Modeling Behavioral Preferences of Cyber Adversaries Using Inverse Reinforcement Learning

Aditya Shinde[1] , **Prashant Doshi**[1]
[1]THINC Lab, School of Computing, University of Georgia
{adityas, pdoshi}@uga.edu

Abstract

This paper presents a holistic approach to attacker preference modeling from system-level audit logs using inverse reinforcement learning (IRL). Adversary modeling is an important capability in cybersecurity that lets defenders characterize behaviors of potential attackers, which enables attribution to known cyber adversary groups. Existing approaches rely on documenting an ever-evolving set of attacker tools and techniques to track known threat actors. Although attacks evolve constantly, attacker behavioral preferences are intrinsic and less volatile. Our approach learns the behavioral preferences of cyber adversaries from forensics data on their tools and techniques. We model the attacker as an expert decision-making agent with unknown behavioral preferences situated in a computer host. We leverage attack provenance graphs of audit logs to derive a state-action trajectory of the attack. We test our approach on open datasets of audit logs containing real attack data. Our results demonstrate for the first time that low-level forensics data can automatically reveal an adversary's subjective preferences, which serves as an additional dimension to modeling and documenting cyber adversaries. Attackers' preferences tend to be invariant despite their different tools and indicate predispositions that are inherent to the attacker. As such, these inferred preferences can potentially serve as unique behavioral signatures of attackers and improve threat attribution.

1 Introduction

Sophisticated cyber attackers are increasingly targeting large organizations and critical infrastructures. These threat actors, also known as advanced persistent threats (APT), are stealthy, resourceful, and often employ novel exploitation techniques to achieve their objectives. Documenting, analyzing, and modeling such threat actors is critical for improving defenses against them. Recently, provenance graphs of audit log data has emerged as a popular computational representation for analyzing attacks by APTs [King and Chen, 2003; Hossain *et al.*, 2017]. Provenance graphs are a causal representation of interactions between kernel-level objects such as processes, threads, and files, which facilitates analysis by connecting together objects that interact. Recent efforts have adopted AI-based techniques to automate detecting APTs [Wang *et al.*, 2022; Milajerdi *et al.*, 2019b].

Tactical information about cyber threats is critical for their detection and timely response. However, the advantage of having such specific intelligence is fleeting as attacker tools and techniques evolve constantly. Approaches to modeling adversaries lack broader insights into an attacker's behavioral characteristics and preferences. At a strategic level, past work has adopted game-theoretic [Ferguson-Walter *et al.*, 2019; Schlenker *et al.*, 2018] and decision-theoretic frameworks [Sarraute *et al.*, 2012; Shinde and Doshi, 2024] to model cyber adversaries. However, most goal and intent recognition approaches for cybersecurity focus exclusively on end-goal recognition. These efforts do not target the broader preferences that the attacker's behavior implicitly reveals. They also rely on assumptions, such as the attacker's intent is restricted to a set of previously-known candidate reward functions [Mirsky *et al.*, 2019; Shinde *et al.*, 2021].

This paper presents a novel, end-to-end approach to modeling adversary preferences from raw forensics data using inverse reinforcement learning (IRL) [Arora and Doshi, 2021]. We use low-level audit logs as they are often the only attack-relevant data source in a post-breach scenario and are commonly used in cybersecurity. Our methodology leverages a provenance graph representation of the system-level audit logs. We model the attacker in a host as an expert decision-making agent in the context of a Markov decision process (MDP). Then, we utilize subgraph isomorphism to map parts of the provenance graph to attacker actions grounded in the popular MITRE ATT&CK matrix. The ATT&CK matrix is a comprehensive catalogue of techniques and tactics used by various attackers [Strom *et al.*, 2018]. In doing so, we bridge the gap between raw security log data and the symbolic action representations that are prevalent in the decision-making models applied to cybersecurity. These mappings enable us to generate trajectories of observed attacker behavior. Subsequently, we infer an attacker's behavioral preferences from these audit logs using IRL. This is a novel application of IRL to adversary modeling, a pertinent goal in modern cyber defense. We model an adversary's preferences through behavioral features such as discoverability, duration, attributability, sophistication, and impact. *These informative features encompass an attacker's*

broader preferences at a level above their tools and techniques and serve as a unique signature of an adversary.

Subsequently, we utilize IRL to compute the weighting of previously mentioned preference features from the trajectories. We test this pipeline on multiple open datasets [Keromytis, 2018], which contain real cyber attacks on different target hosts. *Ground truth on an adversary's preferences is usually not available! We work around this challenge by using two differing IRL techniques and analyzing inter-method agreement.* Our results demonstrate the benefit of this novel approach in extracting broader and likely invariant insights about attacker behavior from low-level log data. This automated approach to recognizing attacker preferences using IRL enables the study of behavioral aspects of cyber attackers without assumptions about their goals.

2 Background

Our approach leverages attack provenance graphs to extract trajectories of attacker behavior for inferring their preferences using IRL. In this section, we provide a brief background on provenance graphs and IRL.

2.1 Attack Provenance Graphs

APTs utilize sophisticated tools and techniques that are difficult to detect with traditional detection mechanisms [Karantzas and Patsakis, 2021]. Recently, the cybersecurity community has successfully applied data provenance to system-level log data for detecting such threats [King and Chen, 2003; Hossain *et al.*, 2017; Milajerdi *et al.*, 2019b; Setayeshfar *et al.*, 2019]. Provenance graphs capture the interactions between system-level subjects such as processes and threads, and objects such as files, sockets, and pipes. The graphical representation aids attack investigations by restricting the search space to events causally connected to an indicator of compromise.

Formally, a provenance graph G is defined as $G = \langle N, E \rangle$ where N is the set of nodes representing abovementioned subjects and objects. $E = \{e_1, e_2, ..., e_t\}$ is the set of *time-stamped edges* representing interactions and the direction of information flow between the nodes. Provenance graphs facilitate impact analysis by forward-tracing the paths causally linked to a suspicious node. Starting from a node n at time t', the set of nodes impacted by n is given by traversing all edges $e_{t \geq t'} \in E$. Similarly, backward search enables root-cause analysis that determines the source of an attack [King and Chen, 2003; Lee *et al.*, 2013a]. The set of nodes that can likely influence a node n at time t' is similarly generated by traversing all edges $e_{t \leq t'} \in E$. In practice, audit logging systems can produce gigabytes of log data per day [Xu *et al.*, 2016]. Several compression techniques have been proposed to address this challenge [Xu *et al.*, 2016; Lee *et al.*, 2013b; Tang *et al.*, 2018]. Figure 1 illustrates this pipeline of generating a provenance graph from audit logs.

In our work, we generate an attack scenario graph starting from the source of the intrusion using forward analysis and a well-known state-based node versioning technique [Hossain *et al.*, 2018]. Subsequently, we map parts of the attack scenario graph to the MITRE ATT&CK matrix using subgraph isomorphism to generate trajectories of attacker behavior.

2.2 Inverse RL

Inverse reinforcement learning (IRL) enables an observer to infer an expert decision-making agent's reward function from observed behavior and, optionally, the agent's policy [Ng and Russell, 2000; Arora and Doshi, 2021]. IRL methods commonly formulate the decision making of the expert agent as a Markov decision process (MDP). The expert's MDP is defined formally as a tuple $\langle S, A, T, \gamma, R \rangle$, where S is the set of states, A the set of actions of the expert, $T : S \times A \times S \rightarrow [0, 1]$ is the transition function, $R : S \times A \rightarrow \mathbb{R}$ is the expert's unknown reward function, and $\gamma \in (0, 1)$ is the discount factor. The expert agent's behavior data is available as a set of M state-action trajectories of length $\mathcal{T}$, $\mathcal{X} = \{(s_1, a_1), (s_2, a_2), ..., (s_\mathcal{T}, a_\mathcal{T})\}$ and $|\mathcal{X}| = M$. For large state spaces, the expert's reward function is approximated linearly as $R(s, a) = w_1\phi_1(s, a) + \cdots + w_k\phi_k(s, a)$, where $s \in S$, $a \in A$, and $\phi_1, \ldots, \phi_k$ are bounded basis functions $\phi : S \times A \rightarrow \{0, 1\}$ [Abbeel and Ng, 2004]. *Basis weights $w_1, \ldots, w_k$ are unknown parameters to be learned.*

Bayesian IRL [Ramachandran and Amir, 2007] is a well-known framework that assumes a prior distribution over the reward functions with i.i.d reward values, $P(R) = \prod_{s \in S, a \in A} P(R(s, a))$, and computes the posterior distribution, $P(R|\mathcal{X})$, using the expert's trajectories as follows,

$$P(R|\mathcal{X}) = \alpha P(\mathcal{X}|R)P(R). \tag{1}$$

Here, α is the normalization constant, and $P(\mathcal{X}|R)$ is the likelihood function expressed as,

$$P(\mathcal{X}|R) = \prod_{m=1}^{M} \prod_{t=1}^{\mathcal{T}} \frac{e^{\beta Q^*(s_t^m, a_t^m; R)}}{\sum_a e^{\beta Q^*(s_t^m, a; R)}}.$$

As computing the partition function contained in α is hard, several approaches exist to compute a point estimate of R distributed according to $P(R|\mathcal{X})$ [Arora and Doshi, 2021]. In our work, we utilize a *maximum-a-posteriori* (MAP) estimation for the reward function within the Bayesian IRL framework [Choi and Kim, 2011]. The MAP-BIRL approach computes a reward, R_{MAP}, that maximizes the log of the posterior in Eq. 1 using a gradient method:

$$\begin{aligned} R_{MAP} &= \arg\max_R \log P(R|\mathcal{X}) \\ &= \arg\max_R \log P(\mathcal{X}|R) + \log P(R) \end{aligned}$$

Value functions $V^*(R)$ and $Q^*(R)$ are convex and differentiable almost everywhere [Choi and Kim, 2011]. These properties enable efficient computation of R_{MAP} using a gradient method with the update rule, $R_{new} \leftarrow R + \delta_t \nabla_R P(R|\mathcal{X})$.

3 Host-Level Cyber Threat Domain

Section 2 reviewed MAP inference in Bayesian IRL for computing the expert agent's reward function. In our work, the expert is the attacker, and the audit logs generated by the attacker's actions are the behavior data that we utilize to infer the attacker's preferences w_i for reward features ϕ_i. We present the MDP that we use to model the attacker agent next.

Action	States affected	Description
InitialAccessUser	AttackerActive	Attacker gets user-level access on the target host
InitialAccessRoot	AttackerActive AttackerPrivs	Attacker establishes privileged access
C2	C2Established	Attacker communicates with command-and-control infrastructure
IngressToolTransfer	IOCGenerated	Attacker downloads a payload on the target host
PrivEsc	AttackerPrivs	Attacker achieves elevated privileges
DataExfil	–	Attacker exfilterates sensitive data from the target
DefenseEvasion	IOCGenerated	Attacker deletes artifacts and evidence
Exit	AttackerActive	Attacker concludes the attack

Table 1: We model the set of attacker actions based on tactics and techniques in the MITRE ATT&CK matrix.

3.1 Attacker Model

We model the attacker's decision-making problem as an $MDP_{/R} = \langle S, A, T \rangle$, where S is the set of states of the target host system, A is the set of actions available to the attacker, and T is the transition function representing the effect of the attacker's actions on the target host. The reward function R capturing the attacker's behavior preferences is unknown to the defender and must be learned.

Elements of the state space represent the attacker's perspective of the state of the target host. The attacker utilizes her actions to manipulate these states based on her preferences. In our model of the attacker's MDP, we define the state space using 4 state features, $S = \{$AttackerActive, AttackerPrivs, IOCGenerated, C2Established$\}$. The AttackerActive state feature indicates whether the attacker has established a presence on the target host. The initial value of AttackerActive is `false`. An attacker may utilize numerous techniques ranging from sophisticated exploits to phishing attacks to get initial access to the target host. We model these techniques using the actions InitialAccessUser and InitialAccessRoot. Upon the attacker's initial access, AttackerActive transitions to `true`. Once inside the target host, an attacker may use the C2 action to reach the command and control (C2) infrastructure. The C2Established state feature represents whether the attacker has achieved communication with their C2 infrastructure. We represent the privilege level of the attacker on the target host using the AttackerPrivs feature. An attacker may have user-level privileges indicated by the feature value `user`, or root privileges indicated by `root`. A sophisticated attacker may achieve root privileges simultaneously while establishing initial access using the InitialAccessRoot action. Attackers may also utilize various privilege escalation techniques after establishing initial access. We model these techniques with the PrivEsc action. We model a noisy transition of AttackerPrivs due to PrivEsc to account for failed escalation attempts. Throughout the attack, the tools employed by the attacker may generate artifacts like log files. An attacker may also download additional tools such as malware stages using the IngressToolTransfer action. Such artifacts, also known as indicators-of-compromise, are subsequently utilized by forensics analysts to reconstruct and investigate the attack. The IOCGenerated state feature indicates whether the attacker's actions generate such indicators. The attacker may also erase indicators of the attack to prevent detection by end-point defense software using the DefenseEvasion action. Table 1 summarizes the complete set of attacker actions that we model.

3.2 Reward Features

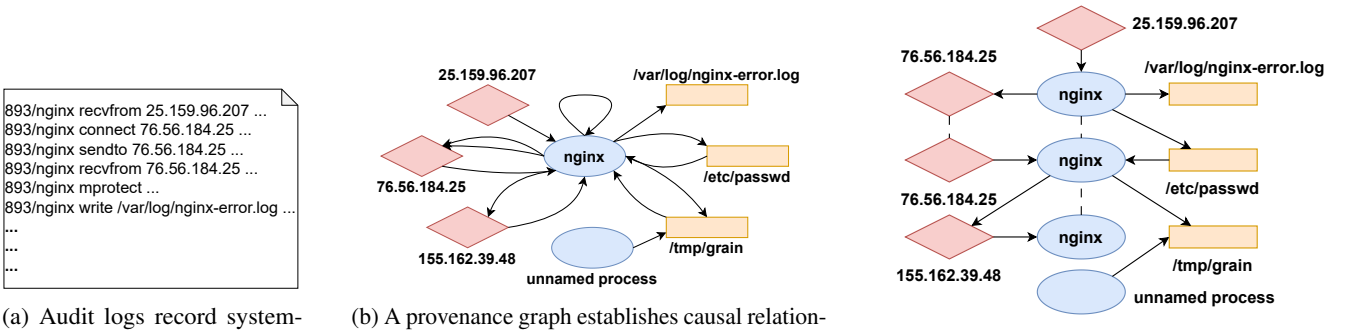
Recall that we approximate the attacker's reward function using a linearly weighted sum of bounded basis functions, $R(s,a) = \sum_{i=1}^{k} w_i \phi_i(s,a)$. The basis functions, $\phi_i(s,a)$, are the features used to characterize the attacker's behavior. Specifically, the preference features that we model are–

1. *Discoverability* is the attacker's use of actions that may produce digital artifacts or other evidence. If an attacker downloads tools or malware stages to the compromised host, the attack can be discovered in an investigation.
2. *Attributability* is the attacker's preference or lack thereof for actions that facilitate the identification of the threat actor group associated with the attack.
3. We model *sophistication* as the attacker's preference for using advanced tools and techniques. For instance, an attacker's ability to exploit privileged processes remotely is indicative of sophistication.
4. *Impact* is the attacker's preference for states and actions that can potentially cause severe consequences for the compromised host.
5. *Evasion* is the attacker's preference for erasing digital artifacts and indicators to avoid detection.
6. *Duration* is the attacker's preference for staying active in the compromised system.

Table 2 summarizes the reward features and their associated state-action values. Though not exhaustive, the features offer deep insights into the attacker's behavior and can be inferred.

Feature name	States	Actions
Discoverability	Att.Active = true	IngressToolTransfer
Attributability	Att.Active = true	C2
Sophistication	Att.Active = false	InitialAccessRoot
Impact	Att.Active = true	PrivEsc
Duration	Att.Active = true	DataExfil
Evasion	Att.Active = true IOCGen. = true	DefenseEvasion

Table 2: We use the reward features ϕ_i to model an attacker's behavioral tendencies. Here, * denotes any action.



(a) Audit logs record system-level interactions between different kernel-level objects

(b) A provenance graph establishes causal relationships between related entities that interact. Here, *nginx* is a Web server process

(c) State-based versioning techniques eliminate redundant edges between entities

Figure 1: Provenance graphs aid forensic investigations by causally connecting related entities even when they are temporally distant in logs.

We utilize the attacker's state-action trajectory to compute the preference weights, w_i of features ϕ_i using IRL. The values of these weights inform us about the preference ordering of each feature of the attacker's reward function. For instance, an attacker who prefers to stay undetected may prioritize deleting evidence of the attack and minimizing their attack duration over the level of impact. A likely preference ordering for such an attacker would be: *evasion* $\succ$ *sophistication* $\succ$ *impact* $\succ$ *duration* $\succ$ *attributability* $\succeq$ *discoverability*. In contrast, an attacker that aims to cause destruction, but is indifferent towards getting detected would prioritize maximizing impact on the compromised host resulting in a preference ordering: *impact* $\succ$ *duration* $\succ$ *sophistication* $\succ$ *attributability* $\succeq$ *discoverability* $\succeq$ *evasion*. Such information about the attacker's implicit preferences is not directly accessible using prevailing log analysis and attack reconstruction techniques on audit log data. By modeling the contextual features of the attacker's intent, IRL enables discerning these underlying behavioral characteristics from log data.

4 Trajectories from Provenance Graphs

In cyberattack scenarios, a breach is usually detected long after the initial intrusion. Consequently, post-attack investigations must rely on log data from compromised systems to analyze the attack. In our work, we first construct provenance graphs from the log data. We then employ subgraph isomorphism to obtain the actions defined in the attacker's MDP model for generating state-action trajectories of the attack trace.

4.1 Obtaining Scenario Graphs from Logs

In post-attack analyses, preliminary forensics investigations often yield information regarding the time of the initial breach, and notable artifacts such as malicious IP addresses and files. We start by processing audit log data recorded during the attack time interval. We then construct a versioned provenance graph from audit events during the attack time window using the state-versioning technique referenced in Section 2.1. Notably, the state-versioning algorithm eliminates redundant events while preserving dependency between subjects and objects. We employ forward and backward search techniques on the versioned provenance graph to extract a scenario graph.

Definition 1 (Attack scenario)**.** *An attack scenario graph* $G_s = \langle N_s, E_s \rangle$ *is a subgraph of the provenance graph* G *containing nodes* N_s *and edges* E_s *causally dependant on known attack-related nodes. It begins with first intrusion and terminates at attacker egress.*

To construct G_s, we traverse G backward from a known detection point– like a file, to a source– such as a network socket. We also collect process nodes that executed malicious files encountered during the backward tracing process. We then search forward from each source node to collect events and nodes causally related to that source node. We construct G_s from the nodes and events traversed during forward search

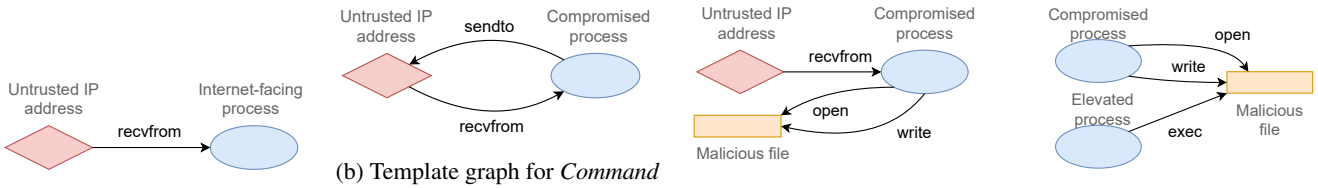
Event	Propagation rule
$S \xleftarrow{RECV} N$	$isUntrusted(N) \rightarrow \text{tag}(S)$
$S \xrightarrow{WRITE} F$	$isUntrusted(S) \rightarrow \text{tag}(F)$
$S \xrightarrow{EXEC} F$	$isUntrusted(F) \rightarrow \text{tag}(S)$
$S_1 \xrightarrow{FORK} S_2$	$isUntrusted(S_1) \rightarrow \text{tag}(S_2)$

Table 3: Propagation rules tag *untrusted* nodes based on their interactions with other nodes. (S = Subject, F = FileObject, N = NetflowObject)

The scenario graph constructed from forward and backward search may still contain subgraphs of benign events causally connected through false dependencies. To avoid false positives while mapping these attack subgraphs to attacker actions, we include an additional preprocessing step that employs tag propagation [Hossain *et al.*, 2017]. We first tag the nodes in the source set as *untrusted*. Then, traverse the attack scenario graph and utilize the following rules to propagate the tag:

1. If a subject node reads data from an untrusted network socket object, the subject node is tagged as untrusted.
2. If an untrusted subject node writes to a file object, the file object node is tagged as untrusted.
3. If a subject process or thread executes an untrusted file, the subject node is marked as untrusted.
4. If a subject node is untrusted, all its child processes are tagged as untrusted.

Table 3 summarizes the events and applicable tag-propagation rules that we utilize to taint untrusted nodes. If a node is tainted, we also taint all of its subsequent versions. We use this information to reduce false positives when matching subgraph templates with the scenario graph to extract attacker actions.



(a) Template for the *Initial Access* tactic shows the initial intrusion from a network socket to a process

(b) Template graph for *Command and Control* shows the attacker-controlled process communicating with an external address

(c) Template *Ingress Tool Transfer* shows the attacker-controlled process creating an untrusted file

(d) Template graph for *Privilege Escalation* shows a privileged process executing an untrusted file

Figure 2: The template graphs represent tactics and techniques in the ATT&CK matrix. We utilize these templates to identify attacker actions in the provenance graph using subgraph isomorphism.

4.2 Extracting Trajectories using Graph Isomorphism

The scenario graph we generate from system-level audit data contains all events causally related to the source node of an attack. We use subgraph isomorphism to match parts of the scenario graph with templates representing the attacker's actions. We call these as *template graphs* and the corresponding isomorphic subgraphs in the scenario graph as *action subgraphs*. Recall from Section 2.1 that the attacker actions we model are grounded in the MITRE ATT&CK matrix. **Consequently, we develop template graphs for the attacker actions based on the tactics and techniques in the ATT&CK matrix.** Specifically, we build template graphs for the *Initial Access*, *Command and Control*, *Execution*, *Privilege Escalation*, and *Defense Evasion* tactics. Figure 2 illustrates the template graphs for some notable actions we model. Notice that most of the attacker's actions span multiple causally related events. In a raw log file, long durations of unrelated activity may separate discrete events representing the same action. A provenance graph facilitates the direct matching of related events using subgraph isomorphism and *unification*, regardless of their temporal separation in a log file. Unification facilitates the matching of template and scenario subgraphs by substituting node information like process privileges and taint tags from the scenario graph into the template graphs and checking for equivalence. We unify each template graph with an isomorphic attack scenario subgraph to get a series of action subgraphs representing the attacker's action sequence.

Feature information from the nodes of the action subgraphs also facilitates the tracking of state values of the attacker's MDP. Information about a tainted subject's privileges indicates the attacker's privilege level, which determines the value of the AttackerPrivs state feature. Similarly, we monitor the filename information from the action subgraphs representing *Ingress Tool Transfer* to track the indicators of compromise that this action generates. We utilize this information to determine the value of the IOCGenerated state feature. C2 IP addresses are collected similarly from the C2 action subgraphs. Subsequently, we retrieve information about the attacker's actions and the MDP state transitions from the action subgraphs.

4.3 Learning Attacker Preferences

We utilize the actions and state information obtained using subgraph isomorphism to define a state-action trajectory of the attacker's observed behavior. An example sequence of an attacker's actions from such a trajectory would be, $\{$*InitialAccessUser, C2, IngressToolTransfer, PrivEsc, ..., C2, DataExfil*$\}$. In this example, the attacker first achieves user-level access and then establishes C2. The attacker then downloads a payload and escalates it to root privileges. Finally, the attacker exfiltrates data from the target system. Our methodology accordingly lifts low-level system call logs to this higher level of abstraction thereby enabling the application of IRL to learn the behavioral preferences of the attacker. The host-level cyber threat domain MDP defined in Section 3 serves as our model of the environment. The model-based MAP-BIRL reviewed in Section 2.2 utilizes this MDP to infer the attacker's preferences that explain the trajectory obtained from the logs. We also use model-free MLE-IRL to learn the preferences and compare the agreement between the two techniques. For the abovementioned example trajectory, a likely preference ordering would be $\{$*attributability, discoverability, impact, duration*$\} \succ \{$*evasion, sophistication*$\}$. We evaluate the performance of IRL at learning the attacker's preference function, using an empirical estimate of inverse learning error (ILE), $||V^{\pi_E} - V^{\hat{\pi}_E}||$, where V^{π_E} is value of the attacker's observed trajectory using the learned Q function, and $V^{\hat{\pi}_E}$ is value of sampled trajectories from the learned policy. In the next section, we evaluate the performance of IRL toward learning attacker preferences directly from real log data.

5 Experiments

We described our model of the attacker's MDP in Section 3, and a general methodology for extracting state-action trajectories from a provenance graph representation of raw audit logs in Section 4. We utilize these trajectories with our model of the attacker's MDP to infer an attacker's hidden preference ordering for the reward features described in Section 3.2.

5.1 Realistic Attack Datasets

We evaluate our approach on large publicly available datasets from DARPA's transparent computing (TC) program [Keromytis, 2018]. The data was collected from a series of red team engagements, of which Engagement 3 was publicly released. The attacks comprised APT simulations on hosts with various provenance capture software which DARPA intended to evaluate. We use the CADETS dataset [Strnad *et al.*, 2019] recorded on a FreeBSD host, and the THEIA dataset [Fazzini, 2017] recorded on a Linux host. We evaluate our approach on **4** provenance graphs automatically constructed from the CADETS logs and **2** from the THEIA logs. These graphs contain data from separate APT attacks. Table 4

shows the size of each attack scenario graph with the attack storyline obtained from log analysis and after-action reports.

Datasets	$\|N_s\|$	$\|E_s\|$	Attack storyline and Ground truth
CADETS-1	8,394	21,394	$RE \rightarrow C2 \rightarrow PE \rightarrow DE$ **(Features: Sop, Att, Imp, Dis)**
CADETS-2	1,734	9,267	$UA \rightarrow C2$ **(Features: Att, Dis)**
CADETS-3	44,974	98,584	$UA \rightarrow C2 \rightarrow PE \rightarrow DE$ **(Features: Att, Imp, Dis, Eva)**
CADETS-4	196,502	469,094	$UA \rightarrow C2 \rightarrow PE$ **(Features: Att, Imp, Dis)**
THEIA-1	824	911	$UA \rightarrow C2 \rightarrow PE$ **(Features: Att, Imp, Dis)**
THEIA-2	30,504	70,300	$UA \rightarrow C2 \rightarrow PE \rightarrow DE$ **(Features: Att, Imp, Dis, Eva)**

Table 4: Size of each dataset's attack scenario graph with the attack storyline (RE = Root exploit, UA = User-level access, C2 = Command and control, PE = Privilege elevation, DE = Defense evasion). Associated behavioral features are also shown (Sop = Sophistication, Att = Attributability, Imp = Impact, Dis = Discoverability, Eva = Evasion, Dur = Duration), which serves as the ground truth.

The DARPA TC dataset is widely used for evaluating provenance graph-based APT detection. While some inconsistencies were reported in the data caused by the target hosts occasionally crashing during process-injection attacks, the dataset is a benchmark in the cybersecurity community due to a lack of realistic attack log data. We could infer attacker preferences from the available data using our approach. We show detailed trajectories extracted from each dataset in Appendix A.

5.2 Learned Preferences

We generated provenance graphs for each APT attack in the CADETS and THEIA datasets and stored them in a Neo4j database. We then extracted state-action trajectories for each attack using Cypher queries for subgraph isomorphism. To infer the attacker's preferences, we use MAP-BIRL as well as *model-free* maximum likelihood estimation (MLE) approach to IRL [Jain *et al.*, 2019]. We then use Mean Shift clustering to group the preference features according to their learned weights. We estimate ILE using 1000 sampled trajectories. Both techniques are effective in learning the preference function as indicated by the low ILE values, with MAP-BIRL performing better on more of the datasets.

Datasets	Spearman's ρ	Inverse learning error	
		MAP-BIRL	MLE-IRL
CADETS-1	$0.94, p < 0.005$	1.75 ± 1.44	1.46 ± 1.66
CADETS-2	$0.82, p < 0.05$	3.74 ± 3.5	6.21 ± 3.45
CADETS-3	$0.94, p < 0.005$	4.18 ± 2.5	9.61 ± 8.76
CADETS-4	$0.77, p < 0.08^{\dagger}$	1.2 ± 1.35	3.1 ± 5.44
THEIA-1	$0.94, p < 0.005$	4.4 ± 4.26	11.77 ± 7.08
THEIA-2	$0.94, p < 0.005$	3.53 ± 3.58	10.71 ± 6.48

Table 5: Spearman's rank correlation coefficient (ρ) measures the agreement between the feature weights learned by MAP-BIRL and MLE-IRL. †– may not correlate. MAP-BIRL exhibits a lower ILE for all but one dataset.

Figure 3 shows the normalized weights representing each attacker's behavioral preferences learned using both methods. Logs from CADETS-1 consisted of **18** attacker actions. The attacker deployed a remote exploit to get *root*-level access via an *nginx* server (InitialAccessRoot) demonstrating high *sophistication*. The attacker's preference to escalate a malware payload, erase evidence and C2 with multiple IP addresses indicated high *impact*, *evasion*, and *attributability* as shown in Table 4. Both MAP and MLE IRL correctly infer these preferences as shown in Fig. 3a. In the CADETS-2 attack, our methodology identified **7** attacker actions. The attacker initially gained user-level access (InitialAccessUser) and immediately established C2. Subsequently, the attacker downloaded a payload (IngressToolTransfer) and concluded the attack. The attacker's failure to elevate privileges and erase the payload was correctly inferred by MAP via negative weights for *impact* and *evasion* as shown in Fig. 3b. The CADETS-3 attack contained **55** actions. Similar to CADETS-2, the attacker gained access by exploiting a Web-facing application. Subsequently, the attacker downloaded multiple payloads for process injection but failed, and instead ran an elevated process. The attacker's preference for subsequent privilege escalation instead of an initial root-level exploit was correctly inferred by MAP-BIRL as a lack of *sophistication*. The CADETS-4 and THEIA-1 attacks were similar, consisting of **17** and **12** actions respectively. The attackers gained user-level access, established C2, and elevated privileges. However, both failed to erase their respective payloads to avoid detection. Both IRL techniques correctly identified this behavioral preference as indicated by lower values for *evasion* in Figs. 3d and 3e. Finally, the THEIA-2 attack contained **11** actions. The attacker gained user-level access and downloaded payloads for injection. However, the attacker promptly deleted all except one payload. Consequently, Fig. 3f shows higher preferences for *discoverability* and *evasion*. Similar to CADETS-3,4 and THEIA-1, a lack of *sophistication* was also observed. None of the attacks contained data exfiltration or similar actions requiring prolonged attacker presence. Consequently, the weights for *duration* were low for all attacks.

Datasets	Ground Truth	Preference Ordering Learned by MAP-BIRL
CADETS-1	Sop, Att, Imp, Dis	$\{Att, Eva, Dis, Imp, Sop\} \succ Dur$
CADETS-2	Att, Dis	$\{Att, Dis\} \succ \{Eva, Imp, Dur, Sop\}$
CADETS-3	Att, Imp, Dis, Eva	$\{Att, Imp, Eva, Dis\} \succ \{Sop, Dur\}$
CADETS-4	Att, Imp, Dis	$\{Att, Dis, Imp\} \succ \{Eva, Dur\} \succ Sop$
THEIA-1	Att, Imp, Dis	$\{Att, Dis, Imp\} \succ \{Eva, Dur, Sop\}$
THEIA-2	Att, Imp, Dis, Eva	$\{Eva, Dis, Att, Imp\} \succ \{Dur, Sop\}$

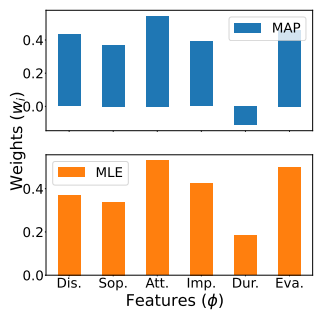
Table 6: The preference orderings learned by MAP-BIRL is consistent with the features emphasized in the ground truth.

Table 5 shows the rank correlation between the preferences learned by both IRL techniques. The table also shows that both techniques were effective in learning the attacker's reward function as indicated by the low values of ILE. Note that the preferences inferred by our methodology are consistent with the ground-truth as shown in Table 6. These weights can serve as unique behavior-generating signatures of the attackers.

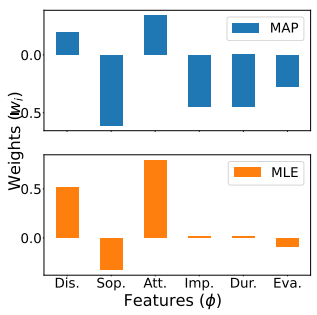
6 Related Work

Recognizing an attacker's intent from forensics data is a topic of much interest at the intersection of cyber security and AI.

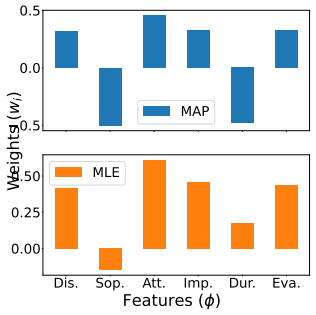
Log Analysis: Several recent works in cybersecurity adopt provenance graphs for APT detection [Hossain *et al.*, 2017;



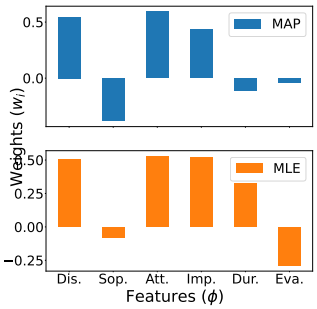
(a) Behavioral signature from MAP for CADETS-1 indicates the ordering, $\{$*attributability, evasion, discoverability, impact, sophistication*$\}$ $\succ$ *duration*



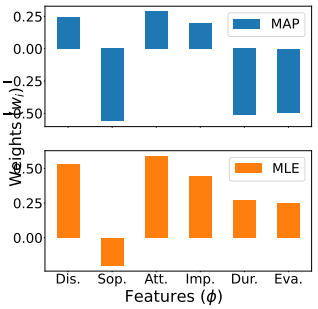
(b) Behavioral signature from MAP for CADETS-2 indicates the ordering, $\{$*attributability, discoverability*$\}$ $\succ$ $\{$*evasion, impact, duration, sophistication*$\}$



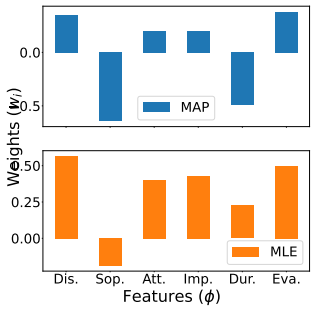
(c) Behavioral signature from MAP for CADETS-3 indicates the ordering, $\{$*attributability, impact, evasion, discoverability*$\}$ $\succ$ $\{$*sophistication, duration*$\}$



(d) Behavioral signature from MAP for CADETS-4 indicates the ordering, $\{$*attributability, discoverability, impact*$\}$ $\succ$ $\{$*evasion, duration*$\}$ $\succ$ *sophistication*



(e) Behavioral signature from MAP for THEIA-1 indicates the ordering, $\{$*attributability, discoverability, impact*$\}$ $\succ$ $\{$*evasion, duration, sophistication*$\}$



(f) Behavioral signature from MAP for THEIA-2 indicates the ordering, $\{$*evasion, discoverability, attributability, impact*$\}$ $\succ$ $\{$*duration, sophistication*$\}$

Figure 3: The reward functions inferred from the trajectories of different attackers using MAP-BIRL show their behavioral preferences.

Milajerdi *et al.*, 2019a; Wang *et al.*, 2022; Cheng *et al.*, 2024]. HOLMES [Milajerdi *et al.*, 2019b] is one such relevant approach that explains APT campaigns at a tactical level. However, these approaches aim to identify APT activity and reconstruct attacks from provenance graphs. Our work goes beyond attack detection and models APT behavior to supplement post-attack investigations with deeper insights into attacker preferences.

AI-based Intent Recognition: Recently, AI-based techniques are also being applied to attacker intent recognition [Kassa *et al.*, 2024]. One such approach proposes an AI-based methodology to identify attack phases from system call logs using an HMM and learned classifiers [AbuOdeh *et al.*, 2021]. Another approach employed an HMM to recognize tactics in the MITRE ATT&CK matrix from sensor alerts [Zhang *et al.*, 2009]. Instead, we model these tactics and phases as actions and learn the intrinsic behavioral preferences of an attacker from them. Another interesting work adopts the I-POMDP$_\mathcal{X}$ framework for attacker intent recognition on a honeypot system [Shinde *et al.*, 2021]. The I-POMDP$_\mathcal{X}$-based defender employs deception to actively infer an attacker's intent from a predefined set specified a priori. In contrast, our work does not make such assumptions about the attacker's intent and learns preferences for abstract behavioral features. As such, our approach to modeling these preferences using IRL differs significantly from conventional approaches in cybersecurity.

7 Conclusion

The lack of representative data on adversarial intent is a known challenge in cybersecurity. Low-level forensics logs are often the only source of attack-relevant data. Conventional approaches to understanding adversary tools and techniques typically do not provide long-term insights into adversary behavior as attacker tools and techniques evolve constantly. However, their behavioral tendencies are long-lasting and independent of their specific tools. Consequently, automated ways of extracting higher-level (deeper) insights into adversary behavior from low-level data is very valuable to modern cyber defense. Our AI-anchored methodology instruments a *novel* use case of IRL to model cyber adversaries. The significant results demonstrate the efficacy of the methodology and IRL's effectiveness in correctly learning adversary behavior from log data.

Insights into an adversary's behavioral tendencies will enable defenders to orient their defenses appropriately to prevent future attacks. Toward this, future work could utilize the learned preferences to engage in forward RL on simulations of various host configurations to predict how an attack from the adversary would unfold.

References

[Abbeel and Ng, 2004] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, 2004.

[AbuOdeh *et al.*, 2021] Muhammed AbuOdeh, Christian Adkins, Omid Setayeshfar, Prashant Doshi, and Kyu H Lee. A novel ai-based methodology for identifying cyber attacks in honey pots. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15224–15231, 2021.

[Arora and Doshi, 2021] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[Cheng *et al.*, 2024] Zijun Cheng, Qiujian Lv, Jinyuan Liang, Yang Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. Kairos: Practical intrusion detection and investigation using whole-system provenance. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024.

[Choi and Kim, 2011] Jaedeug Choi and Kee-Eung Kim. Map inference for bayesian inverse reinforcement learning. *Advances in neural information processing systems*, 24, 2011.

[Fazzini, 2017] Mattia Fazzini. Tagging and tracking of multilevel host events for transparent computing. 2017.

[Ferguson-Walter *et al.*, 2019] Kimberly Ferguson-Walter, Sunny Fugate, Justin Mauger, and Maxine Major. Game theory for adaptive defensive cyber deception. In *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, page 4, New York, NY, USA, 2019. ACM, Association for Computing Machinery.

[Hossain *et al.*, 2017] Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and VN Venkatakrishnan. {SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 487–504, 2017.

[Hossain *et al.*, 2018] Md Nahid Hossain, Junao Wang, Ofir Weisse, R Sekar, Daniel Genkin, Boyuan He, Scott D Stoller, Gan Fang, Frank Piessens, Evan Downing, et al. {Dependence-Preserving} data compaction for scalable forensic analysis. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1723–1740, 2018.

[Jain *et al.*, 2019] Vinamra Jain, Prashant Doshi, and Bikramjit Banerjee. Model-free irl using maximum likelihood estimation. In *AAAI*, pages 3951–3958, 2019.

[Karantzas and Patsakis, 2021] George Karantzas and Constantinos Patsakis. An empirical assessment of endpoint detection and response systems against advanced persistent threats attack vectors. *Journal of Cybersecurity and Privacy*, 1(3):387–421, 2021.

[Kassa *et al.*, 2024] Yidnekachew Worku Kassa, Joshua Isaac James, and Elefelious Getachew Belay. Cybercrime intention recognition: A systematic literature review. *Information*, 15(5):263, 2024.

[Keromytis, 2018] Angelos D Keromytis. Transparent computing engagement 3 data release. *README-E3. md*, 2018.

[King and Chen, 2003] Samuel T King and Peter M Chen. Backtracking intrusions. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 223–236, 2003.

[Lee *et al.*, 2013a] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. High accuracy attack provenance via binary-based execution partition. In *NDSS*, volume 16, 2013.

[Lee *et al.*, 2013b] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. Loggc: garbage collecting audit log. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1005–1016, 2013.

[Milajerdi *et al.*, 2019a] Sadegh M Milajerdi, Birhanu Eshete, Rigel Gjomemo, and VN Venkatakrishnan. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1795–1812, 2019.

[Milajerdi *et al.*, 2019b] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1137–1152. IEEE, 2019.

[Mirsky *et al.*, 2019] Reuth Mirsky, Ya'ar Shalom, Ahmad Majadly, Kobi Gal, Rami Puzis, and Ariel Felner. New goal recognition algorithms using attack graphs. In *Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Beer-Sheva, Israel, June 27–28, 2019, Proceedings 3*, pages 260–278. Springer, 2019.

[Ng and Russell, 2000] Andrew Y Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[Ramachandran and Amir, 2007] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[Sarraute *et al.*, 2012] Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann. Pomdps make better hackers: Accounting for uncertainty in penetration testing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1816–1824, 2012.

[Schlenker *et al.*, 2018] Aaron Schlenker, Omkar Thakoor, Haifeng Xu, Long Tran-Thanh, Fei Fang, Phebe Vayanos, Milind Tambe, and Yevgeniy Vorobeychik. Deceiving cyber adversaries: A game theoretic approach. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2:892–900, 2018.

[Setayeshfar *et al.*, 2019] Omid Setayeshfar, Christian Adkins, Matthew Jones, Kyu Hyung Lee, and Prashant Doshi. GrAALF: Supporting Graphical Analysis of Audit Logs for Forensics. *arXiv e-prints*, September 2019.

[Shinde and Doshi, 2024] Aditya Shinde and Prashant Doshi. Modeling cognitive biases in decision-theoretic planning for active cyber deception. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 1718–1726, 2024.

[Shinde *et al.*, 2021] Aditya Shinde, Prashant Doshi, and Omid Setayeshfar. Cyber attack intent recognition and active deception using factored interactive pomdps. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1200–1208, 2021.

[Strnad *et al.*, 2019] Amanda Strnad, Quy Messiter, Robert Watson, Lucian Carata, Jonathan Anderson, and Brian Kidney. Casual, adaptive, distributed, and efficient tracing system (cadets). Technical report, Technical report, BAE Systems Burlington United States, 2019.

[Strom *et al.*, 2018] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. Mitre att&ck: Design and philosophy. Technical report, MITRE Corp., 2018.

[Tang *et al.*, 2018] Yutao Tang, Ding Li, Zhichun Li, Mu Zhang, Kangkook Jee, Xusheng Xiao, Zhenyu Wu, Junghwan Rhee, Fengyuan Xu, and Qun Li. Nodemerge: Template based efficient data reduction for big-data causality analysis. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1324–1337, 2018.

[Wang *et al.*, 2022] Su Wang, Zhiliang Wang, Tao Zhou, Hongbin Sun, Xia Yin, Dongqi Han, Han Zhang, Xingang Shi, and Jiahai Yang. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security*, 17:3972–3987, 2022.

[Xu *et al.*, 2016] Zhang Xu, Zhenyu Wu, Zhichun Li, Kangkook Jee, Junghwan Rhee, Xusheng Xiao, Fengyuan Xu, Haining Wang, and Guofei Jiang. High fidelity data reduction for big data security dependency analyses. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 504–516, 2016.

[Zhang *et al.*, 2009] Qiang Zhang, Dapeng Man, and Wu Yang. Using hmm for intent recognition in cyber security situation awareness. In *2009 Second International Symposium on Knowledge Acquisition and Modeling*, volume 2, pages 166–169. IEEE, 2009.

8 Appendix

8.1 CADETS-1

The CADETS-1 attacker exploited an internet-facing application running at elevated privileges for initial access. The attacker then established command and control with the IP addresses 78.205.235.65 and 200.36.109.214. Next, the attacker downloaded the file /tmp/vUgefal. This file was an APT stage which the attacker executed with root-level privileges. The attacker then deleted /tmp/vUgefal to avoid detection. Subsequently, the attacker downloaded another APT stage

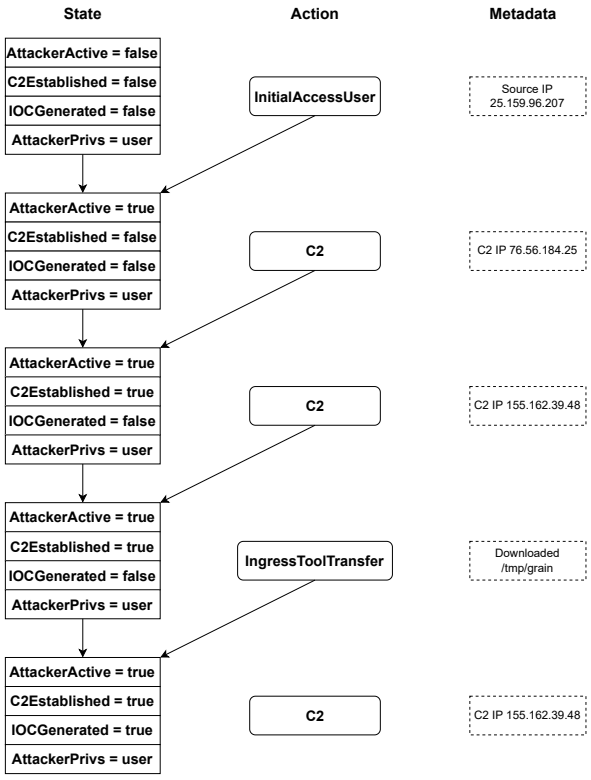


Figure 4: The state-action trajectory for the CADETS-2 attack

/var/log/devc before exiting the system. Figure 5 shows the important state-action pairs for this trajectory.

8.2 CADETS-2

Figure 4 shows the trajectory for the CADETS-2 attacker. The attacker established user-level access by exploiting an internet-facing application. Next, the attacker reached out to the IP addresses 76.56.184.25 and 155.162.39.48. Finally, the attacker downloaded the file /tmp/grain, an APT stage. Figure 6 shows a small provenance subgraph of the CADETS-2 attack and attacker actions obtained using subgraph isomorphism with the template graphs.

8.3 CADETS-3

The CADETS-3 attacker began with user-level access to the target. The attack lasted for 53 steps. Throughout the attack, the attacker communicated with 76.56.184.25, 155.162.39.48, 53.158.101.118, and 192.113.144.28 for C2. Additionally, the attacker downloaded multiple APT stages and attempted to escalate them. As a result, various indicators of compromise were generated. Specifically, /tmp/tmux-1002, /tmp/minions, /tmp/font, /tmp/XIM, /var/log/netlog, /var/log/sendmail, /tmp/main, and /tmp/test were the files containing APT stages that the attacker downloaded. Subsequently, the attacker deleted all files except /tmp/minions. Figure 7 shows the important state-action pairs for this trajectory.

8.4 CADETS-4

The CADETS-4 attacker started the attack with user-level access to the target. The attacker established command and control with the IP addresses 76.56.184.25, 155.162.39.48, and 53.158.101.118. The attacker also downloaded the APT stages /tmp/pEja72mA, /tmp/eWq10bVcx, /tmp/memhelp.so, /tmp/eraseme, and /tmp/done.so. Only /tmp/pEja72mA was escalated to root privileges. The attacker also did not attempt to erase any downloaded APT stages. Figure 7 shows the notable state-action pairs for this trajectory.

8.5 THEIA-1

The THEIA-1 attacker also started the attack by exploiting a user-level application on the target. The attacker used the IP addresses 146.153.68.151 and 161.116.88.72 for C2. The attacker then downloaded the APT stages /home/admin/clean and /home/admin/profile. Both APT stages were elevated to root privileges for execution. Similar to CADETS-4, the attacker did not attempt to avoid detection by deleting the APT stages once they were executed. Figure 9 shows the THEIA-1 trajectory.

8.6 THEIA-2

The THEIA-2 attacker started with user-level privileges by exploiting the Firefox browser. The attacker downloaded the APT stage /etc/firefox/native-messaging-hosts/gtcache. The attacker executed this APT stage with user-level privileges. Subsequently, the attacker downloaded additional APT stages /var/log/wdev, /tmp/memtrace.so, and /var/log/mail. However, the attacker only elevated /var/log/mail to root privileges and deleted all the other files. Throughout the attack, the attacker communicated with 146.153.68.151 for command and control.Figure 10 shows the THEIA-2 trajectory. Figure 11 shows a small provenance subgraph of the THEIA-2 attack and attacker actions obtained using subgraph isomorphism with the template graphs.

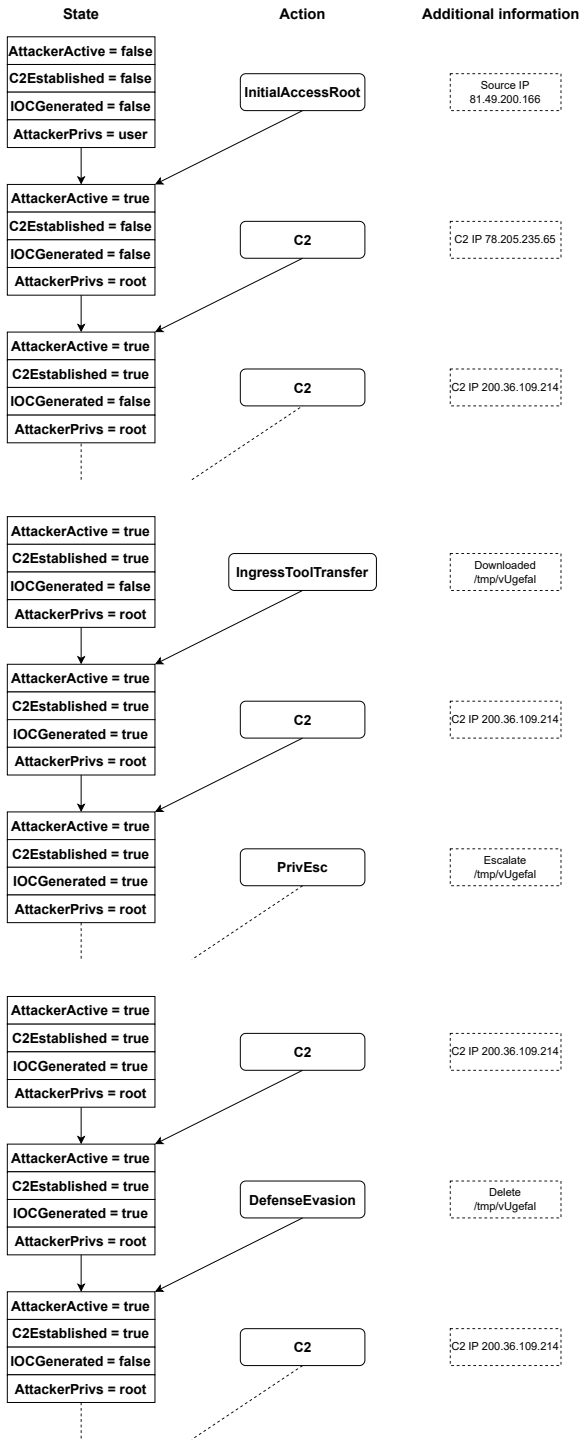


Figure 5: The state-action trajectory for the CADETS-1 attack

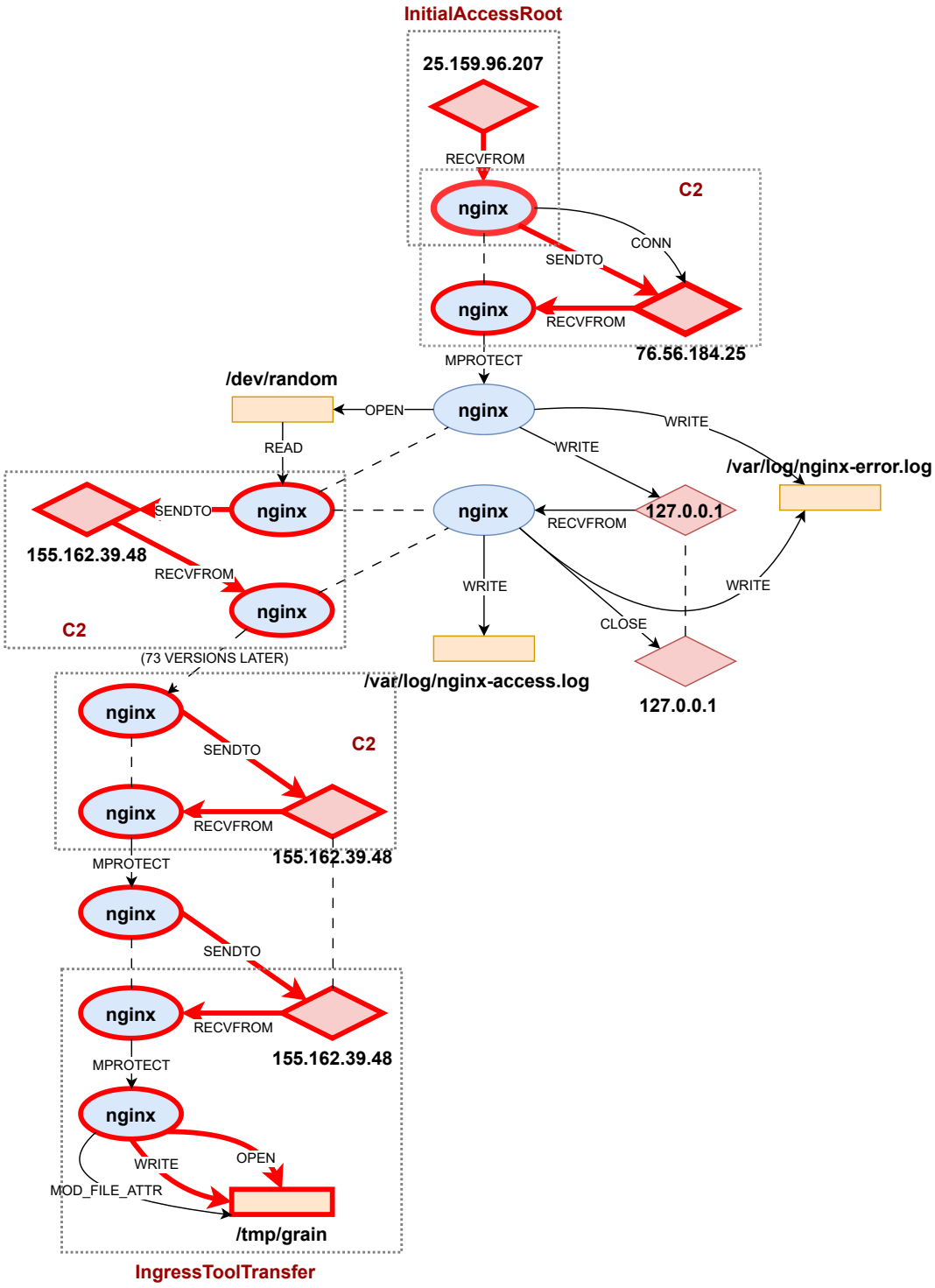


Figure 6: A small subgraph of the state-versioned provenance graph for the CADETS-2 attack shows the attacker's activity on the target system. The nodes and edges highlighted in red match the subgraph templates for attacker actions in the MDP model. The dotted boxes surrounding those subgraphs indicate the action that was identified using subgraph isomorphism

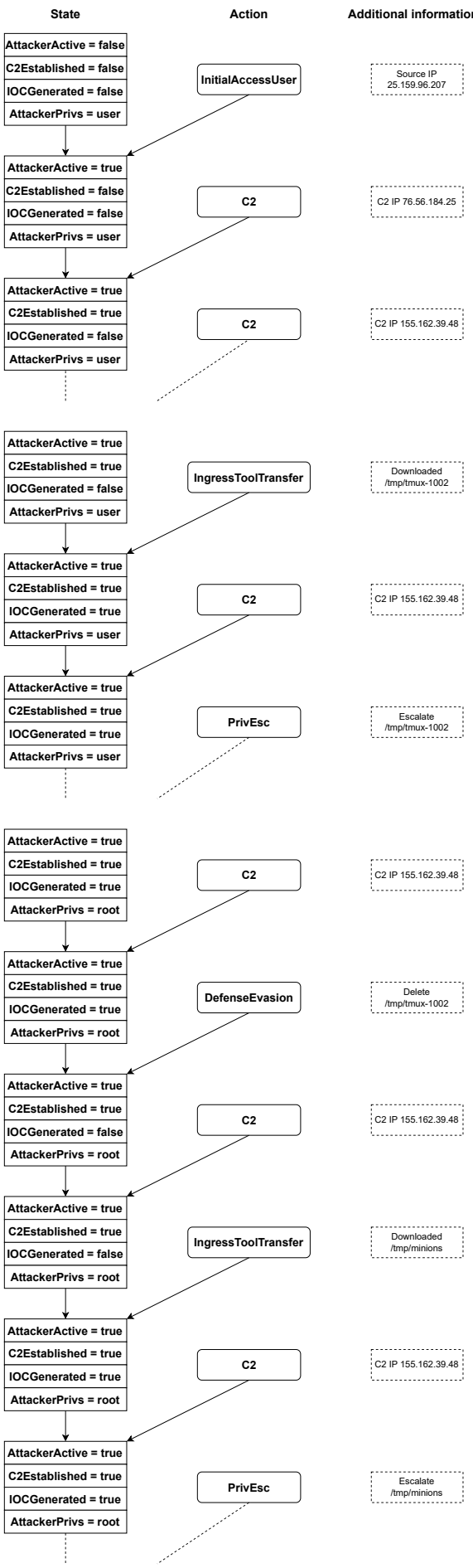


Figure 7: The state-action trajectory for the CADETS-3 attack

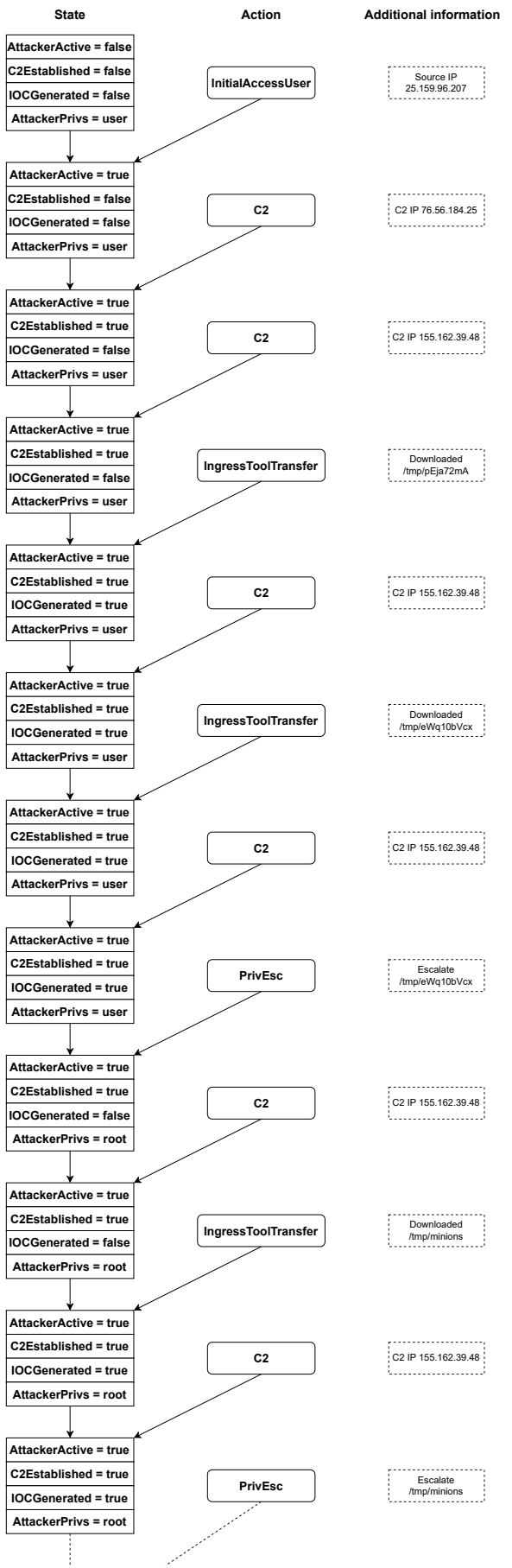


Figure 8: The state-action trajectory for the CADETS-4 attack

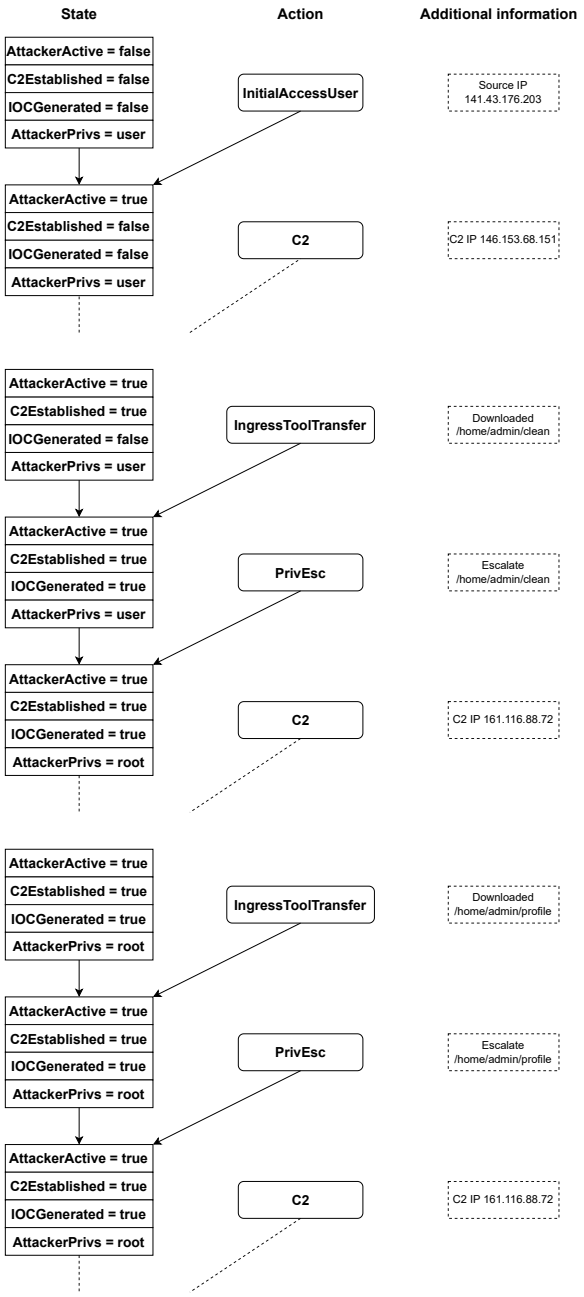


Figure 9: The state-action trajectory for the THEIA-1 attack

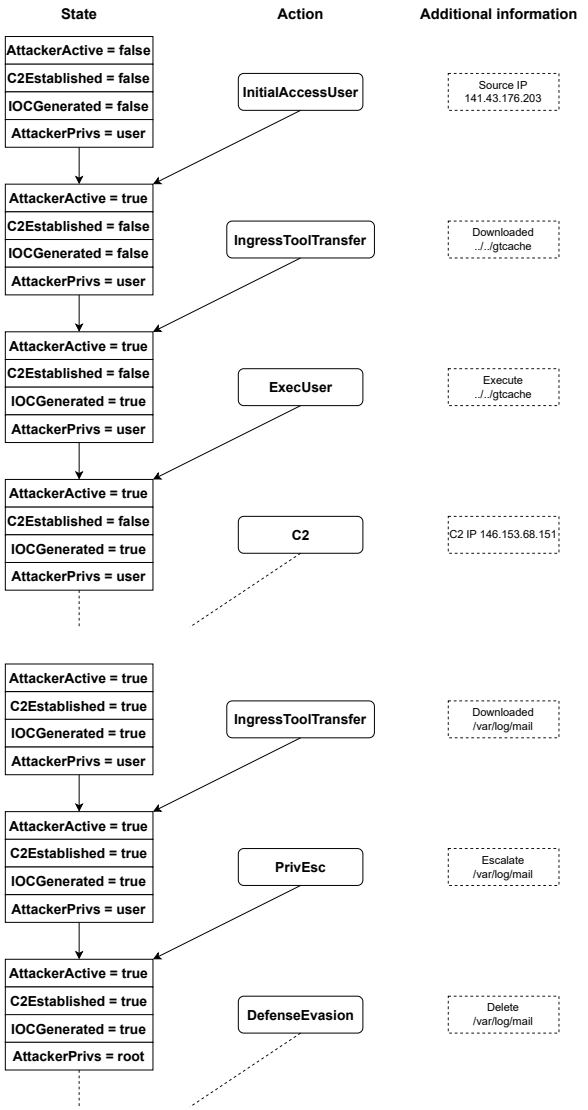


Figure 10: The state-action trajectory for the THEIA-2 attack

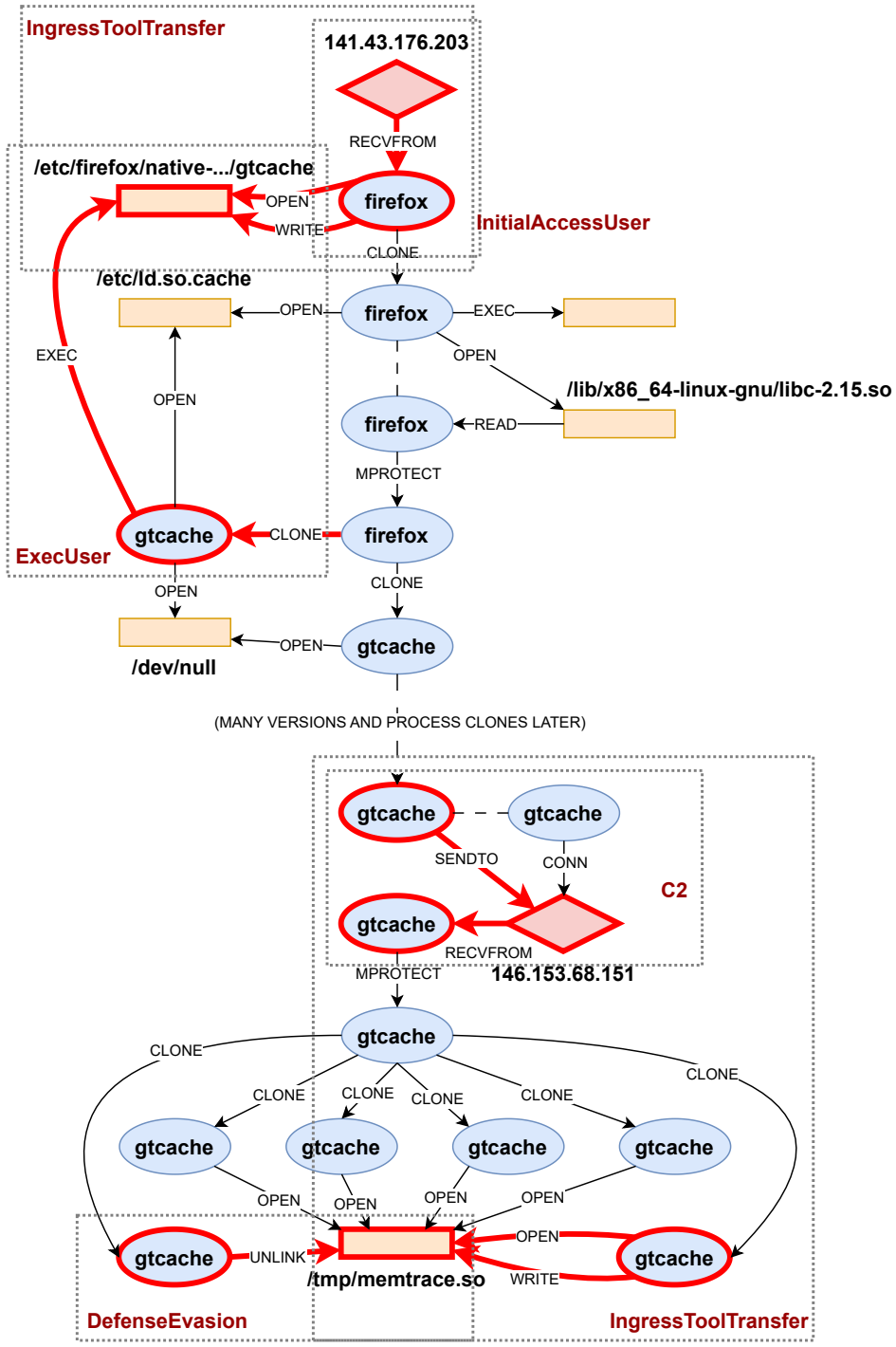


Figure 11: A small subgraph of the state-versioned provenance graph for the THEIA-2 attack shows the attacker's activity on the target system. The nodes and edges highlighted in red match the subgraph templates for attacker actions in the MDP model. The dotted boxes surrounding those subgraphs indicate the action that was identified using subgraph isomorphism