DT-Leak: Process Mining-Based Behavioral Detection of Data Exfiltration

Ailton Cordeiro[1,3], Geilson Silva[1], Fernando Aires [1,2], Wellison R. M. Santos [1,2], Ricardo Lima[3], Milton Lima[1]

[1]Centro Integrado de Segurança em Sistemas Avançados (CISSA), CESAR School, Recife – PE – Brazil

[2]Departamento de Computação, Universidade Federal Rural de Pernambuco (UFRPE), Recife – PE – Brazil

[3]Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife – PE – Brazil

{asc2,gns,wrms,mvml}@cesar.org.br, fernandoaires@ufrpe.br, rmfl@cin.ufpe.br

Abstract. *The increasing sophistication of cyber threats, particularly Advanced Persistent Threats (APTs) focused on data exfiltration, has challenged the effectiveness of traditional perimeter-based and signature-based defenses. Conventional approaches rely heavily on tools such as Security Information and Event Management (SIEM), Intrusion Detection Systems (IDS), and event correlation to identify malicious activities based on isolated Indicators of Compromise (IoCs). However, these methods often fail to detect 'low and slow' or 'live off the land' behavioral sequences that mimic legitimate protocols, due to the lack of systematic mechanisms to transform fragmented log telemetry into verifiable behavioral models. To bridge this gap, this research proposes DT-Leak, a process mining-based approach to discover and analyze the behavioral dynamics of data exfiltration in corporate environments. The solution employs semantic abstraction strategies to mitigate the complexity of raw logs. This work offers a new analytical perspective that views cyberattacks as structured, discoverable processes, providing a formal framework to convert security telemetry into actionable behavioral intelligence. Experimental results demonstrate that the Heuristics Miner achieved the best balance between behavioral fidelity and model interpretability.*

1. Introduction

The society is experiencing an era of unprecedented digitalization, in which the ability to generate, process, and exchange data has become the fundamental engine of the global economy [OECD 2020]. In this information age, data are not merely operational assets but also an organization's most valuable resource, driving corporate efficiency and technological innovation [Schwab 2016].

Despite advances in protection technologies, the market often prioritizes convenience, agility, and price over security, which is rarely integrated natively into the design of applications and infrastructures [Silva et al. 2025]. This scenario of structural fragility has paved the way for the evolution of threats. What were once massive

and noisy attacks have evolved into Advanced Persistent Threats (APTs): targeted cyber campaigns, highly sophisticated and designed to compromise high-value systems [Buchta et al. 2024]. Unlike conventional attacks, APTs operate with a high degree of stealth, seeking to remain undetected for prolonged periods to achieve espionage or sabotage goals [Buchta et al. 2024].

The invisibility of silent exfiltration in *Windows* systems is exacerbated by the fact that victims often do not realize their assets have been compromised until significant and irreversible damage occurs, with average dwell times (*dwell times*) exceeding several months [Alshamrani et al. 2019]. Unlike noisy attacks, the exfiltration process is conducted in a manner that does not generate clear signatures, simulating the behavior of legitimate traffic through common network protocols such as Simple Mail Transfer Protocol (SMTP), Hypertext Transfer Protocol (HTTP), and Domain Name System (DNS) [Pratap Singh and Afzal 2024, ATT&CK 2024].

Considering the context of sophisticated Advanced Persistent Threat (APT) campaigns and the previously discussed limitations of current solutions, this research aims to address the following problem: the inability of traditional approaches based on isolated events to detect complex, procedural data exfiltration patterns in corporate environments.

This work aims to propose, develop, and validate DT-Leak (**D**etection **T**ransfer **Leak**age), a solution grounded in Process Mining capable of discovering, modeling, and analyzing the behavioral dynamics of data exfiltration in *Windows* systems within corporate environments. The intent is not to create a rule-based or signature-based tool, but to explore a distinct analytical paradigm: the understanding of an attack as a dynamic process, composed of deviations, repetitions, and conditional choices, which can be formally discovered and examined.

This work also contributes across two interrelated dimensions:

- In the theoretical-conceptual dimension, it is expected to advance knowledge at the frontier between process mining and cybersecurity, offering a new perspective for APT analysis that transcends the fragmented view based on Indicators of Compromise (IoCs) and embraces a procedural and holistic understanding of the attack.
- In the methodological dimension, the proposal of the DT-Leakmethod, including its design decisions, event abstraction criteria, and strategies for integrating multiple *log* sources, constitutes a contribution that can be replicated and adapted by other researchers in different threat detection contexts.

The remainder of this paper is organized as follows: Section 2 presents the related work. Section 3 introduces the theoretical foundations and fundamental concepts. Section 4 details the proposed DT-Leak approach. Section 5 presents the experimental evaluation and results. Finally, Section 6 concludes the paper with a discussion of the findings and future research directions.

2. Related Work

The application of process mining to the security domain has evolved from theoretical foundations to specialized forensic and threat detection frameworks.

[Van der Aalst and De Medeiros 2005] pioneered this field by demonstrating how the α-*algorithm* could detect anomalies in audit trails, establishing the feasibility of using process discovery for intrusion detection. However, their foundational approach did not address the high granularity and lack of explicit case identifiers prevalent in modern security *logs*.

Recent algorithmic advancements have sought to tackle log complexity. [Augusto et al. 2022] analyzed techniques such as Theory of Regions, *Split Miner*, and *Log Skeleton Miner*. While these offer high precision and handle noise better than basic algorithms, they often struggle with generalization or face high computational complexity when applied to large-scale behavioral data.

Complementing these imperative models, [Di Ciccio and Montali 2022] explored declarative mining using the *Declare* language. Although this provides flexibility in open execution spaces, it is limited by predefined *templates* and may lead to undecidability when incorporating temporal or data perspectives.

In the context of specific threats, several studies have adapted these tools for cyber defense. [Macak et al. 2020] utilized conformance checking to identify insider threats in corporate environments, while [Myers et al. 2018] successfully applied the *Inductive* and *Heuristics Miners* to detect control-flow anomalies in Industrial Control Systems (ICS).

Bridging the gap between raw telemetry and high-level semantics, [Rodríguez et al. 2024] utilized the MITRE ATT&CK *framework* to profile ransomware behavior. Although their semantic lifting is highly relevant, the lack of a systematic activity grouping stage can lead to excessive model variability. Data preparation is central to recent literature. [Adila et al. 2026] addressed the absence of case identifiers by using episode mining to reconstruct incidents, highlighting the necessity of intelligent pre-processing.

Similarly, [Silva et al. 2025] focused on detecting APTs and lateral movement in *Windows* environments using a structured pipeline (*Windows Process Monitor* and *LogsToXes*). While they successfully identified Indicators of Compromise (IoCs) using the *Alpha Miner*, their approach lacked semantic abstraction and formal quality metrics like *fitness*.The DT-Leak extends these efforts by addressing identified limitations.

In light of these gaps, DT-Leak differentiates itself from prior efforts on three fronts: it introduces a systematic semantic abstraction strategy that removes volatile identifiers and consolidates fragmented activities under stable semantic keys, addressing the variability gap noted in [Rodríguez et al. 2024] and [Silva et al. 2025]; it specifically targets the data exfiltration phase of APT campaigns, complementing prior work focused on insider threats [Macak et al. 2020], ICS anomalies [Myers et al. 2018], ransomware [Rodríguez et al. 2024], or lateral movement [Silva et al. 2025]; and, unlike [Silva et al. 2025], it evaluates three discovery algorithms (*Inductive*, *Alpha*, and *Heuristics Miner*) using formal process mining quality metrics (fitness, precision, generalization, and simplicity), also releasing an annotated dataset of traces to support reproducibility.

3. Background

Advanced Persistent Threats (APTs) are targeted and highly sophisticated cyber campaigns designed to compromise high-value systems for the purposes of espionage or

sabotage [Krishnapriya and Singh 2024]. Unlike conventional attacks, APTs are conceived to remain undetected for prolonged periods, requiring a high degree of stealth [Buchta et al. 2024]. Operationally, these characteristics manifest through a life cycle structured into five distinct phases[1] [Alshamrani et al. 2019]:

The cycle begins with **reconnaissance**, which involves the identification of target vulnerabilities and high-value individuals. This is followed by **establishing a foothold**, where attackers obtain initial access through techniques such as phishing. Once inside, they execute **lateral movement** to expand control and privileges within the network. The operation then proceeds to **exfiltration**, consisting of the collection and transfer of sensitive data out of the target infrastructure. Finally, the **post-exfiltration/impact** phase encompasses the removal of traces and the deployment of backdoors to ensure future access.

3.1. Data Exfiltration

In this stage, the attacker uses various techniques to mask data egress, seeking to blend malicious activities with legitimate network traffic:

- **Use of Legitimate Protocols**: Use of protocols such as File Transfer Protocol (FTP), Hypertext Transfer Protocol Secure (HTTPS), SMTP, or DNS to encapsulate exfiltrated data, hindering inspection by traditional security tools [Pratap Singh and Afzal 2024, ATT&CK 2024].
- **Compression and Encryption**: To minimize traffic volume and, consequently, the probability of detection by systems that monitor transfer spikes, data are often compressed before transmission [Lajevardi and Amini 2021].
- **Live off the Land (LoL)**: A particularly stealthy approach involves the use of legitimate processes already present on the target system for data collection and transmission [Krishnapriya and Singh 2024].
- **Low and Slow and Periodic Exfiltration Strategies**: To avoid alarms based on volume thresholds, attackers often fragment exfiltration over time, transferring small amounts of data at regular intervals (T1029 - MITRE ATT&CK) [Lajevardi and Amini 2021].

Figure 1 schematically illustrates an APT operation, from the initial phases to exfiltration and its repetition, contextualizing the stages presented.

3.2. Process Mining

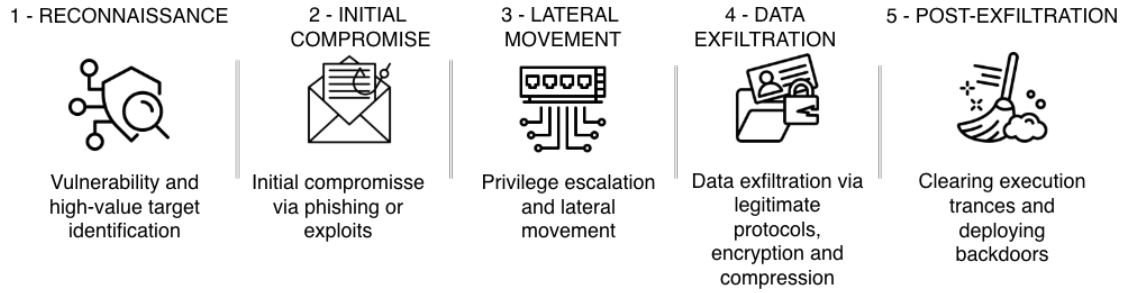
Process Mining is a discipline that emerges at the intersection of data science and process science, using event logs generated by information systems as a starting point [van der Aalst 2016]. It consists of a set of methods for extracting structured knowledge, enabling the analysis of end-to-end processes based on actual execution data.

3.2.1. Mining Techniques

There are three fundamental types of process mining [van der Aalst and Carmona 2022]:

[1]More recent methodologies propose an APT taxonomy composed of up to 14 main objectives, allowing for greater analytical detail

Figure 1. Structured flow of an APT attack with the consolidation of data exfiltration [Singh and Afzal 2024].



- **Process discovery**: Automatic construction of a process model from logs, without prior knowledge of internal structures.
- **Conformance checking**: Evaluation of the degree of alignment between a reference model and the actual behavior recorded in the logs to detect deviations or anomalous patterns.
- **Enhancement**: Extension or improvement of an existing model based on information from the actual process.

3.2.2. Process Discovery Algorithms

Discovery algorithms build formal models from event logs, representing different methodological strategies:

- **Alpha Miner [van der Aalst et al. 2004]**: The foundational approach, identifying ordering relations (succession, causality, parallelism, and choice), but highly sensitive to noise and complex constructs such as short loops.
- **Heuristics Miner [Weijters et al. 2006]**: Uses frequency-based dependency measures robust to real-world noise, applying configurable thresholds to filter weak relations into a Heuristic Net.
- **Inductive Miner [Leemans et al. 2013]**: Employs a divide-and-conquer strategy over the Directly-Follows Graph (DFG), guaranteeing *sound*, *deadlock*-free models while maintaining high *fitness* in noisy contexts.

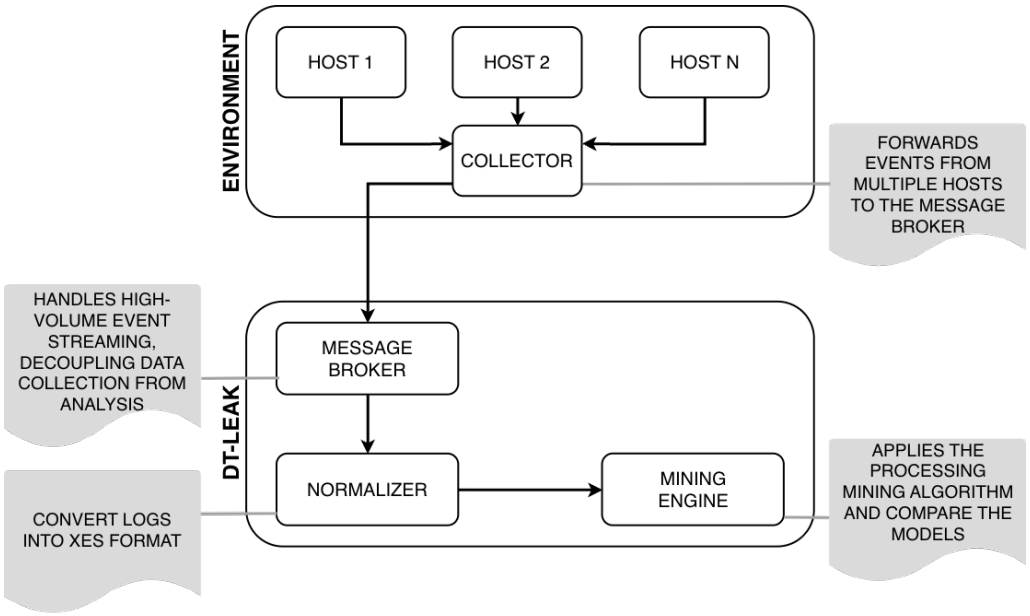
4. DT-Leak (Detection Transfer Leakage)

DT-Leak is a solution for identifying and detecting behavioral patterns in data exfiltration. It implements a processing workflow based on telemetry components and utilizes process mining algorithms as a core element for discovering and modeling malicious activities patterns. Figure 2 presents an overview of the elements directly involved in the detection process. These elements consist of two groups: the *Environment*, which includes the monitored infrastructure; and DT-Leak, which encompasses the processing and mining architecture.

4.1. Environmment

The Environment represents the data source domain and the scenario in which the activities of interest are executed. It comprises elements that generate the telemetry required to

Figure 2. Overview of the proposed architecture for the DT-Leak.



feed the detection *pipeline*. The environment is structured into two fundamental components: the *Hosts* and the Collector.

The *hosts* consist of the workstations or servers where user interactions and system task processing occur. They act as the primary source of events, providing the necessary visibility into operations that may characterize exfiltration behaviors.

The Collector is the component responsible for monitoring the environment and instrumenting the target system in real time. Its primary function is to observe and extract information regarding critical activities, such as process creation, file system access, and network communication. This component ensures that raw actions executed on the Hosts are transformed into structured records, guaranteeing that telemetry is properly forwarded to the subsequent stages of the analysis pipeline.

In the experimental setup of DT-Leak, the environment was instantiated on the *Windows 11* operating system. This choice is justified by the predominance of this platform in corporate infrastructures, as documented in recent threat intelligence reports [European Union Agency for Cybersecurity (ENISA) 2025, CrowdStrike 2026], making it a relevant scenario for studying data exfiltration patterns. The collector functions were implemented using *Sysmon*[2] as the instrumentation agent and *Winlogbeat*[3] as the forwarder.

Sysmon was selected for its forensic-level detail within the chosen environment, while Winlogbeat ensures efficient data forwarding. It is emphasized that the architecture is agnostic to these tools. Any other set of environments or collection and transport agents offering equivalent capabilities can be used without compromising the integrity of the detection flow.

[2]https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon
[3]https://www.elastic.co/en-us/beats/winlogbeat

4.2. Message Broker

The Message Broker is an asynchronous *middleware* that decouples event production from consumption, allowing monitoring agents to transmit records independently of the *pipeline's* immediate processing capacity and ensuring continuity of telemetry capture under high demand. In the experimental setup, Apache Kafka[4] was used for its durable persistence and guaranteed, ordered delivery, although the architecture is tool-agnostic: any messaging service offering equivalent guarantees can be integrated without compromising the pipeline.

4.3. Normalizer

The Normalizer is the component responsible for converting the structured telemetry from the Environment, stored in the Message Bus, into standardized records with consistent attributes. Its function is to ensure information integrity and homogeneity, transforming heterogeneous *logs* into a unified structure that enables comparative analysis across multiple data streams. The normalization process operates through three fundamental logical pillars:

- **Removal of volatile identifiers**. This stage addresses label fragmentation caused by dynamic elements that vary across executions. The component applies substitution rules to remove volatile data, which, although technically unique, are semantically identical and turn semantically identical activities into distinct records.
- **Naming structuring and *templates***. After cleaning the volatile data, each activity is renamed following a structured *template* that preserves the host context and the relationships between processes. The standardized naming organizes the information in the format [hostname] Action_Type: source_process $\rightarrow$ destination_process, allowing a clear distinction of whether the activity occurred on the source or destination host and the causal relationship between the involved entities.
- **Synchronization and attribute mapping**. The remaining attributes are mapped to a common schema, ensuring that fundamental fields, such as system timestamps and category identifiers, are unified.

4.4. Mining Engine

The Mining Engine is the central analytical component of DT-Leak, responsible for discovering formal process models from consolidated event *logs* and extracting quantitative metrics that characterize the observed behavior. This component operates on two complementary fronts: process discovery and the behavioral analysis of executions.

- **Performance perspective:** Evaluates temporal patterns by measuring the average duration between consecutive activities. This perspective is fundamental for identifying execution speed and differentiating manual operations from automated processes.
- **Frequency perspective:** Identifies dominant execution paths by counting the recurrence of transitions between activities.

For the implementation of this analytical layer, the PM4Py library [Berti et al. 2019] was utilized, selected for offering reference implementations of process mining algorithms and support for large-scale *log* manipulation.

[4]https://kafka.apache.org/

5. Experiment

This section describes the experimental evaluation of DT-Leak in detecting data exfiltration. The study utilizes real telemetry data from a controlled environment to assess the effectiveness of process mining in identifying malicious behavior.

To methodologically organize the analysis, the steps adopted are inspired by the recommendations of Jain [Jain 1990]: 1) define the evaluation objective; 2) establish the scenario; 3) select metrics; 4) design the evaluation; and 5) analyze and present the results.

5.1. Evaluation Objective

The objective of this evaluation is to analyze the capability of DT-Leak to model exfiltration behaviors in corporate *Windows 11* environments. To this end, the following specific goals are pursued:

- **Experimental validation:** Verify the technical feasibility of the proposed architecture, integrating attack emulation, structured *log* collection, and analytical processing.
- **Algorithm evaluation:** Analyze the ability of different process discovery algorithms to model exfiltration behaviors.

5.2. The Scenario

The experimental scenario, illustrated in Figure 3, was designed to simulate a typical corporate infrastructure, structured into three main environments that interact to reproduce the complete cycle of an exfiltration attack: the *Attack Environment*, the *Victim Environment*, and the *Destination Infrastructure*.

Attack Environment consists of a Kali Linux 2025.4 machine running an automated orchestrator that emulates *malware* behavior. This orchestrator is responsible for initiating and controlling the attack phases against the target.

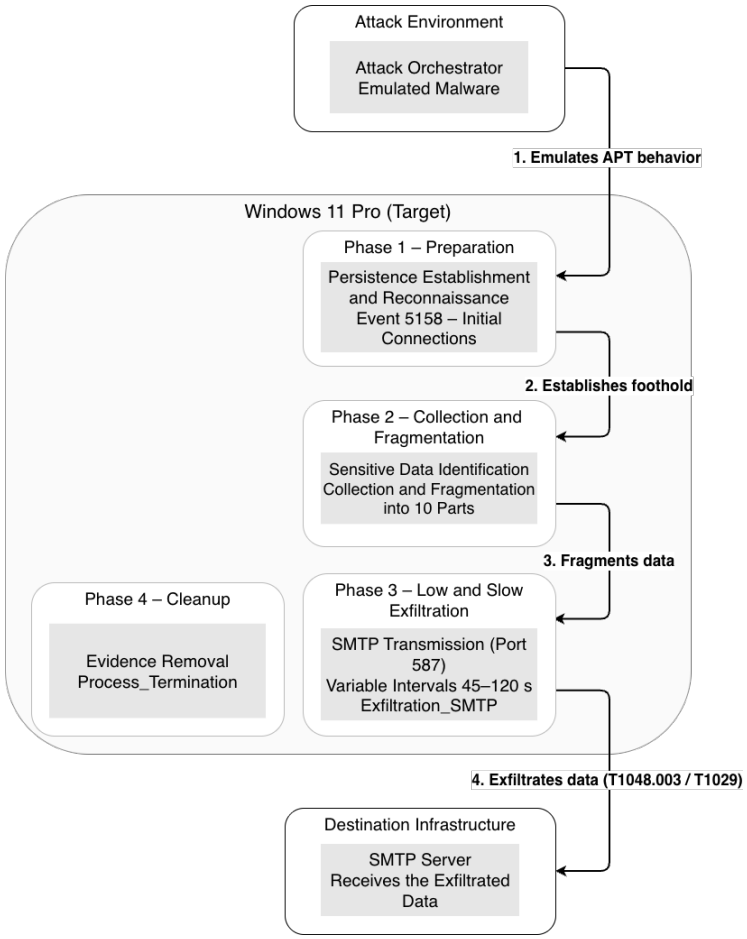
Victim Environment is a *Windows 11 Pro* virtual machine that executes the four sequential phases of the exfiltration attack:

1. **Preparation:** The orchestrator establishes persistence in the system and performs reconnaissance, generating initial network connection events, typically recorded as *Event 5158*.
2. **Collection and Fragmentation:** Sensitive data are identified, collected, and fragmented into 10 parts. This fragmentation aims to evade detection by security systems based on traffic volume.
3. **Low and Slow Exfiltration:** The fragments are sent to the destination server via the SMTP protocol on port 587, using variable intervals between 45 and 120 seconds. This approach characterizes the *low and slow* technique (T1029 and T1048.003 – MITRE ATT&CK), making temporal event correlation more difficult.
4. **Cleanup:** After the transfer, the malicious agent is terminated, generating *Process_Termination* events to remove forensic evidence from the system.

The Destination Infrastructure consists of an SMTP server specifically configured to receive and log exfiltrated data, enabling validation of the simulated attack's effectiveness. The data generated in the experiment are available in an anonymized repository[5].

[5]https://anonymous.4open.science/r/dt-leak-data-2926/

Figure 3. Experimental attack scenario, illustrating the attack, victim, and destination environments.



5.3. Evaluation Metrics

The quality of the discovered process models was evaluated using five fundamental process mining metrics, as defined by [Buijs et al. 2012] and adapted to the context of data exfiltration.

5.3.1. Fitness

Fitness measures the model's ability to replay the event sequences recorded in the *logs*. The token-based "replay" method from PM4Py was used, computed according to Equation 1:

$$\textit{Fitness} = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{n} m_i}{\sum_{i=1}^{n} c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{n} r_i}{\sum_{i=1}^{n} p_i}\right), \tag{1}$$

where, for each case i: m_i = missing tokens (expected but not reproduced); r_i = remaining tokens (produced but not expected); c_i = consumed tokens; p_i = produced tokens; and n = total number of traces in the log. Fitness ranges from 0 to 1, with values close to 1 indicating high capacity to reproduce the observed attack sequences.

5.3.2. Precision

Precision evaluates whether the model allows only behaviors actually observed in the logs [Buijs et al. 2012]. Low precision (*underfitting*) allows additional unobserved behaviors, potentially causing false positives, whereas excessive precision (*overfitting*) may fail to generalize to legitimate variations. Precision is calculated according to Equation 2:

$$\textit{Precision} = \frac{1}{|E_r|} \sum_{e \in E_r} \frac{|A_{\text{log}}(e) \cap A_{\text{model}}(e)|}{|A_{\text{model}}(e)|}, \tag{2}$$

where E_r = set of reproducible repeating events; $A_{\text{log}}(e)$ = activities observed in the log for the context of event e; and $A_{\text{model}}(e)$ = activities allowed by the model in the state reached after that context. This generalizes the *escaping edges* notion of [Buijs et al. 2012] to scenarios where an event involves multiple objects — as in data exfiltration, which simultaneously involves processes, files, and network connections.

5.3.3. Generalization

Generalization indicates the model's ability to adapt to behaviors not observed during training [Buijs et al. 2012], defined as the frequency with which each node of the process tree is visited during the "replay". It is computed according to Equation 3:

$$\textit{Generalization} = 1 - \frac{\sum_{\text{nodes}} \left(\sqrt{\text{executions}}\right)^{-1}}{\text{nodes in the tree}}, \tag{3}$$

where *executions* = number of times a given node is visited during the replay; and *nodes in the tree* = total number of nodes in the process tree.

5.3.4. Simplicity

Simplicity measures the model's structural complexity; simpler models are preferable as they ease interpretation by security analysts. It is calculated according to Equation 4:

$$\textit{Simplicity} = 1 - \frac{\#\text{duplicate activities} + \#\text{missing activities}}{\#\text{nodes in the tree} + \#\text{event classes in the log}}, \tag{4}$$

where *duplicate activities* = activities represented more than once in the model; *missing activities* = event classes present in the log but absent from the model; and the denominator sums the tree nodes and the distinct activity labels in the log. The value approaches 1 when each activity is represented exactly once; lower values indicate structural redundancy that increases complexity without adding observed behavior.

5.3.5. F-Score

The F-Score reported in this work is the harmonic mean of the four process mining quality dimensions:

$$\text{F-Score} = \frac{4}{\frac{1}{\text{Fitness}} + \frac{1}{\text{Precision}} + \frac{1}{\text{Generalization}} + \frac{1}{\text{Simplicity}}} \tag{5}$$

It is essential to distinguish this model-quality F-Score from the detection F1-score based on true and false positives. The former is a composite indicator that summarizes how faithfully a discovered model represents the observed behavior across the four dimensions of [Buijs et al. 2012].

5.4. Evaluation Design

In this experiment, the emulated malware Agent Tesla[6] was used as the attack agent. This choice is justified by its notoriety as an *infostealer* that utilizes the SMTP protocol on port 587 as its primary exfiltration vector [Booij 2019, MITRE ATT&CK 2024, Davidpur 2026]. As documented by the Internet Storm Center, the malware's choice of port 587 is strategic: unlike port 25 (which is frequently blocked), port 587 remains open in most corporate networks as it is intended for authenticated email submission, requiring authentication and allowing malicious traffic to blend in with legitimate communications [Booij 2019]. *Agent Tesla* allows for the simulation of APT behavior adopting a *low and slow* strategy, generating a complex sequence of events.

The attack agent was configured as an automated orchestrator, responsible for executing repetitive exfiltration cycles to ensure the statistical consistency of the data, following Jain's recommendations for experiments with statistical significance [Jain 1990]. The experiment was performed in 30 cycles, each lasting 20 minutes, following a technical protocol divided into four operational stages:

- **Baselining and reconnaissance:** The agent initiates communication with the target *host* to ensure persistence in the system and identify sensitive assets to be extracted. This phase is technically marked by the opening of initial network connections, often recorded by the system as *Event 5158*.
- **Data collection and fragmentation:** Once the files are identified, the orchestrator performs the collection and fragmentation of the data into smaller parts. This fragmentation technique is fundamental to simulate real attacks that seek to evade detection by traffic volume thresholds, creating a sequence of file and process manipulation events.
- **Periodic exfiltration (Low and Slow):** The fragments are sent via the SMTP protocol on port 587. To reduce temporal correlation, the agent uses variable transmission intervals of 45-120 seconds. This iterative behavior generates repetitive structures that are subsequently captured by DT-Leak .

[6]https://www.checkpoint.com/pt/cyber-hub/threat-prevention/what-is-malware/agent-tesla-malware/

- **Trace cleanup:** After the transfer is completed, the agent executes commands to terminate the malicious activities. This stage is characterized by the presence of the *Process_Termination* event in the *logs*, indicating the termination of processes to mitigate the generation of forensic evidence.

For the execution of the tests, the infrastructure was divided into three distinct operational domains, as detailed in Table 1:

Table 1. Detailed configuration of the *hosts* in the experiment.

Resource	Function	Technical Configuration
Target Machine	Hosts the *Windows* 11 Pro system and the collection agents (Sysmon and Winlogbeat)	*Windows* 11 Pro (64-bit), 100 GB storage, 8 GB RAM, 4 CPU cores
Attack Machine	Orchestrates the malware and receives the exfiltrated data	Kali Linux 2025.4, 10 GB storage, 4 GB RAM, 4 CPU cores
Processing Environment	Executes the analytical pipeline and the PM4Py algorithms	macOS Monterey 12.7.4, Intel i7 processor, 250 GB storage, 16 GB RAM

5.5. Results

This section presents findings from processing the collected telemetry, focusing on the characterization of the attack *logs* and the comparative performance of the discovered mining models.

5.5.1. Telemetry Characterization

The execution of the scenario generated a dataset comprising 38,182 events, 1,541 cases (traces), and 251 behavioral variants. After normalization, 77 unique activities were identified. Table 2 summarizes the volume of the main events captured in the *log*:

Table 2. Volume of the main events identified in the *log*.

Event	Occurrences	Semantic Description
Event 5158	29070	Established network connection
Network_Connection	6125	Indicator of network activity
Execution	591	Process and binary execution
Exfiltration_SMTP	392	Attempted transmission via port 587
Authentication	138	Logon attempts (success/failure)

The statistical analysis demonstrated the predominance of network-related events, particularly Event 5158 (established connections) with 29,070 occurrences, followed by the Network_Connection indicator with 6,125 occurrences. The critical event for the scenario, Exfiltration_SMTP, registered 392 occurrences, enabling the direct mapping of data transmission attempts. Regarding temporal performance, a clear distinction was observed between the employed tactics: **Persistence.** Network connection events exhibited an average duration of approximately 1 hour. **Stealthiness.** In contrast, the SMTP exfiltration event exhibited an average duration of only 2 seconds

5.5.2. Qualitative Comparison of the Models

To evaluate the effectiveness of the DT-Leak, three process mining algorithms were applied to the same event *log*. Table 3 presents the quantitative results based on the metrics.

Table 3. Comparison of quality metrics among mining algorithms.

Algorithm	Fitness	Precision	Generalization	Simplicity	F-Score
Inductive Miner	**1,0000**	0,4601	**0,7964**	0,6486	0,6699
Alpha Miner	0,0778	**0,6489**	**0,8659**	0,7647	0,2373
Heuristics Miner	0,9989	0,5638	0,7593	0,5690	**0,6838**

Regarding the quality dimensions, the Fitness analysis reveals that the *Inductive Miner* achieved the maximum score (1.0), indicating that all sequences present in the *logs* can be fully executed by the model. The *Heuristics Miner* also demonstrated excellent performance for this metric (0.9989), whereas the *Alpha Miner* proved inadequate for representing the scenario (0.0778).

With respect to Precision, the *Alpha Miner* achieved the highest score (0.6489), followed by the *Heuristics Miner* (0.5638) and the *Inductive Miner* (0.4601). These findings suggest that more complex models, such as those generated by the *Inductive Miner*, tend to allow additional unobserved behaviors (*underfitting*), whereas excessively simplified models, such as the *Alpha Miner*, may restrict legitimate behaviors by incurring *overfitting*.

Regarding Simplicity, the *Alpha Miner* stood out with a score of 0.7647 due to its minimal structure, followed by the *Inductive Miner* (0.6486) and the *Heuristics Miner* (0.5690). However, the warning from [Buijs et al. 2012] must be considered: simplicity should not be pursued in isolation, as overly simplified models may omit behaviors essential for detection.

Concerning Generalization, the *Alpha Miner* obtained the highest value (0.8659), although this result is a direct consequence of its excessively simplified structure, which allows the occurrence of several behaviors not observed in the *logs*. In contrast, the *Inductive Miner* (0.7964) and the *Heuristics Miner* (0.7593) showed balanced values, indicating good generalization to new variations of the exfiltration attack.

The convergence of the results (F-Score) indicates that DT-Leak is effective in reconstructing the attack chain, with *Heuristics Miner* consolidating itself as the most suitable model for this scenario. By achieving the highest F-Score (0.6838), this algorithm demonstrated the ability to balance near-perfect fitness (0.9989) with moderate precision, enabling the identification of causal dependencies without overwhelming analysts with structural noise. This choice stands out compared to the *Inductive Miner*, which, although faithfully reproducing the *log*, is more complex, and the *Alpha Miner*, which proved insufficient to capture the nuances of security *logs*.

The results demonstrate the feasibility of the DT-Leak for detecting data exfiltration through process mining. The *Heuristics Miner* emerged as the most suitable algorithm for this scenario, balancing the ability to represent process complexity (high fitness) with the restriction to relevant behaviors (moderate precision), resulting in the best

F-Score (0.6838).

5.5.3. Threats to Validity

The evaluation focused on APT-oriented data exfiltration scenarios in a controlled virtualized environment, which may limit the generalization of the results to other attack classes and real-world infrastructures. In addition, the approach depends on detailed telemetry collection (e.g., Sysmon and Winlogbeat) and may introduce computational overhead when processing large-scale logs.

6. Conclusions and Future Works

This work proposes a novel model for the behavioral modeling and attack-chain reconstruction of data exfiltration in the presence of advanced persistent threats, treating these attacks not as isolated events but as structured, dynamic processes composed of multiple deviations, repetitions, and conditional choices.

The results were promising, yielding an annotated dataset comprising 38,182 events, 1,541 traces, and 251 behavioral variants, along with annotated logs and corresponding process models. This provides the academic and professional community with a valuable resource for validation, comparison, and the development of future research on behavioral exfiltration detection. The comparative analysis among the three process discovery algorithms (Inductive Miner, Alpha Miner, and Heuristics Miner) revealed differences in their capability to represent exfiltration behavior. The Heuristics Miner achieved the best balance between fitness and precision, resulting in the highest F-Score among the evaluated algorithms.

This result suggests that heuristic-based approaches are particularly well-suited to handling the complexity and noise inherent in security logs, capturing the main causal dependencies without overgeneralizing. Although the Inductive Miner achieved perfect fitness (1.0000), it exhibited lower precision, indicating a tendency to allow additional behaviors not observed in the logs, a phenomenon known as "underfitting" in process mining. Conversely, despite its high generalization and simplicity, the Alpha Miner proved unsuitable for this domain due to its low fitness, confirming the limitations already documented in the literature regarding its applicability to noisy logs and complex structures.

Several directions are proposed for future work. These include developing graph-based visualizations to facilitate process model interpretation and performing cross-validation against widely adopted SIEM platforms. Additionally, efforts should focus on integrating the MITRE ATT&CK taxonomy for automatic event enrichment and TTP classification, as well as adapting the architecture to improve streaming processing for early exfiltration detection.

Acknowledgment

This work has been fully funded by the project *Identificação e interpretação de desvios de comportamentos de usuários mal-intencionados ou ligados ao cibercrime* supported by Centro Integrado de Segurança em Sistemas Avançados (CISSA), with financial resources from the PPI IoT/Manufatura 4.0 of the MCTI grant number CIS-AFCCT-2024-7-26-2, signed with EMBRAPII.

References

Adila, R., Studiawan, H., and Breitinger, F. (2026). Forensic event reconstruction with process mining. *IEEE Access*.

Alshamrani, A., Myneni, S., Chowdhary, A., and Huang, D. (2019). A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials*, 21(2):1851–1877.

ATT&CK, M. (2024). Exfiltration over alternative protocol: Exfiltration over unencrypted non-c2 protocol. `https://attack.mitre.org/techniques/T1048/003/`. Acedido em: 4 de março de 2025.

Augusto, A., Carmona, J., and Verbeek, E. (2022). Advanced process discovery techniques. In *Process Mining Handbook*, pages 76–107. Springer.

Berti, A., van Zelst, S. J., and van der Aalst, W. M. P. (2019). Process mining for python (PM4Py): Bridging the gap between process- and data science. In *Proceedings of the ICPM Demo Track 2019 co-located with the 1st International Conference on Process Mining (ICPM 2019)*, volume 2374 of *CEUR Workshop Proceedings*, pages 13–16. CEUR-WS.org. Aachen, Germany, June 24-26, 2019.

Booij, D. (2019). Agent tesla: A credential stealer using smtp on port 587. SANS Internet Storm Center Diary. Accessed: 2026-03-16.

Buchta, R., Gkoktsis, G., Heine, F., and Kleiner, C. (2024). Advanced persistent threat attack detection systems: A review of approaches, challenges, and trends. *Digital Threats*, 5(4).

Buijs, J. C., van Dongen, B. F., and van der Aalst, W. M. (2012). On the role of fitness, precision, generalization and simplicity in process discovery. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 305–322. Springer.

CrowdStrike (2026). 2026 global threat report. Technical report, CrowdStrike Holdings, Inc., Sunnyvale, CA, USA. Acesso em: 16 mar. 2026.

Davidpur, A. (2026). Unmasking agent tesla: A deep dive into a multi-stage campaign. `https://www.fortinet.com/blog/threat-research/unmasking-agent-tesla-deep-dive-into-multi-stage-campaign`. Fortinet Threat Research Blog. Accessed: 03-Mar-2026.

Di Ciccio, C. and Montali, M. (2022). Declarative process specifications: reasoning, discovery, monitoring. In *Process mining handbook*, pages 108–152. Springer.

European Union Agency for Cybersecurity (ENISA) (2025). Enisa threat landscape 2025: July 2024 to june 2025. Technical report, ENISA, Attiki, Greece. Acesso em: 16 mar. 2026.

Jain, R. (1990). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons.

Krishnapriya, S. and Singh, S. (2024). A comprehensive survey on advanced persistent threat (apt) detection techniques. *Computers, Materials and Continua*, 80(2):2675–2719.

Lajevardi, A. M. and Amini, M. (2021). Big knowledge-based semantic correlation for detecting slow and low-level advanced persistent threats. *Journal of Big Data*, 8(1):148.

Leemans, S. J. J., Fahland, D., and van der Aalst, W. M. P. (2013). Discovering block-structured process models from event logs - a constructive approach. In Colom, J.-M. and Desel, J., editors, *Application and Theory of Petri Nets and Concurrency*, pages 311–329, Berlin, Heidelberg. Springer Berlin Heidelberg.

Macak, M., Vanát, I., Merjavý, M., Jevočin, T., and Buhnova, B. (2020). Towards process mining utilization in insider threat detection from audit logs. In *2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 1–6.

MITRE ATT&CK (2024). Agent tesla — software s0331. MITRE ATT&CK Framework. Accessed: 2026-03-16.

Myers, D., Suriadi, S., Radke, K., and Foo, E. (2018). Anomaly detection for industrial control systems using process mining. *Computers & Security*, 78:103–125.

OECD (2020). *OECD Digital Economy Outlook 2020*. OECD Publishing, Paris.

Pratap Singh, S. and Afzal, N. (2024). The MESA Security Model 2.0: A Dynamic Framework for Mitigating Stealth Data Exfiltration. *arXiv e-prints*, page arXiv:2405.10880.

Rodríguez, M., Betarte, G., and Calegari, D. (2024). A process mining-based method for attacker profiling using the mitre att&ck taxonomy. *Journal of Internet Services and Applications*, 15(1):212–232.

Schwab, K. (2016). *The Fourth Industrial Revolution*. World Economic Forum, Geneva.

Silva, J., Cordeiro, A., Lima, M., Lins, F., Santos, W. R. M., and Lima, R. (2025). Modelling advanced persistent threats to support cyber incident response. In *2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1–4.

Singh, S. P. and Afzal, N. (2024). The mesa security model 2.0: A dynamic framework for mitigating stealth data exfiltration. *International Journal of Network Security & Its Applications (IJNSA)*, 16(3):1–18.

van der Aalst, W., Weijters, T., and Maruster, L. (2004). Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142.

van der Aalst, W. M. P. (2016). *Process Mining: Data Science in Action*. Springer, Berlin, Heidelberg, 2 edition.

van der Aalst, W. M. P. and Carmona, J., editors (2022). *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*. Springer.

Van der Aalst, W. M. P. and De Medeiros, A. K. A. (2005). Process mining and security: Detecting anomalous process executions and checking process conformance. *Electronic Notes in Theoretical Computer Science*, 121:3–21.

Weijters, A., {Aalst, van der}, W., and {Alves De Medeiros}, A. (2006). *Process mining with the HeuristicsMiner algorithm*. BETA publicatie : working papers. Technische Universiteit Eindhoven.