Towards Cyber Resilience against APTs

Alessandro Gaudenzi[1,*], Lorenzo Nodari[1], Rodolfo Valentim[3], Danilo Giordano[3], Idilio Drago[2], Alessandra Russo[4] and Federico Cerutti[1]

[1]Department of Information Engineering, University of Brescia, Brescia, BS, Italy
[2]Computer Science Department, University of Turin, Turin, TO, Italy
[3]Department of Control and Computer Engineering, Politecnico di Torino, Turin, TO, Italy
[4]Imperial College London, London, UK

Abstract

This paper examines the Uber data breach of September 2022, where the Lapsus$ group exploited multi-factor authentication (MFA) fatigue to compromise contractor credentials. The attackers gained access to internal systems, demonstrating the sophistication and persistence of modern Advanced Persistent Threats (APTs). Using the ACRE framework, which focuses on later stages of the cyber kill chain, we highlight how effective Cyber Threat Intelligence (CTI) can systematically detect and analyse such attacks. The ACRE framework provides tools to collect, process, and analyse threat data, enabling organisations to identify APT activity and mitigate risks proactively. By applying ACRE to the Uber breach, this study demonstrates its capacity to uncover critical intelligence and improve defensive strategies. The case underscores the importance of intelligence-driven approaches in addressing the complexities of contemporary cyber threats and enhancing organisational resilience.

Keywords

Threat intelligence, Attack modelling, Neuro-symbolic machine learning,

1. Introduction

The modern threat landscape, characterised by an evolving adversarial ecosystem and an expanding attack surface, presents a level of complexity that profoundly impacts the interconnected fabric of our digital world. This environment fosters the proliferation of APTs [1, 2], where sophisticated attackers relentlessly seek to infiltrate networks, exfiltrate sensitive information, or establish a foothold for launching subsequent attacks. Today's cyber threats are increasingly powered by artificial intelligence, persistent in nature, and omnipresent across all domains of the digital ecosystem.

Advanced Persistent Threats represent a significant challenge in contemporary cybersecurity. These threats are characterized by their sophisticated orchestration, stealthy execution, extended persistence, and targeting of valuable assets across diverse sectors. APTs typically involve prolonged and targeted cyberattacks, often orchestrated by well-funded and skilled adversaries, including nation-states and organized cybercriminal groups. The complexity and persistence of APTs necessitate advanced detection and defense mechanisms to protect critical infrastructure and sensitive information.

A clear example of the intricacy of APT operations is provided by the Lockheed-Martin cyber kill chain model [3], which outlines six key phases typically employed by attackers. The sequence begins with reconnaissance, where adversaries gather intelligence to identify suitable targets. This is followed by weaponisation, where a tailored payload is created and subsequently delivered to the target system. Once delivered, the attacker exploits vulnerabilities to install a persistent backdoor. At this stage, the malware establishes a command-and-control (C2) channel, enabling the attacker to pursue their mission objectives. The cyber kill chain model also highlights potential defensive strategies, including detection, denial, disruption (e.g., inline anti-virus), degradation (e.g., throttling communication), and deception (e.g., deploying decoys such as honeypots).

The **focus of the ACRE framework** lies in detecting attacks during the later stages of the kill chain. This approach necessitates the collection and analysis of intelligence—often derived from deception-based tools—to develop mechanisms capable of identifying threats at earlier stages.

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT.
*Corresponding author.

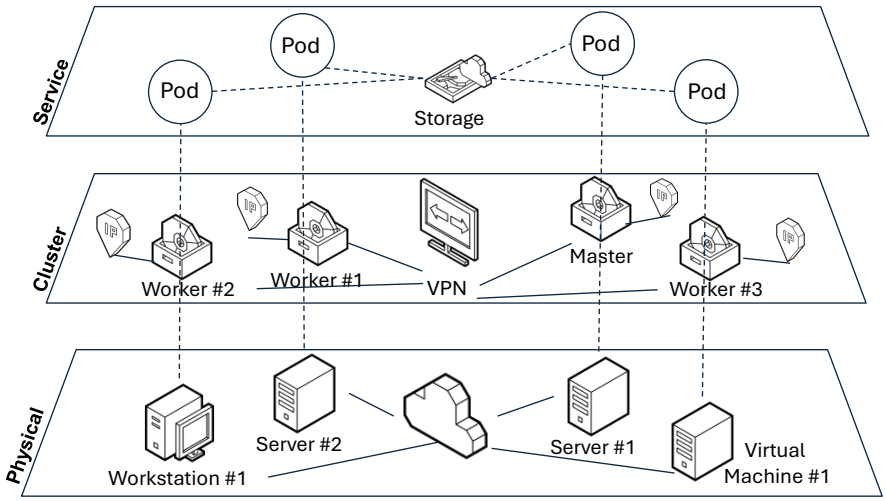

Figure 1: Architecture of data collection platform.

CTI is the iterative process used by analysts to generate actionable insights into the vulnerabilities of organisational assets that adversaries could exploit. Similar to traditional intelligence methodologies [4], CTI involves several distinct phases. The process begins with defining intelligence requirements, such as identifying APT campaigns. Analysts, either human or autonomous, then collect raw data—such as network logs from firewalls—into preliminary storage repositories, often referred to as "shoeboxes." These raw data sets are then organised into an evidence file (Section 2).

Subsequent stages involve data processing, where meaningful semantics are applied to the raw data, creating a structured schema. This enriched dataset becomes the foundation for deeper analysis, hypothesis generation, and verification (Section 3).

Our preliminary experimental analysis (Section 4) builds on top of the Uber data breach by Lapsus$. In September 2022, Uber Technologies Inc. suffered a cybersecurity breach by the Lapsus$ group using *MFA fatigue* to compromise contractor credentials. The attackers accessed internal systems, posted messages on Slack, and altered OpenDNS. Uber reported no access to sensitive user data. By systematically following the methodological steps of CTI, organisations can develop a robust understanding of APT activity and implement proactive measures to mitigate future threats.

2. ACRE Data Gathering and Processing Framework

The ACRE data collection architecture consists of several distributed nodes, each equipped with specialized probes for capturing diverse data feeds from telescopes, honeypots, and CTI crawlers. As seen in Figure 1, each node operates in a virtualized environment for edge computing. This structure is designed to be deployed in multiple network providers, collecting data nearby possible victims of remote attacks, such as servers in datacenters or client/IoT devices with public addresses on the edge of the network. It offers a distributed viewpoint on ongoing attacks. Each node is autonomous, capable of local data processing and storage, and can run specific network sensors. The platform also supports the running of distributed algorithms directly in the nodes, thus providing a framework for federated learning of attacking patterns.

Currently, the infrastructure hosts three types of sensors:

- **Telescope** nodes: Monitor one-way traffic, capturing unsolicited traffic directed to unused IP address spaces. This data provides insights into global scanning activities and potential threats targeting unprotected networks.
- **Honeypots**: The honeypot modules simulate vulnerable systems to attract and record attempted attacks. The current deployment supports containers distributed by the TPot project [5]. The

collected data includes brute-force attack attempts as well as logs of shell sessions that can be used to understand attacking behavior and identify emerging attack vectors.
- **CTI Crawlers**: These crawlers gather data from dark web sources and other CTI platforms, aggregating threat intelligence indicators that are later used for enrichment and threat detection purposes.

The data processing pipeline begins at the node level, where each module gathers data on network activity, detected anomalies, and reports threat intelligence insights. After collection, the logs of the sensors undergo feature extraction, transforming raw inputs into a structured format suitable for machine learning analysis. In particular, embeddings that represent the traffic patterns of different attackers are learned directly on the nodes, relying on a federated learning version of the iDarkvec algorithm [6].

Embeddings and general traffic features are then logged into time-series representations, together with attributes such as attackers' IP addresses, timestamps, and traffic metadata. The parsed data is then centralized for use in various downstream tasks. In particular, these data have been used for learning attacking patterns and textual explanations using Large Language Models (LLMs) as well as to trigger alerts based on anomaly detection algorithms [7].

In the following, we detail one of the applications that can be built upon the data collected in this infrastructure.

3. Reward Machines for CTI

Reward machines are formal structures that extend finite state automata (FSA) by associating rewards or costs with transitions between states. They are widely used in reinforcement learning to provide structured feedback that guides decision-making processes. In the context of cybersecurity, reward machines can model the sequential nature of tasks or attacks, where each state represents a specific step in a process, and observable actions or events trigger transitions. This structured representation is particularly valuable in capturing the procedural nature of APTs, where attackers typically follow a series of well-defined stages to achieve their objectives.

State estimation within a reward machine framework plays a crucial role in CTI. Given that the steps of an attacker can be modelled as an FSA, state estimation allows defenders to infer the current phase of an ongoing attack based on observed system behaviours or detected anomalies. For example, an attacker might progress through reconnaissance, exploitation, and exfiltration, with each step corresponding to a state in the automaton. However, adversaries often obscure their activities, making it challenging to directly observe transitions. State estimation provides the means to infer these hidden states from partial or noisy observations, enabling defenders to assess the attack's progression.

The integration of reward machines into CTI enhances this process by quantifying the impact of potential defensive actions. By associating rewards with actions such as detecting, disrupting, or deceiving the attacker, reward machines help optimise response strategies. For instance, defenders can evaluate the trade-offs between immediately halting an attack versus allowing it to proceed to collect more intelligence about the adversary's objectives and techniques. This approach aligns with the goals of CTI, which seeks not only to understand and monitor threats but also to inform proactive and adaptive defence measures. Consequently, the combination of state estimation, reward machines, and CTI enables a systematic, intelligence-driven defence strategy that improves situational awareness and enhances resilience against sophisticated threats.

3.1. Basics of Reinforcement Learning

We formalize Reinforcement Learning (RL) tasks as *labelled Markov decision processes* (MDPs) [8, 9]. An MDP is defined as the tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \tau, \gamma, \mathcal{P}, \mathcal{L} \rangle$, where:

- $\mathcal{S}$ is the set of states,

- $\mathcal{A}$ is the set of actions,
- $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition probability function,
- $r : (\mathcal{S} \times \mathcal{A})^+ \times \mathcal{S} \to \mathbb{R}$ is the reward function,
- $\tau : (\mathcal{S} \times \mathcal{A}) \times \mathcal{S}^* \to \{\bot, \top\} \times \{\bot, \top\}$ is the termination function,
- $\gamma \in [0, 1)$ is the discount factor,
- $\mathcal{P}$ is a finite set of *propositions* representing high-level events,
- $\mathcal{L} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to 2^{\mathcal{P}}$ is a *(perfect) labeling function* mapping state-action-state triplets to sets of propositions.

These sets of propositions are referred to as *labels*. The transition function p is Markovian, whereas the reward function r and termination function τ may depend on the history (i.e., they are history-dependent).

Given a state-action *history* $h_t = \langle s_0, a_0, \ldots, s_t \rangle \in (\mathcal{S} \times \mathcal{A})^* \times \mathcal{S}$, we define a *trace* $\lambda_t = \langle \mathcal{L}(\emptyset, \emptyset, s_0), \ldots, \mathcal{L}(s_{t-1}, a_{t-1}, s_t) \rangle \in (2^{\mathcal{P}})^+$, which assigns labels to the triplets in h_t. The objective is to find a *policy* $\pi : (2^{\mathcal{P}})^+ \times \mathcal{S} \to \Delta(\mathcal{A})$ that maps traces and states to a probability distribution over actions, maximizing the expected cumulative discounted reward (or *return*) $R_t = \mathbb{E}_\pi \left[\sum_{k=t}^{n} \gamma^{k-t} r(h_k) \right]$, where n is the final step of the episode. To achieve this, traces must accurately represent histories, since the reward and termination functions may depend on traces rather than just the current state.

The agent-environment interaction proceeds as follows. At time t, with trace $\lambda_t \in (2^{\mathcal{P}})^+$, the agent observes the tuple $\langle s_t, s_t^T, s_t^G \rangle$, where $s_t \in \mathcal{S}$ is the current state, $s_t^T \in \{\bot, \top\}$ indicates if the history is terminal, and $s_t^G \in \{\bot, \top\}$ indicates if the task's goal has been achieved. Both s_t^T and s_t^G are determined by the termination function τ. The agent also observes a label $L_t = \mathcal{L}(s_{t-1}, a_{t-1}, s_t)$. If the history is non-terminal, the agent selects an action $a_t \in \mathcal{A}$, and the environment transitions to state $s_{t+1} \sim p(\cdot \mid s_t, a_t)$. The agent then observes the new tuple $\langle s_{t+1}, s_{t+1}^T, s_{t+1}^G \rangle$ and label L_{t+1}, updates the trace as $\lambda_{t+1} = \lambda_t \oplus L_{t+1}$, and receives reward r_{t+1}. A trace λ_t is a *goal trace* if $\langle s_t^T, s_t^G \rangle = \langle \top, \top \rangle$, a *dead-end trace* if $\langle s_t^T, s_t^G \rangle = \langle \top, \bot \rangle$, and an *incomplete trace* if $s_t^T = \bot$.

Learning policies over entire histories or traces is impractical due to their potentially unbounded length. Therefore, we use *reward machines* to encode traces, succinctly facilitating efficient policy learning.

3.2. Reward Machines

A *reward machine* (RM) [10, 11] is a finite-state representation of a reward function. Formally, an RM is a tuple $M = \langle U, \mathcal{P}, \delta_u, \delta_r, u_0, u_A, u_R \rangle$, where:

- U is a set of states,
- $\mathcal{P}$ is the set of propositions (alphabet),
- $\delta_u : U \times 2^{\mathcal{P}} \to U$ is the state-transition function,
- $\delta_r : U \times U \to \mathbb{R}$ is the reward-transition function,
- $u_0 \in U$ is the initial state,
- $u_A \in U$ is the accepting state,
- $u_R \in U$ is the rejecting state.

Reward machines are used during agent-environment interactions. Starting from u_0, the agent transitions through RM states according to δ_u and receives rewards via δ_r. Given an RM M and a trace $\lambda = \langle L_0, \ldots, L_n \rangle$, a *traversal* $M(\lambda) = \langle v_0, v_1, \ldots, v_{n+1} \rangle$ is the sequence of RM states where (i) $v_0 = u_0$, and (ii) $v_{i+1} = \delta_u(v_i, L_i)$ for $i = 0, \ldots, n$. Traversals for goal and dead-end traces should end in u_A and u_R, respectively; incomplete traces end elsewhere.

Reward machines provide compact representations of traces by encoding different task completion stages within RM states. As a result, when rewards are defined over $\mathcal{S} \times U$, they become Markovian. Based on this, [11] propose an algorithm that learns an action-value function (Q-function) over $\mathcal{S} \times U$, estimating the expected return after taking an action from a given state.

Given a transition from state s to s' with action a and label $L = \mathcal{L}(s, a, s')$, the Q-function $q : \mathcal{S} \times U \times \mathcal{A} \to \mathbb{R}$ is updated as:

$$q(s, u, a) \overset{\alpha}{\leftarrow} \delta_r(u, u') + \gamma \max_{a' \in \mathcal{A}} q(s', u', a'), \tag{1}$$

where $u' = \delta_u(u, L)$, and $x \overset{\alpha}{\leftarrow} y$ denotes $x \leftarrow x + \alpha(y - x)$. In the tabular setting, where estimates are maintained for each state-action pair, this algorithm converges to the optimal policy in the limit [11, Theorem 4.1].

3.3. Reward Machine and CTI: the Importance of State Estimation

An essential aspect of leveraging RMs in reinforcement learning is the ability to associate observed traces with the current RM state. This association is particularly critical when only partial traces are available, as in real-world scenarios where agents operate under incomplete or noisy observations. By learning a mapping between partial traces and RM states, one can predict the most likely RM state even when the full trace is unavailable, enabling the agent to act effectively in dynamic and uncertain environments.

The idea stems from the observation that RM states encapsulate task progressions through a sequence of high-level events represented by labels [10, 11]. Each RM state corresponds to a specific configuration of task completion, and transitions between states are governed by the labels observed in the environment. By training a model to infer the current RM state from observed labels and partial trace information, we can enable agents to generalise across similar tasks and handle ambiguous or missing data.

Such learning approaches could involve supervised learning methods where a dataset of traces and corresponding RM states is used to train a classifier or regressor. Alternatively, reinforcement learning agents can integrate trace-to-state prediction into their policy optimisation process, using predictions to guide decisions in real time. This capability enables the agent to navigate complex environments with non-Markovian rewards while maintaining a compact representation of task progress.

Learning trace-to-state associations also facilitates efficient state estimation in partially observable environments. In particular, several complex cyber attacks can be modelled as reward machines, where each state represents a distinct stage of the attack, and transitions between states correspond to specific adversarial actions or system events. For instance, the ultimate reward for an attacker might be the successful exfiltration of sensitive information, with intermediate states corresponding to reconnaissance, exploitation, lateral movement, and persistence. This structured representation captures the sequential and goal-driven nature of APTs, where the attacker progresses systematically through the attack stages.

By analysing the traces left behind by the attacker — such as logs, network anomalies, or other technical intelligence (TECHINT) — defenders can infer both the occurrence of an attack and its current stage. Each trace, composed of high-level events extracted from system logs or behavioural data, corresponds to a label in the RM framework. For example, detecting an unusual spike in network traffic might indicate a transition to a data exfiltration state. Similarly, identifying a previously unknown process running on a critical server might signal lateral movement.

Mapping these traces to RM states enables defenders to reconstruct the adversary's progression, providing actionable intelligence to predict their next moves. This approach allows cybersecurity teams to transition from reactive measures, such as responding to detected anomalies, to proactive strategies that anticipate and mitigate potential threats. Additionally, the RM framework offers a compact and interpretable model of the attack process, facilitating both real-time analysis and retrospective investigations to strengthen overall cyber resilience.

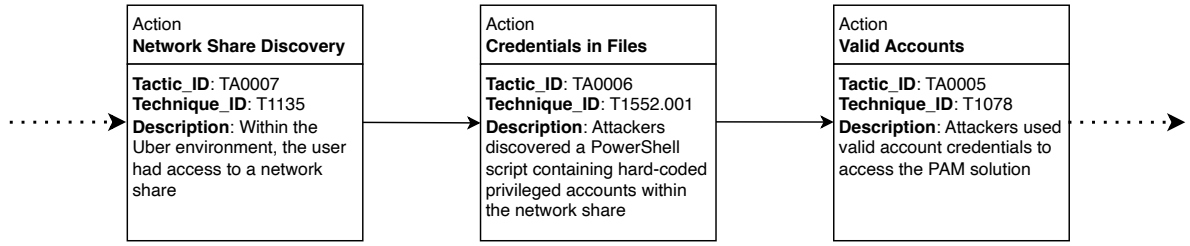

Figure 2: Partial description in MITRE ATT&CK Flow of the Uber Breach.

4. The Uber Breach by Lapsus$

In September 2022, Uber Technologies Inc. experienced a cybersecurity breach attributed to the Lapsus$ hacking group. The attackers gained access by compromising the credentials of an external contractor, employing a technique known as *MFA fatigue* or *MFA bombing*. This method involves inundating the target with multiple multi-factor authentication (MFA) requests until one is approved, thereby granting the attacker access.[1]

Once inside Uber's network, the intruders accessed several internal systems, including G-Suite and Slack. They posted a message on a company-wide Slack channel and altered Uber's OpenDNS to display a graphic image on some internal sites. However, Uber reported no evidence of access to production systems that store sensitive user information, such as personal and financial data.[2]

Uber's investigation, conducted in collaboration with the FBI and the U.S. Department of Justice, concluded that the attackers were affiliated with Lapsus$, a group known for targeting technology companies.

4.1. The Attack Analysed using MITRE ATT&CK Flow

The MITRE ATT&CK Flow[3] is a structured framework designed to visualise systematically and model adversary behaviours and attack sequences. It allows security teams to document, analyse, and communicate the progression of attacks using a flowchart-like representation, connecting individual tactics, techniques, and procedures (TTPs) from the MITRE ATT&CK knowledge base. By mapping these sequences, organisations can better understand how an adversary moves through an attack lifecycle, identify potential defence gaps, and improve detection and response strategies. The framework enhances situational awareness, enabling more robust security postures and collaborative threat analysis.

Figure 2 describes the central part of the Uber breach. Once gained access through the external contractor identity, the attackers had access to network shares. They discovered a PowerShell script containing hard-coded privileged account credentials, which could be used to access the Privileged Access Manager (PAM) solution. That allowed them to access the entire communication network in Uber.

These three steps can thus be described as a small finite-state machine that can be analysed in a hard-coded simulator to assess the possibility of learning the state of the attack from observations.

4.2. From MITRE ATT&CK Flow to a Reward Machine

To prove the viability of detecting the attack's progression based on observable events, we define a three-state reward machine to represent the attack sequence, shown in figure 3.

State 0 (Initial Access): The attacker has gained initial access through the compromised external contractor's credentials.

[1] https://center-for-threat-informed-defense.github.io/attack-flow/ui/?src=..%2fcorpus%2fUber%20Breach.afb (on 27 November 2024).
[2] https://www.uber.com/en-NO/newsroom/security-update/ (on 27 November 2024).
[3] https://center-for-threat-informed-defense.github.io/attack-flow/ (on 27 November 2024).

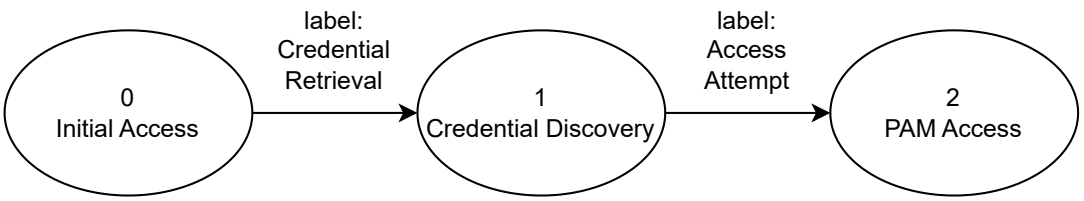

Figure 3: Reward machine matching the partial description of the Uber Breach.

State 1 (Credential Discovery): The attacker has discovered the PowerShell script containing hard-coded privileged account credentials.

State 2 (PAM Access): The attacker successfully accessed the PAM system using the discovered credentials. This state represents the successful completion of the modelled attack sequence.

A crucial component of this model is the labelling function, which maps *observable events* to *labels*. These labels provide evidence that a transition between states has occurred. They could be derived from system logs analysed through LLMs, as explained in Section 2. In this case, we define two possible labels:

Credential Retrieval: This label corresponds to successfully retrieving or accessing the hard-coded credentials from the PowerShell script and thus triggers the transition from state 0 to state 1 in Fig. 3. Evidence for this could be found in logs showing the execution of the PowerShell script or access to the file containing the credentials.

PAM Access Attempt: This label corresponds to an attempt to access the PAM system, triggering the transition from state 1 to state 2 in Fig. 3. Logs showing authentication attempts against the PAM system would trigger this label.

Thus, the labelling function connects the reward machine's abstract state transitions to concrete, observable events within the system logs, providing evidence that the state transitions have occurred.

4.3. Simulated Environment, Trace Collection and State Estimation

In the simulated environment, the nodes represent various devices or systems within a network, and the goal of the agent representative of the attacker is to navigate through the network, probe for vulnerabilities, and access critical credentials in a particular node to access the PAM solution of the system.

The actions the agent can perform in this environment are probing a node and attempting to retrieve credentials by exploiting a vulnerability. Probing involves scanning a node to gather information, such as identifying vulnerabilities or obtaining credentials. Attempting to retrieve credentials occurs when the agent exploits a discovered vulnerability to gain access to sensitive information.

The traces are collected by recording the agent's state, the action taken, the labeling function activated, and the resulting state of the attack at each time step during every try of the agent's training. After collecting the initial traces, we removed the first 20% of the data to avoid making the dataset too biased by observations from the purely exploratory phase.

The agent's state corresponds to the last node it probed, with an additional flag indicating whether it has just attempted to retrieve credentials (without indicating whether the credentials were successfully retrieved). The other information about the environment's state is derived from the labeling function.

We implemented two neural network architectures: a Multi-Layer Perceptron (MLP) and a Long Short-Term Memory (LSTM) network. The MLP is a simple feedforward neural network that processes fixed-size input data, while the LSTM is a type of recurrent neural network designed to handle sequential

Model	Window Size	Accuracy	Precision	Recall	F1
LSTM	1	0.941	0.944	0.941	0.935
	2	0.968	0.967	0.968	0.967
	3	1.000	0.999	1.000	0.999
	4	1.000	0.999	1.000	0.999
	5	1.000	0.999	1.000	0.999
MLP	1	0.887	0.882	0.887	0.884
	2	0.874	0.817	0.874	0.845
	3	0.952	0.952	0.952	0.952
	4	0.920	0.918	0.920	0.918
	5	0.920	0.924	0.920	0.921

Table 1
Performance of MLP and LSTM models in prediction of attack state for varying rolling window sizes

data by capturing temporal dependencies. For the models we used, the input consists of the agent's state and the labeling function. For each time step, we considered a rolling window of the previous n observations, where n can range from 1 to 5. The rolling window ensures that the models have access to a broader context, while ensuring that the model does not have access to too much of the past history which could introduce unnecessary complexity or noise.

The MLP model consists of three fully connected layers: the first layer transforms the input features into a higher-dimensional space, the second layer reduces the dimensionality of the features to half the size of the previous layer and the final layer maps the processed features to the output space. Each of the first two layers is followed by a ReLU activation function to introduce non-linearity and enable the network to model complex relationships. The last layer outputs predictions without any activation.

The LSTM model consists of two components: an LSTM layer that processes input sequences and outputs a sequence of hidden states. This layer is capable of capturing both short-term and long-term dependencies in the data. A dropout mechanism is included within the LSTM to reduce overfitting during training and a fully connected layer that maps the hidden states from the LSTM to the output space. The output of the fully connected layer is passed through a softplus activation function to ensure smooth, non-negative predictions.

Table 1 summarizes the results for both models, measured using accuracy, precision, recall and F1-score metrics. The results demonstrate how each model's performance varies with different window sizes, reflecting the impact of this parameter on the models' ability to process and analyze the input data.

5. Conclusions

This paper has demonstrated how the ACRE framework can address the challenges posed by Advanced Persistent Threats by providing effective tools for data collection, processing, and foresight generation. Using the Uber data breach as a case study, we showed how ACRE enhances the Cyber Threat Intelligence process by systematically analysing threats during the later stages of the cyber kill chain. The framework's ability to integrate and structure raw data enables the extraction of actionable intelligence, supporting organisations in identifying complex attack patterns and mitigating risks proactively.

This research is important because it contributes to bridging the gap between traditional CTI methodologies and modern advancements in threat analysis. By leveraging tools such as ACRE, organisations can process large-scale threat data efficiently and anticipate and respond to adversarial strategies in a

timely manner. Integrating predictive foresight with actionable intelligence is critical in addressing the growing sophistication of cyber threats, enabling a shift from reactive to proactive cybersecurity postures.

Future work will explore the automatic learning of reward machines to enhance the ACRE framework's capabilities further [12]. As part of neuro-symbolic reinforcement learning, reward machines offer a structured means of modelling adversarial behaviours and generating dynamic threat responses. By incorporating automated learning mechanisms, ACRE can adapt to evolving APT tactics in real-time, enabling more effective modelling of complex threat environments.

In addition, we aim to leverage the capabilities of LLMs to strengthen the ACRE framework further. LLMs provide a powerful foundation for extracting meaningful semantics from unstructured and semi-structured threat intelligence data, automating parts of the CTI process such as data enrichment, pattern recognition, and hypothesis generation. By connecting LLMs with the learning mechanisms of reward machines, we can create an adaptive pipeline that processes and interprets large-scale threat data and generates symbolic representations of adversarial behaviours. This integration will enable ACRE to refine its predictive foresight capabilities, facilitating the identification of novel attack patterns and supporting the development of proactive defence strategies.

Combining neuro-symbolic approaches with LLMs' natural language processing abilities will position ACRE as a scalable and adaptive solution for modern cybersecurity challenges, capable of addressing the complexity and volume of data inherent in today's threat landscape.

Acknowledgments

This project was partially funded by the Italian Ministry of University as part of the PRIN: PROGETTI DI RICERCA DI RILEVANTE INTERESSE NAZIONALE – Bando 2022, Prot. 2022EP2L7H This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU, specifically by the project NEACD: Neurosymbolic Enhanced Active Cyber Defence (CUP J33C22002810001). The research reported in this paper was sponsored in part by the DEVCOM Army Research Laboratory via cooperative agreement W911NF2220243. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States government.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. Alshamrani, S. Myneni, A. Chowdhary, D. Huang, A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities, IEEE Communications Surveys & Tutorials 21 (2019) 1851–1877.

[2] Y. Wang, H. Liu, Z. Li, Z. Su, J. Li, Combating advanced persistent threats: Challenges and solutions, IEEE Network (2024).

[3] E. M. Hutchins, M. J. Cloppert, R. M. Amin, Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains, Lockheed Martin Corporation 1 (2011) 1–21.

[4] P. Pirolli, S. Card, Sensemaking processes of intelligence analysts and possible leverage points as identified through cognitive task analysis, Proceedings of the 2005 International Conference on Intelligence Analysis (2005) 1–6.

[5] TPot, The all in one honeypot platform, 2024. URL: https://github.com/telekom-security/tpotce.

[6] L. Gioacchini, L. Vassio, M. Mellia, I. Drago, Z. B. Houidi, D. Rossi, i-darkvec: Incremental embeddings for darknet traffic analysis, ACM Trans. Internet Technol. 23 (2023). URL: https://doi.org/10.1145/3595378. doi:10.1145/3595378.

[7] M. Boffa, I. Drago, M. Mellia, L. Vassio, D. Giordano, R. Valentim, Z. B. Houidi, Logprécis: Unleashing language models for automated malicious log analysis: Précis: A concise summary of essential points, statements, or facts, Computers & Security 141 (2024) 103805. URL: https://www.sciencedirect.com/science/article/pii/S0167404824001068. doi:https://doi.org/10.1016/j.cose.2024.103805.

[8] J. Fu, U. Topcu, Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints, in: Proceedings of the 10th Robotics: Science and Systems Conference (RSS), 2014.

[9] D. Furelos-Blanco, M. Law, A. Jonsson, K. Broda, A. Russo, Hierarchies of Reward Machines, in: Proceedings of the 40th International Conference on Machine Learning (ICML), 2023, pp. 10494–10541.

[10] R. Toro Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning, in: Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 2112–2121.

[11] R. Toro Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning, Journal of Artificial Intelligence Research 73 (2022) 173–208.

[12] R. Parac, L. Nodari, L. Ardon, D. Furelos-Blanco, F. Cerutti, A. Russo, Learning robust reward machines from noisy labels, in: Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning (KR 2024), 2024. Preprint available at arXiv:2408.14871.