end-to-end attack campaign.

In this work, attack life cycle construction is performed after malicious activity events have been extracted and assigned validated MITRE ATT&CK techniques, as described in previous sections. The construction process explicitly acknowledges that CTI reports rarely provide a complete, linear, or temporally explicit narrative of an attack.

doi:10.6342/NTU202600249

Instead, attack activities are often described in fragmented, overlapping, or implicit forms.

To address these challenges, the proposed system constructs attack life cycles using a multi-stage, modular pipeline composed of three distinct LLM-assisted tasks:

1. attack life cycle construction (event ordering, causality inference, and consolidation),
2. attack phase mapping using Mandiant's Attack Life Cycle model, and
3. automated visualization of the resulting life cycle using Mermaid diagrams.

This separation of concerns improves interpretability, controllability, and reproducibility, while allowing each task to focus on a well-defined reasoning objective.

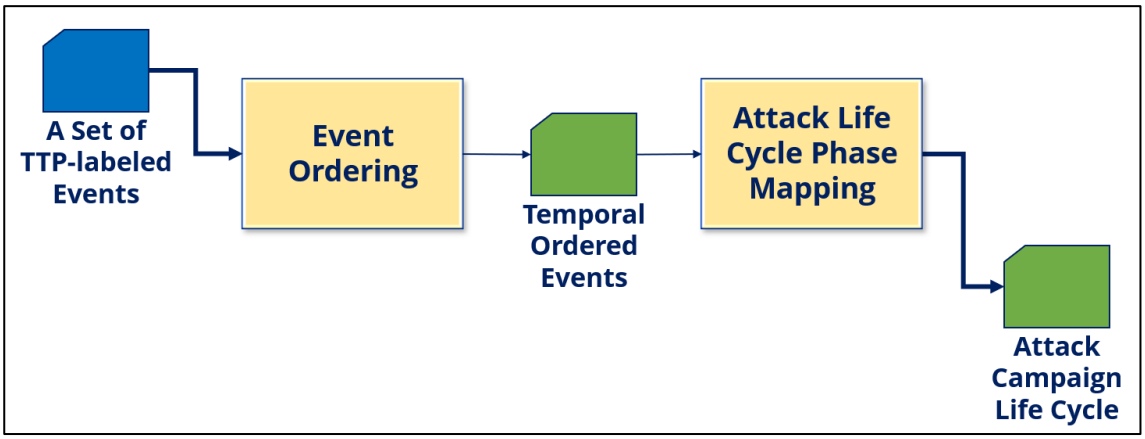


Figure 3.2: Attack Life Cycle Construction Modules

Inputs to Attack Life Cycle Construction

The input to the attack life cycle construction module is a set of validated malicious activity events, where each event consists of:

- the event identifier, concise action summary, and incident-related justification produced by the annotation filter; and
- zero, one, or multiple MITRE ATT&CK techniques validated through the labeling pipeline.

doi:10.6342/NTU202600249

Events are retained even when no ATT&CK technique is assigned, provided the behavior represents a meaningful operational step (e.g., execution flow control, task orchestration, or internal program logic). This design choice avoids artificially constraining the constructed life cycle to the coverage of the ATT&CK framework and preserves behaviors that are relevant for understanding campaign execution.

Event Ordering

The language model is provided with the complete set of validated events associated with an attack campaign and is tasked with ordering these events into a coherent temporal progression. Because CTI reports rarely state the temporal order of individual activities explicitly, the model does not rely solely on temporal markers in the source reports. Instead, it infers event ordering by reasoning about:

- attacker intent,
- functional prerequisites between actions, and
- inter-event causality implied by the event descriptions and corresponding ATT&CK semantics.

For each event included in the attack life cycle, the construction process further identifies causal relationships in terms of:

- **Prerequisite relationships**, indicating which prior events must have occurred for the current event to be possible; and
- **Enablement relationships**, indicating which subsequent events are facilitated by the current event.

These causal relationships encode implicit temporal constraints between events, indicating which actions must precede others and which actions occur subsequently as a result.

Event Merging

doi:10.6342/NTU202600249

CTI reports frequently describe the same attacker behavior multiple times at different levels of abstraction. Without consolidation, such redundancy can distort the constructed life cycle.

Events are merged when they describe the same attacker action and serve the same functional role within the campaign. When merging is performed, the system retains the most concrete and informative representation of the behavior. Higher-level summaries or less detailed descriptions are recorded as merged events, along with explicit justifications for the merge decision.

To ensure completeness and interpretability, the model is required to account for every input event. Each event must be handled in one of two ways:

1. included in the attack life cycle with an explicit position in the ordered sequence; or
2. merged with another event, in which case the model must provide a justification explaining why the event is redundant or subsumed.

This requirement prevents silent omission of evidence and ensures that attack life cycle consolidation decisions are explicit and auditable.

Attack Life Cycle Phase Mapping

After the attack life cycle has been constructed, phase mapping is performed as a second, independent LLM-assisted task. The phase mapping module receives the fully ordered attack life cycle as input, where each event is already annotated with validated ATT&CK techniques and causal relationships.

The objective of this stage is to classify attack life cycle events into Mandiant's Attack Life Cycle phases, while preserving the inferred temporal order established during life cycle construction. Unlike life cycle construction, this task does not reorder or merge events; it functions purely as a labeling layer over an already structured sequence.

doi:10.6342/NTU202600249

Phase mapping uses a fixed set of phase labels derived from Mandiant's Attack Life Cycle model [10], including:

- Initial Reconnaissance
- Initial Compromise
- Establish Foothold
- Escalate Privileges
- Internal Reconnaissance
- Move Laterally
- Maintain Presence
- Complete Mission

Automated Diagram Generation

To improve interpretability and support analyst-facing presentation, the system includes an automated visualization module that converts the structured attack life cycle representation into a diagram.

This module operates as a third, independent LLM-assisted task. It receives the machine-readable life cycle representation (including ordered events, ATT&CK techniques, causal relationships, and phase assignments) and generates a diagram using Mermaid syntax [11].

3.5 Demo Case: Deceptive Development

This section presents a detailed case study to demonstrate the proposed framework in an end-to-end, real-world setting. The objective of the case study is to illustrate how malicious activity events extracted from a Cyber Threat Intelligence (CTI) report can be transformed into a coherent attack life cycle through validated technique labeling, event ordering, phase mapping, and visualization.

Deceptive Development Campaign

doi:10.6342/NTU202600249

The Deceptive Development campaign is a long-running operation conducted by North Korea–aligned threat actors targeting freelance software developers. In this campaign, attackers pose as recruiters on job-hunting, freelancing, and code-hosting platforms, luring victims into fraudulent hiring processes that involve coding challenges or technical assessments.

Victims are induced to download and execute trojanized software projects or fake conferencing tools. These artifacts deliver a first-stage infostealer and downloader, commonly referred to as BeaverTail, which subsequently deploys a more capable Python-based backdoor and infostealer known as InvisibleFerret. The malware targets cryptocurrency wallets, browser-stored credentials, password managers, and system information across Windows, macOS, and Linux platforms, with a strong emphasis on stealing cryptocurrency assets and other financial data.

Following initial compromise, the attackers frequently establish persistent interactive access to victim systems, often through legitimate remote access software such as AnyDesk. This access enables hands-on control, selective data exfiltration, and continued theft operations via attacker-controlled command-and-control infrastructure.

TTP Labeling Result

After preprocessing, the Deceptive Development report published by ESET Research contained 179 sentence segments [12], of which 56 malicious activity events were identified by the annotation filter and labeled independently using the Baseline and open-world strategies. The comparison of their labeling outcomes reveals substantial divergence between the two approaches.

doi:10.6342/NTU202600249

Method	Report: Deceptive Development (41 CTI-listed techniques)	Malicious (total 179)		TTP Labels	TP	FN	FP
Baseline	Sentence segmentation -> LLM annotation filter (summary statement + justification statement) -> **LLM (CTI-listed tech constrained labeling)**	56	36*	21	21	20	-
Open-world	Sentence segmentation -> LLM annotation filter (summary statement + justification statement) -> **LLM (MITRE tech labeling)**	56	56	27	7	34	20
Validation	**LLM validates the TTP labels (explicitly stated evidence with no inference)**	**56**	**53***	**28**	**14**	**27**	**14**

Table 3.1: TTP Labeling Result

Among the 56 events, 9 events received identical technique assignments from both Baseline and open-world. These events typically correspond to behaviors that are explicitly described in the CTI text and map cleanly to well-defined MITRE ATT&CK techniques, leaving little room for alternative interpretations.

The remaining 47 events exhibit differences in labeling outcomes between the two approaches. Within this set, 20 events were labeled only by the open-world approach, while the Baseline approach did not assign any technique. This pattern reflects cases where the described behavior is not covered by the CTI-listed technique but can be mapped to a technique when considering the full ATT&CK taxonomy.

To resolve these disagreements, all 47 events with differing labeling outcomes were subsequently processed by the LLM-based validation stage. The validation outcomes are distributed as follows:

- 25 events where validation agrees with the open-world technique assignment;
- 14 events where validation agrees with the Baseline technique assignment;
- 5 events where validation rejects both candidate techniques and suggests an alternative MITRE ATT&CK technique;
- 3 events where no suitable MITRE ATT&CK technique could be assigned based on the explicit evidence in the event description.

No events in the demo case resulted in validation agreeing with both Baseline and

doi:10.6342/NTU202600249

open-world labels simultaneously, although such cases were observed in other campaigns not discussed in this chapter.

Example Where Validation Agrees with Baseline

One event is summarized as follows:

"Used established-looking profiles and expanded to additional job-hunting and code-hosting websites to appear trustworthy."

The Baseline approach assigns the technique T1585.001 – Establish Accounts, whereas the open-world approach assigns T1656 – Impersonation.

- Example event
  - Summary: "**Used** established-looking profiles and **expanded** to additional job-hunting and code-hosting websites to appear trustworthy."
  - Justification: "This sentence describes an operational change in attacker behavior directly tied to executing the campaign."
- Baseline Technique: **T1585.001 – Establish Accounts**
- Open-world Technique: **T1656 – Impersonation**

#1: Explicit facts
- Attackers **used profiles**.
- The profiles were **established-looking**.
- Attackers **expanded to additional websites**.
- The websites were **job-hunting and code-hosting platforms**.
- The purpose was to **appear trustworthy**.

Figure 3.3: Example Where Validation Agrees with Baseline Labeling

The event description establishes several explicit facts: attackers used profiles that appeared established, expanded their presence to additional job-hunting and code-hosting platforms, and did so with the stated purpose of appearing trustworthy. These actions describe the use and expansion of attacker-controlled accounts across platforms in support of campaign operations.

doi:10.6342/NTU202600249

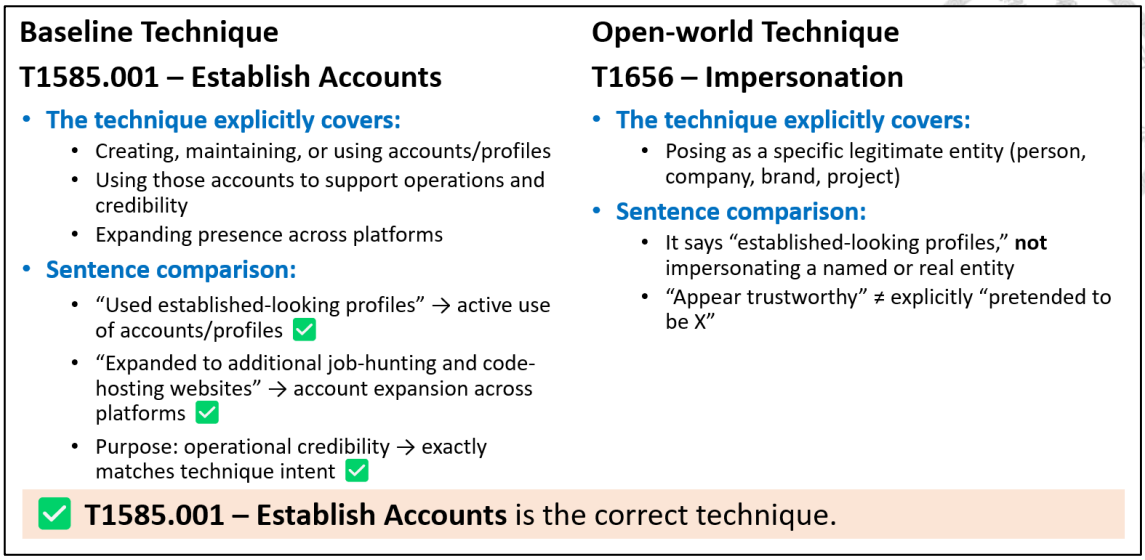


Figure 3.4: Rationale for the Example Where Validation Agrees with Baseline Labeling

Technique T1585.001 – Establish Accounts explicitly covers adversarial behaviors involving the creation, maintenance, or use of accounts and profiles to support operations and build credibility. The use of established-looking profiles and expansion across multiple platforms directly align with this technique's definition and intent.

In contrast, T1656 – Impersonation requires adversaries to pose as a specific legitimate entity, such as a named individual, organization, brand, or project. The event description does not state that the attackers impersonated any identifiable entity. Appearing trustworthy through established-looking profiles does not, by itself, constitute impersonation under the ATT&CK definition.

Accordingly, the validation process agrees with the Baseline assignment and selects T1585.001 – Establish Accounts as the appropriate technique for this event.

Example Where Validation Agrees with Open-World

One event is summarized as follows:

"Inserted malicious code into benign project components by appending it after long comments to conceal it."

doi:10.6342/NTU202600249

The Baseline approach assigns the technique T1140 – Deobfuscate/Decode Files or Information, whereas the open-world approach assigns T1027 – Obfuscated/Compressed Files and Information.

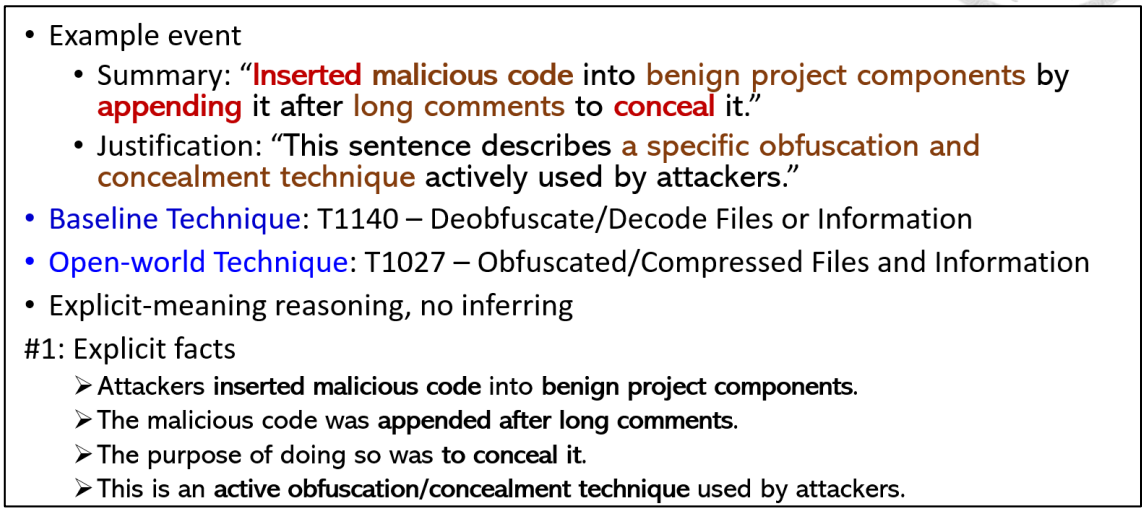


Figure 3.5: Example Where Validation Agrees with Open-World Labeling

The event description establishes several explicit facts: malicious code is inserted into otherwise benign components, the code is appended after long comments, and the stated purpose of this placement is to conceal the malicious logic. Together, these statements describe an active obfuscation or concealment action performed by the attacker.

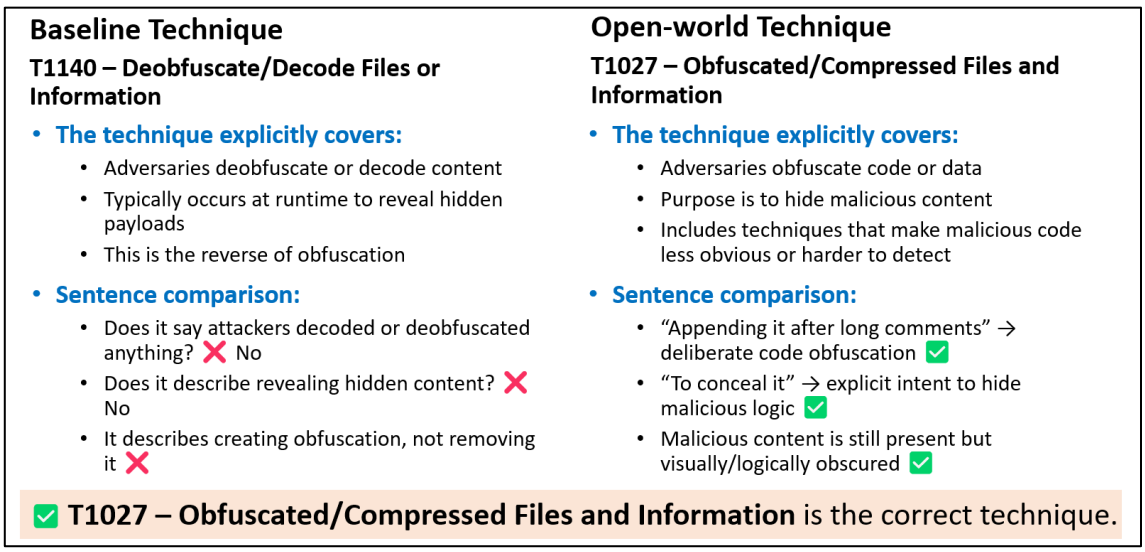


Figure 3.6: Rationale for the Example Where Validation Agrees with Open-World Labeling

doi:10.6342/NTU202600249

Technique T1140 – Deobfuscate/Decode Files or Information explicitly refers to adversaries removing obfuscation or decoding content, typically at runtime, in order to reveal hidden payloads. The event description does not state that any content is decoded, deobfuscated, or revealed. Instead, it describes the creation of obfuscation rather than its removal. As such, the Baseline technique is not supported by the explicit evidence in the text.

In contrast, T1027 – Obfuscated/Compressed Files and Information covers adversarial behaviors in which malicious code or data is deliberately obfuscated to hinder detection or analysis. The described action of appending malicious code after long comments, combined with the explicitly stated intent to conceal it, directly matches the definition of this technique. No additional assumptions are required to support this mapping.

Accordingly, the validation process agrees with the open-world assignment and selects T1027 – Obfuscated/Compressed Files and Information as the appropriate technique for this event.

Example Where Validation Disagrees with Both and Suggests an Alternative

One event is summarized as follows:

"Posed as recruiters and provided software projects containing concealed infostealing malware to targets."

The Baseline approach assigns T1585.001 – Establish Accounts: Social Media Accounts, while the open-world approach assigns T1566.002 – Phishing: Spearphishing via Link.

doi:10.6342/NTU202600249

- Example event
  - Summary: "Posed as recruiters and provided software projects containing concealed infostealing malware to targets."
  - Justification: "The sentence describes a concrete malicious action performed by the attackers as part of the Deceptive Development campaign since early 2024."
- Baseline Technique: T1585.001 – Establish Accounts: Social Media Accounts
- Open-world Technique: T1566.002 – Phishing: Spearphishing via Link
- Explicit facts:
  - Attackers **posed as recruiters** (impersonation / social engineering).
  - They **provided software projects**.
  - Those projects **contained concealed infostealing malware**.
  - This was a **malicious action** in the campaign.

Figure 3.7: Example Where Validation Disagrees with Both and Suggests an Alternative

The event description establishes several explicit facts: attackers posed as recruiters, provided software projects to targets, and those projects contained concealed infostealing malware. The sentence clearly describes a malicious action performed as part of the campaign, but it does not specify how accounts were created or how the software projects were delivered.

Baseline Technique
T1585.001 – Establish Accounts: Social Media Accounts

- **The technique explicitly covers:**
  - The adversary **created or obtained accounts** (e.g., social media, email, platform accounts)
- **Sentence comparison:**
  - Does it say accounts were **created or obtained**? ❌ No
  - "Posed as recruiters" describes a role or pretext, **not** account creation.

Open-world Technique
T1566.002 – Phishing: Spearphishing via Link

- **The technique explicitly covers:**
  - A **phishing attempt**
  - Delivery **via a link**
- **Sentence comparison:**
  - Does the sentence mention a **link**? ❌ No
  - "Provided software projects" does **not** specify the delivery mechanism.

Figure 3.8: Rationale for the Example Where Validation Disagrees with Both and Suggests an Alternativee

Technique T1585.001 – Establish Accounts: Social Media Accounts explicitly covers the creation or acquisition of accounts on platforms such as social media or online

doi:10.6342/NTU202600249

services. The event description does not state that any accounts were created, obtained, or maintained. Posing as recruiters describes a social role or pretext, not the act of establishing accounts. As such, the Baseline technique is not explicitly supported by the text.

Technique T1566.002 – Phishing: Spearphishing via Link requires a phishing attempt delivered via a link. The event description does not mention the use of links, URLs, or other delivery mechanisms consistent with spearphishing via link. Providing software projects does not, by itself, imply link-based delivery, and inferring such a mechanism would exceed the explicit evidence.

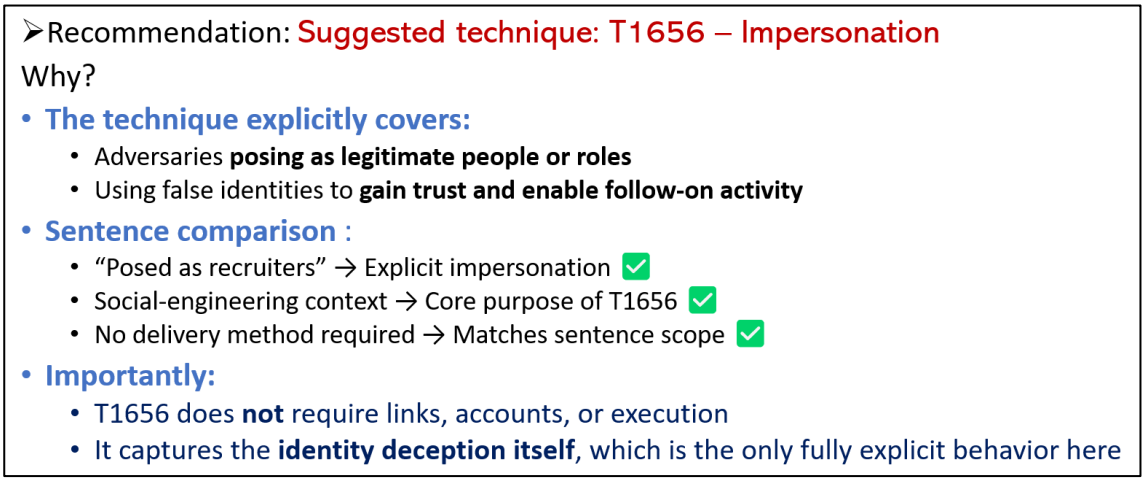


Figure 3.9: Suggestion for the Example Where Validation Disagrees with Both

Given that both candidate techniques require assumptions not supported by the sentence, the validation process rejects both assignments. Instead, it suggests T1656 – Impersonation as the most appropriate technique. This technique explicitly covers adversaries posing as legitimate people or roles to gain trust and enable follow-on malicious activity. The phrase "posed as recruiters" directly and unambiguously describes impersonation, and this behavior is fully supported by the event description without requiring additional assumptions about accounts, links, or execution.

Accordingly, the validation process selects T1656 – Impersonation as the correct

doi:10.6342/NTU202600249

technique for this event.

Example Where No Suitable MITRE ATT&CK Technique Could Be Assigned

One event is summarized as follows:

"Terminated execution on macOS after running the payload module."

The event description establishes several explicit facts: the malware terminated its execution, the behavior occurred on macOS, and the termination took place after the payload module was executed. This describes a control-flow action taken by the malware during its execution.

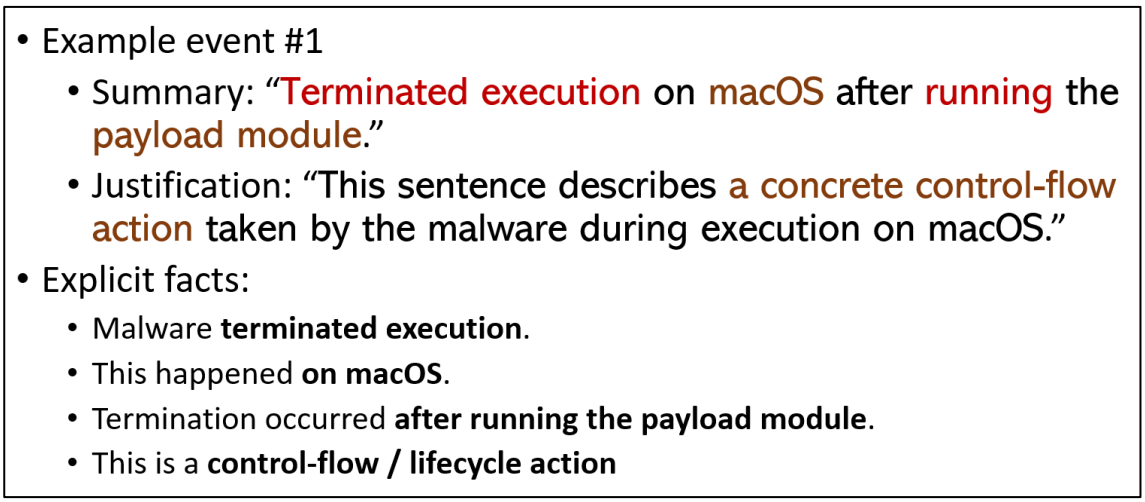


Figure 3.10: Example Where No Suitable MITRE ATT&CK Technique Could Be Assigned

The open-world labeling approach proposes T1070.004 – Indicator Removal on Host: File Deletion, while the Baseline approach does not assign any technique. Technique T1070.004 explicitly covers behaviors related to deleting files or removing artifacts from a host system to evade detection. The event description, however, does not mention file deletion, artifact cleanup, or any form of indicator removal. Terminating execution does not imply deletion of files or removal of on-disk evidence.

doi:10.6342/NTU202600249

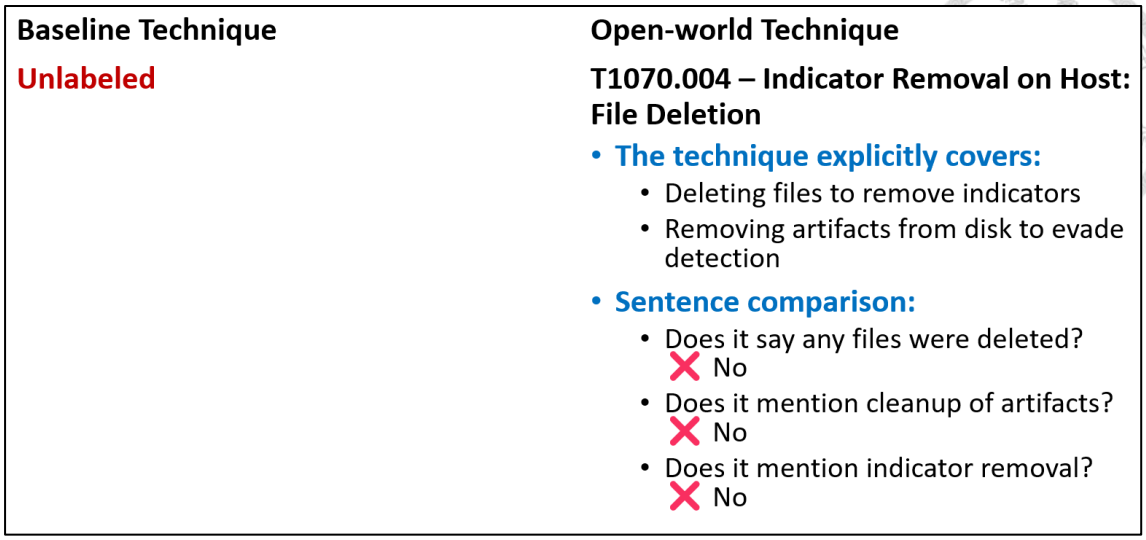


Figure 3.11: Rationale for the Example Where No Suitable MITRE ATT&CK Technique Could Be Assigned

More broadly, the described behavior corresponds to program termination or exit logic. The MITRE ATT&CK framework does not define a technique for self-termination after execution, ending a program's execution flow, or shutting down a payload after completion. These behaviors are implementation-level control-flow decisions rather than adversary techniques in their own right.

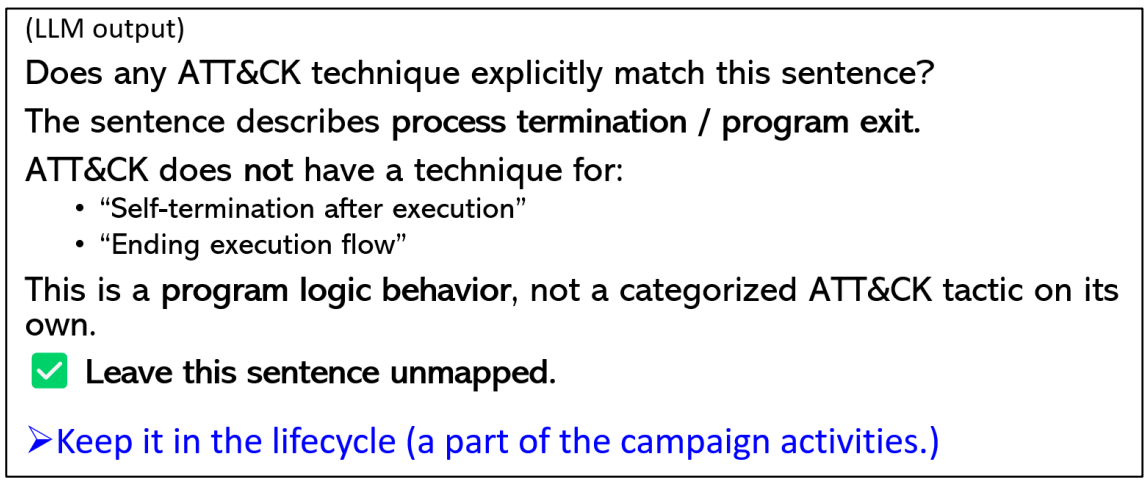


Figure 3.12: Validation Outcome Leaving the Event Unmapped

Accordingly, the validation process leaves this event unlabeled. Importantly, the event is retained for downstream analysis, as it provides useful context about the

doi:10.6342/NTU202600249

malware's execution behavior and contributes to understanding the overall progression of the attack campaign.

Figure 3.13: Second Example Where No Suitable MITRE ATT&CK Technique Could Be Assigned and Figure 3.14: Third Example Where No Suitable MITRE ATT&CK Technique Could Be Assigned show similar unlabeled outcomes where other events describing internal execution logic, such as thread-based command handling within command-and-control loops and process restarts used to reload modified configurations. In these cases, candidate techniques were rejected because the event descriptions lacked explicit evidence for service control, interpreter execution, persistence mechanisms, or defense evasion behaviors defined by ATT&CK. These examples further illustrate that the validation process avoids force-mapping implementation details to techniques when no explicit correspondence exists.

doi:10.6342/NTU202600249

- Example event #2
  - Summary: "Spawned a new thread to execute each received command and immediately resumed listening for further commands."
  - Justification: "This sentence describes concrete remote-command handling behavior as part of C2/backdoor execution."
- Explicit facts:
  1. The malware **received commands**.
  2. For each command, it:
     - **Spawned a new thread**
     - **Executed the command**
  3. It **continued listening** for additional commands.
  4. This is **command-handling logic inside a backdoor/C2 loop**.

Baseline Technique	**Open-world Technique**

Unlabeled	**T1059 – Command and Scripting Interpreter**
	• **The technique explicitly covers:**
	  • Execution **via an interpreter or scripting engine**
	• **This sentence:**
	  • Describes how commands are scheduled/executed internally
	  • Does not describe what mechanism executes them
	  • "Execute command" ≠ "execute via interpreter"

(LLM output)

Why no other ATT&CK technique fits either?

This behavior is:

- **C2 control-flow / concurrency design**
- **Backdoor implementation detail**

ATT&CK does not have a technique for:

- Thread spawning
- Command dispatch loops
- Internal scheduling logic

Those are **implementation details**, not adversary techniques.

✅ **Leave this sentence unmapped.**

➢Keep it in the lifecycle (a part of the campaign activities.)

Figure 3.13: Second Example Where No Suitable MITRE ATT&CK Technique Could Be Assigned

doi:10.6342/NTU202600249

- Example event #3
  - Summary: "**Killed** and **restarted** the AnyDesk process to **load** the modified configuration."
  - Justification: "This sentence describes a concrete process manipulation action to apply attacker-controlled AnyDesk settings."
- Explicit facts:
  - Malware **killed a process**.
  - Malware **restarted a process**.
  - The process was **AnyDesk**.
  - The goal was to **apply modified configuration**.

Baseline Technique
Unlabeled

Open-world Technique
T1569.002 – System Services: Service Execution

- **The technique explicitly covers:**
  - **Starting, stopping, or restarting services**
  - via **service managers** (e.g., Windows SCM)
  - Focus is on **service-level execution control**
- **Sentence comparison:**
  - The sentence says **process**, not **service**
  - Killing/restarting a process can happen without touching service control
  - No service manager or service context is mentioned

Does another ATT&CK technique apply?

ATT&CK does **not** have a technique for:

- Generic process restart
- Reloading configuration by restarting a process

This is **process-lifecycle manipulation**, which ATT&CK does not categorize on its own unless tied to:

- Service control (T1569)
- Autostart persistence (T1547)
- Defense evasion (e.g., killing security tools)

None of those are explicitly stated here.

✅ **Leave this sentence unmapped.**

➢Keep it in the lifecycle (a part of the campaign activities.)

Figure 3.14: Third Example Where No Suitable MITRE ATT&CK Technique Could Be Assigned

doi:10.6342/NTU202600249

TTP Labeling Results Compare with CTI-Listed Techniques

In Table 3.1: TTP Labeling Result, a true positive (TP) denotes a labeled technique that matches a technique listed in the CTI report. A false negative (FN) represents a CTI-listed technique that is not assigned by the labeling process, while a false positive (FP) refers to a technique assigned by the model that does not appear in the CTI-listed technique set. The results show that even under the baseline labeling setting, only slightly more than half of the CTI-listed techniques are identified. This observation empirically supports the claim that the linkage between individual activity descriptions and specific ATT&CK techniques is rarely explicit in CTI reports.

In the validation results, only 14 techniques are counted as true positives, while 27 techniques are classified as false negatives. This high FN count reflects a fundamental limitation of treating CTI-listed techniques as direct ground truth. Because CTI reports often list techniques at a campaign level without explicitly tying them to concrete activity descriptions, the presence of a technique in the CTI list does not guarantee that it is explicitly described in the report text. Consequently, techniques that are not supported by explicit textual evidence are intentionally excluded during validation. At the same time, techniques counted as false positives are explicitly validated during the TTP labeling process and therefore represent techniques that are directly supported by report evidence, even if they are not included in the CTI-listed technique set.

doi:10.6342/NTU202600249

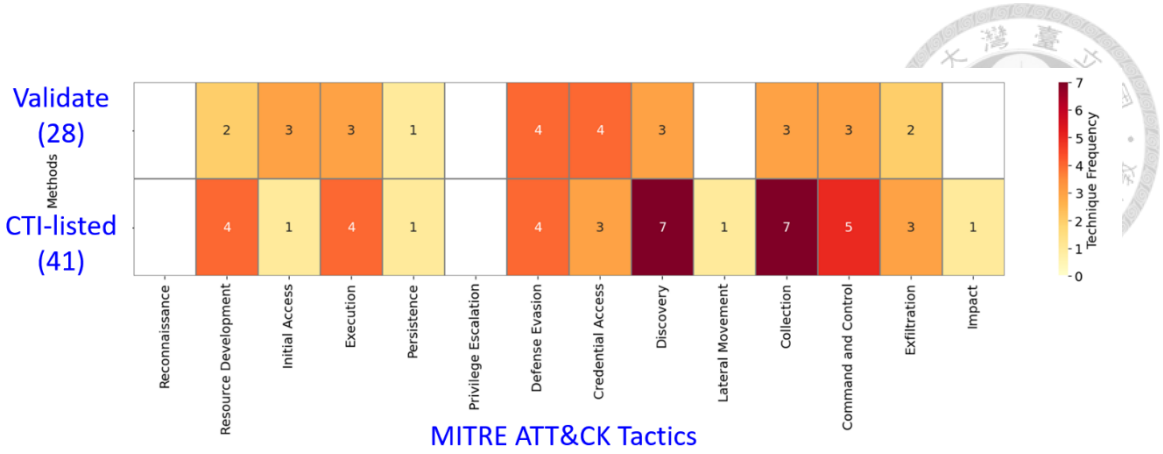


Figure 3.15: TTP Labeling Results Compare with CTI-Listed Techniques

Despite the large number of false negatives at the technique level, Figure 3.15: TTP Labeling Results Compare with CTI-Listed Techniques demonstrates that the validated results still cover most of the ATT&CK tactics—that is, the high-level operational intents—present in the report. This indicates that while fine-grained technique alignment is constrained by reporting ambiguity, the proposed method remains effective at capturing the broader structure and intent of the attack campaign.

Constructed Attack Life Cycle

Figure 3.16: Overview of the Constructed Attack Life Cycle for the Deceptive Development Campaign illustrates the constructed attack life cycle derived from the selected report. The life cycle spans six attack phases, reflecting the progression of the campaign from initial contact with victims to post-compromise objectives. Early phases include social engineering–driven reconnaissance and

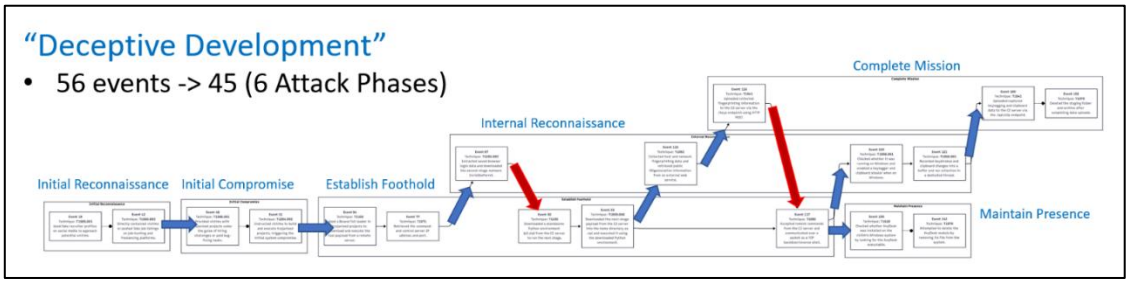


Figure 3.16: Overview of the Constructed Attack Life Cycle for the Deceptive Development Campaign

doi:10.6342/NTU202600249

compromise, followed by establishment of an initial foothold through execution of trojanized software artifacts. Subsequent phases capture internal reconnaissance, maintenance of persistent access, and completion of mission objectives such as data theft.

Importantly, the constructed life cycle is not strictly linear. While events are ordered to reflect their inferred temporal sequence, the resulting structure allows for looping and cross-phase dependencies. This behavior is consistent with real-world attack campaigns and aligns with established attack life cycle models, such as Mandiant's attack life cycle, which emphasize iterative and adaptive attacker behavior rather than a simple linear progression.

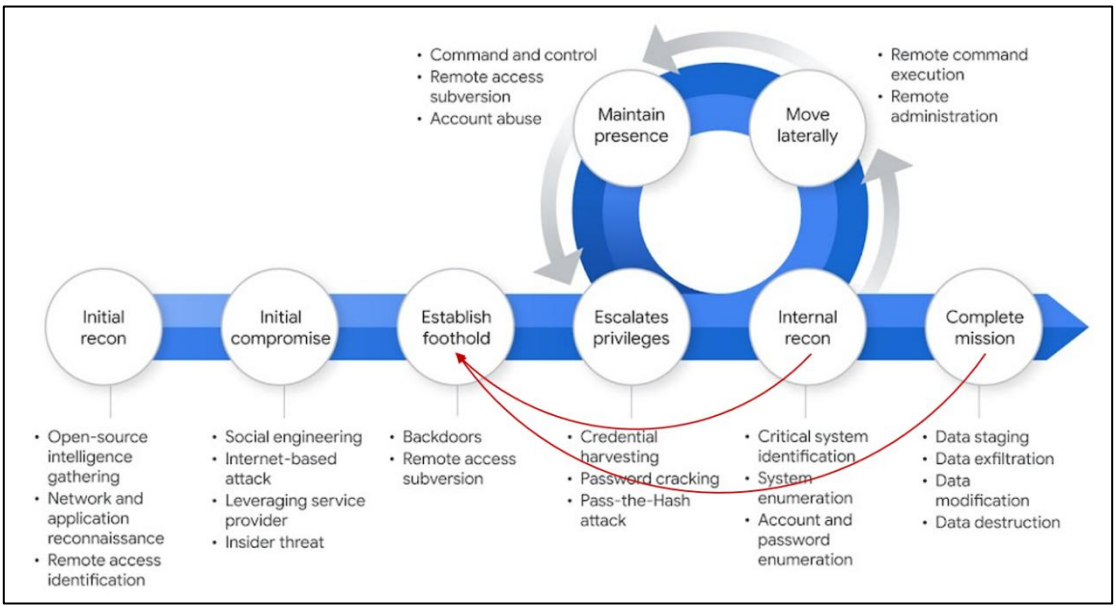


Figure 3.17: Mandiant's Attack Life Cycle.

The red arrows indicate the loop back of the phases in Deceptive Development campaign.

In the constructed attack life cycle, certain events associated with later phases—particularly internal reconnaissance and mission-related activities—support or enable earlier foothold-establishment behaviors.

By explicitly representing these relationships, the constructed life cycle captures aspects of the campaign that are difficult to convey in narrative CTI reports or unordered

doi:10.6342/NTU202600249

lists of techniques. The resulting structure highlights how attacker actions across different phases are interdependent, providing a more faithful representation of the campaign's operational flow as described in the source report.

Event Merging Examples

During attack life cycle construction, the model reasons over the extracted events collectively to infer a coherent progression of attacker activity. In addition to ordering events by inferred temporal relationships, the construction process allows event merging when multiple extracted events describe the same operational step or when differences in granularity would otherwise reduce interpretability.

Two representative merging patterns are observed in the constructed life cycle for the Deceptive Development campaign.

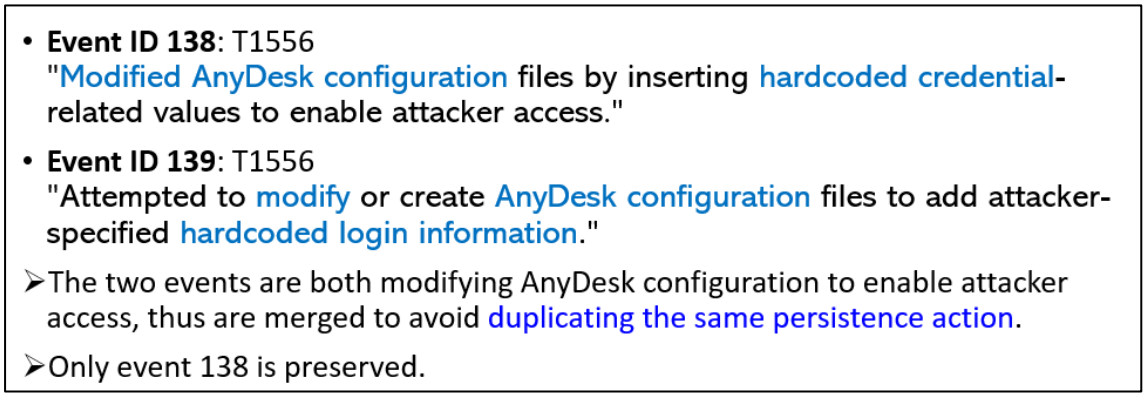


Figure 3.18: Event Merging Example – Duplicate Action

The first pattern, shown in Figure 3.18: Event Merging Example – Duplicate Action, involves duplicate action merging, where multiple events describe the same attacker behavior using slightly different phrasing. For example, two extracted events independently described modification of AnyDesk configuration files to insert attacker-controlled credential values and enable persistent remote access. Although expressed differently in the source report, both events captured the same persistence-related action.

doi:10.6342/NTU202600249

To avoid duplicating this operational step in the life cycle, the events were merged into a single life cycle event representing the modification of AnyDesk configuration to establish attacker access.

The second pattern in Figure 3.19: Event Merging Example – Retaining Detailed Descriptions involves granularity-aware merging, where events differ in their level of detail. In one case, a high-level event described the delivery of trojanized project files that compromised victims' systems, while two other events provided more detailed descriptions of how victims were instructed to build and execute those projects and how the BeaverTail malware was delivered under the guise of legitimate software. Rather than retaining both the abstract summary and the detailed descriptions, the

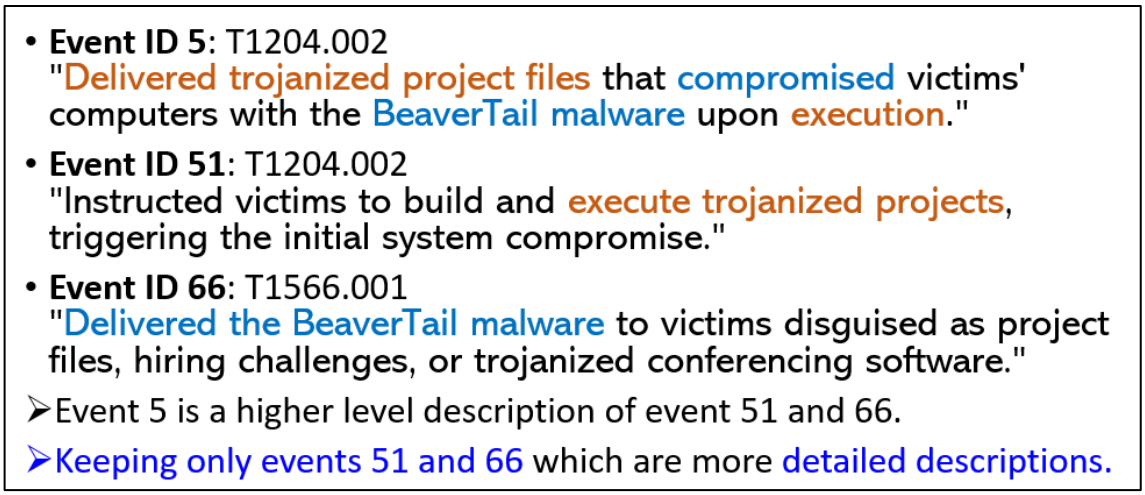


Figure 3.19: Event Merging Example – Retaining Detailed Descriptions

construction process preserved the more specific events and excluded the higher-level summary. This choice ensures that the life cycle reflects concrete attacker actions without losing important operational detail.

Together, these examples illustrate that event merging is applied conservatively and serves to improve clarity rather than reduce informational content. Merging decisions are guided by semantic equivalence and explanatory value, ensuring that each life cycle event

doi:10.6342/NTU202600249

represents a distinct and interpretable step in the campaign.

The Attack Life Cycle Diagram

Figure 3.20: The Whole Visualization of the Deceptive Development Campaign presents a visualization of the constructed attack life cycle for the Deceptive Development campaign derived from the selected CTI report.

In the diagram, malicious activity events are grouped into attack phases to provide a high-level view of campaign progression. Directed edges represent inferred causal relationships between events, indicating how earlier actions enable or support later ones. Importantly, causal links are not constrained to remain within phase boundaries. Cross-phase connections are explicitly retained to reflect the non-linear and iterative nature of real-world attack campaigns.

doi:10.6342/NTU202600249

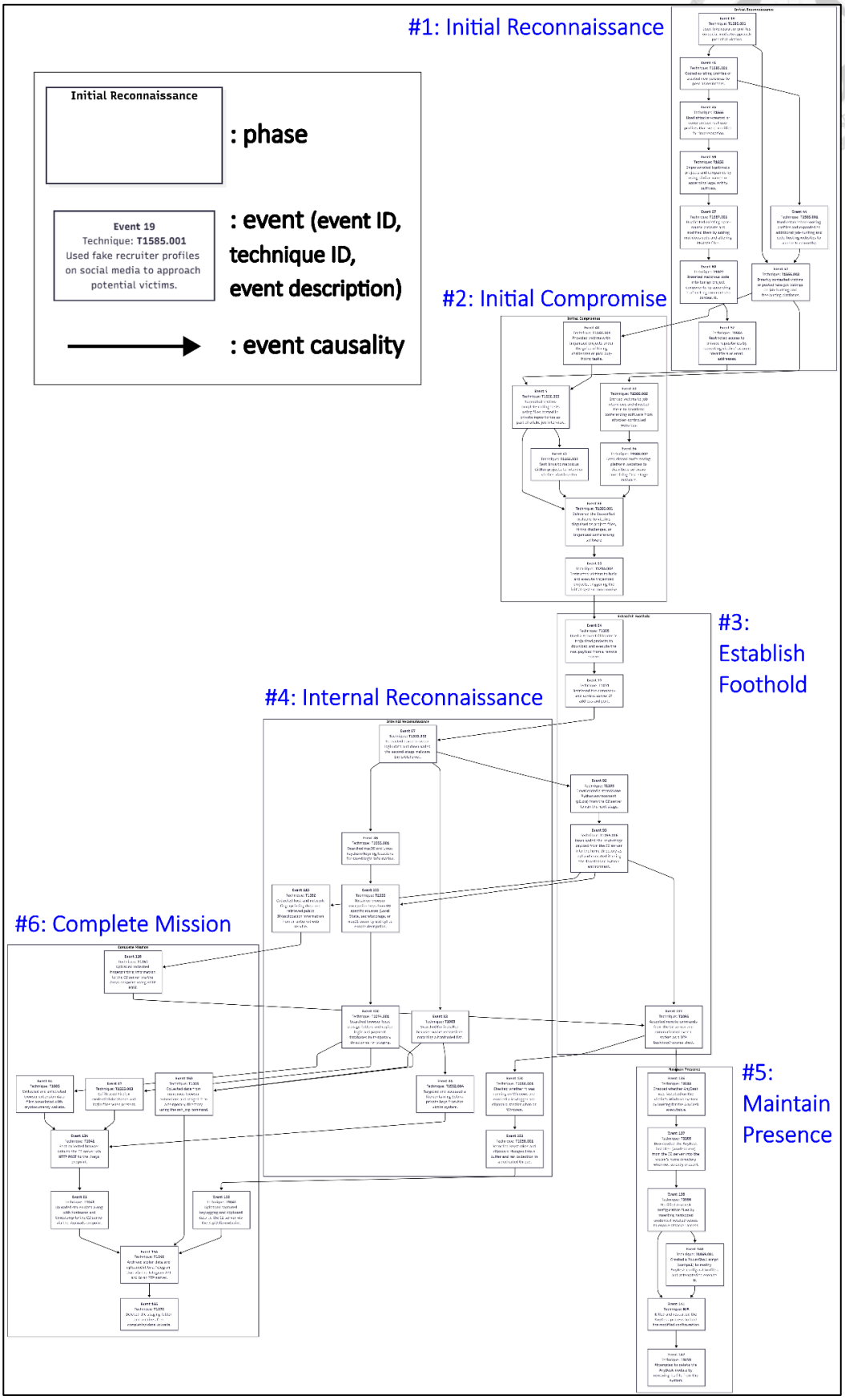


Figure 3.20: The Whole Visualization of the Deceptive Development Campaign

doi:10.6342/NTU202600249

Phase 1. Initial Reconnaissance

The Initial Reconnaissance phase comprises nine malicious activity events and captures the preparatory actions through which the attacker established both social credibility and technical readiness for subsequent stages of the campaign.

At a high level, the attacker established credible recruiter personas and impersonated legitimate companies and projects to build trust with developer targets. These activities allowed the attacker to engage victims in professional contexts where requests to review, build, or execute code appeared reasonable.

In parallel, the attacker prepared weaponized repositories and trojanized software, embedding malicious logic into otherwise benign-looking project components. The attacker also identified suitable victims and positioned delivery channels, including private repositories, fake job listings, and interview-based workflows, that would later be used to distribute malicious artifacts.

Rather than directly compromising victim systems, the actions in this phase collectively served to enable subsequent malware delivery. By the end of Initial Reconnaissance, the attacker had established the trust relationships, technical artifacts, and interaction pathways required to transition into the Initial Compromise phase.

Phase 2. Initial Compromise

The Initial Compromise phase consists of seven malicious activity events and represents the point at which preparatory activities transitioned into direct execution of malicious code on victim systems.

In this phase, the attacker leveraged trusted recruitment and interview workflows established during Initial Reconnaissance to deliver trojanized projects and software to targeted developers. Victims were instructed to build, run, or interact with these artifacts as part of what appeared to be legitimate hiring challenges, paid tasks, or interview

doi:10.6342/NTU202600249

processes.

Execution of these trojanized projects resulted in the successful deployment and execution of the first-stage malware, BeaverTail, on victim systems. This marked the initial establishment of attacker-controlled code execution and created the foundation for subsequent post-compromise activities.

By exploiting trust-based delivery mechanisms rather than overt exploitation, the attacker was able to induce voluntary execution of malicious code, enabling compromise without relying on traditional vulnerability-based attack vectors.

Phase 3. Establish Foothold

The Establish Foothold phase comprises five malicious activity events and captures the attacker's efforts to convert initial code execution into stable, interactive control over compromised systems.

During this phase, the attacker downloaded and executed additional payloads, extending beyond the initial BeaverTail deployment. These actions included establishing command-and-control connectivity, deploying suitable execution environments, and enabling remote command execution through a backdoor. Together, these steps allowed the attacker to interact with victim systems on demand rather than relying solely on the initial execution context.

The activities in this phase transformed a one-time compromise into a durable foothold, providing the attacker with the capability to issue commands, deploy further tooling, and respond dynamically to the victim environment. This foothold served as a prerequisite for both continued operational control and longer-term access mechanisms.

By enabling interactive access early in the campaign, the attacker laid the groundwork for sustained presence and follow-on actions, directly supporting the transition into the Maintain Presence phase.

doi:10.6342/NTU202600249

Phase 4. Internal Reconnaissance

The Internal Reconnaissance phase consists of nine malicious activity events and reflects the attacker's efforts to gain detailed visibility into the compromised victim environment following establishment of interactive access.

In this phase, the attacker collected extensive system and user information, including operating system and network characteristics, credential stores, browser-stored data, encryption keys, cryptocurrency wallets, and indicators of user activity. These activities provided situational awareness necessary to identify valuable targets and prioritize subsequent actions.

In addition to one-time information collection, the attacker enabled continuous surveillance mechanisms, such as keylogging and clipboard capture, allowing ongoing monitoring of user behavior and sensitive input. The combination of broad environment discovery and persistent observation enabled the attacker to both identify high-value data and stage it for later use or exfiltration.

Rather than serving as an isolated step, internal reconnaissance functioned as an enabling phase that informed decisions across later stages of the campaign, including access maintenance and mission execution.

Phase 5. Maintain Presence

The Maintain Presence phase comprises six malicious activity events and captures the attacker's efforts to ensure long-term, reliable access to compromised systems.

In this phase, the attacker installed and configured legitimate remote access software, most notably AnyDesk, to maintain interactive control over victim machines. Configuration files were modified to enable attacker-controlled access, and associated processes or services were restarted to apply these changes. By leveraging widely used legitimate tools, the attacker reduced the likelihood of immediate suspicion while

doi:10.6342/NTU202600249

retaining persistent connectivity.

In addition to establishing remote access, the attacker attempted to reduce detection and operational friction by modifying or removing artifacts associated with their activities. These actions were intended to stabilize access over time rather than to achieve immediate mission objectives.

Rather than representing a terminal stage, the Maintain Presence phase supported both ongoing reconnaissance and mission execution. Persistent access enabled repeated interaction with victim systems, allowing the attacker to adapt tooling, re-collect data, and continue operations as needed.

Phase 6. Complete Mission

The Complete Mission phase consists of nine malicious activity events and represents the attacker's execution of their primary operational objectives following sustained access to victim systems.

In this phase, the attacker collected, staged, and exfiltrated sensitive data from compromised environments. Targeted data included credentials, cryptocurrency wallet information, browser artifacts, keystrokes, and system fingerprints. Collected information was aggregated and prepared for transfer before being delivered to attacker-controlled infrastructure through multiple exfiltration channels.

In addition to data theft, the attacker archived stolen data to facilitate later use and removed or cleaned up staging artifacts associated with data collection and transfer. These actions marked the conclusion of discrete operational cycles while preserving the attacker's ability to re-engage if access remained available.

Although labeled as the final phase, Complete Mission does not imply a definitive end to attacker activity. Instead, mission execution occurred alongside continued access maintenance and surveillance, reflecting the iterative nature of the campaign and the

doi:10.6342/NTU202600249

attackers' emphasis on repeated value extraction rather than a single terminal action.

Cross-Phase Causal Relationships

While grouping events into phases improves interpretability, the constructed attack life cycle reveals several critical causal relationships that span phase boundaries. These cross-phase links illustrate how the campaign progresses as an interconnected sequence rather than a strictly linear chain.

The following subsections walk through the explicit cross-phase causal relationships, explaining how actions in one phase enable or reinforce activities in subsequent phases. Together, these relationships capture the complete set of distinct enablement paths represented in the constructed life cycle.

1. Initial Reconnaissance → Initial Compromise

Figure 3.21: Cross Phase Causal Relationship from Initial Reconnaissance to Initial Compromise highlights several cross-phase causal relationships linking Initial Reconnaissance to Initial Compromise, illustrating how early social engineering

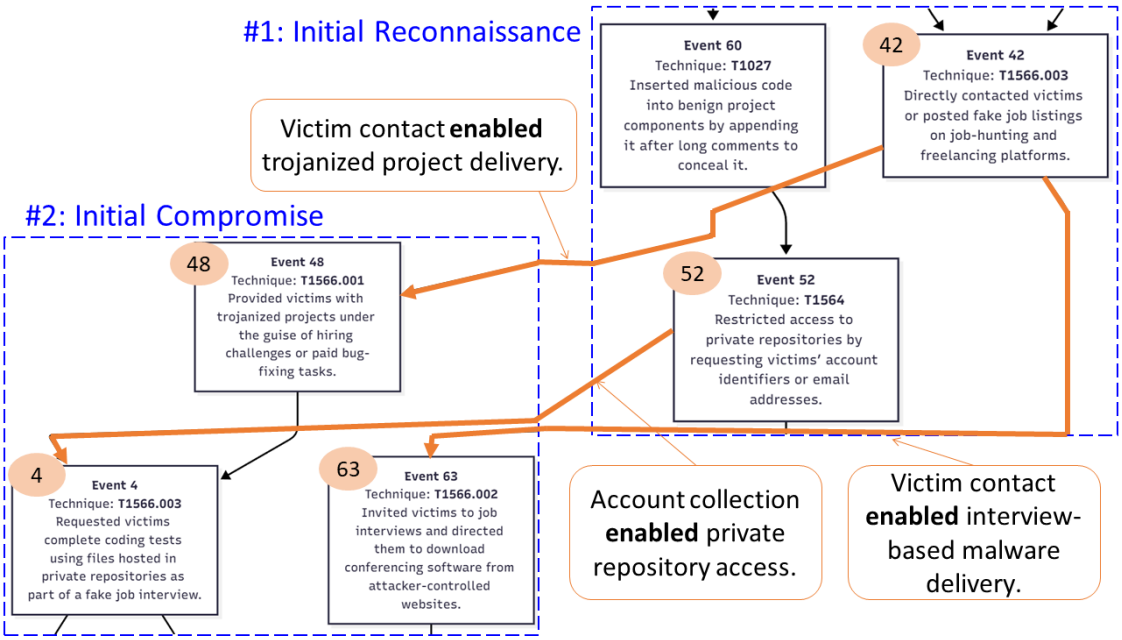
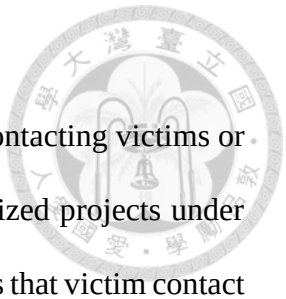


Figure 3.21: Cross Phase Causal Relationship from Initial Reconnaissance to Initial Compromise

and preparation activities directly enabled delivery and execution of malicious artifacts.

doi:10.6342/NTU202600249



First, the diagram shows a causal link from Event 42 (directly contacting victims or posting fake job listings) to Event 48 (providing victims with trojanized projects under the guise of hiring challenges or paid tasks). This relationship indicates that victim contact enabled trojanized project delivery, as engagement through fake recruitment channels created the necessary trust and interaction context for victims to accept and execute attacker-supplied code.

A second cross-phase link connects Event 42 to Event 63, where victims were invited to interviews and directed to download attacker-controlled conferencing software. This arrow represents a distinct delivery pathway in which victim contact enabled interview-based malware delivery, demonstrating how the same reconnaissance-stage social engineering activity supported multiple compromise mechanisms.

Finally, the diagram depicts a causal relationship from Event 52 (requesting victim account identifiers or email addresses) to Event 4 (requesting victims complete coding tests using files hosted in private repositories). This connection captures how account collection enabled access to private repositories, allowing the attacker to control artifact distribution and ensure that malicious projects were accessible only to targeted victims.

Together, these cross-phase relationships demonstrate that compromise-stage execution events were not isolated actions, but rather the direct result of reconnaissance-stage preparation and victim engagement. By making these dependencies explicit, the constructed attack life cycle clarifies how social engineering and access staging enabled multiple malware delivery paths within the campaign.

2. Initial Compromise → Establish Foothold

Figure 3.22: Cross Phase Causal Relationship from Initial Compromise to Establish Foothold also illustrates a direct causal relationship between Initial

doi:10.6342/NTU202600249

Compromise and Establish Foothold, highlighting how user-driven execution events enabled the attacker to transition from initial compromise to stable, interactive access.

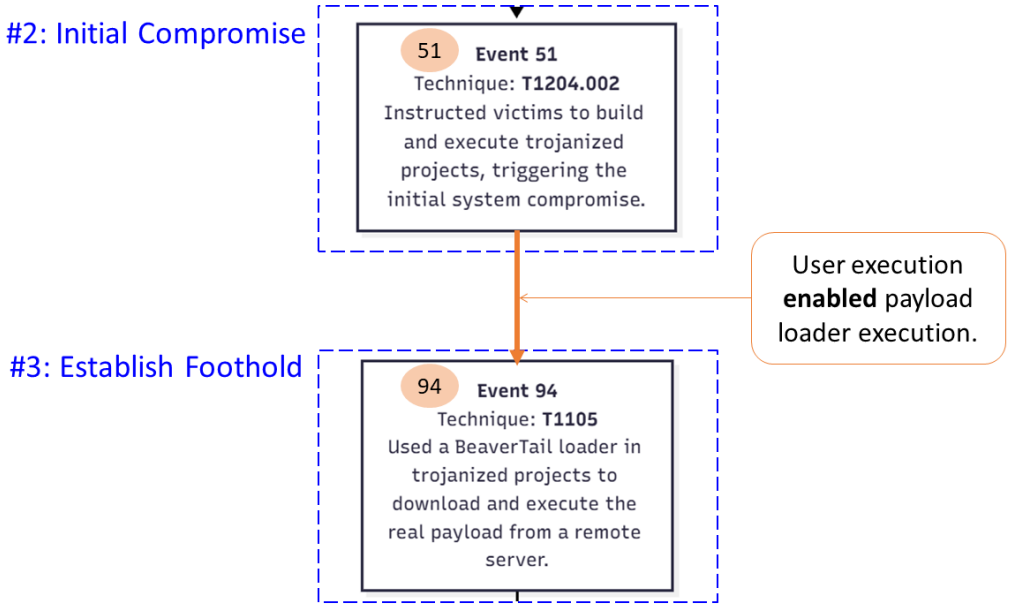


Figure 3.22: Cross Phase Causal Relationship from Initial Compromise to Establish Foothold

In the diagram, Event 51 represents an Initial Compromise activity in which victims were instructed to build and execute trojanized projects, triggering the initial system compromise. A directed arrow connects this event to Event 94 in the Establish Foothold phase, where the BeaverTail loader was used to download and execute the primary payload from a remote server.

This cross-phase link indicates that user execution enabled payload loader execution. The attacker relied on victims' voluntary execution of the trojanized projects to initiate the loader component, which then retrieved and executed additional malicious payloads. Without the initial user-driven execution captured in Event 51, the subsequent loader activity shown in Event 94 could not have occurred.

By explicitly representing this dependency, the diagram clarifies how the campaign progressed from an initial, trust-based compromise to the establishment of attacker-

doi:10.6342/NTU202600249

controlled execution capabilities, marking the transition from transient compromise to a more durable foothold.

3. Establish Foothold → Internal Reconnaissance

Figure 3.23: Cross Phase Causal Relationship from Establish Foothold to Internal Reconnaissance further illustrates how activities in the Establish Foothold phase enabled subsequent Internal Reconnaissance actions by providing connectivity,

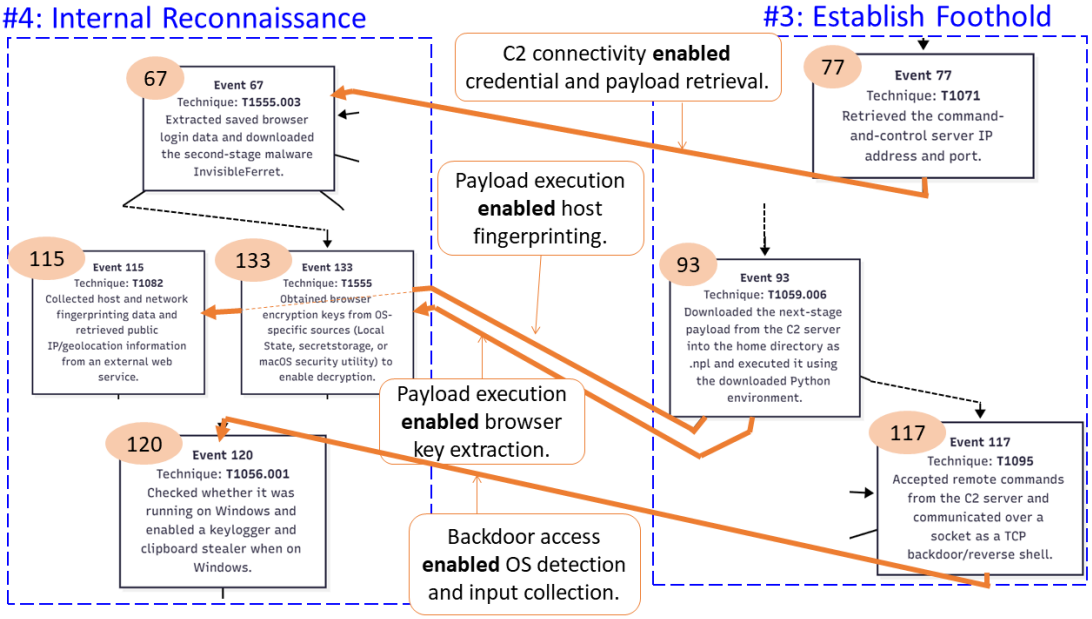


Figure 3.23: Cross Phase Causal Relationship from Establish Foothold to Internal Reconnaissance

execution context, and interactive control over compromised systems.

First, the diagram shows a causal link from Event 77 (retrieval of the command-and-control server IP address and port) to Event 67 (extraction of saved browser login data and retrieval of the second-stage malware). This relationship indicates that C2 connectivity enabled credential and payload retrieval, as the attacker relied on established communication channels to download additional components and access stored credentials.

A second set of cross-phase links connects Event 93 (downloading and executing the next-stage payload) to both Event 115 and Event 133 in the Internal Reconnaissance

doi:10.6342/NTU202600249

phase. These arrows represent that payload execution enabled host fingerprinting and browser key extraction. Once the payload was executed, the malware was able to collect system and network characteristics and obtain encryption keys from OS-specific sources to facilitate decryption of browser-stored data.

Finally, the diagram depicts a causal relationship from Event 117 (accepting remote commands via a backdoor) to Event 120 (checking the operating system and enabling keylogging and clipboard capture). This connection illustrates that backdoor access enabled OS detection and input collection, allowing the attacker to tailor surveillance mechanisms based on the victim environment and continuously monitor user activity.

Together, these cross-phase relationships demonstrate that Internal Reconnaissance activities were not standalone behaviors. Instead, they were directly enabled by foothold-stage actions that established connectivity, executed payloads, and provided interactive control, as explicitly represented by the causal arrows in the constructed attack life cycle diagram.

4. Internal Reconnaissance → Establish Foothold

doi:10.6342/NTU202600249

Figure 3.24: Cross Phase Causal Relationship from Internal Reconnaissance to Establish Foothold illustrates a feedback relationship from Internal Reconnaissance back to Establish Foothold, highlighting how information obtained during reconnaissance directly informed the deployment of additional execution

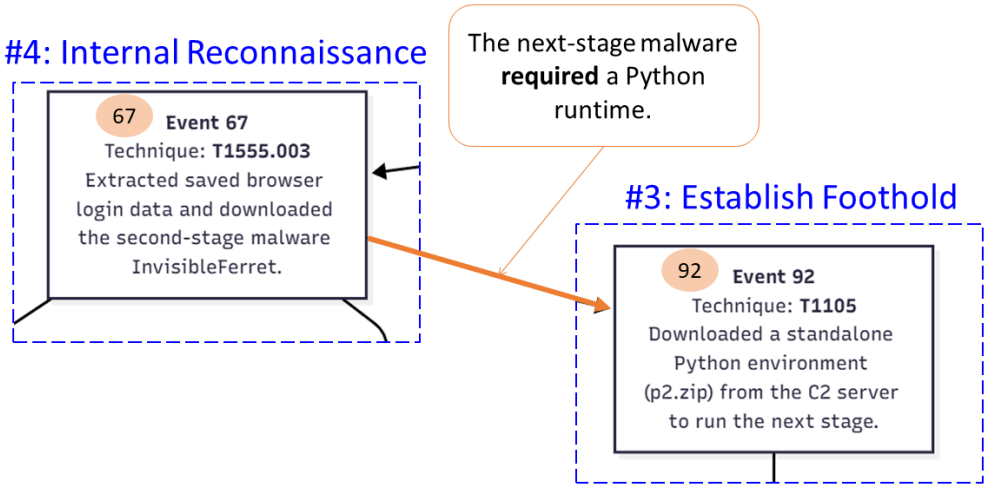


Figure 3.24: Cross Phase Causal Relationship from Internal Reconnaissance to Establish Foothold

infrastructure.

In the diagram, Event 67 in the Internal Reconnaissance phase represents the extraction of saved browser login data and the retrieval of the second-stage malware component. A directed arrow connects this event to Event 92 in the Establish Foothold phase, where the attacker downloaded a standalone Python execution environment from the command-and-control server.

This cross-phase link indicates that execution of the next-stage malware required a Python runtime. After retrieving the second-stage payload, the attacker determined that an appropriate execution environment was not guaranteed to be present on the victim system. As a result, the attacker provisioned a standalone Python environment to ensure reliable execution of subsequent stages.

doi:10.6342/NTU202600249

By explicitly representing this dependency, the diagram shows that foothold-related activities were adaptively extended based on reconnaissance outcomes. Rather than treating foothold establishment as a fixed, one-time step, the constructed life cycle captures how attacker capabilities were incrementally reinforced in response to the technical requirements of later-stage malware.

5. Establish Foothold → Maintain Presence

Figure 3.25: Cross Phase Causal Relationship from Establish Foothold to Maintain Presence illustrates a causal transition from Establish Foothold to Maintain Presence, showing how interactive control over compromised systems enabled deployment of long-term access mechanisms.

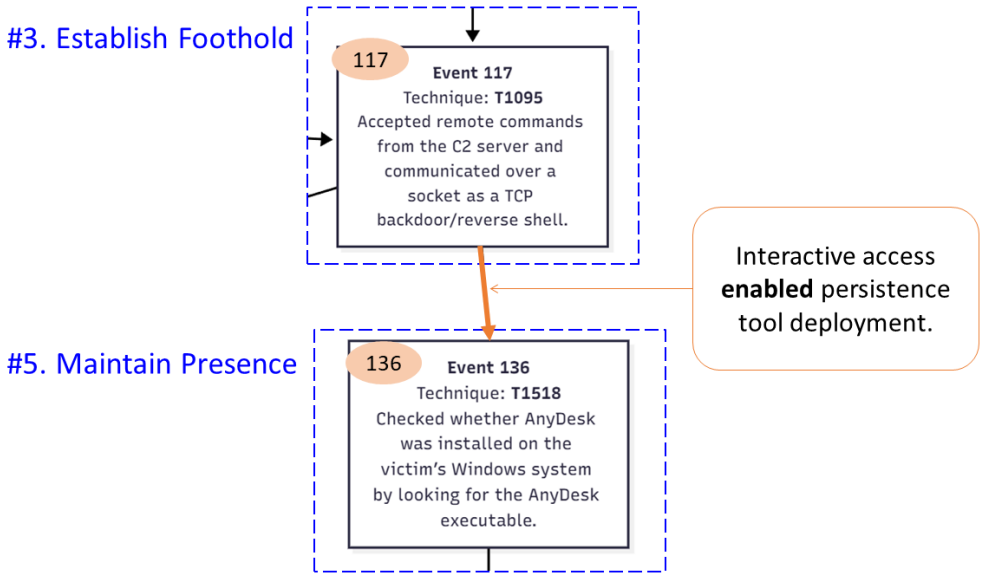


Figure 3.25: Cross Phase Causal Relationship from Establish Foothold to Maintain Presence

In the diagram, Event 117 represents an Establish Foothold activity in which the attacker accepted remote commands from the command-and-control server via a TCP backdoor or reverse shell. A directed arrow connects this event to Event 136 in the Maintain Presence phase, where the attacker checked for the presence of AnyDesk on the

doi:10.6342/NTU202600249

victim's system in preparation for persistent remote access.

This cross-phase link indicates that interactive access enabled persistence tool deployment. Once the attacker obtained an interactive command channel, they were able to inspect the victim environment, determine whether legitimate remote access software was available, and take steps toward configuring it for attacker use. These actions were only possible after the foothold phase established reliable, on-demand command execution.

By explicitly encoding this dependency, the diagram clarifies how the campaign transitioned from transient execution and backdoor access to more durable presence mechanisms, setting the stage for sustained control over victim systems.

6. Internal Reconnaissance → Complete Mission

Figure 3.26: Cross Phase Causal Relationship from Internal Reconnaissance to Complete Mission (1) and Figure 3.27: Cross Phase Causal Relationship from Internal Reconnaissance to Complete Mission (2) depicts a

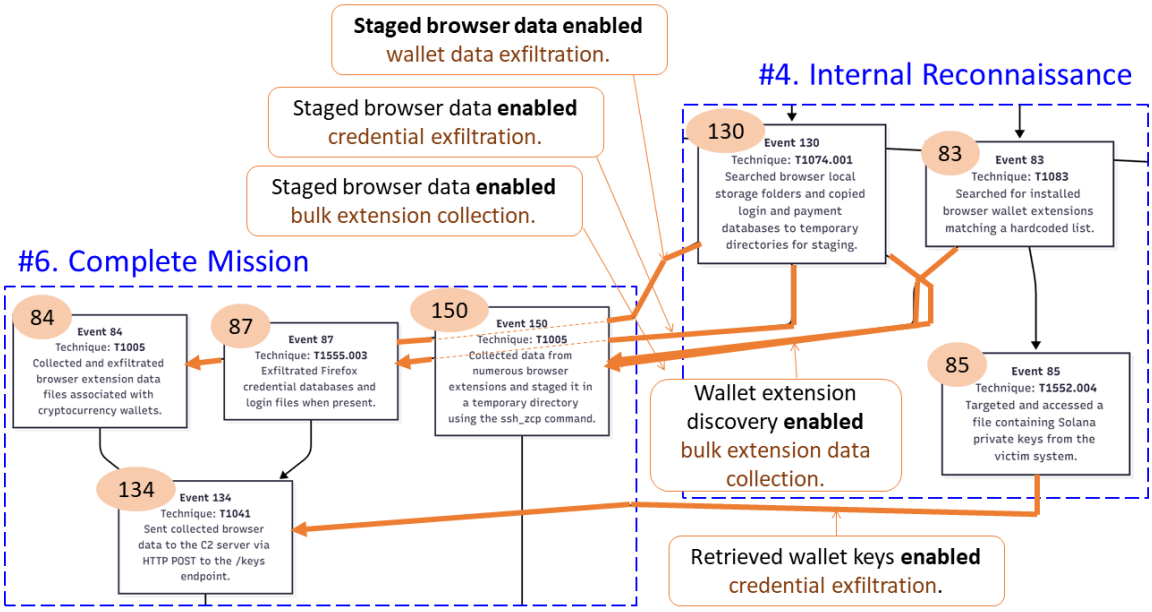


Figure 3.26: Cross Phase Causal Relationship from Internal Reconnaissance to Complete Mission (1)

doi:10.6342/NTU202600249

dense set of cross-phase causal relationships linking Internal Reconnaissance to Complete Mission, illustrating how information discovery, data staging, and continuous surveillance directly enabled multiple exfiltration pathways.

A central enablement pattern shown in the diagram involves browser data staging. In the Internal Reconnaissance phase, Event 130 represents the collection and staging of browser login and payment databases into temporary directories. Multiple directed arrows connect this event to Complete Mission activities, indicating that staged browser data enabled downstream exfiltration actions. Specifically, staged data supported exfiltration of cryptocurrency wallet-related browser artifacts (Event 84), credential databases (Event 87), and bulk browser extension data (Event 150). These links make explicit that exfiltration events were causally dependent on prior staging rather than occurring directly from live collection.

A second enablement path involves wallet-related discovery. As shown by the arrow from Event 83 (discovery of installed browser wallet extensions) to Event 150, wallet extension discovery enabled targeted bulk collection of extension-related data. In addition, Event 85, which captures retrieval of wallet private keys from the victim system, is causally linked to Event 134, indicating that retrieved wallet keys enabled credential

doi:10.6342/NTU202600249

exfiltration to attacker-controlled infrastructure.

The diagram also highlights enablement paths associated with host fingerprinting and continuous surveillance. Information collected during Internal Reconnaissance in

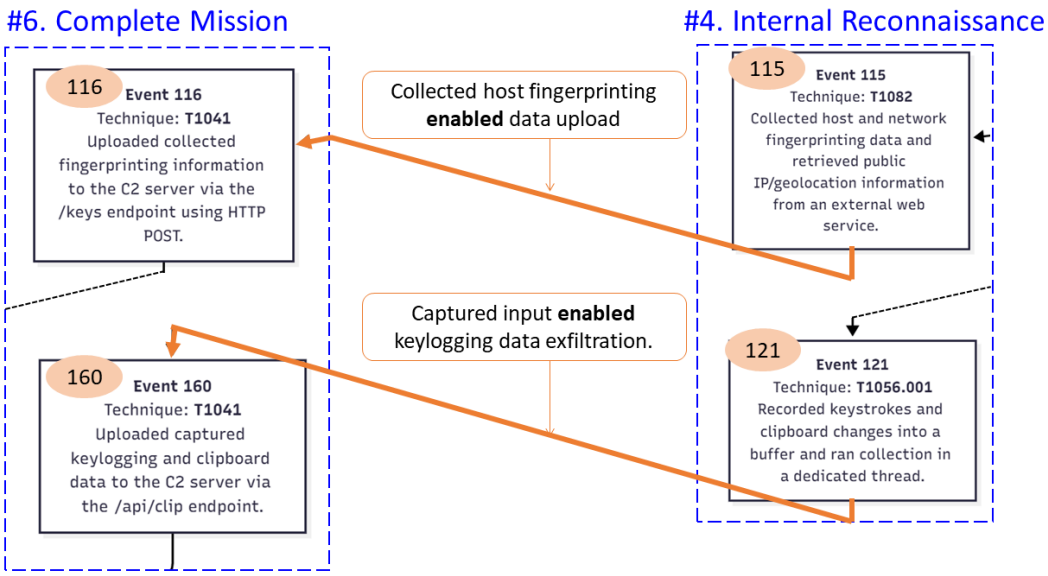


Figure 3.27: Cross Phase Causal Relationship from Internal Reconnaissance to Complete Mission (2)

Event 115 (host and network fingerprinting) is shown to enable subsequent upload of fingerprinting data in Event 116, reflecting a discovery-to-upload relationship. Similarly, Event 121, which records keystrokes and clipboard data, is causally linked to Event 160, where captured input data was uploaded to the command-and-control server. These arrows illustrate how ongoing surveillance directly supported repeated data exfiltration.

Taken together, these relationships demonstrate that the transition from Internal Reconnaissance to Complete Mission is not a single linear step but a set of parallel enablement paths. Discovery activities led to staging, staging enabled multiple forms of exfiltration, and continuous monitoring supported repeated uploads. By explicitly representing these dependencies, the constructed attack life cycle captures the structured and methodical manner in which reconnaissance outputs were transformed into mission-level data theft.

doi:10.6342/NTU202600249

7. Complete Mission → Establish Foothold

Figure 3.28: Cross Phase Causal Relationship from Complete Mission to Establish Foothold also captures a feedback relationship from Complete Mission back to Establish Foothold, illustrating how successful mission-level actions reinforced the attacker's ability to maintain interactive control.

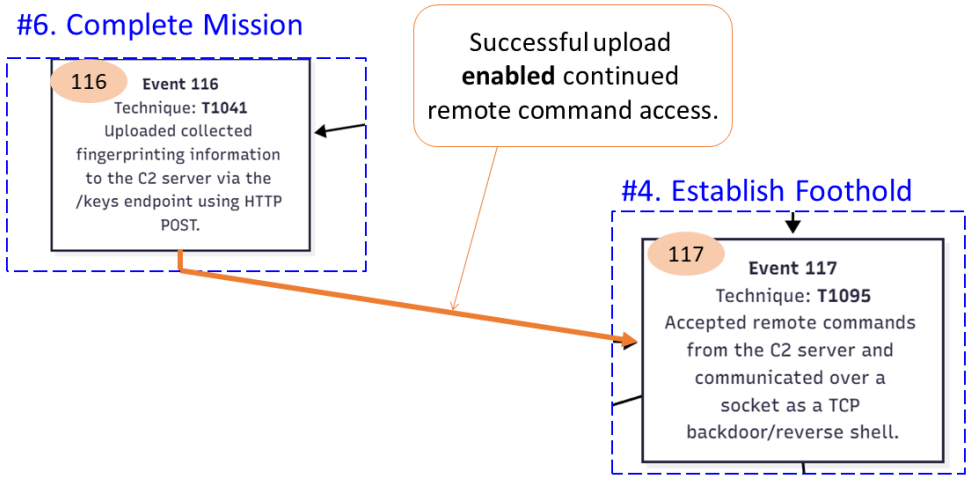


Figure 3.28: Cross Phase Causal Relationship from Complete Mission to Establish Foothold

In the diagram, Event 116 represents the successful upload of collected host fingerprinting information to the command-and-control server. A directed arrow connects this event back to Event 117 in the Establish Foothold phase, where the attacker continued to accept remote commands via a TCP backdoor or reverse shell.

This cross-phase link indicates that successful data upload enabled continued remote command access. By confirming connectivity and the operational status of the command-and-control channel, successful uploads allowed the attacker to sustain interactive access and issue further commands. Rather than marking the end of the campaign, mission completion activities thus fed back into foothold maintenance.

By explicitly representing this feedback loop, the diagram emphasizes that the attack

doi:10.6342/NTU202600249

life cycle is not strictly linear. Mission execution and access maintenance are interdependent, allowing the attacker to iteratively collect data, validate connectivity, and continue operations as long as access remains viable.

This section presented a detailed case study of the Deceptive Development campaign to demonstrate the effectiveness and interpretability of the proposed attack life cycle construction framework when applied to a single Cyber Threat Intelligence report. Starting from unstructured narrative text, the system extracted malicious activity events, aligned them with validated MITRE ATT&CK technique labels, and organized them into a coherent, phase-structured attack life cycle. The resulting representation captures the end-to-end progression of the campaign, from social engineering preparation and initial compromise to foothold establishment, internal reconnaissance, persistent access, and mission execution. The case study establishes a concrete foundation for subsequent chapters, which build upon this representation to explore broader analysis and extensions of the approach.

doi:10.6342/NTU202600249

Chapter 4. Attack Life Cycle Synthesis

4.1 Motivation and Goal

Individual CTI reports often emphasize specific aspects of a campaign, such as initial social engineering tactics, malware internals, command-and-control infrastructure, or persistence mechanisms. As a result, each report provides a partial and incomplete view of the overall attack, even when all reports describe the same underlying campaign. Important behaviors may be omitted, summarized at different levels of abstraction, or described without explicit linkage to other stages of the attack.

For analysts, synthesizing these fragmented observations into a unified campaign narrative requires manual cross-report comparison, reconciliation of overlapping descriptions, and inference of how disparate activities relate temporally and causally. This process is time-consuming and does not scale well as the volume of CTI reporting grows.

Attack life cycle synthesis addresses this challenge by aggregating multiple constructed attack life cycles, each derived from an individual CTI report, into a single, comprehensive campaign representation. By operating on structured life cycles rather than raw text, synthesis builds upon validated event representations, phase assignments, and causal relationships established in earlier stages of the framework. This allows complementary intelligence from different sources to be combined while preserving coherence, interpretability, and traceability.

The goal of synthesis is therefore not to replace individual analyses, but to integrate their strengths into a unified view of the campaign. The resulting synthesized attack life cycle provides broader coverage of attacker behavior, reveals relationships that are not visible within any single report, and supports more informed campaign-level reasoning and analysis.

doi:10.6342/NTU202600249

4.2 Attack Life Cycle Synthesis

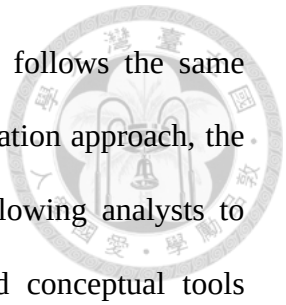
The synthesis process can be understood as a direct extension of attack life cycle construction, operating at a different granularity. While single-report construction organizes malicious activity events extracted from one CTI report into a coherent attack life cycle, synthesis applies the same reasoning logic to integrate multiple such life cycles into a unified, campaign-level representation.

At the core of both processes is the same organizing principle: reasoning over malicious activity events to infer temporal order and merge semantically overlapping actions in order to produce a coherent and interpretable attack life cycle. In single-report construction, this reasoning is applied to events derived from a single source. In synthesis, the same type of reasoning is applied across events originating from multiple constructed life cycles, each representing a partial view of the same campaign.

The primary distinction lies in the input representation. Attack life cycle construction operates on malicious activity events that have been identified, annotated with validated MITRE ATT&CK technique labels, and summarized concisely. Synthesis, in contrast, takes as input the outputs of attack life cycle construction—that is, fully constructed attack life cycles that already include ordered events, inferred causal relationships, and assigned attack phases. This shift in input allows synthesis to reason at the level of campaign structure rather than individual report narratives.

Despite this difference in scope, the reasoning applied during synthesis remains conceptually aligned with single-report construction. Events are reordered based on strategic intent and inter-event dependencies, and events describing the same attacker behavior across different sources are consolidated to improve fluency and reduce redundancy. The resulting synthesized life cycle preserves the structural coherence established during individual constructions while expanding coverage across reports.

doi:10.6342/NTU202600249



Finally, the visualization of the synthesized attack life cycle follows the same process used for individual life cycles. By reusing the same visualization approach, the synthesized representation remains consistent and interpretable, allowing analysts to reason about campaign-level behavior using the same visual and conceptual tools introduced earlier in the framework.

4.3 Demo Case: Deceptive Development Campaign

This case study synthesizes three publicly available Cyber Threat Intelligence (CTI) reports that document the same Deceptive Development campaign targeting software developers through deceptive recruitment workflows.

The primary input is the Deceptive Development report published by ESET Research [12]. This report provides the most comprehensive campaign-level view, consolidating earlier observations and explicitly unifying previously reported variants under a single operational narrative. It describes a near end-to-end attack workflow spanning social engineering, malware delivery, post-compromise activity, and persistent access, and serves as the structural backbone for synthesis. Owing to its breadth and coherence, this report is also used as the single-report demonstration case in earlier sections.

The second report, DEV#POPPER [13], published by Securonix Threat Research in July 2024, captures a narrower but technically detailed snapshot of the same campaign. Its analysis emphasizes execution-stage behaviors, including JavaScript-based loaders, npm-centric delivery mechanisms, and developer-focused infection chains. Although it does not attempt full campaign construction, its temporal position and technical depth surface execution and post-compromise activities that are less explicitly detailed in later, consolidated analyses.

The third report, Contagious Interview [14], produced by Unit 42, represents the

doi:10.6342/NTU202600249

earliest public documentation of the campaign. This report focuses primarily on malware behavior, command-and-control infrastructure, and detection-relevant observations, drawing on distinct telemetry and investigative priorities. While its overall coverage is limited compared to later reports, it contributes explicit descriptions of key behaviors—particularly related to command-and-control establishment—that complement and clarify steps that are only implicitly inferred elsewhere.

Despite differences in publication timing, naming conventions, and analytical focus, all three reports describe the same underlying threat operation. Each documents attackers posing as recruiters, abusing fake hiring processes, delivering trojanized software artifacts, and deploying a multi-stage malware chain that culminates in persistent remote access and data theft. At the same time, the reports are complementary rather than redundant: they emphasize different phases of the campaign, rely on different data sources, and expose distinct subsets of malicious activity. This combination of campaign equivalence and analytical complementarity makes the three reports well suited for synthesis, as their overlap provides grounding while their differences enable a more complete construction of the attack life cycle.

TTP Labeling Results Across Reports

In Table 4.1, the Deceptive Development and DEV#POPPER reports each provide their own sets of CTI-listed techniques, whereas the Contagious Interview report does not include an explicit list of associated techniques. To enable a consistent comparison of labeling results across reports, we aggregate the CTI-listed techniques from the first two reports and remove duplicate entries, yielding a total of 47 unique CTI-listed techniques. The observed increase in true positives for Deceptive Development and DEV#POPPER indicates that the proposed labeling process is capable of identifying additional MITRE ATT&CK techniques beyond those explicitly listed in the original CTI reports.

doi:10.6342/NTU202600249

Report	CTI-listed Techniques (Aggregate: 47)	Labeled Events	TTP Labels	TP	FN	FP	Comment
Deceptive Development	41	53	28	**15 (14)**	32 (27)	**13 (14)**	6 new TTPs
DEV#POPPER	10	13	8	**6 (4)**	41 (6)	**2 (4)**	37 new TTPs
Contagious Interview	47*	5	5	4	43	1	No change

Table 4.1: Individual TTP Labeling Results

To quantitatively examine how the three reports differ in their coverage of adversary behavior, we compare the validated MITRE ATT&CK techniques extracted from each constructed attack life cycle.

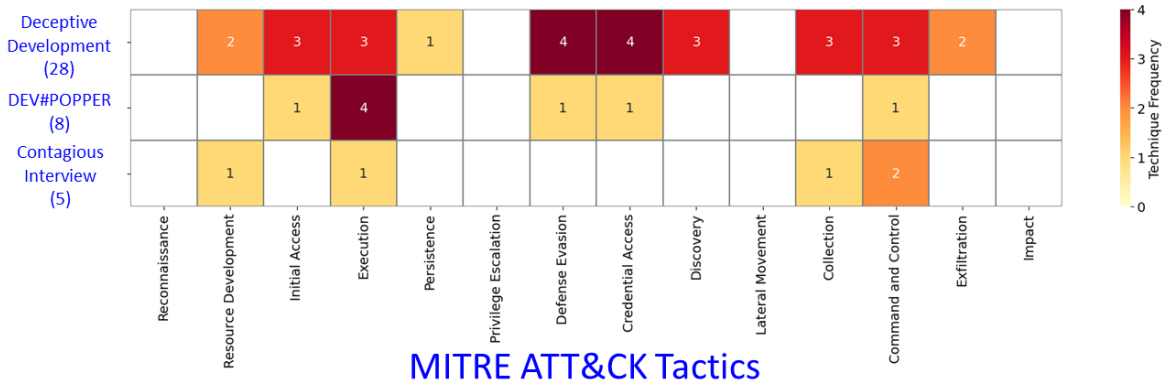


Figure 4.1: Distribution of the Labeled MITRE ATT&CK Techniques across 14 Tactics

Figure 4.1 presents a heatmap summarizing the distribution of validated techniques across MITRE ATT&CK tactics for each report. The Deceptive Development report exhibits the broadest coverage, spanning nearly all major tactic categories, including Resource Development, Initial Access, Execution, Persistence, Defense Evasion, Credential Access, Discovery, Collection, Command and Control, and Exfiltration. This wide distribution reflects ESET's campaign-level perspective and its emphasis on presenting a complete end-to-end view of the operation.

In contrast, the DEV#POPPER report shows a strong concentration in the Execution

doi:10.6342/NTU202600249

tactic, with comparatively limited representation in later-stage tactics. This aligns with its analytical focus on JavaScript loaders, npm-based infection chains, and the mechanics of initial code execution within developer workflows. While narrower in scope, this report provides depth in execution-related behaviors that are less emphasized elsewhere.

The Contagious Interview report contains the fewest validated techniques overall, but places relatively greater emphasis on Command and Control–related behaviors compared to DEV#POPPER. This reflects Unit 42's focus on malware communication patterns, remote command handling, and infrastructure-level observations, even when other campaign stages are described more sparsely.

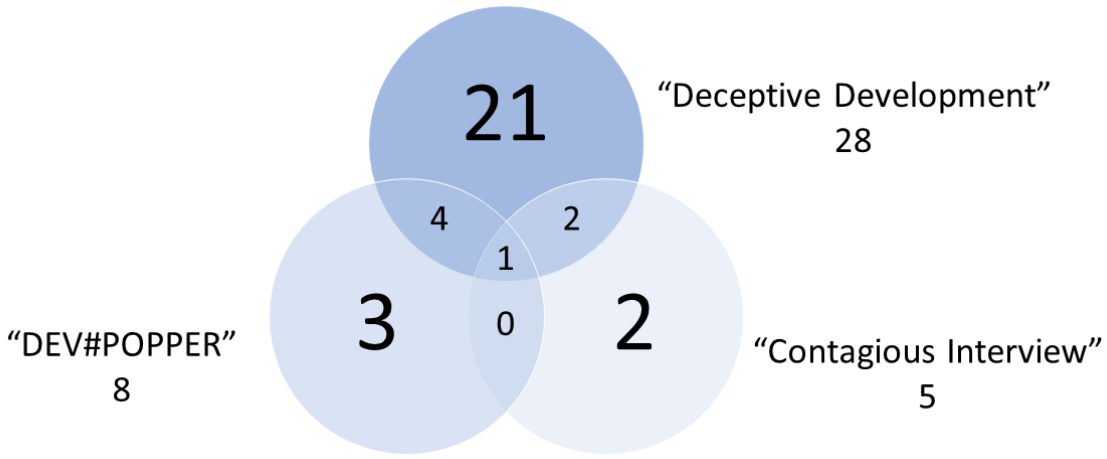


Figure 4.2: MITRE ATT&CK Technique Overlap and Uniqueness Across the Three Reports

Figure 4.2: MITRE ATT&CK Technique Overlap and Uniqueness Across the Three Reports complements this view by illustrating the overlap and uniqueness of validated techniques across the three reports using a Venn diagram. While Deceptive Development contains the largest set of techniques overall, the diagram clearly shows that both DEV#POPPER and Contagious Interview contribute techniques that do not appear in the Deceptive Development report alone. The intersection across all three reports is relatively small, indicating that complete overlap is limited even when the same campaign is

doi:10.6342/NTU202600249

analyzed.

Together, these results provide concrete evidence that the three reports offer partially overlapping but distinct views of the same threat activity. The heatmap highlights differences in tactical emphasis, while the Venn diagram demonstrates that each report contributes unique technique-level observations. This quantitative divergence directly motivates the synthesis process: combining multiple constructed attack life cycles enables a more comprehensive representation of the campaign than any single report can provide in isolation.

In the following sections, these individual attack life cycles are used as input to the synthesis process, illustrating how complementary observations from multiple reports can be integrated into a unified campaign-level life cycle.

Individual Attack Life Cycles

Figure 4.3: Overview of the Individual Attack Life Cycles Constructed from CTI Reports Prior to Synthesis presents the three individual attack life cycles constructed by the proposed system from the selected CTI reports prior to synthesis. Each life cycle was produced using the same event extraction, validated technique labeling, event ordering, and phase assignment pipeline. Differences observed

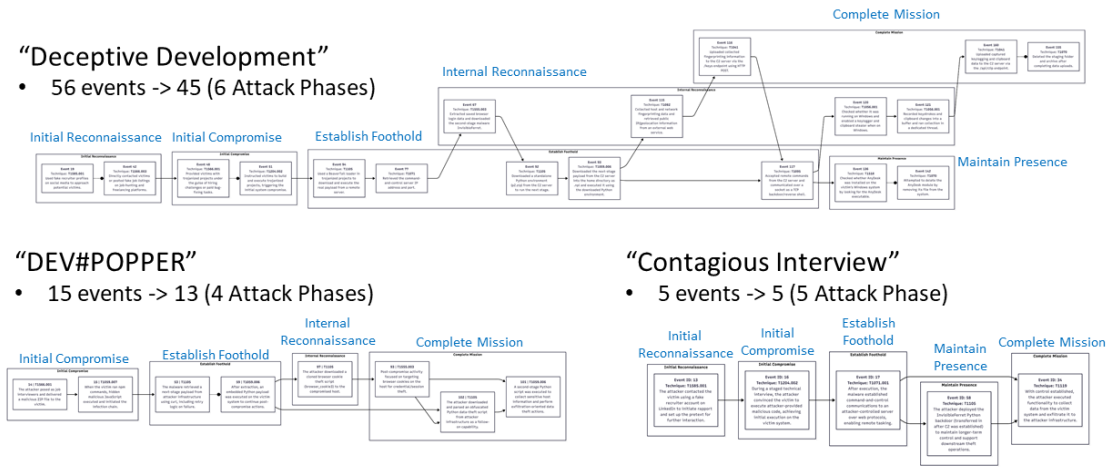


Figure 4.3: Overview of the Individual Attack Life Cycles Constructed from CTI Reports Prior to Synthesis

doi:10.6342/NTU202600249

across the three life cycles therefore reflect variations in reporting scope, analytical focus, and available telemetry in the source reports, rather than methodological inconsistencies.

The constructed attack life cycle for Deceptive Development is the most comprehensive, comprising 56 extracted malicious activity events that were consolidated into 45 events across six attack phases after event merging. The resulting life cycle spans the full campaign progression, from early reconnaissance and initial compromise through post-compromise activity and mission completion. Owing to its breadth and structural completeness, this life cycle serves as the primary reference point for subsequent synthesis.

In contrast, the DEV#POPPER life cycle is notably smaller, containing 15 extracted events that were merged into 13 events distributed across four attack phases. This life cycle focuses predominantly on execution and post-compromise behaviors, with limited visibility into earlier social-engineering or victim-targeting stages. While narrower in scope, it provides detailed coverage of specific technical behaviors that are less emphasized in the Deceptive Development life cycle.

The Contagious Interview life cycle is the most compact, consisting of five events that remain unmerged and span five distinct attack phases. Although minimal in size, its phase distribution indicates that the report captures selected but strategically important actions across the campaign, including initial access mechanisms and command-and-control–related activity. These events highlight behaviors that are either absent or only implicitly described in the other reports.

Viewed together, the diagram illustrates that no single constructed life cycle provides a complete representation of the campaign. Instead, the three inputs exhibit

doi:10.6342/NTU202600249

complementary strengths: Deceptive Development offers broad phase coverage and narrative continuity, DEV#POPPER contributes depth in execution-related behavior, and Contagious Interview surfaces isolated but critical actions across multiple phases. This heterogeneity in event coverage and phase distribution motivates the need for synthesis, which aims to integrate these partial perspectives into a unified and more comprehensive attack life cycle.

Synthesized Attack Life Cycle

Figure 4.4: Overview of the Synthesized Attack Life Cycle Constructed from Multiple CTI Reports presents the synthesized attack life cycle for the Deceptive Development campaign, constructed by combining the individual attack life cycles derived from three CTI reports. The synthesized life cycle contains 50 malicious activity events organized into six attack phases, preserving the same phase vocabulary used throughout earlier chapters: Initial Reconnaissance, Initial Compromise, Establish Foothold, Internal Reconnaissance, Maintain Presence, and Complete Mission.

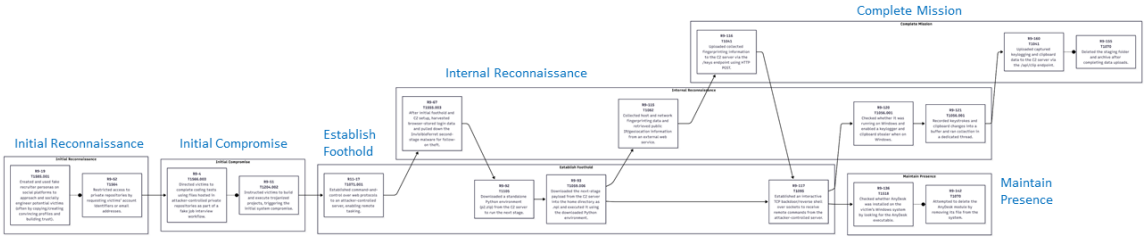


Figure 4.4: Overview of the Synthesized Attack Life Cycle Constructed from Multiple CTI Reports

Compared to the single-report life cycle constructed from the Deceptive Development report alone, the synthesized result expands coverage across multiple phases while maintaining a coherent end-to-end progression. Events from different reports are interleaved based on their inferred temporal and causal relationships, resulting in a unified campaign-level view that reflects both shared and complementary

doi:10.6342/NTU202600249

observations. Importantly, the synthesized life cycle retains a single connected causal graph, rather than a set of parallel or disjoint sequences, enabling direct interpretation of how early-stage actions enable later objectives across the campaign.

Event Contributions to the Synthesized Attack Life Cycle by Report and Phase

Table 4.2: Event Contributions to the Synthesized Attack Life Cycle from Three CTI Reports summarizes the number of events contributed by each report to the synthesized attack life cycle, broken down by attack phase. As expected, the Deceptive Development report contributes the majority of events and serves as the structural backbone of the synthesized life cycle, accounting for 41 of the 50 total events. It provides complete coverage of all six phases and establishes the primary causal structure of the campaign.

Report	Initial Recon	Initial Compromise	Establish Foothold	Internal Recon	Maintain Presence	Complete Mission	Total
After Synthesis	7	5	9	11	6	12	50
Deceptive Development	7	5	5	9	6	9	41
DEV#POPPER	0	0	3	2	0	3	8
Contagious Interview	0	0	1	0	0	0	1

Table 4.2: Event Contributions to the Synthesized Attack Life Cycle from Three CTI Reports

The DEV#POPPER report contributes 8 additional events, primarily enriching the Establish Foothold, Internal Reconnaissance, and Complete Mission phases. These events add detail to execution-stage behaviors and post-compromise activities that are either absent or less explicit in the Deceptive Development report. Although smaller in volume, these contributions meaningfully expand the depth of the middle and late stages of the campaign.

The Contagious Interview report contributes 1 event, which is incorporated into the Establish Foothold phase. While limited in scope, this contribution reflects a distinct observation of foothold-establishment behavior that complements the dominant report

doi:10.6342/NTU202600249

and reinforces the consistency of the campaign's post-compromise tooling.

When compared against the attack life cycle constructed from the Deceptive Development report alone, synthesis increases the total number of events from 41 to 50, with the most notable gains appearing in the Establish Foothold, Internal Reconnaissance, and Complete Mission phases. These are precisely the phases where individual reports tend to vary most in focus due to differences in telemetry and analytical emphasis.

Example of a Synthesis-Added Event Making Implicit Causality Explicit

A concrete example of how synthesis improves campaign-level completeness can be observed in the representation of command-and-control (C2) establishment during the Establish Foothold phase.

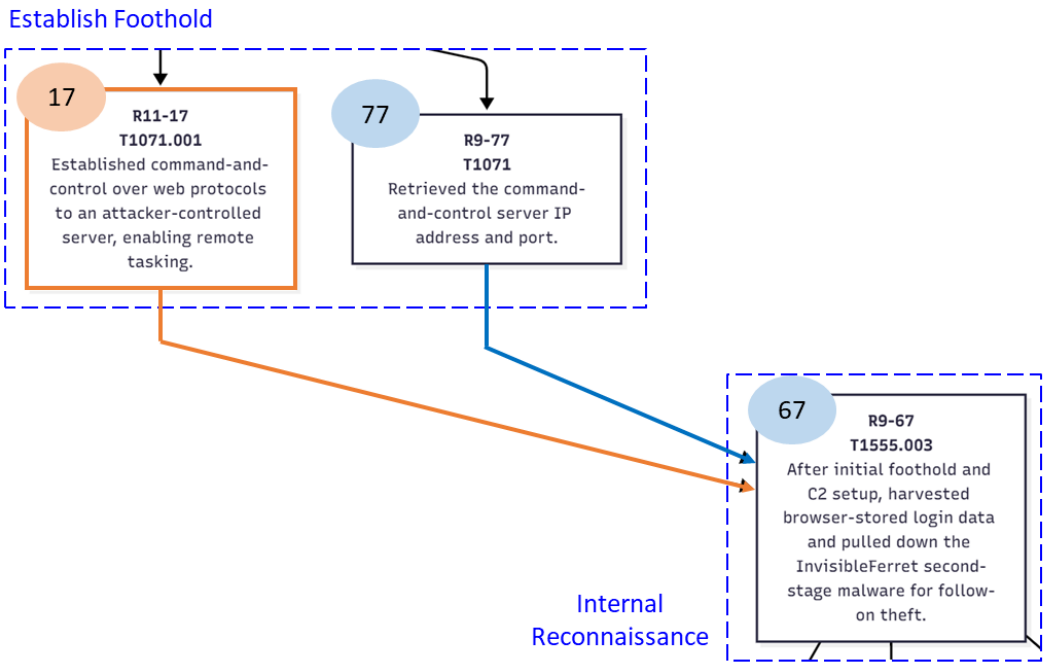


Figure 4.5: Example of a Synthesis-Added Event That Makes Command-and-Control Establishment Explicit

In the attack life cycle constructed from the Deceptive Development report alone, Event 67 describes the harvesting of browser-stored credentials and the download of the second-stage malware InvisibleFerret following initial foothold and C2 setup. While this

doi:10.6342/NTU202600249

event clearly depends on the existence of an operational C2 channel, the prerequisite C2 establishment is only implicitly inferred. The immediately preceding event in the same life cycle, Event 77, retrieves the C2 server IP address and port, which suggests that C2 infrastructure is being prepared, but does not explicitly state that a functional command-and-control channel has been established.

After synthesis, an additional event from the Contagious Interview report, Event 17, is incorporated into the synthesized life cycle. This event explicitly states the establishment of command-and-control over web protocols to an attacker-controlled server, directly enabling remote tasking. By introducing this event, synthesis makes the C2 setup step explicit rather than implied, clarifying the causal dependency between C2 establishment and subsequent credential harvesting and second-stage payload retrieval.

The resulting synthesized representation therefore replaces an inferred assumption with a concrete, explicitly documented action. This not only strengthens the causal linkage between the foothold and internal reconnaissance phases, but also improves interpretability by making the attack progression clearer to analysts. This example illustrates how synthesis can enhance campaign representations by integrating complementary observations from different reports, even when the overall attack flow remains consistent.

doi:10.6342/NTU202600249

Chapter 5. Evaluation

5.1 Cross-Report Structural Evaluation of the Construction Pipeline

This section evaluates the structural behavior of the proposed framework across eleven heterogeneous Cyber Threat Intelligence (CTI) reports. Rather than assessing correctness against unavailable ground truth, the evaluation focuses on how successive pipeline stages—annotation filtering, technique validation, and attack life cycle construction—interact and transform CTI narratives into coherent campaign representations under realistic reporting conditions.

ID	Report Name	Sentences	Annotation Filter	Labeled Events			Attack Life Cycle		TTP
				Baseline	Open-world	Validation	Events	Phases	
1	Lazarus	96	39	39	39	31*	21	6	17
2	CloudScout	106	12	7*	12	8*	12	6	7
3	Cuba Ransomware	105	16	15*	16	10*	14	5	8
4	GoldenJackal	259	24	21*	24	22*	21	5	13
5	Pakistan Navy	94	19	19	19	18*	17	6	13
6	PlushDaemon	69	27	21*	27	23*	27	4	15
7	Sandworm	82	29	24*	29	25*	23	7	15
8	SHROUDEDSLEEP	166	54	33*	54	43*	17	4	14
9	Deceptive Development	179	56	36*	56	53*	45	6	28
10	DEV#POPPER	127	15	15	15	13*	13	4	8
11	Contagious Interview	69	5	5	5	5	5	5	5

Table 5.1: Evaluation across 11 Reports

Narrative Selectivity of Incident-Scoped Annotation Filtering

Across all evaluated reports, the number of sentences contained in the original CTI documents substantially exceeds the number of malicious activity events identified by the annotation filter. In several cases, fewer than one third of sentences are retained for downstream analysis. This observation reflects the narrative nature of CTI reporting, in which descriptions of attacker behavior are interleaved with background context,

doi:10.6342/NTU202600249

historical references, analyst interpretation, and defensive guidance.

The annotation filter consistently isolates sentences that describe concrete attacker actions attributable to the focal campaign, while excluding non-operational content. This selectivity demonstrates that the majority of CTI text does not directly describe actionable malicious behavior at the event level, and that explicit incident scoping is necessary to reduce narrative noise prior to technique labeling or temporal reasoning. The consistency of this pattern across reports of varying length and publisher origin further indicates that the filtering behavior is driven by structural properties of CTI narratives rather than report-specific characteristics.

Prevalence and Role of Unlabeled Events After Validation

Comparison between the annotation filter output and the validated technique labeling results shows that a non-trivial fraction of extracted events remains unlabeled after validation. This outcome is observed consistently across reports, including both large and small campaigns. Validation frequently reduces the number of technique-labeled events relative to both baseline and open-world labeling outputs.

Inspection of these unlabeled events reveals that many describe low-level implementation details, execution flow control, or internal program logic that are operationally meaningful but do not correspond directly to any MITRE ATT&CK technique. This result highlights an inherent abstraction gap between narrative CTI descriptions and technique-level taxonomies. Rather than indicating a failure of the labeling process, the presence of unlabeled events demonstrates the effectiveness of conservative, evidence-based validation in preventing forced or speculative mappings. Importantly, unlabeled events are retained for subsequent life cycle construction, ensuring that operational context is preserved even when technique abstraction is not applicable.

Structural Robustness of Attack Life Cycle Construction

doi:10.6342/NTU202600249

Attack life cycle construction consistently produces compact and stable campaign representations despite substantial variation in report length, narrative detail, and extracted event volume. During construction, multiple extracted events are frequently merged when they describe the same operational behavior at different levels of granularity or from different narrative perspectives. This consolidation reduces redundancy while preserving the diversity of adversary behaviors represented in the attack life cycle. As a result, verbose reports with many extracted events do not yield proportionally more complex life cycles.

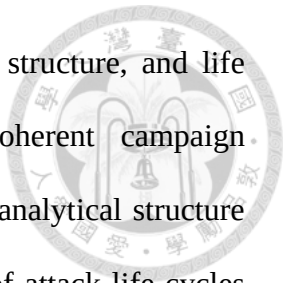
At the same time, the constructed life cycles exhibit stable phase coverage across all evaluated reports. Campaigns described using hundreds of sentences and those documented through sparse narratives alike result in life cycles spanning a comparable number of attack phases. This indicates that phase-level structure is governed by underlying adversary workflows rather than by reporting verbosity or technique labeling density. Even when only a small number of events or validated techniques are available, the framework is able to infer coherent phase progression and maintain interpretable campaign structure.

Together, these observations show that the proposed construction process normalizes both redundancy and sparsity in CTI reporting. By consolidating repeated descriptions and abstracting from low-level narrative variation, the framework produces attack life cycles that are robust to heterogeneity in CTI sources while remaining faithful to the operational behaviors described in the original reports.

Summary of Structural Evaluation Findings

Taken together, these observations demonstrate that the proposed framework behaves predictably and conservatively across diverse CTI sources. Annotation filtering effectively isolates incident-relevant actions from narrative noise, validation enforces

doi:10.6342/NTU202600249



evidence-grounded technique assignment without collapsing event structure, and life cycle construction consolidates redundant descriptions into coherent campaign representations. These properties enable the framework to preserve analytical structure while avoiding overinterpretation, supporting reliable construction of attack life cycles under realistic and heterogeneous CTI reporting conditions.

5.2 Comparison with Expert-Drawn Campaign Diagram

This qualitative evaluation compares the automatically constructed attack life cycle produced by the proposed framework with an expert-drawn campaign diagram published in a CTI report [15].

Demo Case Overview

The Medusa ransomware group conducted a coordinated ransomware campaign by exploiting multiple unpatched vulnerabilities in the SimpleHelp remote monitoring and management (RMM) software (CVE-2024-57726, CVE-2024-57727, CVE-2024-57728). Initial access was achieved through compromised managed service provider (MSP) and third-party supplier infrastructure, allowing attackers to abuse SimpleHelp instances running with SYSTEM-level privileges. Using this access, Medusa operators pivoted into downstream customer environments, conducted reconnaissance, disabled Microsoft Defender, and established high-privilege persistence through SimpleHelp itself and, in some cases, additional remote access tooling such as AnyDesk.

The campaign followed a double-extortion model. In approximately half of the observed incidents, attackers exfiltrated data using renamed and filtered RClone binaries before encrypting victim systems with the ".MEDUSA" extension and deploying "!!!READ_ME_MEDUSA!!!.txt" ransom notes. Ransomware deployment was performed using trusted administrative tooling, including PDQ Deploy and PDQ Inventory, or via direct execution through SimpleHelp, enabling rapid and widespread

doi:10.6342/NTU202600249

encryption across victim environments. Victims were subsequently pressured through Medusa's data leak site and associated communication channels.

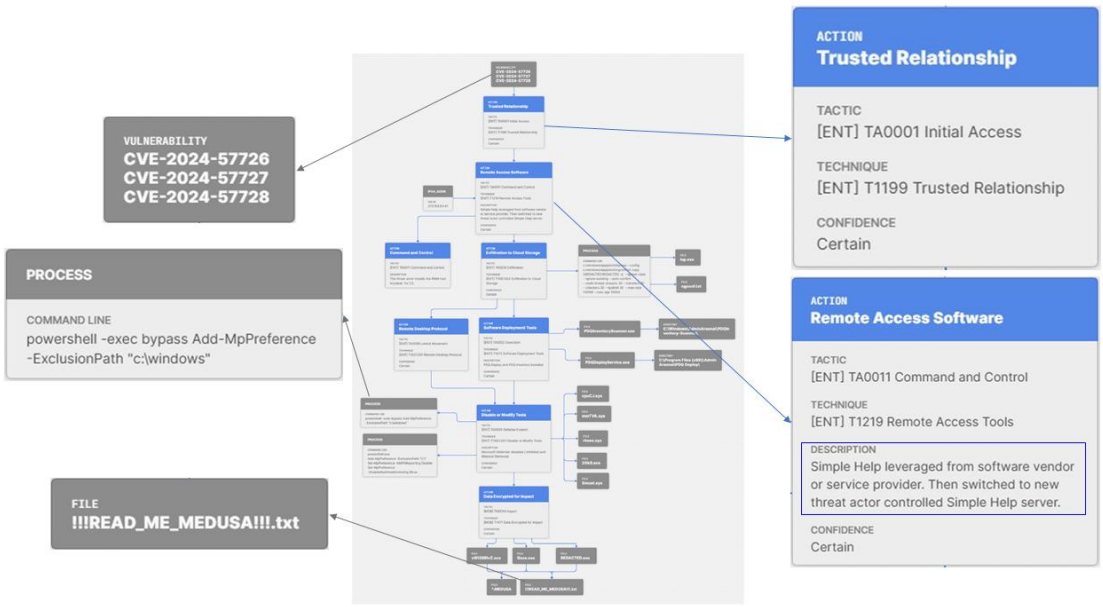


Figure 5.1: Expert-Drawn Campaign Diagram from the CTI Report

The expert-drawn diagram in Figure 5.1: Expert-Drawn Campaign Diagram from the CTI Report provided in the CTI report presents a high-level visual summary of the Medusa ransomware campaign. The diagram focuses on simple relationships between MITRE ATT&CK techniques and enumerates a large number of indicators of compromise (IOCs), such as process command lines and file names. While some techniques are accompanied by brief descriptions, others are listed without explicit contextual explanation. The diagram is primarily artifact- and technique-centric, leaving temporal ordering, causal dependencies, and attacker intent largely implicit.

To illustrate how the proposed framework provides a richer and more structured representation, the following comparisons examine specific portions of the campaign as represented in both the expert diagram and the constructed attack life cycle.

From CVE Enumeration to Intent-Aware Initial Compromise Events

In the expert-drawn diagram, initial access is represented by listing multiple CVE

doi:10.6342/NTU202600249

identifiers associated with vulnerabilities in SimpleHelp. These CVEs are visually connected to a corresponding ATT&CK technique but are not further contextualized in terms of attacker intent or operational outcome. As a result, the reader must infer how these vulnerabilities were leveraged and why they were strategically important to the campaign.

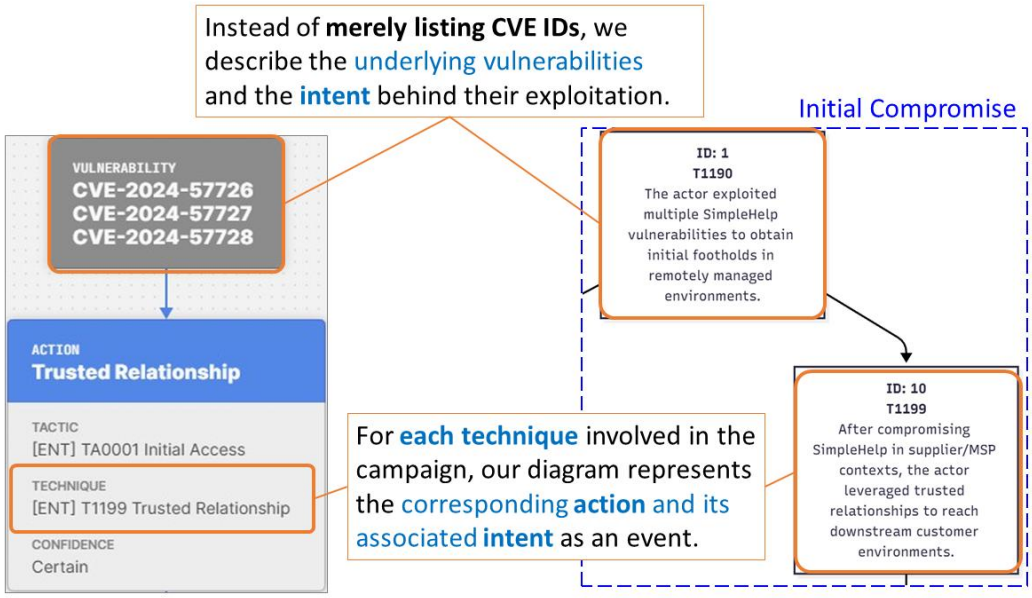


Figure 5.2: From CVE Enumeration to Intent-Aware Initial Compromise Events

In contrast, the constructed attack life cycle explicitly models initial compromise as intent-bearing events. Rather than merely listing CVE identifiers, the system represents the exploitation of SimpleHelp vulnerabilities as attacker actions aimed at establishing initial footholds in remotely managed environments. These events are labeled with the appropriate technique and embedded within the Initial Compromise phase, making the attacker's objective and the operational role of the vulnerabilities explicit. This transformation from vulnerability enumeration to action-and-intent representation enables clearer reasoning about how exploitation directly enables subsequent campaign stages.

Explicit Modeling of Defensive Evasion Actions

doi:10.6342/NTU202600249

The expert diagram enumerates multiple process-level IOCs related to the modification or disabling of Microsoft Defender, including PowerShell commands and associated tooling. These observables are valuable for detection purposes, but they are presented as isolated artifacts without a clear structural relationship between them. The diagram does not distinguish whether these commands represent separate attacker actions, alternative methods, or repeated executions of the same behavior.

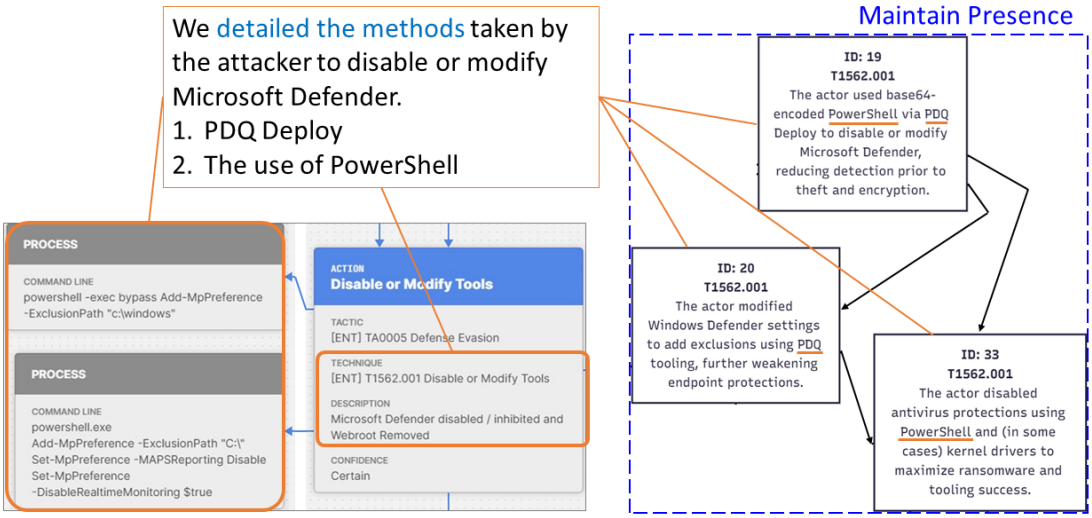


Figure 5.3: Explicit Modeling of Defensive Evasion Actions

By contrast, the constructed life cycle consolidates these low-level observables into a set of explicitly defined attacker actions within the Maintain Presence phase. Separate events capture distinct methods used by the attacker—such as leveraging PDQ Deploy or executing PowerShell commands—to disable or weaken endpoint protection. Each event is associated with the same defensive evasion technique but retains its own operational context and intent. This explicit event modeling clarifies how different implementation methods contribute to a common campaign objective, while preserving causal and temporal relationships between actions.

From File Lists to Role-Aware Impact Representation

doi:10.6342/NTU202600249

In the expert-drawn diagram, the impact phase is illustrated by listing file names associated with ransomware execution, encrypted file extensions, and ransom note artifacts. These files are presented as indicators observed during the campaign, but their functional roles within the attack are not explicitly described.

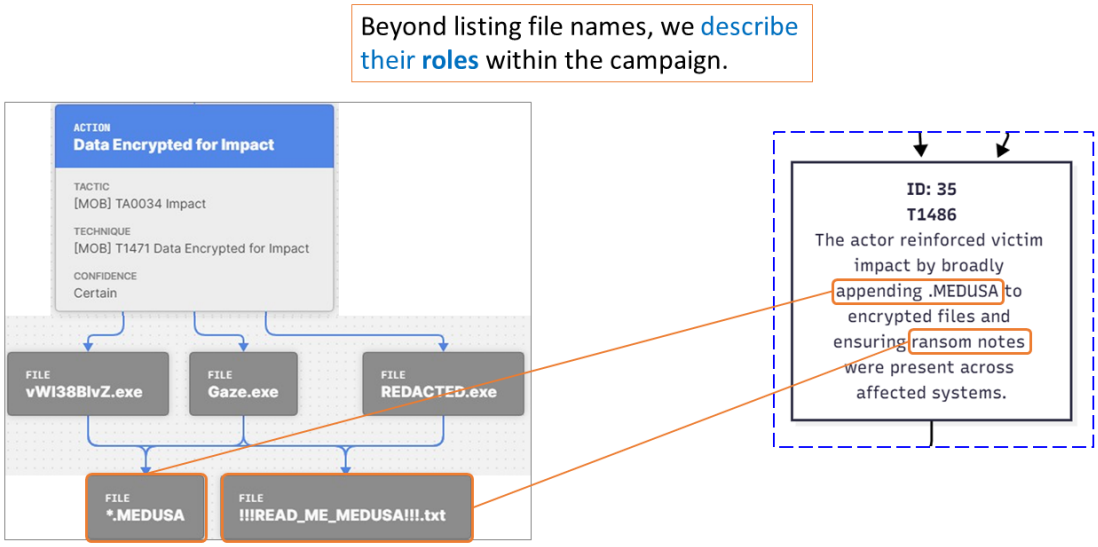


Figure 5.4: Role-Aware Representation of Impact-Related Artifacts

The constructed attack life cycle instead represents these artifacts as components of a higher-level impact event. Rather than simply listing file names, the system describes how the attacker reinforced victim impact by appending the ".MEDUSA" extension to encrypted files and ensuring that ransom notes were deployed consistently across affected systems. By embedding artifacts within intent-driven events, the representation makes clear why these files matter and how they contribute to the campaign's impact objectives, rather than treating them as standalone observables.

doi:10.6342/NTU202600249

Structural Implications of the Comparison

Criteria	System output	Expert Diagram from CTI report
Technique related descriptions	✅	⚠️
Detailed events (action taken by the attacker)	✅	❌
Main artifacts (tools, files)	✅	✅ (more ancillary IOCs)
Causality of events (activities)	✅	❌

Table 5.2: Qualitative Comparison Between System Output and Expert-Drawn Diagram

Taken together, the diagram-based comparisons and the summary in Table 5.2 demonstrate that the proposed framework provides a more structured and behavior-centric representation of attack campaigns than expert-drawn diagrams when the analytical objective is attack life cycle construction. As shown in the table, the system output consistently provides explicit technique-related descriptions, detailed representations of attacker actions, and clear modeling of causal relationships between events—capabilities that are only partially or implicitly supported in the expert diagram. While both representations capture key artifacts such as tools and files, the system emphasizes their operational roles within the campaign rather than enumerating them as standalone indicators. These differences reflect a fundamental shift in representational focus: from technique- and IOC-centric summaries toward explicit modeling of attacker intent, action, and progression. Consequently, the proposed approach offers clear advantages for reasoning about how campaigns unfold over time, how actions enable subsequent phases, and how disparate observations can be integrated into a coherent attack life cycle, complementing but extending beyond the scope of expert-drawn CTI diagrams.

doi:10.6342/NTU202600249

Chapter 6. Conclusion

This work addresses the challenge of constructing coherent attack campaign life cycles from fragmented and heterogeneous CTI reporting. We proposed an LLM-driven framework that decouples malicious activity identification from technique alignment, enabling incident-scoped extraction of concrete adversarial actions grounded in contextual reasoning. By transforming narrative CTI descriptions into structured event representations and aligning them with MITRE ATT&CK techniques through multi-stage labeling and validation, the system enables reliable construction of attack workflows.

Building upon these labeled events, our framework infers temporal ordering and causal dependencies to construct coherent multi-phase campaign life cycles. Furthermore, we demonstrate that life cycles derived from separate CTI reports can be synthesized into unified campaign representations, addressing the inherent fragmentation of intelligence sources. The resulting outputs provide analysts with structured and interpretable views of adversary operations, facilitating campaign understanding and supporting downstream threat analysis.

While current evaluation demonstrates life cycle coverage and consistency using representative CTI reports, further work is needed to scale quantitative validation through larger annotated datasets and analyst-centered evaluation. Additional future directions include improving temporal and causal reasoning under incomplete evidence, enhancing robustness against inconsistent or low-quality reporting, and integrating campaign construction outputs with detection and response systems to support operational defense workflows. Overall, this work demonstrates the potential of LLM-driven reasoning to transform fragmented CTI narratives into coherent and operationally useful representations of adversarial campaigns.

doi:10.6342/NTU202600249

Reference

[1] "MITRE ATT&CK®," MITRE Corporation, 2025. [Online]. Available: https://attack.mitre.org/. [Accessed Oct. 2025].

[2] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D., "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824-24837, 2022.

[3] Chen, Y., Cui, M., Wang, D., Cao, Y., Yang, P., Jiang, B., ... & Liu, B, "A survey of large language models for cyber threat detection," *Computers & Security*, vol. 145, p. 104016, 2024.

[4] Sainz, O., García-Ferrero, I., Agerri, R., de Lacalle, O. L., Rigau, G., & Agirre, E., "Gollie: Annotation guidelines improve zero-shot information-extraction," in *arXiv preprint arXiv:2310.03668.*, 2023.

[5] Xu, D., Chen, W., Peng, W., Zhang, C., Xu, T., Zhao, X., ... & Chen, E., "Large language models for generative information extraction: A survey," in *Frontiers of Computer Science, 18(6), 186357.*, 2024.

[6] Büchel, M., Paladini, T., Longari, S., Carminati, M., Zanero, S., Binyamini, H., ... & van Ede, T., "{SoK}: Automated {TTP} Extraction from {CTI} Reports–Are We There Yet?," *34th USENIX security symposium (USENIX Security 25)*, pp. 4621-4641, 2025.

[7] Krašovec, A., Steri, G., Karopoulos, G., & Trapani, M., "Large Language Models for Cyber Threat Intelligence: Extracting MITRE With LLMs," in *International Conference on Availability, Reliability and Security*, Cham: Springer Nature Switzerland, 2025.

[8] Satyapanich, T., Ferraro, F., & Finin, T., "Casie: Extracting cybersecurity event information from text," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 8749-8757.

[9] "ChatGPT 5.2," OpenAI, [Online]. Available: https://chat.openai.com/. [Accessed Jan. 2026].

[10] "Targeted attack life cycle," Mandiant, [Online]. Available: https://cloud.google.com/security/resources/insights/targeted-attack-lifecycle?hl=en.

[11] "Mermaid: Diagramming and charting tool," Mermaid contributors, [Online]. Available: https://mermaid.js.org/. [Accessed Jan. 2026].

doi:10.6342/NTU202600249

[12] M. Havránek, "DeceptiveDevelopment targets freelance developers," 20 Feb. 2025. [Online]. Available: https://www.welivesecurity.com/en/eset-research/deceptivedevelopment-targets-freelance-developers/.

[13] Den Iuzvyk, Tim Peck, "Research Update: Threat Actors Behind the DEV#POPPER Campaign Have Retooled and are Continuing to Target Software Developers via Social Engineering," 31 Jul. 2024. [Online]. Available: https://www.securonix.com/blog/research-update-threat-actors-behind-the-devpopper-campaign-have-retooled-and-are-continuing-to-target-software-developers-via-social-engineering/.

[14] Unit 42, "Contagious Interview: DPRK Threat Actors Lure Tech Industry Job Seekers to Install New Variants of BeaverTail and InvisibleFerret Malware," 9 Oct. 2024. [Online]. Available: https://unit42.paloaltonetworks.com/north-korean-threat-actors-lure-tech-job-seekers-as-fake-recruiters/.

[15] F. Rondeau, "How RMM abuse fuelled Medusa & DragonForce attacks," Zensec, 30 Oct. 2025. [Online]. Available: https://zensec.co.uk/blog/how-rmm-abuse-fuelled-medusa-dragonforce-attacks/.

doi:10.6342/NTU202600249