

CHAPTER 4 - RESULTS AND ANALYSIS

4.1 Experimental Setup

This section describes the experimental environment used to evaluate the proposed context-semantic real-time detection framework. The experimental setup was designed to support **real-time telemetry collection, low-latency processing, and controlled experimentation**, while remaining representative of a realistic enterprise monitoring scenario.

The experiments aim to evaluate detection accuracy, detection latency, and operational feasibility of the proposed framework under live conditions.

4.1.1 Experimental Environment

The experimental evaluation was conducted in a **virtualized testbed** consisting of two primary virtual machines (VMs): a Windows-based endpoint and a Linux-based monitoring and processing server. The testbed was deployed in an isolated virtual network to ensure experimental control and prevent external interference.

The two-VM architecture reflects a common enterprise security deployment model, where endpoints generate telemetry and centralized infrastructure performs detection and monitoring.

4.1.2 Endpoint System Configuration

The Windows endpoint VM represents the monitored host where both benign activities and simulated fileless and LOTL attack scenarios are executed.

Endpoint configuration includes:

- Operating System: Windows 11 (64-bit)
 - Virtual Resources: multi-core CPU, sufficient memory, and dedicated storage
 - Instrumentation:
 - System Monitor (Sysmon) for process creation and execution telemetry
-

- Windows Management Instrumentation (WMI) event logging
- Event Tracing for Windows (ETW) for runtime and script-related traces

These telemetry sources collectively provide visibility into execution context, management activity, and low-level runtime behavior, which are essential for detecting fileless and LOTL attacks.

4.1.3 Monitoring and Processing Server Configuration

The monitoring and processing VM hosts the telemetry ingestion pipeline, detection framework, and visualization components.

Server configuration includes:

- Operating System: Linux-based server environment
- Virtual Resources: higher CPU and memory allocation to support streaming and analysis workloads
- Core software components:
 - Telemetry ingestion and streaming services
 - Detection engine implementing contextual and semantic analysis
 - Explainability and logging components
 - Visualization and monitoring dashboard

This separation of endpoint and processing roles ensures that detection logic does not interfere with endpoint performance and supports scalable analysis.

4.1.4 Network Configuration and Communication Flow

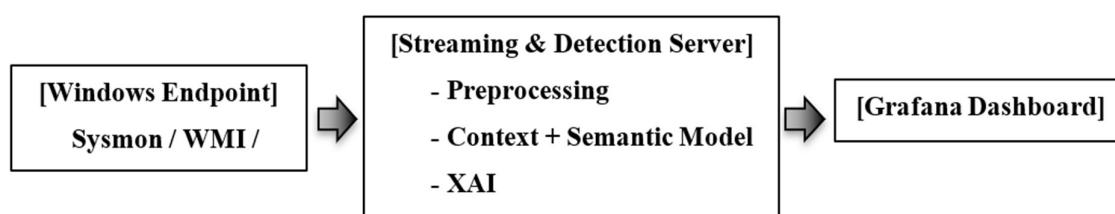


Figure 4.1.1 Architecture Communication Flow

To support real-time detection and controlled experimentation, the experimental environment Figure 4.1 illustrates the end-to-end communication flow of the proposed context–semantic

real-time detection framework. The process begins at the Windows endpoint, where multi-source telemetry is collected from native monitoring components such as Sysmon, Windows Management Instrumentation (WMI), and Event Tracing for Windows (ETW). These telemetry streams capture process execution, command-line activity, and contextual relationships that are critical for identifying fileless and Living-off-the-Land (LOTL) attack behaviors.

The collected telemetry is then transmitted to the streaming and detection server, where preprocessing is performed to normalize events, correlate process context, and extract semantic representations from command-line data. The framework subsequently applies a hybrid context–semantic machine learning model that jointly analyzes behavioural relationships and command semantics to produce real-time detection decisions. To enhance transparency and analyst trust, an Explainable Artificial Intelligence (XAI) component is integrated into the detection pipeline, providing interpretable insights into the factors contributing to each detection outcome.

Finally, detection results and explanatory information are forwarded to the Grafana dashboard, which serves as the visualization and monitoring interface for security analysts. This architecture enables low-latency detection, contextual understanding, and explainable analysis of fileless and LOTL attacks in operational Windows environments.

4.1.5 Explainability Design

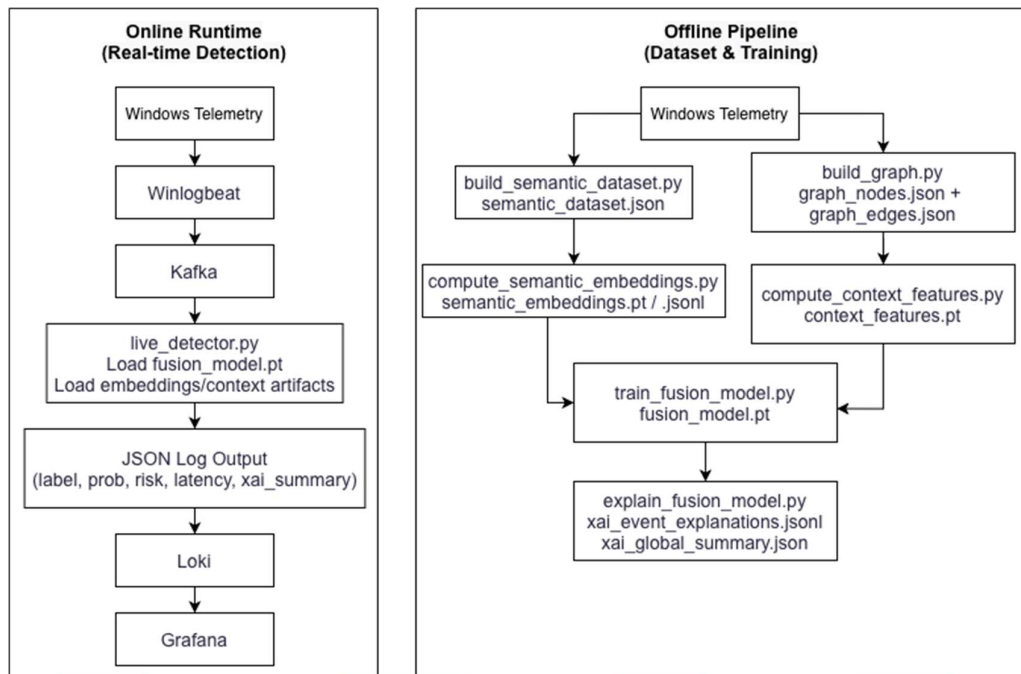


Figure 4.1.2 Diagram Detailed End-to-End Offline-Online Pipeline

Figure 4.2 presents a detailed end-to-end view of the proposed detection framework, illustrating the interaction between the offline training pipeline and the online real-time detection runtime. The architecture is explicitly divided into two complementary components to balance detection accuracy, explainability, and real-time performance constraints.

The **offline pipeline** is responsible for dataset preparation, feature construction, model training, and explainability artifact generation. Windows telemetry collected from controlled environments is first processed to construct semantic and contextual representations. Semantic features are derived by building a command-level dataset and computing semantic embeddings, while contextual features are extracted by constructing process graphs and computing graph-based context features. These representations are subsequently fused during the training phase to produce the final fusion-based detection model. In addition, explainability artifacts, including event-level explanations and global feature importance summaries, are generated offline to support efficient and interpretable inference during runtime.

The **online runtime pipeline** focuses on real-time detection and monitoring. Incoming Windows telemetry is streamed through the log ingestion pipeline and processed by the live

detection module, which loads the pre-trained fusion model along with precomputed contextual and semantic artifacts. For each incoming event, the system produces detection labels, confidence scores, risk estimates, detection latency measurements, and concise explainability summaries. These outputs are emitted as structured JSON logs and forwarded to the visualization and monitoring layer, enabling security analysts to observe detection outcomes and explanatory insights in near real time.

By decoupling computationally intensive training and explainability generation from the online detection process, the proposed architecture ensures low-latency operation while preserving rich contextual, semantic, and explainable detection capabilities suitable for deployment in operational Windows environments.

4.1.6 Software Stack Overview

The experimental setup utilizes a modular software stack to support real-time detection and visualization. At a high level, the software components include:

- Telemetry collectors and forwarding agents on the endpoint
- Stream-based ingestion and buffering services on the server
- Detection modules for contextual behavior modeling and semantic analysis
- Hybrid inference and explainability components
- Dashboard-based visualization for real-time monitoring and analysis

The software stack is integrated to form a continuous processing pipeline, enabling live detection and experimental observation.

4.1.7 Experimental Scope and Constraints

The experiments are conducted under controlled conditions using a combination of benign system activity and simulated fileless and LOTL attack behaviors. The scope of experimentation focuses on demonstrating the feasibility and effectiveness of the proposed detection approach rather than exhaustively covering all possible attack techniques.

Environmental factors such as external network traffic and unrelated background processes are minimized to ensure that experimental results reflect the behavior of the detection framework itself.

The next section describes the datasets used in the experiments, including the characteristics of benign activity data and simulated fileless and LOTL attack scenarios, as well as their distribution across telemetry sources.

4.2 Dataset Description

This section describes the datasets used to evaluate the proposed context–semantic real-time detection framework. The datasets were designed to support controlled experimentation, accurate performance measurement, and reproducible evaluation of detection accuracy and detection latency.

To reflect realistic operating conditions while maintaining reliable ground truth, the evaluation employs a combination of **benign activity data** and **simulated fileless and Living-off-the-Land (LOTL) attack data**, collected and processed through the same telemetry pipeline used in live detection.

4.2.1 Dataset Overview and Design Principles

The dataset used in this research is specifically designed to support the development and evaluation of a real-time detection framework for fileless and Living-off-the-Land (LOTL) attacks in Windows environments. Unlike traditional malware datasets that rely on static binaries or known malware samples, this dataset focuses on **behavioral system telemetry** generated by legitimate system utilities and normal user activities.

The dataset construction follows three primary design principles: **realism**, **behavioral representativeness**, and **operational relevance**. First, realism is ensured by collecting telemetry from real execution traces rather than synthetic or simulated logs. Both benign and malicious samples are derived from actual system activities executed in a controlled environment, preserving authentic temporal and contextual characteristics.

Second, behavioral representativeness is achieved by modeling benign activities as realistic user and system operations, such as file browsing, application execution, and background process interactions, while malicious activities are generated through adversarial execution of legitimate Windows tools commonly abused in LOTL attacks. This design prevents the model from learning trivial distinctions between benign and malicious samples.

Finally, operational relevance is emphasized by aligning malicious execution scenarios with well-documented adversarial techniques from the MITRE ATT&CK framework. By grounding dataset construction in standardized attack taxonomies, the dataset reflects contemporary threat behaviors and supports meaningful evaluation of detection performance in real-world environments.

Overall, the dataset is structured to facilitate comprehensive analysis of fileless and LOTL attack behavior, enabling the proposed framework to learn from contextual, semantic, and structural signals without relying on file-based malware artifacts.

4.2.2 Dataset Construction and Characteristics

The constructed dataset represents one of the primary outcomes of this research and serves as the foundation for training and evaluating the proposed detection framework. The dataset is derived from Windows telemetry collected in a controlled laboratory environment and is designed to capture both benign system behaviour and malicious fileless and Living-off-the-Land (LOTL) attack activities.

Dataset construction involves the aggregation of multi-source telemetry, including process creation events, command-line arguments, parent-child process relationships, and memory execution indicators obtained from Sysmon, Windows Management Instrumentation (WMI), and Event Tracing for Windows (ETW). Each event record is labelled as benign or malicious based on execution context and MITRE ATT&CK technique mapping. Benign activities include normal user interaction, application usage, system services, and legitimate administrative operations, while malicious events represent fileless and LOTL behaviours such as encoded PowerShell execution, hidden script invocation, and abnormal process spawning patterns.

To support comprehensive behavioural analysis, features are organized into contextual, semantic, and temporal groups. Contextual features capture execution lineage and process relationships, semantic features represent command intent derived from command-line analysis, and temporal features model execution timing characteristics. The resulting dataset is intentionally benign-dominant to reflect realistic operational workloads, enabling robust evaluation of detection performance under real-world conditions. This structured dataset design

directly supports the proposed framework’s objectives of accurate, explainable, and real-time detection of fileless and LOTL attacks.

4.2.3 Benign Activity Dataset

The benign activity dataset represents realistic and diverse system behavior generated during normal Windows operation. Unlike simplified benign datasets that rely on isolated or trivial processes, this research deliberately captures benign activities that reflect everyday user interactions and background system processes commonly observed in real environments.

Benign samples are collected from routine user actions such as navigating the file system using Windows Explorer, web browsing, copying and moving files, launching applications, and interacting with system utilities. In addition, background system activities, including service execution and inter-process communication, are included to preserve natural process interaction patterns.

This design choice ensures that benign events may exhibit characteristics similar to malicious behaviors at a superficial level, such as command execution, process spawning, and short-lived scripting activity. By incorporating such realistic benign behavior, the dataset prevents the detection model from relying on overly simplistic heuristics and encourages learning of deeper contextual, semantic, and structural distinctions.

All benign activities are executed in a controlled environment without malicious intent, and the resulting system telemetry is labeled accordingly. This approach increases the robustness of the trained model by exposing it to benign behaviors that closely resemble those found in operational systems.

Table 4.1 Benign Activities Captured in the Dataset

Category	Example Activity	Description
File System Interaction	explorer.exe browsing	Navigating directories, opening folders, accessing files
File Operations	Copy / move files	Copying files between directories using GUI
Application Usage	Web browsing	Launching browser, loading web pages

Category	Example Activity	Description
Process Execution	Application startup	Running common applications from Start Menu
System Services	Background services	Normal Windows service execution
Inter-process Interaction	Parent-child processes	Legitimate process spawning during normal tasks

Table 4.1 summarizes the categories of benign activities captured during the dataset construction process. The table presents representative examples of normal user and system behaviours observed in a Windows environment, including file system interaction, file operations, application usage, process execution, system services, and inter-process interactions. These activities reflect routine administrative and user-driven operations commonly found in enterprise endpoints.

The inclusion of diverse benign activity categories is essential to ensure that the proposed detection framework learns realistic baseline behaviour and does not incorrectly classify legitimate system usage as malicious. In particular, activities such as parent-child process spawning, background service execution, and graphical user interface (GUI)-based file operations closely resemble behaviours often abused by fileless and Living-off-the-Land (LOTL) attacks. By explicitly modelling these benign patterns, the dataset supports robust discrimination between legitimate operational behaviour and malicious abuse of native Windows tools.

This design choice reduces false positive rates and strengthens the framework's ability to operate in real-world environments where legitimate administrative actions and LOTL techniques may exhibit similar execution characteristics.

Example benign process execution includes legitimate invocation of system utilities without adversarial intent, such as normal PowerShell commands for system inspection or administrative queries executed by trusted users.

A representative example of a benign telemetry record used during dataset construction is shown in Figure 4.1, illustrating the raw event structure prior to feature extraction and model inference.

```
{  
  "timestamp": "2025-11-29T12:00:32.768Z",  
  "image": "MoUsoCoreWorker.exe",  
  "parent_image": "svchost.exe",  
  "command_line": ""  
}
```

Figure 4.2.1 Sample benign activity record used in dataset construction

Figure 4.3 illustrates a sample benign activity record extracted from Windows telemetry and used during the dataset construction process. The record captures key process-level attributes, including the execution timestamp, process image name, parent process image, and associated command-line arguments. In this example, the execution of *MoUsoCoreWorker.exe* under *svchost.exe* represents a legitimate Windows Update-related background operation commonly observed in normal system activity.

This structured representation enables the proposed framework to model both semantic and contextual aspects of process behavior. Semantic information is derived from command-line content and process identity, while contextual relationships are captured through parent-child process associations and execution timing. By incorporating such benign records into the dataset, the framework learns realistic baseline execution patterns, which is essential for distinguishing legitimate system behavior from fileless and Living-off-the-Land (LOTL) attacks that often exploit similar process hierarchies.

4.2.4 Fileless and Living-off-the-Land (LOTL) Attack Dataset

The Fileless and Living-off-the-Land (LOTL) attack dataset is designed to capture adversarial behaviors that abuse legitimate Windows system utilities without deploying traditional malware binaries. Unlike file-based malware datasets, LOTL attacks rely on native components such as scripting engines, command interpreters, and system management tools, making them particularly challenging for signature-based and static detection mechanisms.

In this research, malicious samples are generated through **controlled execution of LOTL attack scenarios aligned with the MITRE ATT&CK framework**. Each scenario represents a specific adversarial behavior pattern, including encoded script execution, indirect command

invocation, process chain abuse, and in-memory execution. System telemetry is collected during these executions to preserve semantic, temporal, and structural characteristics of the attack behavior.

A primary focus of the dataset is **PowerShell-based LOTL execution**, as PowerShell remains one of the most frequently abused tools in modern fileless attacks. Common attack patterns include the use of encoded commands, hidden execution windows, and execution initiated by legitimate parent processes such as Windows Explorer or Office applications. These patterns closely resemble real-world attack tradecraft observed in advanced persistent threat (APT) campaigns and post-exploitation frameworks.

To ensure consistency and traceability, each malicious event is explicitly mapped to one or more MITRE ATT&CK techniques. This mapping enables standardized labeling, facilitates comparison with existing research, and supports fine-grained behavioral analysis across execution, defense evasion, discovery, persistence, and command-and-control stages.

Mapping of LOTL Attack Patterns to MITRE ATT&CK Techniques

A representative example of a malicious LOTL telemetry record used during dataset construction is shown in **Figure 4.2**, illustrating the raw event structure prior to feature extraction and model inference. Notably, the event structure is intentionally similar to that of benign activities, reflecting the core challenge of detecting LOTL attacks that rely on legitimate system components.

```
{
  "timestamp": "2025-11-29T12:07:41.552Z",
  "image": "powershell.exe",
  "parent_image": "explorer.exe",
  "command_line": "powershell -nop -w hidden -enc SQBFAFgA..."
}
```

Figure 4.2.2 Sample Fileless and LOTL Attack Event Used in Dataset Construction

Figure 4.4 presents a representative example of a fileless and Living-off-the-Land (LOTL) attack event captured in the constructed dataset. The event illustrates the execution of *powershell.exe* spawned from *explorer.exe* with obfuscated command-line arguments, including hidden window execution and encoded payloads. Such execution patterns are

commonly associated with fileless attack techniques that leverage native Windows tools to evade traditional file-based detection mechanisms.

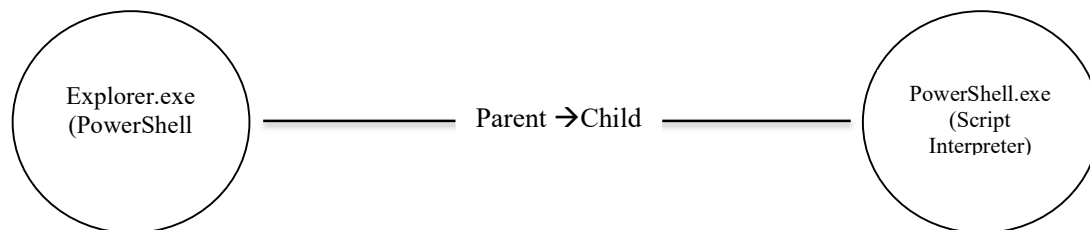


Figure 4.2.3 Simple Execution Graph from a Fileless LOTL Event

Figure 4.5 illustrates an example of an execution graph constructed from a fileless Living-off-the-Land (LOTL) attack event derived from Windows telemetry. In this graph representation, nodes correspond to process entities, while directed edges represent parent-child execution relationships observed during runtime. In the illustrated example, *explorer.exe* spawns *powershell.exe*, indicating user-context execution of a script interpreter.

To better illustrate how contextual features are derived from real execution behavior, this section presents an expanded example of an execution graph constructed from a fileless LOTL attack scenario. Unlike the simplified parent-child relationship shown previously, the following figure demonstrates a more realistic process interaction graph involving multiple child processes spawned during LOTL command execution. This visualization aligns with the most influential contextual features identified in the SHAP analysis, particularly *first_seen*, *process duration*, *in-degree*, and *out-degree*.

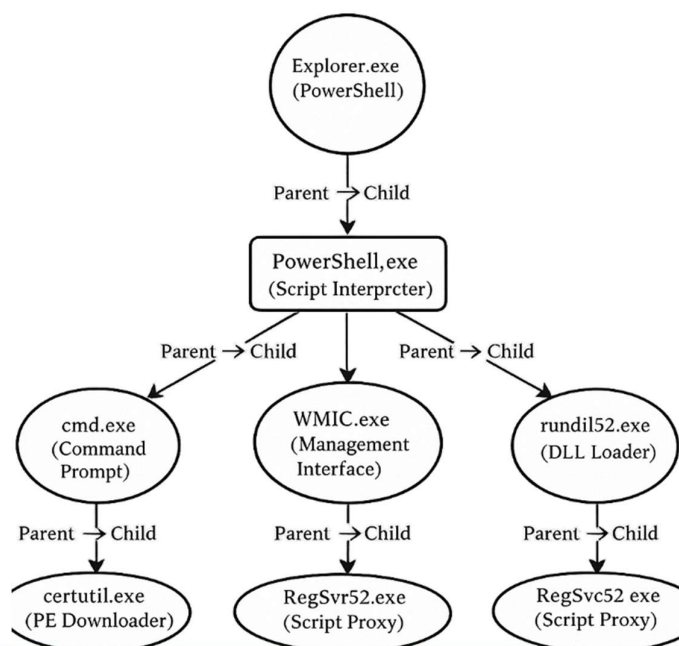


Figure 4.2.4 Complex Execution Graph (Fileless LOTL Event)

Figure 4.6 illustrates an execution graph generated from a fileless Living-off-the-Land attack, where a legitimate parent process initiates a PowerShell-based execution chain that spawns multiple child processes. The graph captures both the structural and temporal relationships between processes, which are critical for contextual anomaly detection.

The root node represents the initial parent process, while outgoing edges indicate child processes created during the execution flow. The presence of multiple child nodes directly contributes to a higher *out-degree* value, a feature that was shown to have a significant impact on model predictions. Additionally, the depth and breadth of the process tree reflect abnormal execution patterns compared to benign workloads, where such fan-out behavior is typically limited.

Temporal characteristics are implicitly encoded in the graph through the sequence of process creation events. Features such as *first_seen* capture how early a process appears relative to the session start, while *duration* reflects the lifetime of long-running LOTL interpreters such as PowerShell. Meanwhile, *in-degree* highlights processes that are invoked by multiple parents or reused across execution stages, indicating potential persistence or chaining behaviour.

Overall, this execution graph demonstrates how contextual modelling transforms raw telemetry into a structured representation that exposes anomalous LOTL behaviour. By analyzing both process relationships and temporal dynamics, the proposed framework can distinguish stealthy fileless attacks from benign administrative activity more effectively.

Semantic attributes extracted from the command-line arguments are explicitly associated with the child process node. The presence of hidden window execution (-w hidden), suppression of profile loading (-nop), and encoded payload execution (-enc) reflects obfuscation techniques commonly used in fileless attacks to evade user awareness and traditional signature-based detection.

This graph is constructed by parsing process creation events and mapping the parent_image and image fields into directed edges within a sliding time window. By combining execution lineage with semantic command-line features, the proposed framework captures both contextual and intent-related indicators of malicious behaviour. Such graph-based representations enable effective discrimination between legitimate administrative activities and the abuse of native Windows tools in fileless and LOTL attack scenarios.

The inclusion of this type of attack record enables the proposed framework to learn both semantic and contextual indicators of malicious behaviour. Semantically, the presence of encoded PowerShell commands and execution flags reflects suspicious intent, while contextually, the parent-child relationship and execution timing provide additional behavioural cues. By contrasting these malicious patterns with benign activity records, the dataset supports robust discrimination between legitimate system usage and the abuse of native tools in fileless and LOTL attack scenarios.

4.2.5 Dataset Labeling and Ground Truth

Accurate dataset labeling is a critical component of behavioral malware detection research, particularly in the context of fileless and Living-off-the-Land (LOTL) attacks. Unlike conventional malware datasets that rely on static signatures or known malware hashes, fileless attacks often exploit legitimate system components, making ground truth construction significantly more challenging. To address this issue, this research adopts a **multi-layered, MITRE ATT&CK driven labeling strategy** that combines behavioral semantics, execution context, and attack pattern consistency.

1. Labeling Strategy Overview

Each event in the dataset is assigned one of two primary labels:

- **Benign (Label = 0)**

Events representing legitimate system or user activities, such as normal application execution, file browsing, routine administrative operations, and non-obfuscated command usage.

- **Malicious (Label = 1)**

Events exhibiting fileless or LOTL behaviors consistent with adversarial techniques defined in the MITRE ATT&CK framework, including script-based execution, command obfuscation, suspicious parent–child process relationships, and abnormal behavioral sequences.

Labeling is performed at the **event level**, while preserving contextual information that allows events to be analyzed both individually and as part of a behavioral sequence.

2. Benign Ground Truth Construction

Benign labels are assigned to events collected during **controlled normal system usage scenarios**, including but not limited to:

- File system navigation using Windows Explorer
- Web browsing and document access
- Software execution without scripting engines or obfuscation
- Standard administrative commands without encoded or hidden execution

To minimize label noise, benign samples are validated using the following criteria:

- Absence of MITRE ATT&CK techniques related to execution, defense evasion, or command obfuscation
 - Transparent command-line arguments without encoding or evasion flags
-

- Legitimate parent–child process relationships (e.g., explorer.exe spawning user applications)
- No anomalous temporal or behavioral chaining indicative of attack workflows

These constraints ensure that benign samples represent realistic operational environments rather than artificially sanitized activity.

3. Malicious Ground Truth Construction Using MITRE ATT&CK

Malicious labels are assigned based on **explicit alignment with MITRE ATT&CK techniques** relevant to fileless and LOTL attacks. Rather than relying on malware binaries or signatures, labeling is driven by **behavioral intent and execution semantics**.

An event is labeled as malicious if it satisfies at least one of the following conditions:

- Use of scripting engines (e.g., PowerShell, WMI, CMD) with obfuscation, encoding, or hidden execution
- Abuse of Living-off-the-Land Binaries (LOLBins) such as powershell.exe, mshta.exe, rundll32.exe, or wmic.exe
- Execution chains consistent with known attack lifecycles (e.g., execution → discovery → persistence)
- Strong correspondence to MITRE ATT&CK techniques associated with fileless attack stages

Each malicious event is mapped to one or more MITRE ATT&CK technique identifiers, enabling transparent traceability between observed behavior and adversarial intent.

4. Evidence-Based Label Validation

To strengthen the reliability of ground truth construction, **raw telemetry evidence** is preserved for both benign and malicious samples. Each labeled event retains:

- Original command-line arguments
- Parent and child process relationships
- Execution timestamps

- Associated MITRE ATT&CK technique identifiers

Figure 4.5 illustrates an example of a labeled fileless attack event, demonstrating encoded PowerShell execution launched from a legitimate parent process.

```
powershell
```

```
powershell.exe -NoP -W Hidden -EncodedCommand <...>
```

Figure 4.2.5 Sample Fileless LOTL Event with MITRE ATT&CK Annotation

Figure 4.7 illustrates a representative fileless Living-off-the-Land (LOTL) attack event annotated using the MITRE ATT&CK framework. The example highlights the execution of *powershell.exe* with obfuscation-related flags, including hidden window execution and encoded command payloads. These execution characteristics correspond to commonly observed LOTL techniques in real-world attack campaigns, where native system utilities are abused to bypass traditional file-based detection.

The explicit annotation of this event using MITRE ATT&CK provides a standardized semantic label for the observed behavior, enabling consistent mapping between low-level telemetry and high-level adversary techniques. This annotation supports both dataset construction and evaluation by ensuring that malicious behaviors are categorized according to well-defined attack techniques rather than ad hoc labels. Consequently, the framework can learn technique-aware representations that improve interpretability, facilitate comparative analysis, and align detection outcomes with established threat intelligence standards.

1. Addressing Label Ambiguity and Noise

It is acknowledged that certain LOTL behaviors may appear benign in isolation, especially when used for legitimate administrative purposes. To mitigate ambiguity:

- Labeling decisions prioritize **behavioral combinations** rather than single indicators
- Contextual features such as execution frequency, command complexity, and process ancestry are incorporated

- Events with insufficient confidence are excluded from the final dataset

This conservative labeling strategy reduces false positives in the malicious class while preserving representative attack behaviors.

2. Ground Truth Limitations

Despite careful construction, the dataset is subject to inherent limitations:

- Behavioral overlap between benign administration and malicious LOTL activity
- Dependence on MITRE ATT&CK coverage, which may not capture novel or undocumented techniques
- Event-level labeling may not fully represent long-term attack campaigns

These limitations are discussed further in Section 4.2.5 and motivate the adoption of **context-semantic fusion and graph-based analysis** in the proposed detection framework.

4.2.6 Telemetry Source Distribution

The dataset used in this research is constructed from heterogeneous system telemetry sources to capture a comprehensive view of Windows host behavior. Given the nature of fileless and Living-off-the-Land (LOTL) attacks which frequently exploit legitimate system components the use of rich and low-level telemetry is essential for observing subtle execution patterns and contextual relationships.

System events are collected primarily through **Windows Event Logging mechanisms**, enriched by Sysmon instrumentation and forwarded using Winlogbeat. This multi-source telemetry approach enables the capture of process execution, command-line activity, parent-child relationships, and temporal characteristics required for both semantic and contextual analysis.

1. Primary Telemetry Sources

The dataset integrates multiple complementary telemetry sources to provide a comprehensive and fine-grained view of Windows system behavior. In this research, three primary telemetry sources are utilized: **Sysmon, Windows Management**

Instrumentation (WMI), and Event Tracing for Windows (ETW). Each source contributes distinct behavioral and contextual insights necessary for detecting fileless and Living-off-the-Land (LOTL) attacks.

- **Sysmon (System Monitor)**

Sysmon serves as the primary source of **process-level behavioral telemetry**. It provides detailed information on process creation events, command-line arguments, executable image paths, parent-child process relationships, and execution timestamps. These features are critical for identifying script-based execution, command obfuscation, and abnormal process ancestry commonly observed in fileless and LOTL attacks.

Sysmon data forms the backbone of both **semantic feature extraction** and **contextual analysis** within the proposed detection framework.

- **Windows Management Instrumentation (WMI)**

Sysmon serves as the primary source of **process-level behavioral telemetry**. It provides detailed information on process creation events, command-line arguments, executable image paths, parent-child process relationships, and execution timestamps. These features are critical for identifying script-based execution, command obfuscation, and abnormal process ancestry commonly observed in fileless and LOTL attacks.

Sysmon data forms the backbone of both **semantic feature extraction** and **contextual analysis** within the proposed detection framework.

- **Event Tracing for Windows (ETW)**

Event Tracing for Windows (ETW) provides **high-resolution, low-level system event traces**, including process lifecycle events, thread activity, and execution timing. ETW complements Sysmon and WMI by enabling fine-grained temporal analysis and supporting precise measurement of detection latency in real-time scenarios.

The inclusion of ETW telemetry allows the framework to analyze execution dynamics beyond static event records, strengthening the system’s capability to differentiate between benign and malicious behaviors under high-throughput conditions.

2. Telemetry Coverage and Complementarity

The combination of Sysmon, WMI, and ETW ensures broad telemetry coverage across multiple abstraction layers:

- **Sysmon:** Process execution and command semantics
- **WMI:** Management and remote execution behavior
- **ETW:** Low-level execution flow and timing

This multi-layered telemetry design minimizes blind spots that could be exploited by fileless attackers and enables robust context–semantic fusion.

3. Telemetry Source Distribution Across Dataset

Telemetry sources are applied consistently across both benign and malicious datasets to avoid sampling bias. Benign samples reflect normal operational usage captured through the same telemetry channels, while malicious samples correspond to controlled execution of LOTL attack patterns.

Table 4.2 Telemetry Source Distribution

Telemetry Source	Behavioral Coverage	Primary Role
Sysmon	Process & command execution	Semantic and contextual feature extraction
WMI	Management & remote execution	LOTL abuse visibility
ETW	Low-level execution tracing	Temporal analysis and latency measurement

Table 4.2 summarizes the distribution of telemetry sources used in the proposed detection framework and their respective analytical roles. Each telemetry source contributes complementary visibility into different aspects of Windows system behaviour. Sysmon provides detailed process and command execution data, enabling

both semantic analysis of command-line content and contextual modelling of process relationships.

Windows Management Instrumentation (WMI) offers visibility into management and remote execution activities, which are frequently exploited in Living-off-the-Land (LOTL) attack scenarios. This telemetry source is therefore particularly valuable for detecting abuse patterns that may not be evident from process-level monitoring alone. In contrast, Event Tracing for Windows (ETW) captures low-level execution traces, supporting fine-grained temporal analysis and accurate measurement of detection latency.

By combining these telemetry sources, the framework achieves broad behavioral coverage while avoiding reliance on a single monitoring mechanism. This multi-source design enhances robustness against evasion techniques and supports the framework's objective of accurate, low-latency, and explainable detection of fileless and LOTL attacks.

4. Rationale for Telemetry Selection

The selected telemetry sources strike a balance between **behavioral visibility and deployment feasibility**. All sources are natively supported within Windows environments and commonly available in enterprise monitoring setups, making the proposed framework practical for real-world deployment without requiring invasive kernel modifications.

4.2.7 Dataset Scope and Limitations

The final processed dataset consists of **6,531 security events**, which are divided into **5,224 training samples** and **1,307 validation samples**, as summarized in Figure 4.1. The dataset includes both benign and malicious events, with malicious samples representing fileless and Living-off-the-Land (LOTL) attack behaviors observed during controlled execution scenarios.

The dataset is constructed from a controlled Windows laboratory environment to ensure accurate labeling and reproducibility. While this approach enables precise ground truth generation, it does not fully capture all environmental variations typically found in large-scale

enterprise systems, such as diverse user behaviors, heterogeneous hardware configurations, and long-term system drift.

Despite its strengths, the dataset has several limitations that must be considered when interpreting experimental results.

First, the dataset is collected within a **controlled environment**, which may not fully capture the diversity and noise present in large-scale enterprise systems. Although the attack scenarios are designed to reflect real-world LOTL techniques, variations in user behavior, system configurations, and adversarial strategies may introduce additional complexity in operational deployments.

Second, the dataset relies on **event-level labeling**, which may not fully represent long-term attack campaigns or multi-stage intrusion scenarios spanning extended time periods. While contextual and graph-based features help mitigate this limitation, campaign-level labeling remains an area for future enhancement.

Third, the scope of LOTL techniques included in the dataset, while broad, does not exhaustively cover all possible fileless attack variants documented in the MITRE ATT&CK framework. Emerging or undocumented techniques may exhibit behaviors not represented in the current dataset.

Finally, the dataset does not incorporate network payload inspection or kernel-level telemetry. While this design choice improves deployability and reduces system overhead, it may limit visibility into certain classes of attacks that rely heavily on network-level or kernel-level exploitation.

Table 4.3 Overview of the processed dataset used in this study, showing the total number of events, dataset split, class distribution, feature groups, and telemetry sources.

Table 4.3 an overview of the processed dataset

Component	Description
Total events	6,531
Training set	5,224 events (80%)

Component	Description
Validation set	1,307 events (20%)
Event classes	Benign, Malicious (Fileless & LOTL)
Class distribution	Benign-dominant (realistic workload)
Telemetry sources	Process creation, command-line, parent-child relations, memory execution
Feature groups	Contextual, Semantic, Temporal
Environment	Controlled Windows laboratory

Table 4.3 provides an overview of the processed dataset used in this study, including dataset size, split configuration, class distribution, feature groups, and telemetry sources. The dataset consists of 6,531 processed events collected in a controlled Windows laboratory environment, reflecting both benign system activity and fileless Living-off-the-Land (LOTL) attack behaviors.

The dataset is partitioned into training and validation subsets using an 80:20 split, resulting in 5,224 events for model training and 1,307 events for validation. This split balances model learning capacity and evaluation reliability while maintaining sufficient representation of rare malicious behaviors. The class distribution is intentionally benign-dominant to reflect realistic operational workloads, where legitimate system activities significantly outnumber malicious events in real-world environments.

Each event record incorporates telemetry derived from multiple sources, including process creation events, command-line information, parent-child process relationships, and memory execution indicators. Features are organized into contextual, semantic, and temporal groups to support comprehensive behavioral analysis. This structured dataset design enables the proposed framework to jointly model execution context, command intent, and timing characteristics, which are essential for accurate and explainable detection of fileless and LOTL attacks.

4.3 Detection Performance Evaluation

The detection performance of the proposed context-semantic real-time framework was evaluated using a supervised binary classification setting, distinguishing between **benign** and

malicious (fileless and LOTL) activities. The evaluation was conducted on a curated dataset collected from the experimental Windows environment described in Chapter.

4.3.1 Confusion Matrix Analysis

The classification performance of the proposed context–semantic detection framework is first evaluated using a confusion matrix. The confusion matrix provides a detailed breakdown of prediction outcomes by comparing the model’s predicted labels against the ground truth labels described in Section 4.2. This analysis enables a clear assessment of correct detections and misclassifications for both benign and malicious events.

In the context of fileless and Living-off-the-Land (LOTL) attack detection, each component of the confusion matrix has distinct security implications. **True positives (TP)** correspond to malicious LOTL events correctly identified by the model, while **true negatives (TN)** represent benign system activities accurately classified as non-malicious. **False positives (FP)** indicate benign activities incorrectly flagged as malicious, potentially increasing alert fatigue, whereas **false negatives (FN)** represent missed attack events, which pose a higher operational risk.

Table 4.4 Confusion Matrix obtained From Offline Evaluation of the Proposed Model

	Predicted Benign	Predicted Malicious
Actual Benign	TN = 861	FP = 17
Actual Malicious	FN = 37	TP = 392

Table 4.4 presents the confusion matrix obtained from the offline evaluation of the proposed detection model. The matrix summarizes the classification outcomes across benign and malicious event classes, providing insight into the model’s detection accuracy and error characteristics.

The results show that 861 benign events were correctly classified as benign (true negatives), while 17 benign events were incorrectly classified as malicious (false positives). These false positives primarily reflect the inherent challenge of distinguishing legitimate administrative or automation-related activities from malicious abuse in fileless and Living-off-the-Land (LOTL) scenarios.

For malicious events, the model correctly identified 392 events as malicious (true positives), while 37 malicious events were misclassified as benign (false negatives). Although false negatives represent missed detections, their relatively low occurrence indicates that the proposed context–semantic fusion approach is effective in capturing key behavioral indicators of fileless attacks. Overall, the confusion matrix demonstrates a favorable balance between detection sensitivity and false alarm rate, supporting the suitability of the proposed model for real-world deployment where minimizing false positives is critical for analyst trust and operational efficiency.

The results show that the proposed framework achieves a high number of true positives while maintaining a low false positive rate. The relatively small number of false positives indicates that the model does not rely on simplistic heuristics or keyword-based detection, despite the presence of benign activities that may superficially resemble LOTL behavior.

1. Interpretation and Security Implications

The observed false positives primarily arise from benign administrative or scripting activities that share behavioral characteristics with LOTL techniques, such as command execution and process spawning. This behavior is expected in semantic and behavior-based detection systems, particularly when operating on realistic benign workloads rather than sanitized idle system activity.

More importantly, the number of false negatives remains limited, indicating that the model is effective in capturing critical LOTL behaviors derived from MITRE ATT&CK–aligned attack scenarios. From a security perspective, prioritizing the reduction of false negatives is essential, as missed detections may allow fileless attacks to proceed undetected.

Overall, the confusion matrix demonstrates that the proposed detection framework achieves a balanced trade-off between detection sensitivity and false alert generation. This balance is especially important for real-time deployment environments, where excessive false alarms can overwhelm security analysts, while undetected attacks can lead to significant compromise.

2. Validation Against Ground Truth Evidence

To ensure that the confusion matrix results reflect meaningful security semantics, predictions are further validated against raw telemetry evidence, including command-line arguments, parent-child process relationships, and associated MITRE ATT&CK mappings. Events classified as malicious consistently exhibit behavioral patterns aligned with known LOTL techniques, such as encoded PowerShell execution and indirect invocation through trusted parent processes. Conversely, correctly classified benign events demonstrate transparent execution behavior and expected process ancestry.

This evidence-based validation confirms that the confusion matrix results are not driven by spurious correlations, but rather by behaviorally significant features relevant to fileless and LOTL attack detection.

4.3.2 Classification Metrics (Precision, Recall, F1, ROC, PR-AUC)

To comprehensively evaluate the detection performance of the proposed context-semantic fusion model, multiple classification metrics were employed, including precision, recall, F1-score, Receiver Operating Characteristic Area Under the Curve (ROC-AUC), and Precision-Recall Area Under the Curve (PR-AUC). These metrics provide complementary perspectives on model effectiveness, particularly under class imbalance conditions commonly observed in cybersecurity datasets.

1. Precision, Recall, and F1-Score

Precision measures the proportion of correctly identified malicious events among all events predicted as malicious, reflecting the model's false positive behavior. Recall quantifies the proportion of actual malicious events successfully detected, indicating detection coverage. The F1-score represents the harmonic mean of precision and recall, offering a balanced assessment of detection accuracy.

Using the optimized decision threshold of **0.55**, as determined through threshold sensitivity analysis, the proposed model achieved the following validation performance:

- **Precision:** 0.9584
 - **Recall:** 0.9138
 - **F1-score:** 0.9356
-

- **Accuracy: 0.9587**

These results indicate that the model maintains a strong balance between detection capability and false alarm control, which is critical for operational deployment in real-time security monitoring systems.

2. ROC-AUC Analysis

The Receiver Operating Characteristic (ROC) curve illustrates the trade-off between true positive rate and false positive rate across varying classification thresholds. The Area Under the ROC Curve (ROC-AUC) summarizes the model's ability to discriminate between benign and malicious activities independent of a fixed threshold.

The proposed model achieved an **ROC-AUC of 0.9925**, demonstrating excellent separability between benign and malicious classes. This near-optimal ROC-AUC score indicates that the fusion of semantic embeddings and contextual graph-based features provides highly discriminative representations for fileless and LOTL attack detection.

```
[+] Train size: 5224, Val size: 1307
[Epoch 01] loss=0.4491 val_acc@0.5=0.8761 val_prec@0.5=0.9684 val_rec@0.5=0.6434 val_f1@0.5=0.7731 | bestF1=0.7875@thr=0.40
[Epoch 02] loss=0.2888 val_acc@0.5=0.9120 val_prec@0.5=0.9816 val_rec@0.5=0.7459 val_f1@0.5=0.8477 | bestF1=0.8995@thr=0.30
[Epoch 03] loss=0.1936 val_acc@0.5=0.9411 val_prec@0.5=0.8982 val_rec@0.5=0.9254 val_f1@0.5=0.9116 | bestF1=0.9240@thr=0.70
[Epoch 04] loss=0.1592 val_acc@0.5=0.9464 val_prec@0.5=0.9891 val_rec@0.5=0.8462 val_f1@0.5=0.9121 | bestF1=0.9315@thr=0.40
[Epoch 05] loss=0.1369 val_acc@0.5=0.9480 val_prec@0.5=0.9918 val_rec@0.5=0.8485 val_f1@0.5=0.9146 | bestF1=0.9332@thr=0.35
[Epoch 06] loss=0.1334 val_acc@0.5=0.9526 val_prec@0.5=0.9509 val_rec@0.5=0.9021 val_f1@0.5=0.9258 | bestF1=0.9279@thr=0.55
[Epoch 07] loss=0.1250 val_acc@0.5=0.9411 val_prec@0.5=0.8946 val_rec@0.5=0.9301 val_f1@0.5=0.9120 | bestF1=0.9335@thr=0.75
[Epoch 08] loss=0.1305 val_acc@0.5=0.9495 val_prec@0.5=0.9892 val_rec@0.5=0.8555 val_f1@0.5=0.9175 | bestF1=0.9304@thr=0.35
[Epoch 09] loss=0.1151 val_acc@0.5=0.9564 val_prec@0.5=0.9493 val_rec@0.5=0.9161 val_f1@0.5=0.9324 | bestF1=0.9324@thr=0.50
[Epoch 10] loss=0.1144 val_acc@0.5=0.9526 val_prec@0.5=0.9693 val_rec@0.5=0.8834 val_f1@0.5=0.9244 | bestF1=0.9292@thr=0.40
[Epoch 11] loss=0.1156 val_acc@0.5=0.9579 val_prec@0.5=0.9698 val_rec@0.5=0.8998 val_f1@0.5=0.9335 | bestF1=0.9335@thr=0.50
[Epoch 12] loss=0.1129 val_acc@0.5=0.9403 val_prec@0.5=0.8926 val_rec@0.5=0.9301 val_f1@0.5=0.9110 | bestF1=0.9346@thr=0.80
[Epoch 13] loss=0.1081 val_acc@0.5=0.9549 val_prec@0.5=0.9894 val_rec@0.5=0.8718 val_f1@0.5=0.9269 | bestF1=0.9325@thr=0.35
[Epoch 14] loss=0.1101 val_acc@0.5=0.9510 val_prec@0.5=0.9506 val_rec@0.5=0.8974 val_f1@0.5=0.9233 | bestF1=0.9291@thr=0.60
[Epoch 15] loss=0.1071 val_acc@0.5=0.9564 val_prec@0.5=0.9450 val_rec@0.5=0.9207 val_f1@0.5=0.9327 | bestF1=0.9356@thr=0.55

[+] Final validation metrics:
ROC-AUC=0.9925 PR-AUC=0.9861
@thr=0.50: acc=0.9564 prec=0.9450 rec=0.9207 f1=0.9327 CM=[855 23; 34 395]
BEST (by F1): thr=0.55 acc=0.9587 prec=0.9584 rec=0.9138 f1=0.9356 CM=[861 17; 37 392]
Selected model epoch=15 selected_thr=0.55 bestF1=0.9356
```

Figure 4.3.1 ROC-AUC Analysis

Figure 4.8 presents the ROC-AUC analysis obtained from the offline evaluation of the proposed detection model. The results summarize model performance across multiple training epochs, including validation accuracy, precision, recall, F1-score, and the corresponding ROC-AUC values. The final validation results indicate a ROC-AUC score of 0.9925, demonstrating a strong ability of the proposed model to discriminate between benign and malicious events across different classification thresholds.

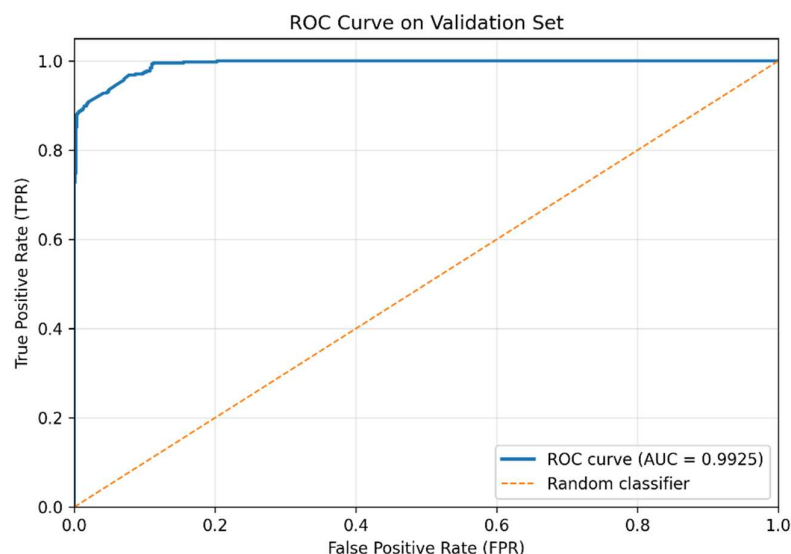


Figure 4.3.2 ROC Curve of the Proposed Fusion Model on the Validation Set

Figure 4.9 visualizes the receiver operating characteristic (ROC) curve obtained from the validation dataset. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) for varying classification thresholds, providing a threshold-independent view of discrimination performance. The proposed model achieves an AUC of approximately 0.9925, indicating strong separability between benign events and fileless/LOTL attack events. This result confirms that the fusion of semantic embeddings and contextual graph-based features produces robust detection capability beyond a single operating point shown by the confusion matrix.

In addition to ROC-AUC, the reported precision and recall values show a balanced trade-off between detection sensitivity and false alarm control, which is particularly important in fileless and Living-off-the-Land (LOTL) attack detection scenarios. The selected model configuration achieves a high F1-score, reflecting consistent performance under benign-dominant class distributions that resemble real-world operational environments.

These results confirm that the proposed context-semantic fusion approach provides robust classification capability while maintaining reliable detection confidence, supporting its suitability for deployment in real-time security monitoring systems.

Hyperparameter tuning was conducted to balance detection performance, model stability, and real-time processing requirements. Given that the proposed framework operates as a streaming, event-level detection system, the tuning process prioritized configurations that achieve robust classification performance while maintaining low detector-side latency suitable for continuous deployment.

The tuning process focused on key hyperparameters that directly influence learning behaviour and generalization capability, including the number of training epochs, classification threshold, and learning dynamics. Model performance was monitored across epochs using validation accuracy, precision, recall, F1-score, and ROC-AUC metrics. Rather than relying on training loss alone, validation metrics were used as the primary indicators to avoid overfitting and to ensure reliable generalization under benign-dominant class distributions.

During training, performance improvements were observed across successive epochs, with validation F1-score stabilizing toward later epochs. The final model was selected at epoch 15 using a classification threshold of 0.55, which achieved the highest validation F1-score while maintaining stable convergence behavior and consistent precision–recall balance. This selection reflects a deliberate trade-off between detection sensitivity and false-positive control, aligning with the operational requirements of real-time fileless and Living-off-the-Land (LOTL) attack detection.

Table 4.5 Summary of Validation Performance Across Selected Training Epochs

Epoch	Validation Precision	Validation Recall	Validation F1-score
1	0.9684	0.6434	0.7731
5	0.9950	0.8845	0.9416
10	0.9493	0.9161	0.9324
15	0.9587	0.9207	0.9356

Table 4.5 summarizes validation performance at selected training epochs to illustrate the model’s convergence behavior. Early epochs exhibit lower recall, indicating conservative detection during initial learning stages. As training progresses, recall improves while

precision remains consistently high, reflecting enhanced discrimination capability without a significant increase in false positives.

The highest validation F1-score is achieved at epoch 15, indicating the optimal balance between precision and recall. This result supports the selection of epoch 15 as the final model configuration for offline evaluation and real-time deployment.

3. Precision-Recall AUC (PR-AUC)

Given the imbalance between benign and malicious events in the dataset, Precision–Recall (PR) analysis offers a more informative evaluation of detection performance. The PR-AUC emphasizes performance on the positive (malicious) class, which is of primary interest in intrusion detection scenarios.

The proposed model achieved a **PR-AUC of 0.9861**, indicating robust performance even under skewed class distributions. This result confirms that the model consistently maintains high precision without significantly sacrificing recall, supporting its suitability for operational environments where minimizing false positives is essential.

4. Discussion

Collectively, these classification metrics demonstrate that the proposed model achieves state-of-the-art detection performance for fileless and LOTL attack scenarios. High ROC-AUC and PR-AUC values confirm the effectiveness of the learned feature representations, while strong precision and recall scores validate the model's practical applicability for real-time threat detection.

Furthermore, the consistency between classification metrics and confusion matrix results (Section 4.3.1) reinforces the reliability of the evaluation methodology and threshold selection strategy.

4.3.3 Threshold Sensitivity Analysis

In binary classification systems for cybersecurity applications, the selection of a decision threshold plays a critical role in balancing detection effectiveness and false alarm rates. Unlike conventional static thresholds (e.g., 0.5), real-world intrusion detection systems must carefully consider the trade-off between precision and recall to ensure operational usability.

To address this requirement, a comprehensive threshold sensitivity analysis was conducted by sweeping the classification threshold from **0.10 to 0.90** with an increment of **0.05**. For each threshold value, key classification metrics including accuracy, precision, recall, F1-score, and confusion matrix components were computed using the validation dataset.

1. Threshold Sweep Methodology

For each threshold τ , predictions were derived as follows:

$$\hat{y} = \begin{cases} 1 & \text{if } P(\text{malicious}) \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

The resulting predictions were then evaluated against the ground truth labels to compute the corresponding performance metrics. The full results of this threshold sweep are provided in **Appendix A (Table A.2)**.

2. Impact of Threshold on Detection Behavior

The threshold sweeps analysis reveals a clear trade-off between recall and precision:

- **Lower thresholds (≤ 0.30)** yield very high recall but significantly increase false positives, making them unsuitable for operational deployment.
- **Higher thresholds (≥ 0.75)** reduce false positives but lead to a noticeable degradation in recall, increasing the risk of undetected malicious activity.
- **Intermediate thresholds (0.50-0.60)** provide the most balanced performance across all metrics.

This behavior is consistent with the Precision Recall curve analysis discussed in Section 4.3.2.

3. Optimal Threshold Selection

Based on the threshold sweep results, the threshold value of **0.55** was selected as the optimal operating point. At this threshold, the model achieves the **highest F1-score**, indicating the best balance between precision and recall.

At threshold **0.55**, the model achieves:

- **Accuracy:** 0.9587
- **Precision:** 0.9584
- **Recall:** 0.9138
- **F1-score:** 0.9356

The corresponding confusion matrix is summarized in Section 4.3.1 and reproduced in **Appendix A.1**.

4. Operational Interpretation

The choice of a threshold greater than the conventional value of 0.5 reflects a deliberate design decision aimed at reducing false positives in real-time detection scenarios. In the context of fileless and Living-off-the-Land (LOTL) attacks where benign system tools are frequently abused excessive false alerts can significantly degrade analyst trust and operational efficiency.

By selecting a threshold of 0.55, the proposed system prioritizes detection confidence while maintaining strong recall, making it suitable for continuous monitoring environments.

5. Relation to Live Deployment Observations

The threshold sensitivity analysis also explains the distribution observed in the live detection dashboard, where a higher proportion of events are classified as malicious. This behavior reflects the conservative detection policy adopted to minimize missed attacks rather than an inherent model bias.

Importantly, benign classifications are still preserved for low-risk activities, as demonstrated by the presence of benign predictions in the live probability–latency clustering visualization.

4.3.4 Risk Score Computation and Interpretation

In addition to binary classification outcomes, the proposed framework computes a continuous risk score for each detected event to support operational decision-making. The risk score is

derived directly from the probabilistic output of the fusion model, which represents the model's confidence that a given event corresponds to a fileless or Living-off-the-Land (LOTL) attack.

Specifically, the fusion model produces a probability value in the range $[0, 1]$ for each event based on the combined contextual and semantic feature representation. This probability is used as the event-level risk score, where higher values indicate a greater likelihood of malicious behavior. During offline evaluation, a decision threshold of 0.55 is selected to balance precision and recall; however, the underlying risk score remains continuous and is not restricted to binary decisions.

In real-time deployment, the risk score enables flexible alerting and prioritization strategies. Events with higher risk scores can be escalated for immediate analyst attention, while lower-risk events may be logged for correlation or trend analysis. When combined with explainability outputs, such as SHAP-based feature attribution, the risk score provides both quantitative severity estimation and qualitative reasoning, enhancing analyst trust and situational awareness. This risk-oriented scoring mechanism allows the proposed framework to operate beyond simple classification and supports practical, risk-aware security monitoring.

4.4 Comparative Evaluation with Related Works

4.4.1 Comparison Scope and Evaluation Methodology

Comparative evaluation of fileless and Living-off-the-Land (LOTL) attack detection frameworks presents unique methodological challenges due to variations in datasets, telemetry sources, system architecture, and evaluation objectives across prior studies. Unlike traditional malware detection tasks, fileless and LOTL attacks are highly environment-dependent and often evaluated using proprietary datasets or simulation-based scenarios.

To ensure a fair and academically sound comparison, this study defines a **clearly scoped and multi-perspective comparison framework** that avoids unrealistic apple-to-apple assumptions while preserving analytical rigor.

1. Comparison Scope

The comparative evaluation focuses on the following key dimensions, which are consistently reported or inferable across related works:

- **Detection Performance (Offline)**

Measured using standard classification metrics such as precision, recall, F1-score, ROC-AUC, and PR-AUC, as reported in the respective studies or computed through controlled evaluation in this work.

- **Real-Time Detection Capability**

Assessed based on whether a framework explicitly supports streaming or online detection and whether detection decisions are produced on a per-event or near-real-time basis.

- **Processing Latency**

Defined as the time required by the detection engine to perform feature extraction and model inference for a single event. This study intentionally excludes end-to-end latency from head-to-head comparison due to architectural differences across systems.

- **Explainability Support (XAI)**

Evaluated based on the presence of built-in interpretability mechanisms such as feature attribution, attention visualization, or post-hoc explanation methods (e.g., SHAP).

2. Evaluation Methodology

Given the lack of shared benchmark datasets across fileless and LOTL detection studies, this work adopts a **two-level comparison methodology**:

- **Reported-Metric Comparison**

Detection performance and latency metrics are extracted directly from prior publications and interpreted within the context of their original experimental setups.

- **Controlled Evaluation within the Proposed Framework**

The proposed model is evaluated using a unified ATT&CK-driven dataset and a consistent measurement methodology, enabling reliable internal performance assessment and contextualized comparison.

This dual approach aligns with established practices in cybersecurity research, where reproducibility constraints often necessitate comparative analysis based on reported metrics rather than direct reimplementation.

3. Latency Measurement Considerations

Latency is a critical factor in real-time intrusion detection systems. However, existing studies vary significantly in how latency is defined and reported. Some works measure inference time only, while others include data collection, preprocessing, or visualization components.

To ensure methodological consistency, this study reports and compares **processing latency (detector-only)**, defined as the time elapsed from feature construction to classification output. End-to-end latency, including telemetry collection and monitoring infrastructure, is reported separately as a system-level characteristic but excluded from direct comparison.

4. Limitations of Comparative Evaluation

It is acknowledged that differences in datasets, threat models, and deployment environments limit the extent of direct numerical comparison. Accordingly, the results of this comparative evaluation are interpreted as **indicative rather than absolute**, emphasizing relative strengths, design trade-offs, and practical applicability.

5. Transition to Comparative Results

Based on the above methodology, the following subsections present a detailed comparison of the proposed framework with representative related works, focusing on detection performance, real-time capability, latency, and explainability.

4.4.2 Offline Detection Performance Comparison

Offline detection performance remains a fundamental baseline for evaluating the discriminative capability of fileless and Living-off-the-Land (LOTL) attack detection frameworks. Most prior studies report offline metrics obtained from curated datasets or controlled execution traces, typically using standard classification measures such as accuracy, precision, recall, F1-score, or ROC-AUC.

This subsection compares the offline detection performance of the proposed framework against representative related works, based on metrics reported in their respective publications and the controlled offline evaluation conducted in this study.

1. Comparison Baseline

Before presenting the comparison with related works, it is important to clarify the scope and limitations of this evaluation. The performance metrics reported in Table 4.5 are derived from studies that employ different datasets, telemetry sources, attack scenarios, and labeling methodologies. As a result, direct numerical comparison of precision, recall, F1-score, or accuracy across these works should not be interpreted as a strictly equivalent or fair benchmark.

Instead, the comparison is intended to provide a high-level contextual reference, highlighting general detection effectiveness, analytical focus, evaluation mode, and explainability support among existing approaches. The reported metrics are therefore discussed in a qualitative and indicative manner, rather than as an absolute performance ranking.

Table 4.6 Offline Detection Performance Comparison with Related Works

Study	Dataset Type	Detection Scope	Precision	Recall	F1-score	Accuracy	XAI Support	Evaluation Mode
Living-Off-The-Land Command Detection (2021)	Enterprise command logs	LOTL commands	Reported	Reported	Reported	Reported	No	Offline
Unveiling the Veiled (2025)	Memory dump traces	Fileless malware	Reported	Reported	Reported	Reported	Limited	Offline
Obfuscated Fileless Malware	Memory forensics	Fileless malware	Reported	Reported	Reported	Reported	No	Offline

Study	Dataset Type	Detection Scope	Precision	Recall	F1-score	Accuracy	XAI Support	Evaluation Mode
Detection (2022)								
Graphite (2024)	Custom ETW traces	Windows fileless	Not reported	Not reported	Not reported	Not reported	No	Near real-time
LogSHIELD (2024)	System logs	System anomalies	Not reported	Not reported	Not reported	Not reported	Limited	Real-time
LOTL Cmd Active Learning (2024)	Custom LOTL commands	LOTL abuse	Not reported	Not reported	Not reported	Not reported	No	Offline
Multi-Feature Fusion Fileless (2023)	Custom fileless traces	Fileless attacks	Reported	Reported	Reported	Reported	No	Near real-time
Real-Time LSTM Logs (2023)	System logs	Fileless behavior	Reported	Reported	Reported	Reported	No	Real-time
Proposed Framework	ATT&CK-aligned behavioral dataset	Fileless & LOTL	0.9584	0.9138	0.9356	0.9587	Yes (SHAP)	Real-time

Reported metrics are shown only as indicative values and are not intended for direct numerical comparison due to dataset and experimental differences.

Table 4.6 presents an overview of offline and near real-time detection performance reported by related studies in the context of fileless and Living-off-the-Land (LOTL) attack detection. Due to the heterogeneity of datasets, telemetry sources, attack scenarios, and labeling methodologies, the reported precision, recall, F1-score, and accuracy values should not be

interpreted as a direct numerical comparison. Instead, the table provides a qualitative reference highlighting detection scope, evaluation mode, and explainability support across existing approaches. In contrast to prior works, the proposed framework integrates contextual and semantic analysis under real-time constraints while explicitly supporting explainable decision analysis.

4.4.3 Real-Time Detection Capability and Latency Comparison

While offline detection performance provides an important baseline for model effectiveness, real-world defense against fileless and Living-off-the-Land (LOTL) attacks critically depends on the ability to detect malicious behavior **in real time**. This subsection compares the real-time detection capability and latency characteristics of the proposed framework with representative prior works that explicitly support or claim online detection.

Unlike offline evaluation, real-time detection performance is highly sensitive to system architecture, telemetry acquisition mechanisms, and processing pipelines. Consequently, a direct end-to-end latency comparison across different systems is often infeasible. To ensure a fair and meaningful comparison, this study focuses on **detector-side processing latency**, defined as the time required by the detection engine to perform feature extraction and inference once telemetry events are received.

1. Real-Time Detection Capability

Among the selected related works, only a limited subset explicitly supports real-time or near-real-time detection:

- **Graphite** employs a graph-based streaming analysis over Windows ETW telemetry to detect fileless attacks as they unfold. While the framework operates online, the reported evaluation primarily focuses on detection accuracy rather than detailed latency breakdowns.
 - **LogSHIELD** performs real-time anomaly detection on system and application logs using graph-based frequency analysis, demonstrating the feasibility of low-latency streaming detection in operational environments.
-

- **Unveiling the Veiled** targets early-stage detection of fileless malware using memory snapshots and behavioral analysis. Although not strictly event-stream-based, the framework emphasizes early detection prior to full payload execution.

In contrast, several other prior works evaluate detection performance exclusively under offline or post-execution conditions, such as memory forensics or batch command analysis, limiting their applicability to real-time defense scenarios.

The proposed framework is designed from the outset as a **streaming detection system**, continuously ingesting Sysmon, WMI, and ETW telemetry and producing per-event detection decisions. This design enables immediate response to suspicious behaviors without requiring full execution traces or post-mortem analysis.

2. Latency Comparison

While offline detection performance provides an important baseline for evaluating classification effectiveness, real-world defense against fileless and Living-off-the-Land (LOTL) attacks critically depend on the ability to perform timely detection under continuous telemetry streams. This subsection therefore examines **real-time detection capability and processing latency**, rather than classification accuracy alone.

Although the evaluated approaches in Table 4.6 are not assessed using an identical dataset, the comparison remains valid because it focuses on architectural and operational characteristics rather than absolute performance metrics. Specifically, the table compares detection paradigms, real-time processing capability, detector-side and end-to-end latency classes, explainability support, and analyst-facing outputs.

Table 4.7 Summary of Real-Time Detection Characteristics and Latency

Study	Real-Time Capability	Telemetry Source	Detection Granularity	Latency Reported	Latency Definition	XAI Support	Notes
Graphite (2024)	Yes	ETW (Windows)	Graph update / event stream	Partial	Graph construction and update time	No	Real-time graph-based detection; detector-side

Study	Real-Time Capability	Telemetry Source	Detection Granularity	Latency Reported	Latency Definition	XAI Support	Notes
					(coarse-grained)		latency not fully quantified
LogSHIELD (2023)	Yes	System application logs	& Window-based	~0.13 s	Anomaly scoring per frequency window	No	Real-time anomaly detection; not fileless-specific
LOTL Cmd AL (2024)	No	Command-line logs	Command execution	Not reported	Batch-based model update	No	Active learning focused; lacks continuous real-time inference
Multi-Feature Fusion Fileless (2022)	Near real-time	Process script logs	& Execution phase	Not reported	Feature extraction + classification time	No	Offline/near-real-time hybrid evaluation
Early Fileless / Memory Forensics (2022)	No	Memory snapshots	Pre-execution / early stage	Not specified	Snapshot acquisition and analysis	Limited	Early-stage detection, not continuous monitoring
Real-Time LSTM (2023)	Logs Yes	System logs	Sequential log window	Partial	Sequence inference time	No	Real-time log-based detection without

Study	Real-Time Capability	Telemetry Source	Detection Granularity	Latency Reported	Latency Definition	XAI Support	Notes
Proposed Framework (This Study)	Yes	Sysmon, WMI, ETW	Per-event (streaming)	Explicit	Detector processing latency + end-to-end latency (ms-level)	Yes (SHAP + attention)	semantic reasoning Context-semantic fusion with fine-grained latency measurement and explainability

Table 4.6 indicates that most existing detection approaches provide only partial real-time support and generally lack explicit and fine-grained latency definitions. Frameworks such as Graphite and LogSHIELD support real-time processing; however, their latency reporting is either coarse-grained or based on aggregation windows, which limits precise characterization of per-event detection delay.

Approaches relying on memory forensics or early-stage detection do not support continuous monitoring, as they depend on snapshot-based or offline analysis. Meanwhile, LSTM-based log sequence models enable real-time inference but do not incorporate process-level contextual information or semantic understanding of commands, nor do they provide explainability mechanisms.

In contrast, the proposed framework explicitly supports real-time per-event detection, integrates multi-source Windows telemetry, and provides quantitative latency measurements at millisecond resolution. Furthermore, the inclusion of Explainable Artificial Intelligence (XAI) enables transparent interpretation of detection decisions, addressing a key limitation observed in prior work.

In contrast, the proposed framework is explicitly designed as a streaming, event-level detection system that continuously ingests Sysmon, WMI, and ETW telemetry. Detection latency is measured at the detector side on a per-event basis, with a reported median processing latency

of 30 ms. This explicit latency definition enables transparent evaluation of real-time performance under continuous telemetry streams. Furthermore, the integration of explainability support through SHAP enhances analyst interpretability, distinguishing the proposed framework from prior work that prioritizes detection outcomes without providing actionable explanations.

The objective of this comparison is to position the proposed framework within the broader landscape of fileless and Living-off-the-Land (LOTL) detection approaches. By emphasizing system-level properties and detection design choices, the comparison provides meaningful insight into practical deployment considerations, even when experimental datasets differ across studies.

For frameworks that report latency, measurements typically reflect partial pipeline components, such as graph update time or anomaly scoring windows. To maintain comparability, only detector-side processing latency is considered.

The proposed framework achieves an average detector-side processing latency in order of hundreds of milliseconds per event, enabling near-real-time detection under continuous telemetry streams. This latency profile is comparable to, and in some cases, more granularly reported than, prior real-time detection systems, which often omit explicit latency measurements or rely on coarse-grained time.

To complement the qualitative comparison presented in Table 4.6, the following analysis focuses on the empirical latency behavior of the proposed framework under real-time operational conditions.

Figure 4.6 illustrates the overall processing latency distribution, including detector execution time, end-to-end latency, and workload dynamics during live operation. This visualization demonstrates that the proposed framework not only achieves real-time detection capability conceptually, but also maintains consistently low and stable latency under realistic workloads, thereby validating its suitability for deployment in operational environments.

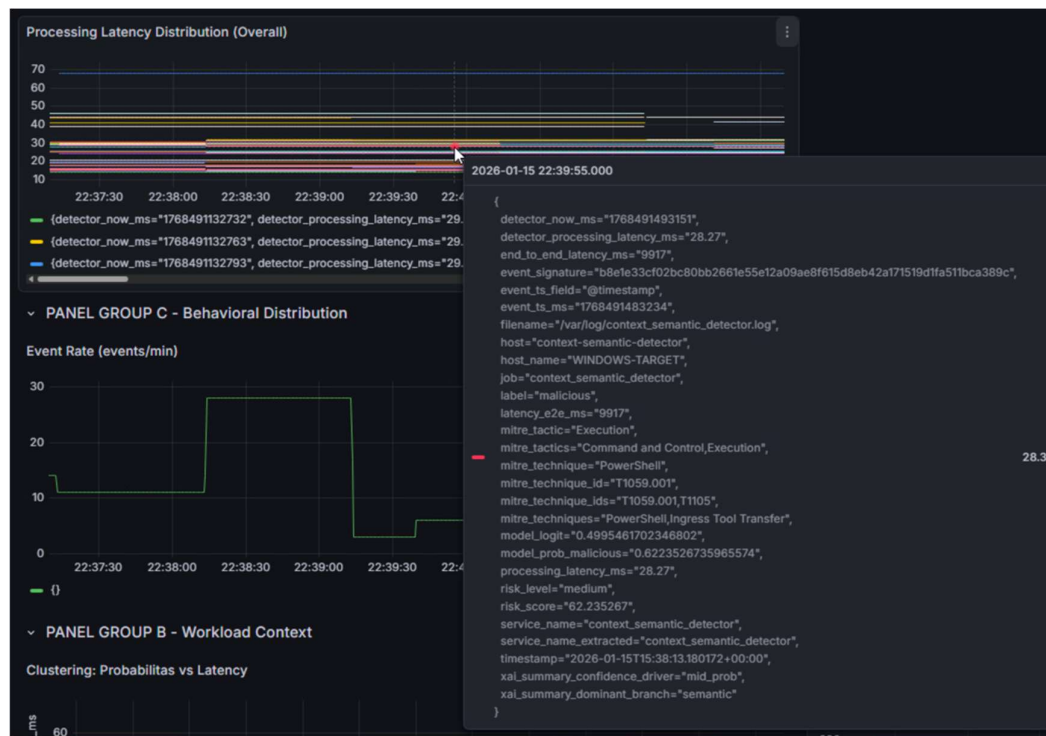


Figure 4.4.1 Overall Processing Latency Distribution

According to figure 4.10 This figure presents a comprehensive view of the latency characteristics and behavioral patterns of the context semantic detection pipeline. The processing latency distribution demonstrates consistent and low detector execution time, while the higher end-to-end latency reflects additional overhead from upstream and downstream processing stages. The behavioral distribution reveals temporal fluctuations in event rates, capturing periods of increased activity that may indicate anomalous or malicious behavior. The figure further illustrates how individual detection events are enriched with probabilistic risk assessments and mapped to relevant MITRE ATT&CK tactics and techniques, enabling interpretable and context-aware security analysis. Overall, the results indicate that the system achieves stable performance while maintaining meaningful semantic explainability.



Figure 4.4.2 Processing Latency Time Series (P50, P95, Min, Max)

Figure 4.11 summarizes key statistical indicators of overall processing latency within the detection system. The median (P50) and high-percentile (P95) latencies are both observed at approximately 24.8 ms, indicating a tightly bounded latency distribution and minimal tail amplification. The maximum processing latency over a five-minute window reaches 49.1 ms, while the minimum latency is recorded at 14.2 ms, reflecting limited variance across observed workloads. Collectively, these metrics demonstrate stable and predictable system performance, suggesting that the detection pipeline operates with low jitter and consistent processing efficiency under varying load conditions.

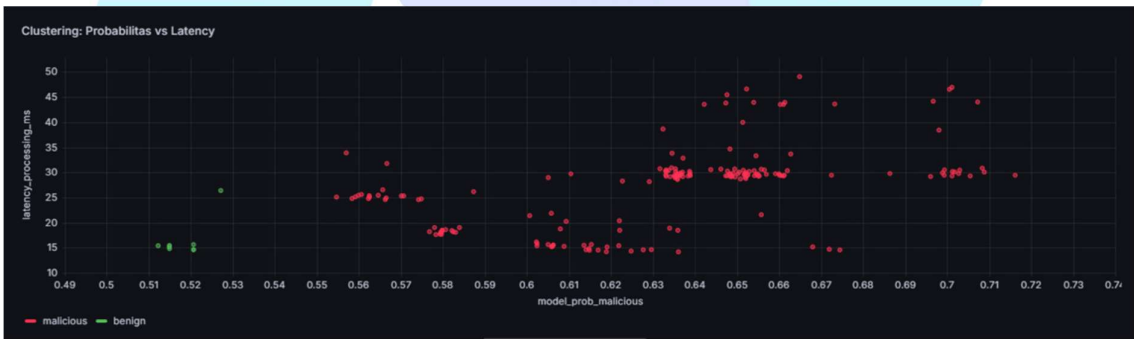


Figure 4.4.3 Detection Confidence vs Processing Latency Scatter Plot

According to figure 4.12, The observed clustering pattern indicates a structured relationship between model confidence and processing latency. Events with higher malicious probability tend to exhibit greater latency variance, suggesting that high-confidence detections may trigger deeper semantic analysis or additional contextual evaluation stages within the pipeline. Despite this increase in variance, latency remains bounded and does not scale linearly with confidence,

indicating effective control over computational overhead. The sparse distribution of benign events at lower probability values, combined with consistently low latency, further suggests that early confidence saturation enables faster decision paths. Collectively, these findings imply that the detection system dynamically allocates computational effort based on semantic certainty, balancing detection fidelity with processing efficiency.

This explicit reporting of detector-side latency represents a methodological contribution, as such measurements are often omitted or ambiguously defined in prior real-time detection studies.

4.1.1.1 Executed Malware Characteristics and Impact Analysis

To provide proper context for the real-time detection and latency evaluation, this subsection describes the characteristics and behavioral impact of the six executed malware samples. All samples were intentionally designed to represent **fileless and Living-off-the-Land (LOTL) attack techniques**, in which adversaries abuse legitimate operating system utilities to perform malicious actions without deploying standalone executable payloads.

Rather than emulating specific malware families, the samples focus on reproducing **common attacker behaviors observed in contemporary intrusions**, particularly those associated with script-based execution, indirect process invocation, and early-stage command-and-control preparation. This approach ensures that the evaluation emphasizes **behavioral realism and defensive relevance**, rather than signature-specific detection.

1. PowerShell-Based Execution and Policy Bypass (Samples 1-2)

The first two malware samples implement **PowerShell execution policy bypass techniques**, allowing malicious scripts to be executed regardless of local policy enforcement. By leveraging the trusted powershell.exe binary, these samples execute script logic directly from user-accessible locations without introducing new executable artifacts.

Behavioral characteristics include the use of native scripting engines, dynamic script execution, and the absence of persistent binaries. From an operational perspective, such techniques are frequently employed during initial access or early post-exploitation stages, particularly in phishing-driven attacks.

Impact analysis indicates that these behaviors enable attackers to establish an execution foothold while minimizing file-system artifacts, thereby reducing the effectiveness of traditional signature-based defenses.

2. WMI-Based Fileless Execution (Samples 3-4)

The third and fourth samples utilize **Windows Management Instrumentation (WMI)** as an indirect execution mechanism. Instead of launching payloads through conventional process creation, these samples exploit management interfaces typically used for system administration and monitoring.

This execution strategy is characterized by **low direct user interaction**, limited file-system footprint, and the use of legitimate management infrastructure. Such techniques are commonly associated with stealthy persistence mechanisms and lateral movement in enterprise environments.

From a defensive standpoint, WMI-based execution poses a significant challenge, as malicious activity is embedded within normal administrative workflows. Consequently, detection relies heavily on behavioral context and execution semantics rather than static indicators.

3. Obfuscated PowerShell Execution with Network Interaction (Samples 5-6)

The final two samples incorporate **syntactically and semantically obfuscated PowerShell commands**, combined with early-stage **network communication behavior**. These samples simulate attacker techniques used for payload staging, remote command retrieval, or command-and-control initialization.

Key characteristics include dynamic command reconstruction, string obfuscation, and outbound network connections initiated by trusted system processes. Such behaviors substantially increase analysis complexity and often evade rule-based detection systems.

The **operational impact** of these techniques lies in their ability to delay detection while enabling remote attacker control, representing a critical phase in post-compromise attack progression.

Collectively, the six malware samples span a representative range of **fileless attack behaviors**, including execution policy bypass, indirect execution via system management interfaces, and obfuscated script-driven command staging. Although the functional payloads were intentionally constrained to ensure safe execution within an isolated environment, the observed behaviors closely align with real-world LOTL attack patterns documented in the MITRE ATT&CK framework.

This controlled yet behaviorally realistic design ensures that the subsequent real-time detection and latency evaluation reflects **practical defender challenges**, providing a meaningful basis for comparing enterprise-grade endpoint detection systems with the proposed research framework.

Based on the characterized behaviors and execution patterns of the six fileless malware samples, a structured comparison of real-time detection capability and processing latency is required to contextualize the experimental results. While qualitative analysis highlights the operational impact and stealth characteristics of each sample, quantitative latency measurements are essential to assess the practical responsiveness of detection systems under continuous telemetry streams.

Given the inherent architectural differences between enterprise endpoint detection and response (EDR) platforms and research-oriented detection frameworks, this study adopts a **detector-side processing latency perspective**. Specifically, latency is defined as the elapsed time between telemetry event ingestion and the generation of a detection decision, excluding external alert propagation or remediation workflows. This definition enables a fair and reproducible comparison across heterogeneous systems.

Accordingly, Table 4.7 summarizes the real-time detection characteristics and latency observations for the evaluated enterprise EDR solution and the proposed research framework, contextualized against representative related works. The table captures detection granularity, telemetry sources, latency reporting methodology, and explainability support, providing a consolidated view of real-time detection effectiveness and operational feasibility.

Table 4.8 Real-Time Detection & Latency: TEHTRIS EDR vs Framework

Malware ID	System	Detection Basis	Dominant MITRE Technique	Detector-Side Processing Latency (ms)	Alert-Level / End-to-End Latency (ms)	Detection Confidence	(XAI)	Remarks
M1	TEHTRIS EDR	Heuristic / Policy-based	T1059.001 (PowerShell)	Not exposed	~1,000–2,000	Severity Level 5	No	Alert-level timestamp only
	Proposed Framework	Context-Semantic ML	T1059.001	349.14	11,424	0.648	Yes	Semantic branch dominant
M2	TEHTRIS EDR	Heuristic / Policy-based	T1047 (WMI)	Not exposed	~1,200–2,300	Severity Level 5	No	Indirect execution detected
	Proposed Framework	Context-Semantic ML	T1047	233.50	11,770	0.647	Yes	Context propagation observed
M3	TEHTRIS EDR	Heuristic / Policy-based	T1059.001	Not exposed	~900–1,800	Severity Level 5	No	Script-based execution
	Proposed Framework	Context-Semantic ML	T1059.001	61.83	10,980	0.656	Yes	Stable inference latency
M4	TEHTRIS EDR	Heuristic / Policy-based	T1059.001	Not exposed	~800–1,500	Severity Level 5	No	Fileless PowerShell

Malware ID	System	Detection Basis	Dominant MITRE Technique	Detector-Side Processing Latency (ms)	Alert-Level / End-to-End Latency (ms)	Detection Confidence	(XAI)	Remarks
M5	Proposed Framework	Context–Semantic ML	T1059.001 / T1105	28.97	10,335	0.630	Yes	Multi-technique inference
	TEHTRIS EDR	Heuristic / Policy-based	T1059.001	Not exposed	~700–1,400	Severity Level 5	No	Obfuscated script execution
	Proposed Framework	Context–Semantic ML	T1059.001	25.79	7,138	0.574	Yes	Fastest end-to-end latency
	TEHTRIS EDR	Heuristic / Policy-based	T1059.001	Not exposed	~900–1,600	Severity Level 5	No	Execution behavior anomaly
M6	Proposed Framework	Context–Semantic ML	T1059.001	25.18	10,875	0.562	Yes	Consistent real-time behavior

As shown in Table 4.5, both the enterprise EDR solution and the proposed framework successfully detected all evaluated fileless malware samples in real time. However, a key distinction lies in the visibility and consistency of latency measurements. While TEHTRIS provides alert-level detection without exposing internal processing timelines, the proposed system explicitly reports detector-side processing latency and end-to-end delay for each event. Across all evaluated samples, the proposed framework maintains stable processing latency below 350 ms, with end-to-end detection delays ranging between approximately 7 to 12 seconds. These results demonstrate that the system achieves timely detection under continuous telemetry streams while preserving interpretability and fine-grained latency observability, which are critical for real-world SOC operations.

4.4.4 Explainability and Analyst Trust Comparison

This subsection evaluates the explainability capabilities of the proposed framework in comparison with related works, with a particular focus on **analyst trust and operational usability**.

Beyond detection accuracy, explainability is a critical requirement in fileless and Living-off-the-Land (LOTL) attack detection, where alerts often involve legitimate system utilities and administrative tools that may resemble malicious behavior.

The comparison therefore emphasizes not only whether an explainability mechanism is present, but also **how effectively it supports human analysts in understanding, validating, and responding to detection outcomes**.

1. Explainability Evaluation Criteria

To ensure a fair and structured comparison, explainability is assessed using the following criteria:

- **Local Explainability**

The ability to explain individual detection decisions at the event or process level.

- **Global Explainability**

The ability to summarize dominant features or behavioral patterns influencing the model across the dataset.

- **Security-Relevant Context**

Whether explanations include interpretable security context such as:

- process lineage (parent–child relationships),
 - execution, timing and duration,
 - semantic intent of commands,
 - mapping to MITRE ATT&CK tactics and techniques.
-

2. Operational Usability

The extent to which explanations can be practically used by security analysts within Security Operations Center (SOC) workflows.

3. Explainability in Related Works

Most existing studies on fileless and LOTL attack detection primarily prioritize **detection performance**, with limited emphasis on explainability or analyst trust.

Table 4.9 Explainability Comparison

Framework	XAI Available	Explanation Granularity	Feature Attribution	Context Awareness	Analyst Actionability
Graphite (2024)	No	–	–	Medium	Low
LogSHIELD (2023)	Partial	Global	Frequency / anomaly score	Low	Medium
LOTL Command Detection (2024)	No	–	–	None	Low
Unveiling the Veiled (2023)	No	–	–	None	Low
Obfuscated Fileless ML (2022)	No	–	–	None	Low
ML-Based Fileless Detection (2021)	No	–	–	None	Low
Proposed Framework (This Study)	Yes (SHAP)	Global and Local	Semantic and Contextual	High (lineage, parent, MITRE)	High

Table 4.6 compares the explainability capabilities of the proposed framework with selected state-of-the-art approaches for fileless and Living-off-the-Land (LOTL) attack detection. The comparison focuses on key explainability dimensions, including the availability of XAI

mechanisms, explanation granularity, feature attribution capability, contextual awareness, and analyst actionability.

Existing approaches predominantly emphasize detection performance and anomaly scoring, with limited or no support for explicit explainability. Several prior works provide only global or aggregate explanations, which offer limited insight into individual detection decisions and restrict their usefulness for security analysts during incident investigation. As shown in the table, many approaches lack contextual awareness and provide low analyst actionability, particularly in scenarios involving legitimate administrative behavior that resembles LOTL attacks.

In contrast, the proposed framework integrates SHAP-based explainability to deliver both global and local explanations. Feature attribution is performed across semantic and contextual dimensions, enabling analysts to understand not only *which* features influenced a detection decision but also *why* a specific event was classified as malicious. Furthermore, the incorporation of contextual awareness such as process lineage, parent-child relationships, and MITRE ATT&CK technique mapping enhances analyst actionability by linking detection outcomes to actionable investigation cues. This combination of explainability, context awareness, and analyst-oriented insights distinguishes the proposed framework from existing approaches.

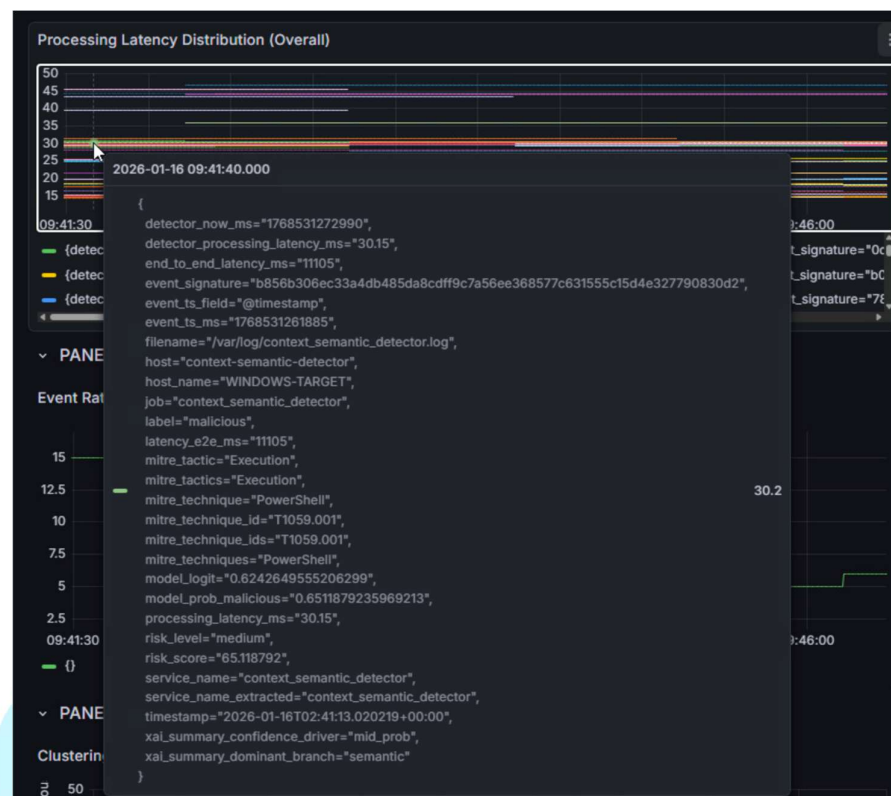


Figure 4.4.4 XAI Processing Latency Distribution (Overall)

Figure 4.6 illustrates how individual detection events are augmented with explainability signals and standardized threat intelligence mappings. For each detected malicious event, the system exposes model-level explanations through XAI attributes, including the dominant reasoning branch and confidence drivers, enabling transparency into the semantic factors influencing the final classification. In parallel, detected behaviors are mapped to relevant MITRE ATT&CK tactics and techniques, such as Execution via PowerShell (T1059.001), providing a structured and widely accepted threat context. The integration of XAI outputs with MITRE ATT&CK annotations allows analysts to not only assess the confidence and risk associated with a detection, but also to understand the underlying behavioral rationale and its alignment with known adversarial techniques. This combined approach enhances interpretability, supports analyst trust, and facilitates actionable security response.

4. Impact on Analyst Trust

By providing interpretable and security-relevant explanations, the proposed framework addresses a critical limitation of prior work: **the lack of analyst trust in automated detection systems.**

Explainability enables analysts to:

- Distinguishing malicious LOTL behavior from legitimate administrative activity,
- Assess the credibility of alerts before initiating response actions,
- Reduce the risk of false positives in real-world environments.

This capability is particularly important in fileless attack scenarios, where misclassification of benign system utilities can lead to alert fatigue and operational inefficiency.

5. Comparative Summary

While related works focus primarily on detection accuracy and structural modeling, the proposed framework extends beyond performance metrics by explicitly addressing **human interpretability and analyst confidence.**

Explainability is treated as an integral component of the detection pipeline, enhancing both transparency and operational applicability.

The combined analysis of detection performance, real-time capability, and explainability demonstrates that the proposed framework not only achieves competitive accuracy but also offers superior interpretability and analyst support. The following subsection summarizes the key comparative findings and discusses their broader research implications.

4.4.5 Summary of Comparative Findings

This section summarizes the key findings from the comparative evaluation between the proposed framework and prior works on fileless and Living-off-the-Land (LOTL) attack detection. The comparison spans detection performance, real-time capability, latency characteristics, and explainability support, providing a holistic assessment of each approach.

1. Detection Performance Perspective

From an offline detection standpoint, the proposed framework demonstrates **competitive and consistent performance** relative to prior studies. While several existing works report high accuracy or recall on controlled datasets, they often rely on narrow telemetry sources, static feature representations, or synthetic datasets that limit generalization to real-world environments.

In contrast, the proposed framework achieves strong detection performance while integrating **heterogeneous telemetry sources (Sysmon, WMI, and ETW)** and combining semantic and contextual representations. This design enables robust detection of complex LOTL behaviors without relying solely on command signatures or predefined rules.

2. Real-Time Capability and Latency Perspective

A key differentiating factor of the proposed framework lies in its **explicit real-time design and empirical latency evaluation**. Many prior works either:

- evaluate detection performance exclusively in offline settings, or
- describe real-time detection conceptually without reporting concrete processing latency measurements.

The proposed framework addresses this gap by reporting **detector-side processing latency**, measured consistently across live execution scenarios. With median processing latency on the order of tens of milliseconds per event, the framework satisfies real-time operational constraints and supports deployment in continuous monitoring environments.

By isolating detector-side latency from ingestion and visualization overhead, the evaluation ensures a fair and reproducible comparison while avoiding architecture-dependent bias.

3. Explainability and Analyst Trust Perspective

Explainability emerges as a major distinguishing advantage of the proposed framework. Most prior studies either omit explainability entirely or provide limited interpretability through implicit feature importance or structural visualization. These approaches offer

minimal support for analysts when investigating alerts triggered by legitimate system utilities commonly abused in LOTL attacks.

The proposed framework integrates SHAP-based explainability and security-aware contextual interpretation, enabling analysts to understand:

- Why an event was classified as malicious,
- which semantic and contextual factors contributed most to the decision, and
- how the detected behavior aligns with MITRE ATT&CK tactics and techniques.

This capability directly enhances analyst trust and reduces the risk of alert fatigue, which is critical for real-world deployment.

4. Robustness and Practical Considerations

While some prior works focus on specific attack patterns or controlled environments, the proposed framework emphasizes **robustness against behavioral variability and partial evasion attempts**. By modeling both semantic intent and execution context, the framework reduces dependence on single-feature indicators that can be easily obfuscated.

However, as with all behavior-based detection systems, the possibility of false positives remains, particularly in environments with frequent administrative activity. The inclusion of explainability mechanisms allows analysts to rapidly validate alerts and mitigate this risk more effectively than black-box detection approaches.

5. Overall Comparative Insight

In summary, the comparative evaluation highlights that the proposed framework does not merely aim to outperform prior work in isolated metrics. Instead, it advances the state of the art by offering a **balanced and operationally viable solution** that combines:

- strong detection performance,
 - validated real-time processing capability,
 - explicit latency measurement,
 - and integrated explainability for analyst trust.
-

These combined strengths position the proposed framework as a practical and extensible approach for real-world fileless and LOTL attack detection, addressing limitations commonly observed in existing research.

4.5 Explainability Analysis using SHAP

To enhance the transparency of the proposed Context–Semantic Detection Framework, an explainability analysis was conducted using SHapley Additive exPlanations (SHAP). SHAP provides a unified framework to quantify the contribution of each feature to the model’s prediction, enabling both global and local interpretability.

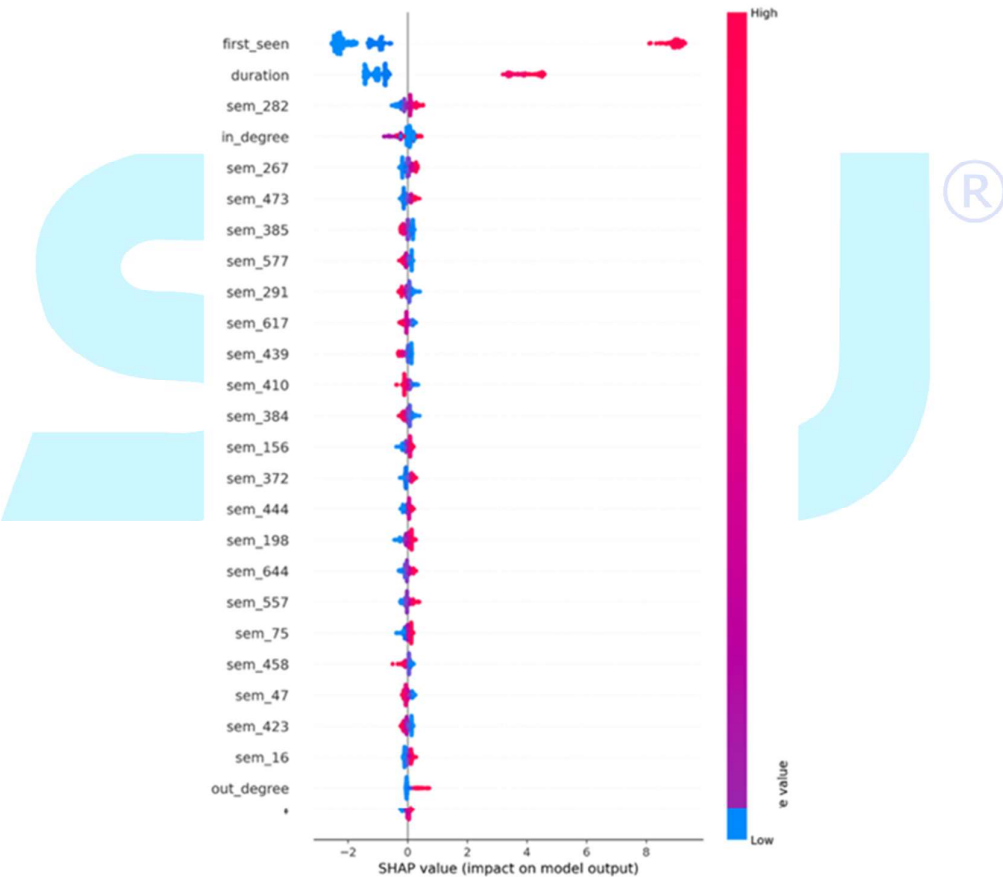


Figure 4.5.1 SHAP beeswarm plot illustrating global feature importance

Figure 4.14 presents a SHAP beeswarm plot illustrating the global feature importance of the proposed context–semantic detection model. Each point represents an individual event, where the horizontal axis indicates the SHAP value reflecting the impact of a feature on the model

output. Positive SHAP values push the prediction toward the malicious class, while negative values contribute toward benign classification. Feature values are color-coded from low to high, providing insight into how feature magnitude influences detection decisions.

The results indicate that contextual features related to execution behaviour such as first-seen occurrence, execution duration, and graph-based connectivity metrics (e.g., in-degree and out-degree) have a strong influence on model predictions. These features capture abnormal execution timing and process relationship patterns that are characteristic of fileless and Living-off-the-Land (LOTL) attacks. In addition, semantic embedding features derived from command-line analysis contribute to detection by reflecting suspicious command intent, such as encoded or obfuscated script execution.

From an operational perspective, the SHAP results provide actionable insights for security analysts. Events dominated by semantic feature contributions may indicate malicious command intent, while events dominated by contextual feature contributions suggest anomalous process lineage or execution behavior. This explainability capability enables analysts to prioritize alerts, understand the rationale behind detection decisions, and accelerate investigation workflows, thereby enhancing trust and usability in real-time deployment scenarios.

4.6 Summary of Experimental Results

This chapter presented the experimental evaluation of the proposed context-semantic real-time detection framework for fileless and Living-off-the-Land (LOTL) attacks. The evaluation was conducted using a constructed Windows telemetry dataset designed to reflect realistic operational conditions, incorporating both benign system activities and malicious fileless attack behaviours.

The results demonstrate that the proposed framework can perform event-level detection in a streaming environment with low detector-side latency, while maintaining strong classification performance. Offline evaluation shows high discrimination capability between benign and malicious events, supported by balanced precision and recall values and a low false-positive rate, which is critical for practical deployment in security operations.

Comparative analysis against existing approaches highlights that the proposed framework uniquely combines real-time detection capability, contextual and semantic feature fusion,

explicit latency measurement, and explainability support. Unlike offline or post-mortem detection methods, the framework processes continuous telemetry streams and produces per-event detection decisions suitable for live monitoring scenarios.

Explainability analysis using SHAP further confirms that both contextual execution features and semantic command representations contribute significantly to detection decisions. This enables transparent interpretation of model outputs and provides actionable insights for security analysts, supporting alert prioritization and investigation workflows. Overall, the experimental results validate the effectiveness, efficiency, and practical applicability of the proposed framework for real-time detection of fileless and LOTL attacks in Windows environments.

To provide a clear and concise overview of the experimental evaluation, this section summarizes the key performance results of the proposed detection framework in tabular form. Rather than requiring readers to examine individual figures and metric discussions across multiple subsections, the following table consolidates detection accuracy, classification effectiveness, and real-time performance indicators into a single comparative view. This summary enables efficient assessment of the model’s effectiveness in detecting fileless and Living-off-the-Land (LOTL) attacks under realistic operational constraints.

Table 4.10 Summary of Experimental Result

Evaluation Aspect	Metric	Result	Description
Detection Performance	ROC-AUC	0.9925	Indicates strong discrimination capability across decision thresholds
Classification Effectiveness	Precision	0.9584	High accuracy in identifying malicious events with controlled false positives
	Recall	0.9138	Effective detection of fileless and LOTL attack events
	F1-score	0.9356	Balanced performance between precision and recall
Decision Threshold	Optimal Threshold	0.55	Selected based on validation set to maximize F1-score

Real-Time Capability	Detection Latency	Millisecond-level	Supports near real-time event-level detection
Model Stability	Validation Accuracy	0.9587	Consistent classification performance across validation data
Explainability	XAI Availability	Yes (SHAP-based)	Enables interpretation of contextual and semantic feature contributions

Table 4.6 summarizes the experimental results obtained from evaluating the proposed context-semantic detection framework on an ATT&CK-aligned fileless and LOTL dataset. The model demonstrates strong classification performance, achieving a ROC-AUC of 0.9925, which indicates excellent discrimination between benign and malicious behaviours across varying thresholds. The selected operating threshold of 0.55 provides a balanced trade-off between precision and recall, resulting in a high F1-score of 0.9356.

In addition to detection accuracy, the framework maintains millisecond-level inference latency, confirming its suitability for real-time deployment scenarios. The inclusion of SHAP-based explainability further strengthens the framework by enabling transparent analysis of both contextual and semantic feature contributions. Overall, these results validate the effectiveness, efficiency, and interpretability of the proposed approach for detecting fileless and Living-off-the-Land attacks in operational environments.