informatics

MDPI

Article

Active Defense and Deception Technology Against Insider Threats: A Serverless and Cryptographically Verifiable Framework

Licia Pia Zichella, Camilla Zambetti *, Simone Tavilla, Gianvito Matarrese and Marina Mongiello

Department of Electrical and Information Engineering, Polytechnic University of Bari, 70126 Bari, Italy; l.zichella@studenti.poliba.it (L.P.Z.); s.tavilla@studenti.poliba.it (S.T.); gianvito.matarrese@poliba.it (G.M.); marina.mongiello@poliba.it (M.M.)
* Correspondence: c.zambetti@studenti.poliba.it

Abstract

Traditional perimeter security models, such as the "castle-and-moat" paradigm, are structurally ineffective against insider threats—including malicious employees, compromised accounts, and external vendors—as these actors already possess legitimate credentials and traverse authorized paths. Traditional passive monitoring systems rely on retrospective access log analysis, which typically identifies data exfiltration only after it is completed. This creates a critical temporal asymmetry that is unsustainable under recent European cybersecurity regulations, such as GDPR, NIS2, and DORA, which mandate incident notification within 24 to 72 h. To bridge this gap, this paper proposes a serverless Active Defense and Deception framework tailored for cloud environments. By strategically deploying realistic, dynamically generated decoy assets (honeyfiles and honeytokens) alongside multi-format web beacons (PDF, DOCX, XLSX) and cryptographic PAdES digital signatures, the system enables real-time detection, forensic attribution, and automated, targeted remediation. We validated a prototype of the system on an emulated Amazon Web Services (AWS) infrastructure using LocalStack and Docker. Experimental results demonstrate a highly stable automated-response latency (approximately 2.16 s, measured as the time to issue the IAM revocation request in the emulated environment), a zero false positive rate (0.00%) over 500 normal traffic events, 100% recall on modeled malicious behavioral patterns, and beacon activation across all tested desktop PDF readers. The proposed framework successfully maps to the MITRE ATT&CK and D3FEND frameworks, offering a verifiable, proactive solution for corporate data protection.

Keywords: active defense; cyber deception; insider threat; cloud security; web beacon; digital signature; NIS2; GDPR

Check for updates

Academic Editor: Guangjie Han

Received: 21 September 2026
Revised: 11 November 2026
Accepted: 12 November 2026
Published: 16 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

1.1. Context, Motivation, and the Regulatory Time-Window Paradox

Traditional cybersecurity architectures have historically relied on perimeter defenses—the "castle-and-moat" paradigm—assuming implicit trust for internal entities [1]. However, this topological assumption is structurally blind to insider threats, which currently account for 57% of global cybersecurity incidents [2]. Operating with legitimate credentials and traversing authorized access paths, malicious insiders or compromised accounts bypass perimeter firewalls entirely, blending seamlessly with honest corporate users [3,4].

This vulnerability is severely compounded by a critical temporal asymmetry we define as the "Dwell Time Paradox". Globally, the median time an adversary remains undetected inside a compromised network is 14 days [5], as conventional passive log analysis typically identifies exfiltration long after the damage is finalized [6]. This operational delay directly violates strict European cybersecurity mandates: Article 33 of the General Data Protection Regulation (GDPR) requires breach notification within 72 h [7], while the NIS2 Directive mandates a 24-h "early warning" for significant incidents, backed by substantial administrative sanctions [8]. Furthermore, the Digital Operational Resilience Act (DORA) demands near-real-time operational resilience for financial institutions [9]. To bridge the gap between sluggish passive monitoring and strict regulatory compliance windows, organizations must abandon implicit trust and transition toward serverless Active Defense systems engineered to autonomously deceive, detect, and contain internal threats in near real time.

1.2. Research Gap and Main Contributions

While the theoretical value of active defense and cyber deception is well-established in the literature, a significant gap remains between academic concepts and practical, cloud-native deployments. Deception frameworks like MITRE Engage [10] catalog offensive and defensive engagement strategies, but existing implementations are often siloed, focusing on coarse-grained network honeypots [11] rather than serverless architectures integrated with enterprise identity and cloud-hosted data-at-rest [3]. Furthermore, conventional deception tools rarely incorporate a legally defensible digital chain of custody, leaving collected telemetry vulnerable to repudiation during internal investigations, regulatory audits, or legal disputes [1].

To address this gap, this paper introduces a serverless Active Defense and Deception framework validated against a real-world financial enterprise document portal emulated via LocalStack and Docker. The primary contributions of this work are as follows:

1. We design a dynamic decoy generator that produces highly realistic multi-format files (PDF, DOCX, XLSX) populated with metadata and author identities drawn from the organizational registry of the simulated enterprise, preventing attackers from identifying traps through static signatures.
2. We embed invisible remote web beacons exclusively within legitimate documents to detect unauthorized access even after the files leave the corporate network.
3. We develop a two-tier detection engine ("Radar") that separates a host-side mediator—responsible for the operations requiring local filesystem access, namely geolocation and cryptographic signature verification—from a stateless serverless function performing threat classification, logging and response: a partition dictated by the isolation of the serverless execution container.
4. We formulate an explainable, sliding-window behavioral rule engine that includes a specialized `recon_pattern` rule to capture the critical transition from legitimate navigation to decoy interaction.
5. We provide an automated, targeted remediation mechanism that instantly revokes the IAM permissions of the compromised account, isolating the threat without disrupting the work of unaffected colleagues.
6. We establish a digital chain of custody by signing both decoy and authentic documents with the PAdES (RSA-2048) standard and storing system events in versioned S3 bucket logs to auto-generate ready-to-print NIS2 and GDPR compliance audits.

1.3. Research Questions

To rigorously evaluate the engineering feasibility, performance guarantees, and overall security posture of our proposed serverless deception and active defense architecture, this

research is guided by three fundamental Research Questions (RQs) that systematically map to the experimental evaluation presented in Section 6:

- **RQ1: Operational Latency and Alert Fatigue Mitigation.** Can a serverless active defense framework achieve a deterministic, near-real-time detection and automated containment loop within emulated cloud environments, and can it eliminate alert fatigue by maintaining a zero false positive (0.00%) rate under typical, high-volume legitimate workflows?
- **RQ2: Offense-Defense Tactical Coverage.** How effectively does the strategic deployment of multi-format decoy assets (honeyfiles, honeytokens, and web beacons) map to standard security frameworks, specifically covering the tactical spectrum of the MITRE ATT&CK matrix while validating defensive countermeasures within the MITRE D3FEND framework?
- **RQ3: Technical Boundaries and Client-Side Security Policies.** What are the inherent boundaries, client-side security constraints, and bypass vectors of document-level remote beaconing mechanisms across contemporary software readers (e.g., PDF and Office clients), and what compensatory active defense controls can mitigate these limits?

The remainder of this paper details the theoretical background (Section 2), the proposed active defense architecture, detection, and forensic mechanisms (Sections 3–5), the experimental evaluation (Section 6), and concludes with a critical discussion (Sections 7 and 8).

2. Related Work and Theoretical Background

2.1. Traditional Perimeter Security vs. Active Defense

For decades, cybersecurity architectures have rested on a heavily asymmetrical premise: defense-in-depth is conceptualized almost exclusively as a series of barriers designed to prevent unauthorized entry. Firewalls, intrusion detection and prevention systems (IDS/IPS), security information and event management (SIEM) platforms, and endpoint antivirus software focus on identifying and blocking malevolent actions at the border or on local hosts, in line with the protective and detective functions codified by international cybersecurity frameworks [12]. Yet, once an adversary manages to bypass these perimeters, or when they already operate within them, traditional controls offer no subsequent layer of friction. There is no structural mechanism to make the attacker doubt the authenticity of the assets they have compromised, nor is there a proactive layer that forces them to make tactical errors.

To address this fundamental limitation, the paradigm of Active Defense has emerged, shifting the strategic focus from passive barrier maintenance to proactive engagement [10]. As formalized in the MITRE Engage framework, active defense is categorized into two primary defensive operations: Cyber Denial and Cyber Deception [10]. Cyber Denial is defined as the capability to impede, degrade, or compromise an adversary's operational maneuvers by systematically limiting their movement or access to critical system assets. Conversely, Cyber Deception involves the deliberate presentation of fictions (such as fake configurations, decoy documents, or honey-credentials) paired with the strategic concealment of genuine corporate assets.

The primary difference between traditional passive defense and active cyber deception is not merely a matter of degree, but of nature. Conventional security tools operate directly upon the adversary's actions, seeking to physically restrict or block them. Active deception, however, operates upon the adversary's cognitive processes and decision-making reasoning [11]. By manipulating the information environment, deception induces attackers to behave in a predictable manner that directly benefits the defender [13]. This is particularly valuable against sophisticated insiders who possess legitimate credentials: while they may

easily circumvent static perimeter thresholds, they cannot differentiate between authentic production data and deceptive traps.

Importantly, active defense operations are strictly confined within the boundaries of the defender's own corporate network. As emphasized in the literature, active defense does not encompass offensive "hacking back" or any operations within the attacker's administrative domain [10]. This strict boundary ensures that deception-based frameworks remain fully compliant with national and international legal guidelines, providing a defensible security posture that avoids the liabilities associated with offensive countermeasures.

2.2. Cyber Deception and Honey-Artifact Taxonomy

Within the broader domain of cyber deception, contemporary literature categorizes deceptive assets into three distinct functional abstractions: *honeypots* (isolated network environments or decoy services), *honeyfiles* (deceptive documents mimicking high-value enterprise data), and *honeytokens* (fake operational credentials, such as API keys or database strings, embedded within production repositories) [11,14]. To operate as high-fidelity detection sensors, these deceptive artifacts must satisfy key operational design principles: high credibility, attractiveness to adversaries, visibility during internal reconnaissance, and structural variability [15]. Crucially, if decoy assets exhibit invariant cryptographic signatures, static metadata, or uniform structural templates, sophisticated attackers can easily script automated filters to bypass the deceptive layer [15].

Building upon these principles, modern deception leverages high-interaction honeyfiles that contain realistic text content to withstand deep semantic inspection, focusing heavily on enticement and context-aware camouflage [14,16]. To extend detection beyond the host server, contemporary architectures transition from passive decoy documents to active tracking sensors equipped with embedded network telemetry, which independently initiate a callback (web beacon) to a remote detection server upon execution [15].

2.3. Document Tracking: From Watermarking to Remote Web Beacons

Historically, corporate document protection relied on digital watermarking to embed ownership metadata directly into files [17]. However, watermarking is a passive, post-mortem forensic mechanism that offers no real-time notification capabilities, making it structurally incapable of preventing data exfiltration in progress.

In modern distributed and cloud-native environments, corporate assets frequently traverse diverse administrative domains, escaping the static control points of perimeter firewalls [18]. To address this, researchers proposed the concept of "sticky policies"—machine-readable access and usage rules cryptographically attached to the data itself, which travel with the asset throughout its life cycle [18]. While sticky policies attempt to enforce remote access control, they require specialized, trusted client environments to interpret and enforce the policies.

To achieve proactive, client-agnostic monitoring, active defense systems leverage remote Web Beacons (commonly referred to as tracking pixels or canary tokens). Operating on HTTP callback mechanics, a web beacon typically consists of an invisible, single-pixel image or a remote resource link embedded within a document's structure [19]. When a client application renders the document, it executes an outbound HTTP request to retrieve the remote resource. This single request transmits critical telemetry to the listening server, including the client's external IP address, timestamp, and user-agent string, without requiring the client to maintain an active session state [19]. By embedding these lightweight callback URLs into multi-format office documents (PDF, DOCX, XLSX), the document is effectively transformed into an active sensor, capable of alerting the security operations center the moment it is opened outside the authorized corporate perimeter.

2.4. User and Entity Behavior Analytics (UEBA) in Insider Threat Detection

While active deception assets provide highly reliable, binary alerts, they must be complemented by behavioral modeling to capture sophisticated threats that evade simple traps. Traditional access controls are structurally blind to malicious actions executed by legitimate users who possess valid permissions to read sensitive directories. To detect these quiet, authorized-yet-malicious activities, modern Security Operations Centers deploy User and Entity Behavior Analytics (UEBA) [20].

When modeling internal threats, the literature distinguishes between two primary adversary profiles: the Masquerader and the Traitor [21]. A Masquerader represents an external attacker who has compromised a legitimate employee's credentials and attempts to impersonate them. A Traitor, conversely, is an authentic member of the organization who abuses their legitimate privileges for personal gain, financial motivation, or espionage. This behavioral distinction is crucial: because "behavior is not something that can be easily stolen" [21], a Masquerader will inevitably display behavioral anomalies, such as accessing unusual directories or logging in at irregular hours, which can be captured via statistical profiling. A Traitor, however, already knows the enterprise systems intimately and can easily blend their exfiltration activities within normal operational workflows, making active deception assets (lures) the primary mechanism for their detection. Recent work has begun to combine the two strategies explicitly, coupling decoy documents with automated classifiers to discriminate masquerade activity [22].

According to the Software Engineering Institute (SEI) Common Sense Guide to Mitigating Insider Threats, a robust detection posture must establish a clear behavioral baseline and actively monitor for indicators such as high-volume data downloads and out-of-hours file system access [1]. However, translating these indicators into reliable alerts is historically challenging. As surveys of anomaly detection have long emphasized, the boundary between normal and anomalous behavior is rarely precise, and models calibrated on one operational context degrade when transferred to another [23]. Classic statistical experiments, such as Schonlau et al.'s analysis of UNIX command patterns, confirm this empirically: purely statistical anomaly detection models often struggle to maintain low false positive rates, frequently causing alert fatigue for SOC analysts [24].

For these reasons, this framework intentionally utilizes deterministic, rule-based sliding-window triggers rather than complex, black-box machine learning anomaly models. This design choice is guided by three core engineering requirements:

1. **Data Availability Constraints:** Building an accurate, multi-user behavioral baseline requires months of real, consistent enterprise telemetry, which is fundamentally absent in newly deployed, experimental, or emulated cloud environments.
2. **Explainability and Legal Accountability:** In forensic investigations, regulatory compliance audits, and legal proceedings, every alarm must be fully explainable [1]. A deterministic trigger based on an explicit, configurable threshold provides clear, auditable evidence (e.g., stating exactly which user downloaded more than 10 files in 5 min), whereas a statistical anomaly score cannot be easily translated into legally defensible proof.
3. **Reproducibility and Determinism:** Configuring explicit thresholds allows administrators to systematically tune, test, and predict the system's sensitivity, ensuring reproducible performance across different organizational departments.

2.5. Positioning Relative to Existing Deception Systems

Decoy documents instrumented with remote beacons were introduced by Bowen et al. [15]. Open-source services such as Canarytokens [25] make beaconed documents and cloud-credential honeytokens widely accessible. A common limitation among these approaches

is their exclusive focus on the detection event. The contribution of this work lies not in a novel deceptive primitive, but in the orchestration of these mechanisms into a serverless pipeline spanning detection to verified containment. Detection is coupled with identity-based attribution: a beacon activation is resolved to the internal principal who downloaded that exact copy. Table 1 summarizes this positioning.

Table 1. Comparative positioning against existing deception approaches (— indicates that the capability is not provided).

Capability	This Work	Bowen et al. [15]	Canarytokens [25]
Generated, camouflaged decoy documents	Yes	Partial	—
Multi-format document beacons	Yes	Partial	Yes
Credential honeytokens	Yes	—	Yes
Identity-based attribution to the internal principal	Yes	—	—
Automated, targeted response (IAM revocation)	Yes	—	—
Cryptographic chain of custody	Yes	—	—
Explainable behavioral analytics	Yes	—	—

3. Proposed Active Defense System Architecture (Materials and Methods)

To transition active defense from a theoretical paradigm into a robust cloud-native utility, we designed and implemented a serverless deception-based active defense framework. This framework was validated against a high-fidelity enterprise environment representing the document repository of a financial firm, *Aurea Capital S.p.A.* This section details the system requirements, the emulation environment, access control models, dynamic decoy generation and structural variability, multi-format beacon injection techniques, and the overall system integration.

3.1. Design Requirements and Threat Model Formalization

The architecture is governed by 14 functional requirements (RF1–RF14) and 10 non-functional requirements (RNF1–RNF10) designed to enforce real-time deception, automated containment, and forensic accountability. Functional requirements cover the dynamic provisioning of multi-format honeyfiles and decoy credentials (RF1–RF2), the instrumentation of authentic documents with invisible beacons, their cryptographic signing and their tracked distribution through the enterprise portal (RF3–RF5), instant alerting on decoy retrieval and out-of-band exfiltration detection (RF6–RF7), sliding-window behavioral analysis, forensic attribution and targeted account containment (RF8–RF10), real-time out-of-band notification and immutable audit logging through bucket versioning and encryption at rest (RF11–RF12), and programmatic compliance reporting on a unified security monitoring dashboard (RF13–RF14). Non-functional requirements establish rigorous cloud access control and encryption at rest (RNF1), cryptographic document non-repudiation (RNF2), near-real-time detection and containment with computationally costly operations kept off the critical path (RNF3), a fail-open posture under communication uncertainty (RNF4), full explainability of behavioral security alerts (RNF5), a reproducible localized testbed (RNF6), code portability between emulated and production environments (RNF7), centralized maintainability of endpoints and rules (RNF8), high usability of the SOC analytical dashboard (RNF9), and native compliance support for the NIS2 and GDPR directives (RNF10).

The corresponding threat model formalizes three primary threat actors:

1. **Compromised User (External Adversary)**: An external attacker who has acquired valid corporate credentials (e.g., through phishing, session hijacking, or credential stuffing) and seeks to perform systematic data discovery and silent exfiltration.

2. **Malicious Insider (The Traitor)**: An authorized corporate employee acting with hostile intent—espionage, financial gain, or sabotage being the motivations most frequently reported as a concern by security practitioners [26]—who leverages their legitimate, broad read-only access to browse the confidential repository, searching for proprietary intelligence.
3. **Careless Insider**: A well-meaning employee who unintentionally violates data security policies, downloads confidential assets to personal devices, or falls prey to social engineering, inadvertently triggering web beacons.

To lure these actors without impacting legitimate personnel, the trust boundaries are defined at the S3 API level. Since the adversary holds legitimate credentials, traditional perimeter firewalls and endpoint detection systems are blind to their queries. However, because decoy documents are accessible using standard read-only IAM policies, any access to these files represents a boundary violation that triggers an instant alert. Figure 1 summarizes the resulting architecture.

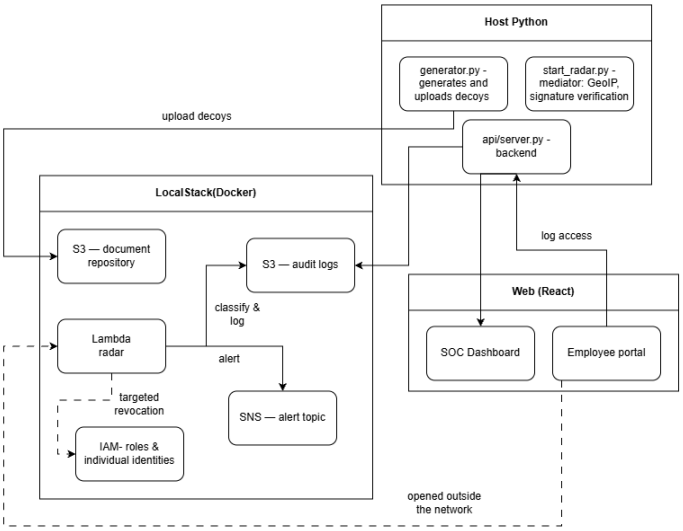


Figure 1. Overall block diagram of the proposed serverless Active Defense and Deception framework, detailing the containerized cloud emulation stack (LocalStack), host-side mediator service, and web-based security analytics and user portals. Solid arrows denote provisioning, logging, and data-access flows; dashed arrows denote the out-of-band exfiltration path (a document opened outside the corporate network) and the resulting targeted IAM revocation.

3.2. LocalStack IAM Enforcement Limitations and Application-Layer Compensating Control

Due to architectural limitations in LocalStack Community Edition regarding the active enforcement of AWS Identity and Access Management (IAM) policy restrictions, native Amazon S3 *Access Denied* exceptions are not generated out-of-the-box upon policy revocation. To preserve the observable behavior and telemetry of the automated containment workflow during local emulation, we implemented a compensating permission verification mechanism at the application layer. Prior to serving any requested document, the host-side backend explicitly queries the LocalStack IAM service to inspect and validate the policies attached to the requesting principal. This application-level check serves strictly as an emulation compensating control; within a production AWS deployment, the native IAM access evaluation engine enforces policy detachments automatically, rejecting unauthorized requests directly at the cloud API gateway and rendering this application-level verification entirely redundant.

3.3. Identity and Access Management (IAM) and Decoy Accessibility

The core philosophy of cyber deception dictates that for a trap to be effective, it must be visible and easily accessible to the target. If decoy documents are locked behind highly restrictive security groups or require complex multifactor authorization, a malicious insider performing routine data discovery will bypass them.

Consequently, we designed a balanced access control scheme:

- **Shared Repository Integration**: Decoy files are stored side-by-side with real corporate assets in a single S3 bucket (`company-secure-documents`) adopting a flat logical layout, with no directory prefix separating the two classes. An adversary enumerating the repository therefore observes one undifferentiated listing in which decoys and authentic documents are interleaved.
- **Broad Read-Only Policies**: All emulated corporate accounts (representing employees in Finance, HR, and Administration) are granted broad read-only access (`s3:GetObject`) across the entire bucket. This is a deliberate departure from the least-privilege principle normally advocated for cloud deployments [27]: restricting visibility would also conceal the traps from the very actor they are designed to engage.
- **No Implicit Exceptions**: The IAM policies contain no rules that explicitly isolate or tag the decoy assets. The enterprise portal through which documents are browsed likewise presents them without any distinguishing marker, so that to a user enumerating the repository the decoys appear indistinguishable from authentic corporate documentation.

3.4. Decoy Generation, Camouflage, and Structural Variability

To prevent a sophisticated attacker from using static file properties or automated content-matching signatures to distinguish between real assets and decoy documents, we formulated an algorithmic decoy generator. This generator enforces both semantic and structural camouflage.

The Python (v3.14)-based generator implements this through a three-phase pipeline. Phase (i), *Semantic Enticement*, relies on the *Faker* library to construct realistic financial and administrative documents. These are localized in Italian and populated with appropriate domain jargon and coherent monetary values. Phase (ii), *Metadata Camouflage*, maps the author properties of each generated file to an employee from the simulated organization's registry, ensuring that an adversary inspecting the decoy encounters a plausible member of the correct department. Phase (iii), *Structural Variability*, regenerates every artifact from scratch at each execution. Because narrative content, financial figures, internal dates, counterparties, and authors are independently drawn, no two decoys of the same category are identical, nor do they share constant text strings. A decoy set is inherently compromised if its members share an invariant feature, as this provides a predictable signature for adversarial scripts [15]. Consequently, file sizes are not imposed as fixed targets but emerge naturally from the generated content. Furthermore, no static boilerplate, constant payload, or repeated binary footer is embedded, leaving no invariant elements for automated filters to detect.

Finally, to ensure complete consistency, the generator automatically applies the organization's official *PAdES digital signature* (using an RSA-2048 private key and self-signed X.509 certificate) and a visual "Confidential" watermarking layer to both real and decoy documents, as detailed in Section 5. In summary, the algorithmic provisioning pipeline consists of five sequential stages: content generation, multi-format beacon injection, cryptographic PAdES digital signing, deployment to the S3 repository, and persistence of the beacon-to-document mapping.

3.5. Multi-Format Beacon Injection Techniques

Our framework tracks unauthorized document openings by injecting invisible remote telemetry sensors (web beacons) that perform out-of-band HTTP GET requests to our serverless ingestion API. We developed and tested injection scripts across three dominant corporate document formats:

3.5.1. Portable Document Format (PDF)

To maximize the probability of triggering a callback, we implement a dual-layer injection strategy within PDF structures:

1. **Visible/Invisible Clickable Annotations**: Using the *ReportLab* library, we inject an invisible, transparent link annotation (a `linkURL` action) spanning the exact coordinate bounding box of the entire document page. If an attacker opens the document and clicks anywhere on the page to scroll, select text, or focus the window, the client reader interprets this as a link activation and initiates an HTTP connection to the target URL.
2. **Zero-Click Document-Level JavaScript**: Using the *pypdf* library, we register a document-level JavaScript action, stored in the catalog's `/Names` JavaScript name tree, containing the following script:

```
// Zero-click PDF Web Beacon trigger
app.launchURL(``http://api.aurea-capital.local/v1/beacon/trigger?file_id=
    UUID_PLACEHOLDER'', true);
```

When a compatible PDF viewer (such as Adobe Acrobat Reader or Foxit Reader) opens the file, it executes this document-level JavaScript on open without requiring any click or interaction from the attacker, triggering an instantaneous, stealthy alert.

3.5.2. Word Documents (DOCX)

Microsoft Word documents use the Office Open XML (OOXML) standard, which stores document components as compressed XML files inside a ZIP archive. To inject web beacons into DOCX files, our Python script performs OpenXML manipulation:

- It extracts the archive and parses the primary document structure file (`word/document.xml`).
- It injects a fields tag containing an `INCLUDEPICTURE` field, pointing to our remote API server URL:

```
<!-- XML Snippet for DOCX Remote Web Beacon Ingestion -->
<w:p><w:r>
  <w:fldChar w:fldCharType="begin"/>
  <w:instrText xml:space="preserve"> INCLUDEPICTURE "http://api.aurea-capital
      .local/v1/beacon/trigger?file_id=UUID_PLACEHOLDER" \d </w:instrText>
  <w:fldChar w:fldCharType="end"/>
</w:r></w:p>
```

- When Microsoft Word or any compatible XML-compliant word processor opens the document, it attempts to render the layout. The rendering engine encounters the `INCLUDEPICTURE` field and sends an HTTP GET request to download the remote image, effectively acting as a silent web beacon.

3.5.3. Excel Spreadsheets (XLSX)

Microsoft Excel suppresses external image links and XML-level picture inclusions, displaying prominent security warnings to the user. To bypass this and avoid raising suspicion, we utilize a formula-based approach. We inject programmatic `HYPERLINK` functions inside critical cells, disguised as necessary corporate connections to external financial databases:

```
=HYPERLINK("http://api.aurea-capital.local/v1/beacon/trigger?file_id=
    UUID_PLACEHOLDER",
          "Update Financial Data")
```

When the attacker clicks on the spreadsheet cell to refresh calculations or follow the reference, the spreadsheet engine triggers an immediate out-of-band HTTP GET request, alerting our detection engine.

3.5.4. Decoy Credentials (Honeytokens)

Beyond documents, the repository includes fake system configurations and DevOps files containing honeytokens. These files, such as `aws_credentials.txt` or `.env.production`, contain fake API credentials, database strings, and private SSH keys. Since these keys are stored exclusively in the decoy files, any attempt to download these credentials (recorded in the portal download log, as discussed in Section 4) provides immediate, irrefutable proof of data discovery and credential compromise.

3.6. Overall System Block Diagram and Use Cases

To orchestrate these components into a unified defense system, we developed the *Radar* active defense server. It operates as a stateless host-side mediator that bridges the emulated cloud endpoints (LocalStack) and the security analysis frontend.

The system defines four key roles in the active defense loop:

1. **Regular Employee**: Performs authorized read/write actions on real corporate documents, experiencing no performance impact, and never interacting with decoy assets during their routine, approved tasks.
2. **Insider Attacker**: Browses S3 storage, downloads intermingled decoy documents, and opens them outside the network, triggering invisible web beacons and honeytoken alerts.
3. **Automated System**: Serves as the primary controller. The host-side mediator geolocalizes beacon callbacks via GeoIP2 and verifies document signatures; the serverless function matches source addresses against Tor and VPN reference sets and executes the targeted revocation of the compromised IAM identity. The behavioral rules, which operate on the accumulated access log rather than on a single callback, are evaluated by the backend analytics service.
4. **SOC Analyst**: Accesses the secure React dashboard to visualize real-time geolocation of active threat vectors, inspects automatically generated forensic compliance dossiers, and manages credential lockouts.

4. Real-Time Detection, Attribution, and Auto-Remediation Mechanisms

4.1. The Active Defense Lifecycle

The architecture presented in Section 3 constitutes, by itself, only latent defense: decoys are provisioned, authentic documents are instrumented, and notification channels are configured, but no observation has yet been converted into an action. This section describes what occurs from the instant an adversary retrieves a decoy asset or opens an instrumented document outside the corporate perimeter.

The distinction between a conventional detection system and an active defense framework lies not in what is observed, but in what follows the observation. A passive system records the event and delegates every subsequent decision to a human analyst. Our framework instead articulates its response into four consecutive stages: **detection**, recognizing that a significant event has occurred; **attribution**, tracing that event back to the identity that originated it; **response**, revoking the offending principal's permissions before the

exfiltration completes; and **forensic proof**, recording every step in a form that resists subsequent alteration.

This decomposition is not arbitrary. The Zero Trust model does not grant trust permanently, but verifies it continuously and admits that it may be *dynamically revoked* in response to anomalous behavior [28]; the third stage is the operational translation of that principle. Likewise, the detection–response–documentation sequence mirrors the incident-handling lifecycle codified by NIST [29], with the crucial difference that the stages here are automated and chained rather than delegated to a subsequent manual intervention.

4.2. Three-Tier Detection Scenarios

The framework recognizes a compromise through three distinct scenarios. **Scenario A** (**Honeyfile**) and **Scenario C** (**Honeytoken**) detect the retrieval of a decoy document or credential file from the enterprise portal, respectively. These share the logic of pure deception: serving no legitimate productive purpose, the mere act of retrieving them is an anomaly by construction, requiring no contextual privilege assessment [11]. To preserve camouflage effectiveness, classification relies on an explicit asset-type attribute rather than lexical filename inspection, ensuring traps remain completely indistinguishable from authentic assets.

Conversely, **Scenario B** (**Instrumented authentic file**) detects the opening of a legitimate document outside the corporate network. Since employees are fully entitled to retrieve authentic files, the access itself carries no anomalous signal; rather, the anomaly arises contextually. The embedded beacon contacts the detection engine upon opening, triggering an exfiltration alarm strictly if the source address falls outside the corporate network space. Finally, one boundary of Scenario C must be explicitly stated: the framework currently detects the *retrieval* of the decoy credential rather than its subsequent *use*, as instrumenting external third-party systems lies outside the scope of this work.

4.3. Serverless Detection Engine: The Radar Architecture

The component that realizes Scenario B is the *Radar*, deliberately partitioned into two parts executing in different environments. This partition is not an incidental complication but the resolution of a concrete constraint of serverless execution, and it recurs as a design pattern throughout the system.

The function that processes the event runs inside an isolated container which, by construction, has no access to the host filesystem: it can therefore read neither the geolocation database nor the X.509 certificate required to verify document signatures. For this reason the Radar is articulated into a **host-side mediator** and a **serverless decision function**. The former receives the beacon's HTTP callback while emulating the role of an API gateway, performs the operations requiring local filesystem access—geolocation of the source address and cryptographic verification of the document signature—and injects the results into the event payload forwarded to the serverless function. The latter contains the decision logic proper: discrimination between internal and external origin, threat classification, immutable logging, notification dispatch and permission revocation.

A security property of the mediator deserves note: the client address is extracted from the underlying TCP connection rather than from HTTP forwarding headers. Headers such as `X-Forwarded-For` are attacker-controllable and would permit trivial source spoofing; reading the connection endpoint removes that bypass vector.

A final element concerns what the adversary perceives. Upon completing its evaluation, the serverless function returns a corporate-branded "403—Access Denied" page simulating an ordinary authorization failure. This is the outermost layer of the deception: while the adversary believes they lack the privileges required to view the content, they have in fact

just raised the alarm, disclosed their geographic origin and triggered the revocation of their own access. The honeyfile acts as an attractor, the beacon as a sensor, and the error page as a deterrent, realizing a multi-layered deception.

A methodological clarification is due regarding how Scenario B is exercised experimentally. In a laboratory environment all addresses are private and therefore not geolocatable, and operating a genuine Tor exit node is not possible. The framework consequently supports a second trigger path: in addition to the beacon fired by a real document opening, the portal can simulate an out-of-perimeter opening by injecting the address of an actual Tor exit node. Both paths produce the same event type and the same downstream behavior; only the trigger differs. The sole simulated element is the adversary's source address, whereas the geolocation, threat classification, notification and permission revocation that follow are genuine.

4.4. Threat Intelligence: Tor, VPN, and Geolocation Enrichment

Once an opening is established as originating outside the corporate network, the framework does not merely record it as external but enriches it with context, because not all external addresses carry the same meaning. An employee opening a document from a residential connection is materially different from an adversary operating behind an anonymization network. The classification module compares the source address against two reference sets and returns one of three labels: *normal*, *Tor*, or *VPN*.

The Tor classification warrants a technical note. Tor routes traffic through a chain of encrypted relays such that each relay knows only its immediate predecessor and successor, but never the full path [30]. The property relevant to a defender is that, from the server's perspective, traffic emerging from the Tor network always exits through a publicly enumerable *exit node*: comparing the source address against that list therefore reveals that a request arrived through Tor. VPN classification follows an analogous principle, matching the address against the network blocks of major commercial providers. Geolocation resolves the address to an approximate position through the MaxMind GeoLite2 database, allowing the event to be plotted alongside the corporate headquarters.

The value of this enrichment is to assign a gradation of severity to the alarm: an origin identified as the Tor anonymization network is a considerably stronger indicator of intent than a generic external address, and, as shown in Section 4.6, it qualifies the justification recorded with the automated response.

Two limitations must be declared. The Tor exit-node and VPN range lists employed in the prototype are **static snapshots**: a production deployment would require periodic synchronization with official feeds or replacement by a commercial threat-intelligence service, and a VPN self-hosted on a private server would appear in no provider list. As for geolocation, GeoLite2 offers high accuracy at the country level but appreciably lower accuracy at the city level; it should therefore be treated as an investigative indicator rather than as forensic evidence.

4.5. Behavioral Rule Engine and Sliding-Window Analytics

The three scenarios described so far operate on the individual event. However, as established in Section 2.4, many hostile behaviors emerge only when a sequence of actions is observed, each of which appears legitimate in isolation. The behavioral module addresses this level, translating the risk indicators recognized by the insider-threat literature [1] into four threshold-based rules, summarized in Table 2. Unlike the Radar, which reacts to a single callback, this module operates on the accumulated access log and is therefore evaluated by the backend analytics service rather than by the serverless function.

Table 2. The four behavioral detection rules, with their configured thresholds and severity.

Rule	Window	Condition	Severity
`download_burst`	5 min	Same user exceeding 10 document retrievals	High
`off_hours`	—	Retrieval performed outside working hours (08:00–19:00)	Medium
`mass_access`	30 min	Same department exceeding 15 distinct retrievals	High
`recon_pattern`	10 min	Decoy access shortly after an authentic-file access, by the same user	Critical

The first three rules share a **sliding-window** mechanism. Let $E_u = \langle e_1, \ldots, e_n \rangle$ be the access events of principal u ordered by timestamp $t_1 \leq \cdots \leq t_n$, let Δ be the window length and θ the configured threshold. For each index i the module maintains the left boundary $j(i) = \min\{\, j \leq i : t_i - t_j \leq \Delta \,\}$, so that the number of events falling inside the window is

$$c_i = i - j(i) + 1. \tag{1}$$

An alert is emitted at i if and only if $c_i > \theta$ and no alert is currently latched; the latch is released as soon as $c_i \leq \theta$. This state variable prevents a single sustained peak from producing one alert per event, a common source of alert fatigue. The `mass_access` rule applies the same predicate with u ranging over organizational departments rather than individuals, capturing coordinated activity that no single account would reveal.

The fourth rule follows a different predicate. Denoting by R_u and H_u the accesses of u to authentic and decoy assets respectively, `recon_pattern` fires when

$$\exists\, r \in R_u,\ h \in H_u \ :\ 0 < t_h - t_r \leq \Delta, \tag{2}$$

that is, when a decoy interaction follows an authentic-file access within the configured interval. This is the characteristic signature of reconnaissance, in which the adversary explores authentic files and inadvertently engages a trap, and it is the point at which the two complementary strategies of the framework—deception and behavioral analysis—converge, which is why it carries the highest severity. Consistently with Section 4.2, decoy recognition within this rule relies on the declared asset attribute rather than on the filename, so as not to nullify the camouflage.

The thresholds of all rules are declared in a centralized configuration file rather than hard-coded, which operationalizes the explainability requirement (RNF5): an analyst can read, justify and adjust each threshold and observe its effect on the number of alerts produced—a transparency that a trained model would not afford. Two boundaries delimit what these rules are and are not. First, they primarily intercept the *Traitor* profile: the legitimate user whose conduct becomes anomalous in volume, timing or sequence, rather than the *Masquerader* operating under stolen credentials [21]. Second, the framework performs **no psychological profiling**: it assigns no personality scores and infers no individual traits—an inferential direction explored elsewhere in the literature [31]—restricting itself to observable and measurable behavior, namely how many files, at what time, and in which sequence. This is a deliberate choice whose privacy implications are examined in Section 7.3.

4.6. Identity-Based Attribution and Targeted Auto-Remediation

This subsection addresses the stage at which the defense, having detected, reacts. Attribution and response are inseparable: permissions cannot be revoked from a principal who has not been identified.

Attribution resolves a correlation problem between identifiers generated at different stages of the document lifecycle. The system binds the beacon identifier to the individual retrieval event rather than to the document. Upon the retrieval of an authentic file, the backend personalizes the served copy: it injects a freshly generated beacon identifier, re-signs the file (in the case of PDF documents), and persists a mapping that associates this identifier

with the requesting principal and the retrieval timestamp. When a beacon activates, the attribution module resolves the identifier directly to the principal who received that specific copy. The opening of one copy can never be misattributed to an innocent colleague who retrieved the same file. This guarantee holds for distinct downloaded copies; it does not extend to the case in which a single copy is forwarded and subsequently opened by a different individual. In that scenario the automated IAM revocation targets the original downloader rather than the eventual opener, a boundary discussed in Section 7.2.

The same correlation yields a metric of independent operational value. For an exfiltration observed through beacon b at t_{exf}, the **dwell time** is defined as

$$D(b) = t_{\text{exf}} - t_{\text{dl}}(b), \tag{3}$$

where $t_{\text{dl}}(b)$ is the retrieval timestamp persisted with the mapping of b, i.e., the interval between the internal retrieval of that specific copy and its opening outside the perimeter. This quantity instantiates, at the granularity of a single document, the dwell-time paradox introduced in Section 1.1: a short interval suggests an automated operation or a determined adversary, whereas an interval of hours or days suggests an insider deliberating, or data forgotten on an unmanaged device.

Response consists of revoking the identified principal's permissions. The module detaches the access policies bound to the individual identity of the employee; should that identity not exist, it falls back to the shared role. The decisive property is that the revocation is **targeted**: it affects the single compromised principal and leaves colleagues' access intact—the difference between reacting and penalizing indiscriminately. The response is furthermore **proportionate**: the justification recorded alongside the revocation incorporates the threat classification of Section 4.4 and the verification status of the document signature, so that the enrichment becomes not merely a dashboard label but an element qualifying the severity of the incident within the response record. The same revocation, besides executing automatically, may be issued manually by the analyst from the dashboard, with an identical technical effect.

The efficacy of the containment is itself observable. Should the isolated principal attempt a further retrieval, the request is refused and the refusal is recorded as an event in its own right. The audit trail therefore contains not only the decision to respond, but the evidence that the response took effect—the final link of a chain running from detection to verified containment.

Detection and response are of value only if they reach an operator in useful time. Every significant event—the exfiltration of an authentic document, the access to a honeyfile or a honeytoken, and each behavioral alert—generates a dedicated out-of-band notification, colour-coded by severity, delivered to a corporate messaging channel. The notification component is deliberately *pure*: it receives the destination endpoint as a parameter and never interrupts the detection pipeline, since an unconfigured channel must not block the system—notification is ancillary, detection is not. Compressing the interval between an event and the analyst's awareness of it is not an operational nicety: it is what makes the 24- and 72-h notification windows discussed in Section 1.1 practically attainable.

Finally, a boundary of the emulated environment must be restated. As noted in Section 3.2, the Community Edition records policy revocations but does not enforce access control, so revocation alone would be an administrative action without observable effect. Permission checking is therefore replicated at the application layer, adopting a deliberately *fail-open* posture: where the system cannot establish with certainty that access has been revoked, it does not block, because a defense must not obstruct legitimate work under uncertainty. In a production AWS deployment, IAM itself would reject the request and this check would be redundant. Declaring this boundary does not weaken the result: the

framework demonstrates the complete *logic* of active response—attribution, targeted revocation and traceability—which on production infrastructure would be enforced by the native IAM engine.

5. Forensic Custody, Document Integrity, and Security Operations Center (SOC)

While the real-time detection and remediation loops described in Section 4 provide rapid containment, an active defense system remains operationally incomplete without a verifiable mechanism to establish legal admissibility and provide security analysts with high-fidelity situational awareness. A system that detects and reacts but cannot prove what transpired is unusable during formal internal investigations, regulatory audits, or judicial proceedings [1]. This section details the cryptographic document integrity mechanisms, the immutable audit trail that serves as a digital chain of custody, the automated forensic dossier and compliance report compiler, and the architecture of the Security Operations Center (SOC) dashboard.

5.1. Cryptographic Document Integrity via PAdES Digital Signatures

To ensure both non-repudiation and structural mimetics, every document in the repository—whether authentic or decoy—undergoes a standardized cryptographic provisioning process. Real corporate documents require cryptographic signatures to certify authenticity, meaning that decoy files must also be signed; otherwise, a sophisticated attacker could easily identify trap-based documents by sifting for unsigned files [15].

The framework utilizes RSA-2048 keys and self-signed X.509 certificates to achieve compliance with ETSI PAdES (PDF Advanced Electronic Signatures) baseline specifications [32]. During the system's initialization bootstrap, a corporate Certificate Authority (CA) is established on the host system to generate an X.509 certificate for *Aurea Capital S.p.A.—Cloud Active Defense* using a SHA-256 digest.

When a PDF decoy or real document is generated, the cryptographic signature is applied using an incremental update methodology. Rather than rewriting the entire PDF binary, which would disrupt existing cross-reference tables and trigger file modification alerts, the PAdES digital signature block is appended directly to the end of the file structure.

This incremental signing model ensures that any post-download modification of the file's content (such as an attacker attempting to strip out embedded web beacons, modify financial digits, or sanitize binary strings) will alter the document's byte range. During subsequent verification, the cryptographic hash computed on the modified file will fail to match the signature hash protected by the public key, rendering any tampering immediately detectable. Under the eIDAS regulation [33], this implementation is structurally equivalent to an Advanced Electronic Signature (AdES). While it provides absolute mathematical integrity and non-repudiation within the organization's trust boundary, we acknowledge that a fully legally qualified signature (QES) would require integration with a certified external Trust Service Provider (TSP), which is designated as future work.

Verification returns one of three states, which the SOC dashboard renders distinctly: VALID, when the recomputed digest matches the signed byte range; INVALID, when the document has been altered after signing; and MISSING, when no signature field is present at all—itself a meaningful signal, since every document leaving the repository is signed by construction.

5.2. Immutable Audit Trail and Chain of Custody

A core requirement for forensic admissibility is the preservation of telemetry under a strict "Chain of Custody" protocol, documenting that log data has remained untampered

throughout its lifecycle [34]. To achieve this without the overhead of physical write-once media, our framework establishes an immutable audit trail utilizing Amazon S3 logical configurations.

All system events—including portal downloads, beacon exfiltration callbacks, behavioral anomalies, dynamic remediation executions, and unauthorized access attempts—are serialized as structured JSON records and written directly to a dedicated S3 log bucket. The following S3 configurations approximate write-once semantics:

- **Bucket Versioning**: Enforcing versioning ensures that any delete or modify API request on an existing log object merely generates a new logical version marker, preserving the historic record intact.
- **AES-256 Server-Side Encryption (SSE-S3)**: Protects the confidentiality of the audit trail at rest using keys managed directly by the S3 service [27].
- **Access Policy Constraints**: Restricts S3 write access solely to the serverless Lambda execution roles, while restricting read access to the security administrator role.

We note explicitly that this configuration provides *logical* rather than cryptographic immutability: it is enforced through version retention and access policy rather than through S3 Object Lock, which is not available in the emulated Community Edition. A production deployment seeking regulatory-grade WORM guarantees would enable Object Lock in compliance mode on the log bucket. This provides tamper-evidence, not WORM immutability against a privileged administrator; regulatory-grade immutability additionally requires trusted timestamps from a Qualified Trust Service Provider.

To guarantee interoperability and seamless ingestion into standard Security Information and Event Management (SIEM) platforms, the log schema is designed to be fully compatible with AWS CloudTrail formats [27]. Each JSON log payload explicitly records the complete operational context, including the user's IAM identity, precise timestamp, source IP, the requested S3 key, and the remediation state (e.g., 'ISOLATED' for revoked credentials).

This structured approach provides complete tracing: the log captures who downloaded the file, when, from what IP, and records that subsequent unauthorized attempts were actively blocked by the Security Mediator. This complete ledger complies directly with the forensic custody principles defined in NIST SP 800-86 [34].

5.3. Forensic Dossier Generation and Automated Compliance Auditing

To translate raw JSON events into human-readable evidence, the system implements an automated report-generation engine utilizing the ReportLab library [35]. Upon a security event or analyst request, it compiles six distinct forensic outputs into standardized PDFs:

- **Incident Dossier (Scenario A):** Employee IAM metadata, downloaded decoy name, timestamp, and source IP.
- **DLP Report (Scenario B):** Beacon UUID, external IP, GeoIP coordinates, Tor/VPN status, original downloader ID, dwell time, and remediation outcome.
- **Honeytoken Dossier (Scenario C):** Listing of exfiltrated decoy credentials and access timeline.
- **Behavioral Risk Report:** Breakdown of triggered sliding-window anomalies and employee risk ranking.
- **Compliance Report (NIS2/GDPR):** Live system auditing of S3 versioning, encryption, and remediation state.
- **Compliance Bundle:** ZIP archive containing all PDF reports and raw JSON CloudTrail events for judicial validation.

The *NIS2/GDPR Compliance Audit Report* represents a critical engineering contribution: instead of relying on subjective questionnaire responses, it dynamically queries the live emulated cloud infrastructure to verify security controls. It checks the versioning state of S3, verifies SSE-AES256 configurations, monitors the status of active Lambda functions, and evaluates the operational status of the outbound webhook notification channel. The final report computes an objective control coverage score, directly supporting regulatory audits under GDPR Article 25 and NIS2 Article 21 [7,8].

5.4. SOC Incident Dashboard and Situational Awareness

All detection telemetry, cryptographic verification states, and automated remediation logs converge into the Security Operations Center (SOC) dashboard. The interface is engineered to provide security analysts with high-fidelity situational awareness, defined as the ability to perceive anomalous events, comprehend their security impact, and project threat vectors in real time. Figure 2 shows the main view.

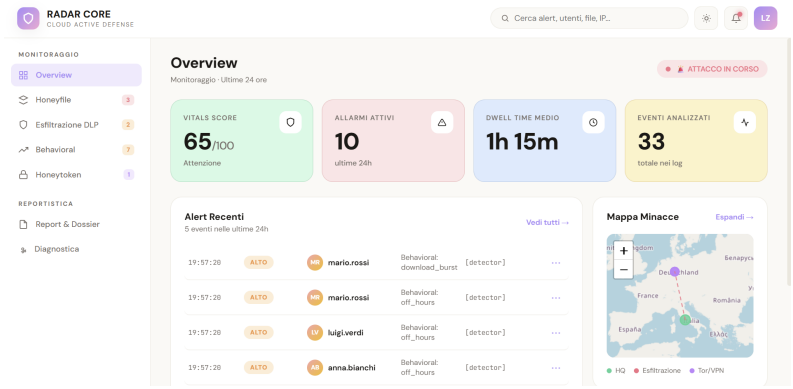


Figure 2. The SOC Dashboard Overview page. The central interface displays real-time key performance indicators (KPIs), an active threat alert panel, a chronological event log, and an interactive geographic map charting the external IP addresses of active web beacons. Card colors distinguish the four KPIs (green: vitals score; red: active alarms; blue: mean dwell time; yellow: analyzed events); the map legend identifies the marker types (corporate HQ, exfiltration endpoints, Tor/VPN origins).

The system is constructed upon a modern single-page-application (SPA) architecture utilizing React [36] and FastAPI [37]. Real-time geolocation of exfiltrated documents is visualized natively by plotting IP callbacks on interactive geographic tiles via the Leaflet library [38]. To ensure complete data consistency across the seven sections of the dashboard (Overview, Honeyfile, DLP Exfiltration, Behavioral Analytics, Honeytoken, Reports, and Diagnostics), the frontend employs a single-source-of-truth polling architecture: the root component executes a synchronized, periodic request to the FastAPI aggregator API, updating a single state tree propagated to every section. This prevents UI race conditions where different sections display conflicting counts or out-of-order logs.

The *Diagnostics Section* is crucial for operational transparency: it queries the real, live state of the cloud resources (such as testing read/write API reachability on LocalStack, checking the GeoIP2 binary database mount, and reading IAM policy bindings). This ensures that the analyst is always aware of system health, proving that active defense operates as a predictable, verifiable enterprise utility.

6. Experimental Evaluation and Performance Results

6.1. Experimental Setup and Reproducibility Methodology

To evaluate the performance, feasibility, and defensive posture of the proposed serverless deception framework, a fully functional prototype was validated on a containerized local

cloud emulation stack using LocalStack Community Edition (v3.8) [39] and Docker [40]. Conducting active defense experiments on production cloud tenants poses unnecessary operational risks and costs; thus, local emulation was chosen. The setup represents the document infrastructure of a financial firm, *Aurea Capital S.p.A.*, containing both authentic operational documents and interleaved decoy assets.

The local cloud stack emulates core Amazon Web Services (AWS) REST APIs to interact natively with key infrastructure components. Amazon S3 provides storage by hosting a secure vault (`company-secure-documents`) and an audit log repository (`company-secure-logs`) protected by SSE-AES256 encryption and bucket versioning. Access control is managed via AWS IAM, which enforces granular, identity-based permissions for individual emulated accounts rather than shared roles. The stateless "Radar" remediation and threat classification logic is executed through AWS Lambda. For notifications, Amazon SNS is provisioned as the native alerting topic; however, since the Community Edition lacks actual message delivery capabilities, alerts are routed via an outbound webhook to a corporate messaging channel. Finally, a host-side ingestion listener acts as an emulated API gateway, receiving and enriching remote web beacon callbacks prior to invoking the serverless function, as detailed in Section 4.3.

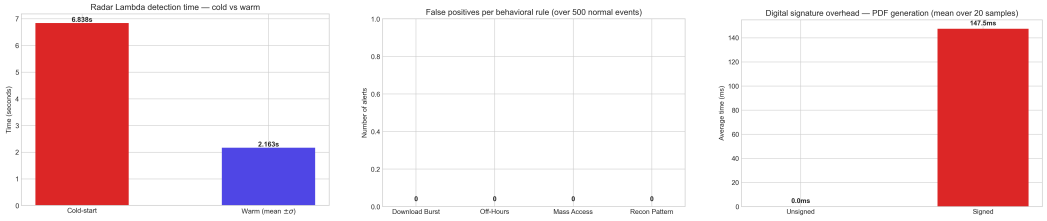
For scientific reproducibility, a Python orchestration script automates the entire system bootstrap, including container initialization, Lambda packaging, and dynamic decoy provisioning. To prevent signature detection by sophisticated attackers, the decoy generator uses seed-controlled instances of the Faker library [41] to generate realistic content, metadata, and author details dynamically, ensuring structural and semantic camouflage. Testing was managed by a benchmarking module executing 71 automated test cases across 12 operational components.

6.2. System Latency and Cold-Start Analysis

In active defense architectures, minimization of latency is paramount to prevent data exfiltration. We benchmarked the "Radar" Lambda detection loop over 11 consecutive execution trials, measuring the time elapsed between intercepting a web beacon callback, querying a local MaxMind GeoIP2 database, classifying the IP risk profile, writing a JSON audit log to S3, and detaching the IAM policies of the compromised user.

The benchmark revealed two distinct operational regimes (Figure 3a):

- **Cold-Start Latency**: The initial container execution requires **6.838 s**, representing the physical overhead of initializing the Python runtime and importing packages (such as `boto3` and GeoIP2 libraries) within the local containerized virtualization layer.
- **Warm-Start Latency**: Subsequent warm-start invocations drop to a stationary mean of **2.163 s** with a low standard deviation of **0.033 s**, demonstrating highly deterministic execution.



(**a**) Latency Breakdown (**b**) Classification Metrics (**c**) PAdES Signing

Figure 3. Experimental evaluation results: (**a**) AWS Lambda cold-start versus warm-start execution latency (2.163 s median); (**b**) classification specificity across 500 ordinary benign transactions showing a 0.00% false-positive rate; (**c**) offline PAdES (RSA-2048) digital signature provisioning overhead per document (147.5 ms median).

This sub-three-second response demonstrates that the detection-to-remediation loop—from the interception of the beacon callback to the IAM policy detachment—completes well within the window relevant to an ongoing data exfiltration. Within the LocalStack Community environment, this measurement captures the time required to issue the revocation, rather than the instant of policy enforcement.

6.3. Classification Metrics: Specificity, Sensitivity, and Alert Fatigue Mitigation

Traditional intrusion detection systems suffer from alert fatigue caused by false positives [24]. To demonstrate system specificity, we subjected the behavioral rule engine to a baseline dataset of **500 normal traffic events** representing routine, legitimate file access patterns from 6 employees during working hours (08:00–19:00). This baseline generated **zero false alerts (0.00% FPR)**, proving high resilience to noise, since no legitimate access pattern in the baseline ever crossed a configured threshold (Figure 3b).

To evaluate sensitivity, targeted malicious events were executed to trigger our four threshold-based rules:

- `download_burst`: A compromised account downloading > 10 files within a 5-min sliding window.
- `off_hours`: Document access outside the 08:00–19:00 window.
- `mass_access`: Multi-user accounts within the same department retrieving > 15 files in 30 min.
- `recon_pattern`: Quick sequential access to an authentic file and a decoy within 10 min.

The behavioral engine achieved a **100% recall rate (zero false negatives)** on all simulated attacks. Using deterministic thresholds instead of probabilistic models ensures complete auditable explainability. On this dataset the engine also attains an F1-score and specificity of 1.00; these figures reflect correctness on controlled data, not enterprise-wide generalization (future work). By design, legitimate-but-atypical activity (bulk or out-of-hours downloads) raises an analyst-facing alert, and accidental decoy access generates a forensic dossier for review—never an automatic sanction, which is reserved for the out-of-perimeter activation of a beacon on an authentic document (Scenario B). Low-frequency theft, beyond volumetric rules, is covered by the deception layer.

6.4. Cryptographic Signature Overhead

To prevent sophisticated insiders from distinguishing decoy assets from authentic ones, both files must carry identical cryptographic markings [15]. If only legitimate files were signed, the lack of a signature would expose the decoy. Our system utilizes an RSA-2048 private key and X.509 certificates to apply ETSI PAdES (PDF Advanced Electronic Signatures) baseline signatures [32].

We benchmarked the computational cost of applying these signatures across 20 trials. The results, plotted in Figure 3c, show that PAdES signing adds an average overhead of **147.5 milliseconds** per document. This overhead is incurred at provisioning and, for authentic PDFs, again at each retrieval, when the personalized copy is re-signed (Section 4.6), adding a comparable cost to the delivery of that copy.

6.5. Beacon Activation Rates Across PDF and Office Clients

To evaluate tracking capabilities, we embedded multi-format web beacons across different document types. PDF tracking used (i) transparent click-based link annotations and (ii) zero-click document-level JavaScript executed on open. Word files contained an XML-level `INCLUDEPICTURE` field, while Excel files contained an interactive `HYPERLINK` formula. These assets were opened in six standard application environments.

The experiments activated the beacon in all four PDF clients—two automatically and two on a user click—and in five of the six clients overall (Table 3).

A review of the blocked vectors shows that Microsoft Edge blocks external outbound requests to local emulated domains for security protection. Additionally, Microsoft Word suppresses the remote `INCLUDEPICTURE` field automatically to mitigate the "Follina" remote code execution vulnerability (CVE-2022-30190) [42], which abused external XML relations. Thus, beaconing must be combined with server-side download tracking to maintain reliable coverage. Beacon injection is imperceptible, but its activation is generally perceptible: desktop PDF readers raise a security prompt, and click-based paths open a browser navigation. The beacon is therefore a high-signal trigger rather than a covert sensor. When assessing detection effectiveness, the two activation modes should be considered separately: only the two automatic activations (Adobe Reader DC and Foxit) behave as genuine zero-interaction sensors, whereas the click- and navigation-based activations (Firefox, Edge, and Excel) depend on adversary interaction and thus provide weaker guarantees as passive sensors.

Table 3. Web beacon activation outcomes and rates across various clients.

Client Reader	Format	Activation Type	Outcome	Visibility
Adobe Reader DC	PDF	Automatic	Activated	Security prompt
Foxit PDF Reader	PDF	Automatic	Activated	Security prompt
Firefox PDF Viewer	PDF	Click	Activated	Browser nav.
Microsoft Edge	PDF	Click	Activated	Browser nav.
Microsoft Word 365	DOCX	None	Blocked	—
Microsoft Excel 365	XLSX	Click	Activated	User click

6.6. Security Coverage Mapping (MITRE ATT&CK & MITRE D3FEND)

To contextualize the defensive capabilities of our system, we mapped its operational components against the MITRE ATT&CK [43] and MITRE D3FEND [44] frameworks. In an insider threat scenario, adversaries already possess valid credentials, rendering early-stage defenses ineffective; consequently, our coverage focuses strictly on the Collection and Exfiltration phases. The system provides complete mitigation for *Data from Cloud Storage (T1530)* and *Unsecured Credentials (T1552)* through S3 access logging and honeytoken placement, while automated Tor/VPN classification addresses *Multi-hop Proxy (T1090.003)* and *External Proxy (T1090.002)*. Embedded web beacons and behavioral rules (`mass_access`, `recon_pattern`, `off_hours`) offer partial to compensative coverage against *Exfiltration to Cloud Storage (T1567.002)*, *Exfiltration over C2 (T1041)*, *File and Directory Discovery (T1083)*, and the abuse of *Valid Accounts (T1078)*. Correspondingly, these active controls map cleanly to D3FEND countermeasures: *File Hashing* and *Document Integrity Enforcement* are achieved via PAdES SHA-256 signatures; *Decoy Files/Tokens* are realized through high-fidelity S3 honeyfiles; *Network Traffic Filtering* relies on automated IP classification; and *Identity-based Authentication Action* paired with *Account Locking* is executed via real-time IAM policy revocation.

7. Discussion and Critical Analysis

7.1. Interpretation of Results and Comparison with Traditional Intrusion Detection Systems

Our findings demonstrate that combining serverless active deception with a threshold-based behavioral rule engine addresses the structural vulnerabilities of passive intrusion detection models, such as standard SIEMs and statistical UEBA platforms. Traditional passive methods rely on post-hoc analysis of massive event logs [20], leading to severe alert fatigue and high false-positive rates [24].

Furthermore, traditional perimeter defenses fail to address insider threats, as malicious actors traverse legitimate paths with valid credentials. This produces the "Dwell Time

Paradox" [5], where threats remain undetected for weeks, conflicting with strict European regulations (GDPR, NIS2, DORA) that require incident notification within 24–72 h.

Our framework bypasses this temporal asymmetry. Deploying high-fidelity decoy assets (honeyfiles and honeytokens) introduces a zero-use operational constraint [11]. Any interaction with a decoy represents an unambiguous threat signal, eliminating false positives (0.00% FPR over 500 events). Simultaneously, the serverless "Radar" function issuing the containment request (IAM policy revocation) for compromised accounts in 2.163 s in the emulated environment reduces the detection-to-response window from days to seconds, yielding legally auditable and compliant system logs.

7.2. System Limitations and Bypass Vectors

A comprehensive STRIDE-based [45] evaluation reveals key structural boundaries. First, beacon telemetry is inherently neutralized by **offline execution** (air-gapped environments) or **client-side sandboxing** (e.g., Office 365 or Edge suppressing cross-origin connections to prevent exploits like CVE-2022-30190 [42]). Second, our current **static network intelligence**, which relies on Tor/VPN snapshots, is vulnerable to dynamic routing or private VPS bypasses. Third, **cryptographic gaps** exist because PAdES signatures are restricted to PDFs; Word and Excel documents lack native PAdES structures, making them susceptible to silent beacon sanitization. Finally, static beacon URLs are vulnerable to **replay and DoS attacks**. In addition, attribution is bound to the principal who downloaded a given copy and not to a third party who may later receive and open it; when a document is shared before being opened outside the perimeter, the automated revocation therefore acts on the original downloader rather than on the individual who performed the exfiltration. Future iterations will address these boundaries by implementing single-use signed URLs, real-time threat intelligence APIs, and hybrid integration with endpoint DLP agents.

7.3. Legal and Privacy Implications

Under European frameworks [7,8], proactive security must enforce data minimization (GDPR Art. 25). Unlike invasive literature proposals relying on sentiment analysis or psychological profiling [31]—which are highly subjective and legally restricted—our framework evaluates strictly **observable operations** (e.g., download volumes, timestamps). It utilizes deterministic, **transparent thresholds** rather than black-box models, ensuring that any automated account suspension is fully explainable. **Data minimization** is guaranteed by triggering IP geolookups exclusively for out-of-perimeter access. Finally, the forensic evidence is cryptographically verifiable within a defined scope. The PAdES signatures guarantee the integrity and authenticity of the instrumented documents, rendering any post-signing alteration detectable. However, this property does not extend to the beacon callback. Similarly, the versioned audit trail provides tamper-evidence rather than strict WORM immutability. Within these boundaries, the integration of signed artifacts with a version-controlled log establishes a chain of custody [34] that aligns with GDPR Art. 32 and NIS2 Art. 21.

8. Conclusions and Future Directions

8.1. Conclusions

This research was initiated to address a fundamental architectural contradiction in modern enterprise cloud environments: while European regulatory frameworks—such as GDPR [7], NIS2 [8], and DORA [9]—impose strict, rapid incident notification windows (ranging from 24 to 72 h), traditional passive monitoring infrastructures require days or weeks to detect internal data exfiltration. Passive systems are retrospectively focused, attempting to identify malicious anomalous behavior within massive volumes of legitimate

transaction logs after the exfiltration has already occurred. This study has demonstrated that resolving this temporal asymmetry requires shifting the defensive paradigm from passive, boundary-based monitoring to proactive content-level deception and active defense. By designing and implementing a serverless deception layer that blends dynamically generated decoy assets (honeyfiles and honeytokens) with remote tracking capabilities (invisible web beacons) and rigorous cryptographic validation (PAdES digital signatures), we successfully inverted the operational dynamic, forcing the attacker to reveal their presence through the simple, unambiguous act of interacting with zero-use corporate resources [11].

The core scientific contribution of this work lies not in the invention of individual security mechanisms, but in their verifiable, systematic composition into a unified, serverless active defense fabric. The containerized LocalStack prototype representing the financial workflows of *Aurea Capital S.p.A.* proved that these isolated technologies can be securely chained together: an out-of-perimeter beacon activation triggers real-time out-of-band detection, leading to rapid, identity-based IAM auto-remediation and the immediate generation of forensic evidence. Empirically, the framework demonstrated a highly stable warm-start response latency of **2.163 s** (the time to issue the IAM revocation request in the emulated environment), a **0.00% false positive rate** over 500 normal transaction events, a **100% recall rate** on modeled malicious behavioral patterns, and beacon activation across all tested desktop PDF readers. By declaring system limitations—such as client-side sandboxing, browser security policies, and offline execution—through formal STRIDE threat modeling, this paper establishes a realistic, robust, and transparent benchmark for future implementations of cyber deception in enterprise cloud storage.

8.2. Future Directions

Based on the experimental evaluation and identified system boundaries, we propose a concrete roadmap for future technical developments, organized across three distinct operational areas:

- **Cryptographic and Regulatory Hardening**: To transition the prototype to a production-ready, highly secure posture, the primary objective is to migrate private cryptographic keys from the host filesystem to a managed hardware security module (HSM) or AWS Key Management Service (KMS) [27], ensuring the cryptographic root of trust is never exposed in memory. Additionally, integrating trusted, independent timestamps and replacing the self-signed enterprise certificate with one issued by a Qualified Trust Service Provider (QTSP) would promote the appended digital signatures to Qualified Electronic Signatures (QES) under the eIDAS Regulation (Regulation 910/2014) [33], ensuring absolute judicial admissibility. Finally, extending cryptographic signing capabilities from PDF to XML-based OOXML formats (DOCX and XLSX) would resolve current document validation gaps.
- **Deception and Telemetry Robustness**: The tracking capabilities of the embedded web beacons can be significantly reinforced by replacing static URLs with single-use, dynamically signed token parameters to prevent replay and denial of service (DoS) attacks. Furthermore, migrating the correlation key from static filenames to cryptographic content hashes (such as SHA-256) would prevent attackers from bypassing the dwell-time calculation and detection mapping by simply renaming the exfiltrated documents. On the network side, integrating real-time, commercial threat intelligence APIs would resolve the latency of static Tor and VPN exit node lists, providing the serverless engine with dynamic, high-fidelity IP classification metrics.
- **Observation Scope and Anomaly Analytics**: To bypass the physical limitations of network-dependent beacon callbacks, future research should investigate pairing the cloud serverless architecture with lightweight, host-based enterprise Data Loss Pre-

vention (DLP) agents on endpoints. This hybrid approach would capture offline document openings and track internal document read events that do not trigger S3 download logs. Additionally, the honeytoken framework can be transformed from an access-based trigger to a use-based sensor by actively monitoring and correlating AWS CloudTrail API call logs for the unauthorized utilization of decoy credentials against cloud services. Finally, once real-world, labeled enterprise datasets become available, the static, threshold-based behavioral rules can be augmented with unsupervised machine learning models to detect subtle, distributed masquerader anomalies without sacrificing the core explainability of the detection output.

Author Contributions: Conceptualization, L.P.Z., C.Z. and M.M.; methodology, C.Z. and S.T.; software, L.P.Z., C.Z. and S.T.; validation, S.T.; formal analysis, L.P.Z.; investigation, C.Z.; resources, M.M.; data curation, S.T.; writing—original draft preparation, L.P.Z. and C.Z.; writing—review and editing, C.Z. and M.M.; visualization, S.T. and G.M.; supervision, M.M.; project administration, M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: During the preparation of this manuscript, the authors used Gemini 3.1 Pro strictly for language polishing and structural formatting of the English text. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Software Engineering Institute. *Common Sense Guide to Mitigating Insider Threats*, 7th ed.; Report; Software Engineering Institute, Carnegie Mellon University: Pittsburgh, PA, USA, 2022.
2. Orange. Distribution of Threat Actors Detected in Cyber Incidents Worldwide as of September 2025, by Type. Available online: https://www.statista.com (accessed on 20 August 2026).
3. Cybersecurity Insiders; Gurucul. *Insider Threat Report 2024*; Report; Cybersecurity Insiders: Baltimore, MD, USA, 2024. Available online: https://gurucul.com/2024-insider-threat-report (accessed on 20 August 2026).
4. Gopalakrishna, R.; Bhaskaran, R. Experience paper: Uncovering a privileged insider threat in action. In Proceedings of the 2025 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Venice, Italy, 30 June–4 July 2025; pp. 656–661.
5. Mandiant. *M-Trends 2026 Report: Real-World Investigations and Actionable Defense Insights*; Report; Mandiant, Google Cloud: Reston, VA, USA, 2026. Available online: https://cloud.google.com/security/resources/m-trends (accessed on 20 August 2026).
6. Verizon. *2025 Data Breach Investigations Report*; Report; Verizon: New York, NY, USA, 2025. Available online: https://www.verizon.com/business/resources/reports/dbir/ (accessed on 20 August 2026).
7. European Parliament and Council of the European Union. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data (GDPR). *Off. J. Eur. Union* **2016**, *L 119*, 1–88. Available online: https://eur-lex.europa.eu/eli/reg/2016/679/oj (accessed on 20 August 2026).
8. European Parliament and Council of the European Union. Directive (EU) 2022/2555 on measures for a high common level of cybersecurity across the union (NIS 2). *Off. J. Eur. Union* **2022**, *L 333*, 80–152. Available online: https://eur-lex.europa.eu/eli/dir/2022/2555/oj (accessed on 20 August 2026).
9. European Union. Regulation (EU) 2022/2554 on digital operational resilience for the financial sector (DORA). *Off. J. Eur. Union* **2022**, *L 333*, 1–79. Available online: https://eur-lex.europa.eu/eli/reg/2022/2554/oj (accessed on 20 August 2026).
10. The MITRE Corporation. *MITRE Engage Handbook*; Handbook; MITRE: McLean, VA, USA, 2022. Available online: https://engage.mitre.org (accessed on 20 August 2026).
11. Spitzner, L. Honeypots: Catching the insider threat. In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC), Las Vegas, NV, USA, 8–12 December 2003; pp. 170–179.

12. National Institute of Standards and Technology. *The NIST Cybersecurity Framework (CSF) 2.0*; NIST Cybersecurity White Paper CSWP 29; U.S. Department of Commerce: Washington, DC, USA, 2024.
13. Yuill, J.; Zappe, M.; Denning, D.; Feer, F. Honeyfiles: Deceptive files for intrusion detection. In Proceedings of the Fifth Annual IEEE SMC Information Assurance Workshop, West Point, NY, USA, 10–11 June 2004; pp. 116–122.
14. Whitham, B. Automating the generation of enticing text content for high-interaction honeyfiles. In Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS), Hilton Waikoloa Village, HI, USA, 4–7 January 2017; pp. 6069–6078.
15. Bowen, B.M.; Hershkop, S.; Keromytis, A.D.; Stolfo, S.J. Baiting inside attackers using decoy documents. In *Security and Privacy in Communication Networks (SecureComm 2009)*; Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Springer: Berlin/Heidelberg, Germany, 2009; Volume 19.
16. Timmer, R.C.; Liebowitz, D.; Nepal, S.; Kanhere, S.S. Honeyfile camouflage: Hiding fake files in plain sight. In Proceedings of the 3rd ACM Workshop on the Security Implications of Deepfakes and Cheapfakes (WDC '24), Singapore, 1–5 July 2024.
17. Kamaruddin, N.S.; Kamsin, A.; Por, L.Y.; Rahman, H. A review of text watermarking: Theory, methods, and applications. *IEEE Access* **2018**, *6*, 8011–8028. [CrossRef]
18. Miorandi, D.; Rizzardi, A.; Sicari, S.; Coen-Porisini, A. Sticky policies: A survey. *IEEE Trans. Knowl. Data Eng.* **2020**, *32*, 2481–2499. [CrossRef]
19. Sim, K.; Heo, H.; Cho, H. Combating web tracking: Analyzing web tracking technologies for user privacy. *Future Internet* **2024**, *16*, 363. [CrossRef]
20. Homoliak, I.; Toffalini, F.; Guarnizo, J.; Elovici, Y.; Ochoa, M. Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Comput. Surv.* **2019**, *52*, 30.
21. Ben Salem, M.; Hershkop, S.; Stolfo, S.J. A survey of insider attack detection research. In *Insider Attack and Cyber Security: Beyond the Hacker*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 39.
22. Berdychowski, M.; Salinas Monroy, S.A. Automated detection of masquerade attacks with AI and decoy documents. In Proceedings of the 2024 Cyber Awareness and Research Symposium (CARS), Grand Forks, ND, USA, 28–29 October 2024.
23. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 1–58. [CrossRef]
24. Schonlau, M.; DuMouchel, W.; Ju, W.H.; Karr, A.F.; Theus, M.; Vardi, Y. Computer intrusion: Detecting masquerades. *Stat. Sci.* **2001**, *16*, 58–74. [CrossRef]
25. Thinkst Applied Research. Canarytokens. Available online: https://canarytokens.org (accessed on 15 August 2026).
26. Cybersecurity Insiders; Securonix. What Motivations Behind Malicious Insider Threats Are You Most Concerned About? Available online: https://www.statista.com (accessed on 21 August 2026).
27. Amazon Web Services. *Security Pillar: AWS Well-Architected Framework*; Whitepaper; Amazon Web Services, Inc.: Seattle, WA, USA, 2024. Available online: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/ (accessed on 20 August 2026).
28. National Institute of Standards and Technology. *Zero Trust Architecture*; NIST Special Publication (SP) 800-207; U.S. Department of Commerce: Washington, DC, USA, 2020.
29. National Institute of Standards and Technology. *Computer Security Incident Handling Guide*; NIST Special Publication (SP) 800-61 Rev. 3; U.S. Department of Commerce: Washington, DC, USA, 2024.
30. Dingledine, R.; Mathewson, N.; Syverson, P. Tor: The second-generation onion router. In Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA, 9–13 August 2004; pp. 303–320.
31. Levy, M.; Horneman, A.; Ditmore, R.; Motell, C. Predicting the threat: Investigating insider threat psychological factors with advanced natural language processing. In Proceedings of the 55th Hawaii International Conference on System Sciences (HICSS), Maui, HI, USA, 4–7 January 2022.
32. *EN 319 142-1 V1.2.1*; Electronic Signatures and Infrastructures (ESI); PAdES Digital Signatures; Part 1: Building Blocks and PAdES Baseline Signatures. ETSI: Sophia Antipolis, France, 2024.
33. European Parliament and Council of the European Union. Regulation (EU) no 910/2014 on electronic identification and trust services for electronic transactions in the internal market (eIDAS). *Off. J. Eur. Union* **2014**, *L 257*, 73–114.
34. Kent, K.; Chevalier, S.; Grance, T.; Dang, H. *Guide to Integrating Forensic Techniques into Incident Response*; NIST Special Publication (SP) 800-86; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2006.
35. ReportLab Europe Ltd. ReportLab PDF Toolkit Documentation. Available online: https://docs.reportlab.com (accessed on 20 August 2026).
36. Meta Platforms Inc. React: The Library for Web and Native User Interfaces. Available online: https://react.dev (accessed on 20 August 2026).
37. Ramírez, S. FastAPI Documentation. Available online: https://fastapi.tiangolo.com (accessed on 21 August 2026).
38. Agafonkin, V. Leaflet: An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps. Available online: https://leafletjs.com (accessed on 20 August 2026).
39. LocalStack. LocalStack—A Fully Functional Local Cloud Stack. Available online: https://github.com/localstack/localstack (accessed on 21 August 2026).

40. Docker, Inc. Docker Desktop. Available online: https://www.docker.com (accessed on 21 August 2026).
41. Faker. Faker Python Package Documentation. Available online: https://faker.readthedocs.io (accessed on 21 August 2026).
42. Microsoft. Guidance for CVE-2022-30190 Microsoft Support Diagnostic Tool Vulnerability. Available online: https://msrc.microsoft.com (accessed on 21 August 2026).
43. The MITRE Corporation. MITRE ATT&CK Framework. Available online: https://attack.mitre.org (accessed on 20 August 2026).
44. The MITRE Corporation. MITRE D3FEND KB: A Knowledge Graph of Cybersecurity Countermeasure Techniques. Available online: https://d3fend.mitre.org (accessed on 21 August 2026).
45. Microsoft. The STRIDE Threat Model. Available online: https://learn.microsoft.com (accessed on 21 August 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.