elastic security labs

2025 State of Detection Engineering at Elastic

Table of contents

Introduction

The global threat landscape is dynamic, reactive, and overwhelming. As threat actors launch attack after attack, they force security teams to shift their attention and resources on a regular basis. The best way to address, and even preempt these threats, is detection engineering — a practice that teams rarely have enough resources for.

Effective detection engineering demands a detailed understanding of the attack surface and tools in an organization. Security teams must rely — sometimes exclusively — on their tools to make up for gaps in resources. In an industry that is so demanding of expertise, security vendors have an obligation to deliver and continuously innovate upon quality products that alleviate security teams from a siloed approach.

Elastic Security Labs remains dedicated not only to innovating upon our Elastic Security solution with research and protections, but also empowering the security community at large by driving discussions and sharing practices. This report will give our users an in-depth look at how we create, maintain, and assess our rulesets within Elastic Security, providing broader context into detection engineering and how teams may benefit. Our expertise is made available for the entire security community, regardless of familiarity with or patronage of our technology.

The 2025 State of Detection Engineering at Elastic explores a full year of our detection engineering efforts: October 2023 - October 2024. This time frame was chosen to incorporate our work following the 2023 Elastic Global Threat Report and elicit enough data for patterns to emerge.

As such, the way we manage detection engineering for the Elastic Security solution will look different from your own practices, but we hope that by sharing what we do, we set the stage for an in-depth discussion on the practice of detection engineering. We hope that you'll join us in this conversation.

Part 1

Detection engineering in practice

The State of Detection Engineering at Elastic begins where we begin: with our rulesets. Elastic Security Labs maintains SIEM Detection and Endpoint Protection rulesets. Our SIEM Detection rules cover a broad range of threats across operating systems and 3rd party platforms like AWS, GitHub and Kubernetes. Our Endpoint Protection rules prevent ransomware and malware, detect advanced threats, and arm responders with vital context using our endpoint agent.

Regardless of the ruleset, there are many different strategies for developing rules. We'll begin by sharing how we utilize real-world threat campaigns — sometimes exposed by our very own colleagues! — to shape our focus. From there, we'll transition into our proactive designs for future-proofing these rules: our development lifecycle. We'll explore how a multi-pronged approach to detection engineering, combining real-world threat analysis and robust rule development, can enhance threat detection capabilities.

1.1 Real-world threat analysis

Elastic's detection engineering strategies are deeply influenced by the retrospective analysis of real-world threat behavior, security telemetry from triggered alerts, and detonated malware samples. This hindsight analysis and historical data help us to refine our existing detections, develop new rules in response to evolving adversaries, and identify previously undetected threats. We'll highlight a couple key examples of our approach.

The CUPS vulnerability

On September 26, 2024, security researcher Simone Margaritelli (@evilsocket) disclosed a group of critical vulnerabilities in CUPS (Common Unix Printing System) utilities. These vulnerabilities — CVE-2024-47076, CVE-2024-47176, CVE-2024-47175, and CVE-2024-47177 — enabled unauthenticated remote code execution (RCE) on widely used UNIX-based systems, including most GNU/Linux distributions, BSDs, ChromeOS, and Solaris.

In response, we promptly assembled our team of detection engineers and researchers to methodically analyze the emerging threat. Prioritizing user protection, we temporarily redirected focus from other projects to conduct a thorough assessment of the vulnerabilities.

Our team:

- Reviewed available threat intelligence to fully understand the exploit chain.
- Obtained a proof of concept (PoC) to simulate real-world exploitation.
- Tested the PoC in a controlled test environment with vulnerable endpoints to determine detection strategies.

We focused on two primary attack scenarios:

1. Using living-off-the-land (LOL) techniques to establish a reverse shell
2. Retrieving and executing a remote payload to achieve RCE

While emulating these attack scenarios, we assessed the impacts on our system and analyzed activity logs for process activity and command execution patterns. These insights informed our detection strategy and remediation steps for customer guidance.

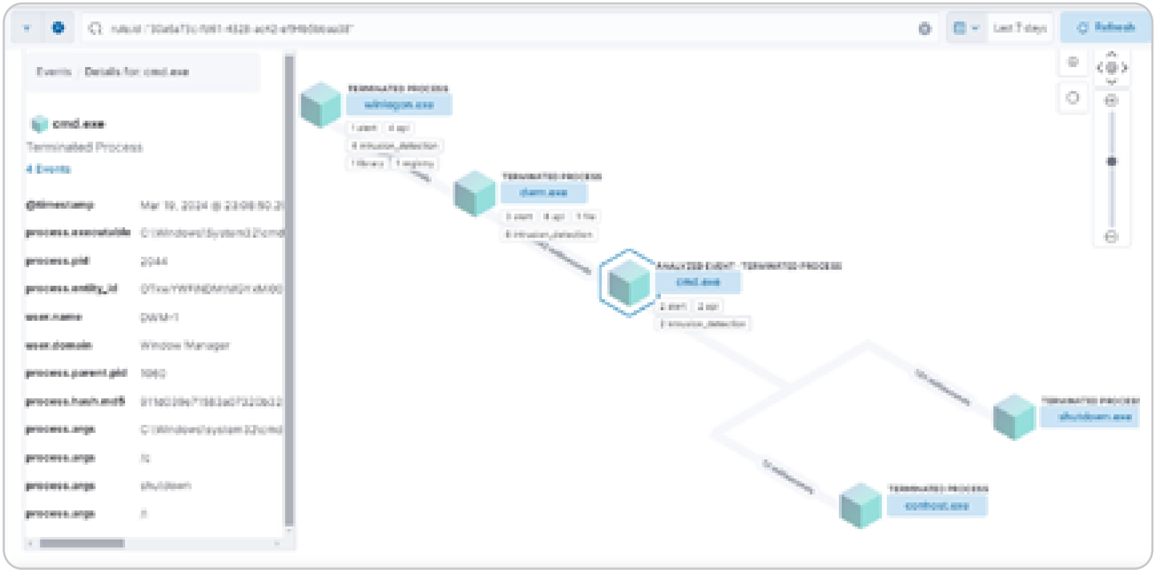
By leveraging our streamlined processes for rule development, deployment, and content publishing, we delivered detection coverage and public guidance for the community. While part of the team worked on writing and validating detection rules, others documented the findings for an informative article on Elastic Security Labs — which included remediation recommendations. Rules like *Cupsd or Foomatic-rip Shell Execution*, which detects shell executions from the foomatic-rip parent process, captured all 33 of our PoC attempts.

Local Privilege Escalation on Windows

Last March, our team examined publicly disclosed zero-day vulnerabilities targeting the Windows Common Log File System (CLFS) and Desktop Window Manager (DWM) Core Library. We analyzed exploitation methods to identify behavioral patterns indicative of Privilege Escalation attempts. By focusing on the dynamic behaviors of these exploits — such as the creation of BLF files followed by unexpected system-level activities — we developed a resilient detection strategy for these complex vulnerabilities that included both behavioral-based detection rules and signature-based YARA rules.

Specifically, we designed high-level behavioral detections by correlating file manipulation events involving `clfsw32.dll` APIs (`CreateLogFile` and `AddLogContainer`) with subsequent unexpected system integrity-level process activity (i.e. spawning a system child process, API call, file, or registry manipulation with system privileges). In parallel, we leveraged YARA to hunt for unsigned Portable Executable (PE) files that import the same user mode APIs and an atypical number of functions from `clfsw32.dll`. While targeted detections for highly exploited Windows components like CLFS and win32k provide valuable coverage, detecting Privilege Escalation attempts across a broad range of exploits requires a strategy that extends beyond individual vulnerabilities.

Adversaries frequently reuse core exploitation techniques across different vulnerabilities. As a result, investing in behavior-driven detection mechanisms — such as identifying Kernel address space layout randomization (KASLR) bypass attempts, token swapping, and PreviousMode abuse — offers wider coverage and long-term resilience against evolving privilege escalation techniques. More on how we explored these samples can be found in In-the-Wild Windows LPE 0-days: Insights & Detection Strategies.


Events Details for cmd.exe

cmd.exe
Terminated Process
4 Events

```
@timestamp            Mar 19, 2024 @ 23:08:50.2
process.executable    C:\Windows\System32\cmd
process.pid           2044
process.entity_id     [illegible]
user.name             DWM-1
user.domain           Window Manager
process.parent.pid    1060
process.hash.md5      [illegible]
process.args          C:\Windows\system32\cmd
process.args          /c
process.args          shutdown
process.args          /r
```

Figure 1: We can also look for unusual activity in DWM by baselining child processes and file activity

Based on our telemetry visibility, `dwm.exe` rarely spawns legitimate child processes. Figure 1 is an example of `dwm.exe` spawning `cmd.exe` as a result of exploitation. To further elevate privileges, the shellcode triggers a logoff by executing the `shutdown /l` command, which triggers the execution of the `LogonUI.exe` process running as a SYSTEM user. The two main detection points here occur when `dwm.exe` drops a PE file to disk and when `LogonUI.exe` loads a DLL, with the call stack pointing to `dcomp.dll` — an indicator of marshaling/unmarshaling Direct Composition objects.

Retrospective detection engineering is a cycle of research, analysis, validation, and adaptation. It goes beyond developing better detections — it depends on continuous engagement with exploit and vulnerability research. Engineers must understand not just how an exploit works, but how its effects manifest within system telemetry across diverse threat scenarios. By closely examining in-the-wild threat behavior, analyzing past incidents for detection gaps, and leveraging public repositories like VirusTotal to validate and refine detection logic, we ensure that our defenses remain adaptable and forward-looking.

1.2 Robust rule development

Our detection engineering strategy, while rooted in retrospective analysis, has evolved to be future-focused in nature. We have developed techniques to automate rule schema validation, refine our rule development process, and proactively detect emerging threats. This approach shapes everything from our internal rule deployment processes to novel uses of industry frameworks — ultimately ensuring that our detection capabilities evolve in step with a changing threat landscape.

Rule validation and detections-as-code

A major part of the work we do as detection engineers is maintaining our rules through a lifecycle. We've implemented automated query and rule validation into our Continuous Integration (CI) workflow in GitHub. We automatically validate rule syntax across multiple stack and integration versions, test detection efficacy against sample datasets, and ensure rules are fully functional before deployment. This is a process we have done for years across all of Elastic's supported query languages, and we recently added Elasticsearch Query Language (ES|QL) — Elastic's piped query language — into many of these routines. We committed to doing the same with ES|QL because automated rule validation reduces the risk of introducing noisy or inaccurate rules into production environments for our users. Furthermore, the streamlined validation process helps our team iterate faster by closing the feedback loop between rule creation, testing, and deployment.

In addition to query and rule validation, we've always applied a detections-as-code (DaC) approach to rule management. DaC allows detection rules to be treated as software artifacts, subject to the same robust development practices used in traditional software engineering. Key actions here include automated testing, peer review, and version control. These help catch errors and inefficiencies before they impact our users and enable better collaboration and consistency across environments. Version control at the rule level streamlines updates allowing individual rule changes to be tracked and rolled back if necessary.

There has been significant refactoring and updating to decouple components originally designed for internal use only, to allow for adoption by the community and users. For more guidance, Detections as Code Reference describes principles to adopt a DaC approach to Elastic Security rule management.

Together, these practices allow us to manage our ever-growing ruleset at scale, accelerate our internal release cycles, minimize disruptions from erroneous rules, and facilitate a more agile response to evolving threats.

Maturing rules with the DEBMM

Even with advanced CI and DaC, detection engineering can suffer from misaligned priorities and an inconsistent approach to rule maturity. To systematically address these pitfalls, we introduced the Detection Engineering Behavioral Maturity Model (DEBMM) — a universal model used to assess and mature processes and behaviors of security teams. Spanning five maturity stages (Foundation, Basic, Intermediate, Advanced, and Expert), the DEBMM provides benchmarks for behaviors like telemetry integration, rule management, and continuous refinement based on adversary insights.

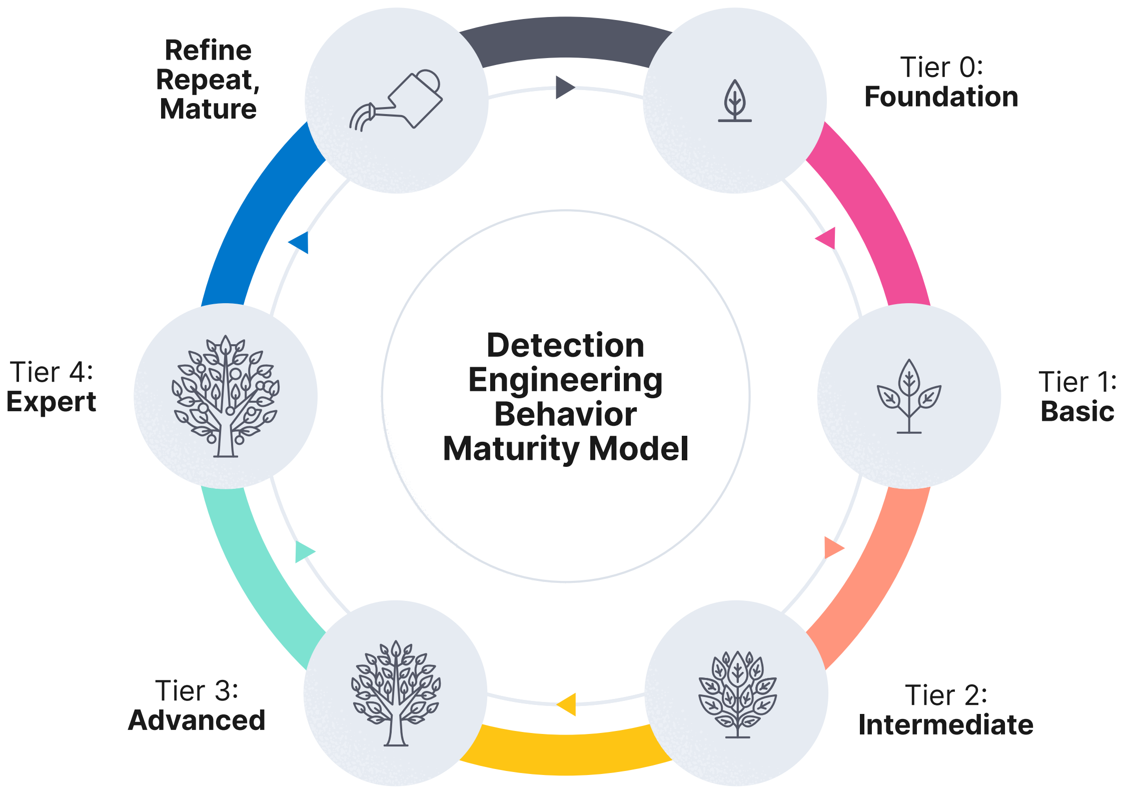


Figure 2: The Detection Engineering Behavior Maturity Model

We are currently using this model to build a framework for how we track the maturity of our rulesets, starting with Azure. While we have a good set of standard rules available, we know this ruleset needs some attention. Our assessment began with an acknowledgement of its current state: not actively maintained, with sparse documentation, and thus initially assessed as Tier 0. Despite this, the existence of standard rules from community contributions and Elastic researchers provided a foundation for evaluation.

The assessment focused on four key criteria: structured rule development and management, creation and maintenance of detection rules, roadmap documentation, and threat modeling. Both qualitative and quantitative measures were evaluated, revealing areas where existing practices, such as schema validation and peer reviews within the Detection Rules repository, provided some maturity, while other areas, like regular rule creation and updates, lacked consistency. This process identified specific improvements needed and led to the development of a detailed plan with prioritized tasks to progress toward Tier 1 maturity.

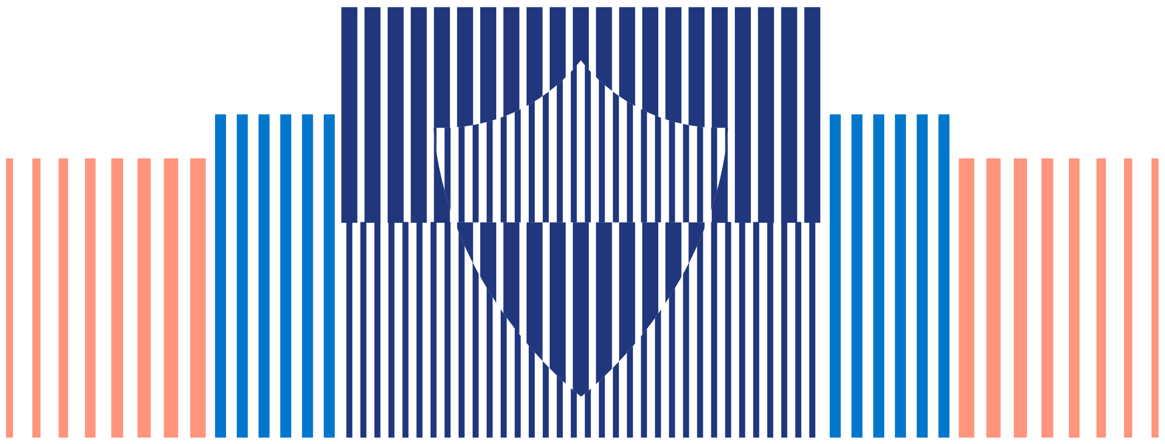
Advancing rule development through proactive threat hunting

While we dedicate substantial effort to building robust detections, we recognize that alerting on malicious behavior is only one part of an effective overall approach. Proactive threat hunting offers a deeper understanding of an environment by uncovering tactics and anomalies that might otherwise go unnoticed. Internally, we use a curated library of hunting queries, which are routinely run against our alert telemetry clusters. When these queries return potentially suspicious leads to our dedicated Slack channel, our detection engineers investigate further. From there, we may refine existing rules or develop new ones based on the patterns we discover.

Additionally, we share a public set of threat hunting queries alongside our detection rules, giving the community a structured approach aligned with the methods our team relies on. We view rules and hunting queries as complementary: once a rule fires an alert, correlated hunting queries can offer additional context or act as pivot points during incident triage. Likewise, some hunting logic may capture potentially malicious behaviors that don't yet meet the threshold for a universal detection rule — but still provide valuable insight for analysts. Our goal in publishing these hunting queries is to enable other security teams to tailor them to their own environments.

One notable example occurred in September 2024, when one of our researchers spotted suspicious behavior in our Linux alert telemetry. Further investigation revealed an active threat (later designated activity group REF6138) had compromised a customer's Linux server. By partnering with our threat research team, we reverse-engineered the newly discovered malware, created YARA signatures to detect the malware family, and enhanced our behavioral SIEM and Endpoint Protection rules to cover its unique adversary tactics, techniques, & procedures (TTPs); as described under the MITRE ATT&CK framework. We also published a comprehensive report, Betting on Bots, that detailed the campaign's methods and provided defense recommendations.

Robust rule management processes and DaC provide the technical backbone for efficiently deploying and managing detection rules, while the DEBMM provides a systematic approach to consistently improve and mature those rulesets. What sets Elastic apart is not just the introduction of these tools, but how we strategically incorporate these elements into the detection engineering lifecycle as an operational approach. Query validation and CI workflows ensure precision at the foundational level, DaC enables scalable rule management, and threat hunting delivers both enhanced context and a proactive approach to threat detection. These future-proof methodologies provide not just better tools but a smarter, more scalable approach to detection engineering.



Part 2

Enhancing Elastic Security

Over the past year we've significantly expanded our detection engineering coverage by utilizing various integrations available within the Elastic Security ecosystem and improving data sources within our endpoint agent. A key enabling factor in these efforts has been Elastic's commitment to developing and enhancing both Elastic's core protection tools — like the Elastic Defend integration for our endpoint agent — and widely adopted third-party platforms, such as AWS, Okta, and GitHub.

As detection engineers and security researchers, we collaborate with our endpoint agent and integration developers to enrich existing data sources with the most pertinent context required for effective threat detection. This section of the SDEE will explore some of those enhancements and their impact on detection engineering at Elastic.

2.1 Integration enrichment

The detection capabilities of our SIEM solution are rooted in both Elastic's detection engine — which allows us to build out our rulesets — and the diverse landscape of supported integrations. With more than 450 to choose from, we can collect, visualize, and query any data from any source.

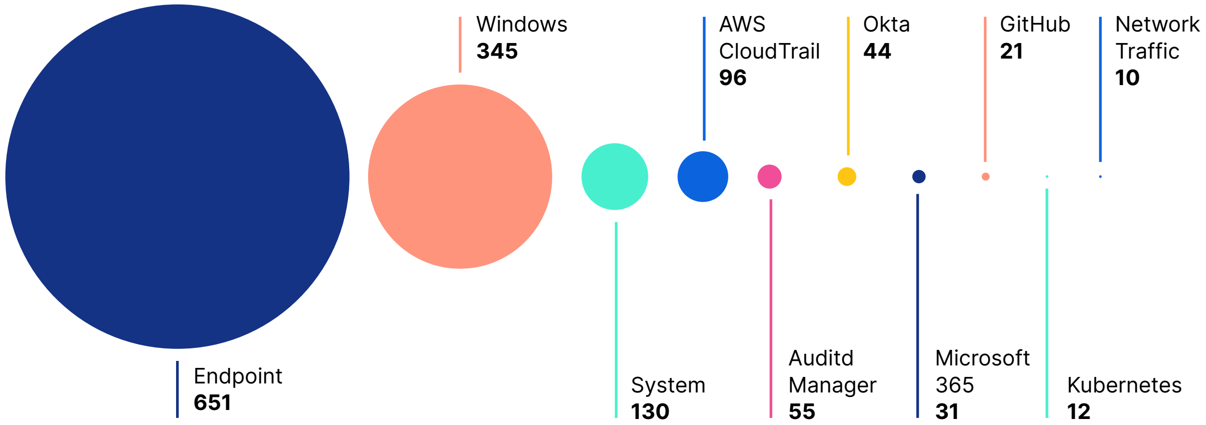


Figure 3: Top 10 integrations used for new and tuned rules

We focused our efforts this year on a broad range of security topics related to endpoint, cloud, and software-as-a-service (SaaS). As new threats emerged, we emulated techniques with existing frameworks, detonated malware, exploited new vulnerabilities, and created threat simulation tools for internal use. These tools include Panix, a Linux Persistence tool developed by Ruben Groenewoud, and SWAT, a Google Workspace red teaming tool developed by Terrance DeJesus and Justin Ibarra. Both of these are maintained by their creators, but are available for public use. Let's explore some of the integrations we've used this year, a few of the enhancements made to those integrations, and the resulting detection engineering outcomes.

Cloud and SaaS integrations

Cloud and SaaS was a major area of focus for us this year — 15% of new and tuned rules were related to these platforms, with AWS CloudTrail and Okta integrations accounting for nearly 70% of those rules.

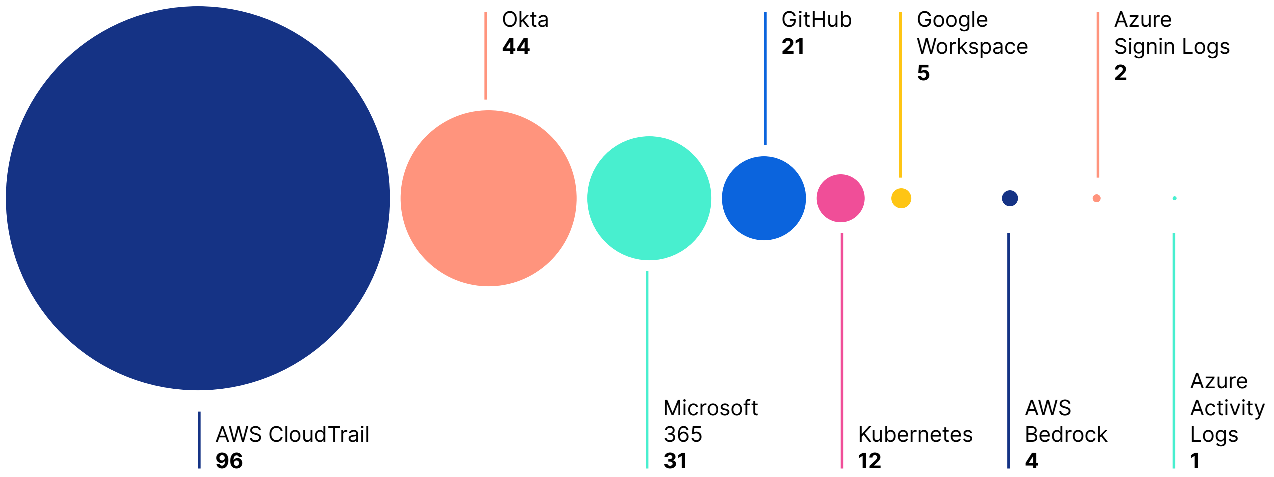


Figure 4: Distribution of new and tuned rules for Cloud and SaaS integrations

AWS CloudTrail

Last year there were many cloud-specific enhancements made across the Elastic ecosystem. To mature our largest existing cloud platform ruleset, we comprehensively audited our existing AWS CloudTrail detection ruleset. The goal of the audit was to look for areas of improvement in existing detection logic, analyze alert telemetry to find tuning opportunities, update MITRE tactic and technique assignments, and build new rules to fill coverage gaps.

When determining our current coverage, we took a couple of approaches. First we researched the current threat landscape for Cloud, both generally and specifically for AWS. Using resources like MITRE ATT&CK's Cloud Matrix, Wiz's Cloud Threat Landscape database, and other recent cloud-related breach reports, we compared what threat actors were doing to what our ruleset was detecting. Then we looked at some of the most popular cloud threat emulation tools, like Stratus Red Team and the AWS Cloud Incident Response Team (CIRT) Workshops, compared those emulations with our ruleset, and executed a strategic plan to fill in the gaps.

By October 2024 we had significantly improved the existing ruleset with over 50 rule tunings, more than 40 new rules, and 17 threat hunting queries. These rules uncover high-risk activities within AWS environments related to compromised access keys, abused lambda functions, overly permissive IAM policies, and more. We've continued regular maintenance of our AWS ruleset and made plans to extend this comprehensive audit to include both Azure and GCP cloud platforms as our next priority.

With Okta's numerous breach reports, we had previously conducted initial research to develop Dorothy, our threat emulation tool for the platform, and established a basic ruleset. Our next step was to identify which parts of the Okta threat landscape were already covered by existing detections and then pivot to threat research to uncover any gaps for in-the-wild attacks, such as those from the Scattered Spider threat group known for targeting Okta users. We replicated these attack scenarios in our lab environment and addressed the gaps with new or tuned detections.

Collaboration with our integrations and telemetry teams was crucial to ensure robust global alert telemetry, allowing us to measure the effectiveness of our rules and make necessary adjustments.

To support advanced detection techniques, we requested enhancements to the Okta integration for system logs and the Okta Entity Analytics integrations. The Okta System Log API provides an audit trail of system events, while the Entity Analytics integration offers contextual user data via the Okta Users and Devices APIs. Together, these integrations tell us what happened and give context on who did it. The Entity Analytics integration initially pulled user data such as `group memberships` and `status` from the Okta Users API. We enriched this data with additional metadata related to `roles` and `authentication factors` to enhance our querying capabilities, allowing us to write detections based on both actions and user context.

For example, if a user, “Alice,” is assigned admin rights by “Bob,” we can use System Logs to view this action and answer questions like “Does Bob typically assign user roles?” and “Is he authenticated from a typical location?” We can then use additional context from Entity Analytics to answer questions like “Does Bob belong to an overly permissive group?” or “Does he have MFA disabled?” These questions can be used for anomaly-based detection, threat hunting, and machine learning (ML) to automate the detection of potential threat activity.

Securing LLM workflows via integrations and standardized fields

A subset of generative AI (GenAI), large language models (LLMs) introduce a layer of interpretive reasoning to workflows by analyzing complex relationships between seemingly disparate data points. As the world was diving headfirst into GenAI, our team realized the need to embed security into these LLM workflows. We used Elastic AI Assistant as an example third-party application — AI Assistant empowers analysts by offering dynamic threat insights, anomaly detection, and contextual analysis across vast datasets. Initially, we developed a prototype proxy solution to extract security-relevant fields from interactions with the AI Assistant.

This approach allowed us to ingest and analyze data from vendor solutions that lacked built-in security auditing capabilities. Our proof of concept validated the need for deeper monitoring and highlighted opportunities to enhance integration processes for broader usability. In collaboration with our Integrations team, we developed a new integration for AWS Bedrock model invocation logs, enabling seamless ingestion of LLM-related activity for threat monitoring. Alongside this, we developed standardized field mappings for LLM interactions, aligning them with the Elastic Common Schema (ECS) and OpenTelemetry (OTel) standards. This allows for consistent data ingestion across various LLM platforms, and makes it easier to maintain detection rules by minimizing the need for separate rules for each LLM vendor. As we continue to add and enhance our integrations, we are strategizing to align other LLM-based integrations to the new standards we’ve set, paving the way for a unified experience across the Elastic ecosystem.

By strategically enhancing cloud and SaaS integrations — and pairing them with advanced detection methods — we worked to deliver deeper visibility into the most critical layers of modern IT environments. From strengthening coverage for cloud platform and identity providers to monitoring LLM-based applications, our initiatives transformed multiple data sources into coherent, high-fidelity detections while enabling a more context-aware security posture for our users.

2.2 Expanding endpoint visibility

Maintaining our endpoint visibility is crucial on multiple fronts: it allows us to detect threats and anomalies and returns detailed telemetry data for us to investigate. Enhancing these capabilities benefits our product and users, but also allows us to respond more thoroughly to evolutions in the threat landscape.

Evolving Windows in-memory threat detection

In-memory threat detection is a pivotal aspect of endpoint security, given that many threats now operate entirely in memory for purposes of Defense Evasion — often by manipulating process memory or indirectly executing kernel-level syscalls. Over time, our approach to in-memory detection has evolved considerably through new technologies and innovative engineering strategies. By collaborating closely with our endpoint agent development team, we have enriched data sources to enable more advanced detection techniques.

Early detection methods for in-memory threats often relied on malware signatures and user-mode hooks — techniques that advanced attackers learned to circumvent. These approaches offered limited insight into in-memory activity and over time were insufficient against sophisticated threats. Preempting these shortcomings, we identified the need for kernel-level visibility improvements to effectively monitor memory manipulations. Starting in May 2023, we began adding significant capabilities to Elastic's kernel telemetry to capture more robust data, including:

- Collecting detailed call stack data directly from the kernel — Elastic 8.8
- Extending Microsoft's Event Tracing for Windows (ETW) Microsoft-Windows-Threat-Intelligence provider for near real-time visibility, including call stacks, into critical syscalls like `VirtualAlloc`, `VirtualProtect`, and `WriteProcessMemory` — Elastic 8.11
- Enhanced Keylogger detection capabilities — Elastic 8.12
- TCP `connect` call stacks — Elastic 8.14
- `DeviceIoControl` driver events with call stacks — Elastic 8.16
- Added visibility for Windows Management Instrumentation (WMI), a commonly abused administration service — Elastic 8.16
- Added Antimalware Scan Interface (AMSI) events to provide deep runtime inspection of Microsoft's native script engines and beyond — Elastic 8.18

As we continue to develop and refine Elastic Security, we will be delivering more impactful enhancements to our visibility and detection capabilities:

Token Impersonation API events to detect privilege escalation

Expanding OpenProcess API events to detect cookie stealer tradecraft

ResumeThread API events and an "early-bird" behavior heuristic to detect process hollowing

ProcessFreeze API events and an "active-debugger" behavior heuristic to detect abuse of Windows Debugging APIs

The release and timing of any features or functionality described in this post remain at Elastic's sole discretion. Any features or functionality not currently available may not be delivered on time or at all.

Our continuous enhancements to Kernel ETW telemetry paired with the incorporation of cutting-edge detection techniques allow us to more effectively identify and mitigate advanced threats in near-real time. These advancements are pivotal in safeguarding our users against sophisticated adversaries and their evolving tactics.

Adding new data sources for macOS

Our team identified a gap in macOS threat coverage because there was no data source equivalent to the Windows Dynamic Link Library (DLL) load event that could be used to detect the loading of dynamic libraries (dylibs) in macOS. After exploring the Apple Endpoint Security API, we discovered there was no dedicated dylib load event for us to subscribe to. However, we noticed that mmap events are generated any time a process memory-maps a binary, dylib, or file. Collaborating closely with our endpoint developers, we filtered mmap events to capture only dylib loads, enriching the resulting telemetry with code signature data and dylib hashing to form a dedicated first-of-its-kind dylib load event for macOS.

New dylib load event enables detection of advanced Evasion on macOS

Once this new event type was created, we built and tested rules around it. During our research, we examined various Command-and-Control (C2) frameworks featuring in-memory JavaScript for Automation (JXA) execution — a technique that loads and executes JXA scripts entirely within process memory, effectively bypassing many conventional detection mechanisms. By monitoring the dylib load events for the rapid, consecutive loading of the JavaScript and StandardAdditions libraries, our team observed a unique pattern essential for executing JXA scripts in memory.

We developed a sequence rule, *In-Memory JXA Execution via ScriptAdditions*, that detects these two dylib loads by the same process in quick succession. Testing this rule against multiple tool implementations confirmed its reliability; it generated surprisingly few false positives in our environment, so we confidently released it to our users. In doing so, we established a first-of-its-kind detection for in-memory JXA loading and execution using a novel dylib load event. This development not only expands our macOS detection capabilities but also demonstrates the value of engineering custom telemetry to capture advanced threat behaviors.

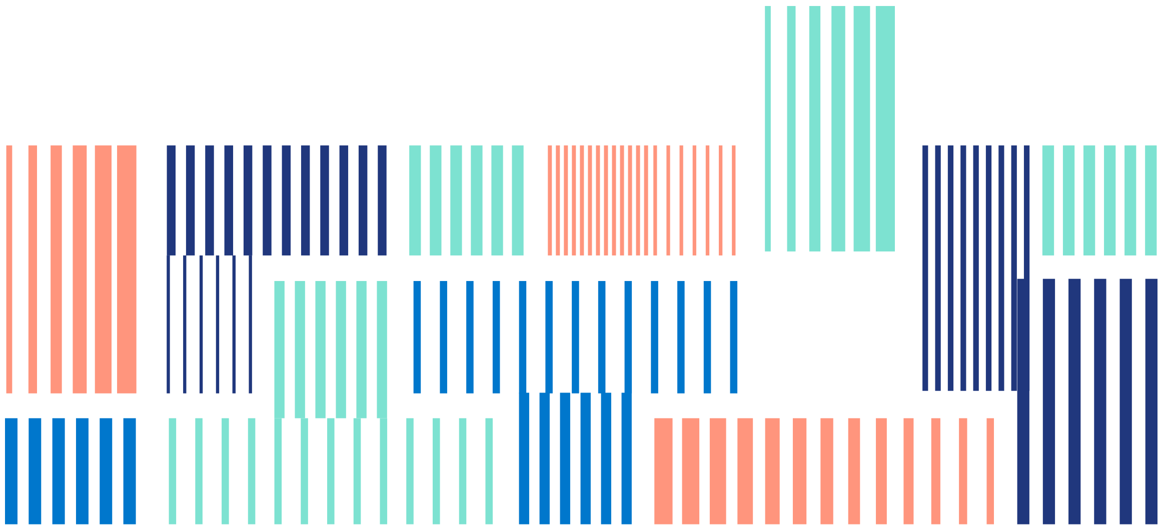
New DNS telemetry uncovers first party intelligence findings on macOS

In addition to dylib load visibility, we recently enriched macOS network events with domain data and introduced a custom DNS event, another first-of-its-kind data source for macOS. By capturing DNS requests alongside other network indicators, we broadened our detection capabilities for Initial Access and C2 methods. Almost immediately, these enhancements helped us surface a North Korean (DPRK) campaign targeting developers via malicious Node Package Manager (NPM) packages – reusable pieces of code stored in a public registry, which can be easily integrated into Node.js projects. By correlating DNS event data with package service processes making suspicious queries, we traced malicious NPM packages to IP lookup domains used for C2. The DNS telemetry also offers insight into Exfiltration techniques, as some macOS malware leverages DNS channels to sneak data out. This new DNS event further underscores how expanding and enriching endpoint telemetry can yield tangible benefits in uncovering real-world attacks.

Broadening Linux visibility

Elastic's expansive ecosystem of integrations significantly amplifies our detection capabilities for Linux systems by offering in-depth visibility into a wide array of OS activities. For instance, the System integration facilitates the collection and analysis of crucial authentication logs and syslog events, providing insights into user activity and system processes. The Auditd Manager integration focuses on audit events, enabling the tracking of changes and potential security violations within the system. Additionally, the File Integrity Monitoring (FIM) integration plays a key role in maintaining system integrity by monitoring and alerting on modifications to critical files.

By leveraging these integrations, we broaden our ability to detect and respond to potential threats within Linux environments. For example, System may log a suspicious login event, while Auditd Manager captures an unexpected syscall related to file access, and FIM detects modifications to a critical system file; all together indicative of an attacker attempting to modify system files to maintain Persistence, escalate privileges, or cover their tracks.



Elastic Defend

Our endpoint detection strategy is tightly coupled with the Elastic Defend integration, which allows us to monitor host-based event categories like file, process, network, and many more across Windows, Linux, and macOS endpoints. A large portion of our detection engineering efforts rely solely on this integration, as the dataset provides deep-level visibility suitable for most threat scenarios.

A major enhancement we made (in collaboration with our integrations and endpoint development teams) was the addition of data fields capturing effective and permitted capabilities for Linux processes. Capabilities are a subset of root level privileges that can be individually assigned to processes in order to limit excessive permissions. We worked together with our endpoint developers to add the most relevant effective and permitted capabilities for running processes across all Linux-specific integrations, including Elastic Defend.

> Root level kernel permissions bypass all permission checks, so they are essential to monitor. These are often assigned to container processes and can be abused for container breakout techniques.

With this enhanced data, we were able to fill coverage gaps related to Linux Privilege Escalation techniques. We developed detection rules such as *Privilege Escalation via CAP_SETUID/SETGID Capabilities*; which identifies sequences where a process with `CAP_SETUID` or `CAP_SETGID` capabilities executes and subsequently elevates its access to root (`UID/GID 0`). These capabilities enable processes to manipulate user and group identity, potentially allowing attackers to exploit misconfigurations and escalate their privileges to root.

Auditd Manager

Auditd is a user-space component of the Linux Auditing System that hooks into the Linux kernel space and captures detailed information about syscalls and other security-related events. With our Auditd Manager integration, we can establish a subscription to the kernel to receive these events as they occur. Multiple messages sent for a single auditable action are consolidated into one easily digestible event that includes the different aspects of the activity (the syscall itself, file paths, current working directory, process title, and more). As we've explored, Auditd is a powerful tool for detection engineering and we use this integration to expand visibility in some areas where Elastic Defend is limited.

One of our most effective Auditd rules, *Kernel Driver Load*, detects when a Linux loadable kernel module (LKM) is loaded through system calls. An LKM is a piece of code that can be dynamically loaded into the Linux kernel to extend its functionality without the need to reboot the system. Threat actors can load a rootkit using LKMs, giving them total control of the system and the ability to hide from security products. As other rules monitor for the addition of LKMs through system utilities or .ko files, this rule is designed to catch attempts by rootkits to evade those detections by monitoring for kernel module additions at the system call level. Using Auditd Manager, this rule monitors the `init_module()` and `finit_module()` syscalls, capturing all LKM loads and making bypass attempts by threat actors very hard, if not impossible.

File Integrity Monitoring (FIM) integration

Detecting built-in shell functionality such as echo and pipes/redirects presents unique challenges, as these operate within the shell process itself rather than invoking distinct executables. As a result, traditional process-based detection methods may not capture file modifications made through these utilities. To ensure comprehensive visibility at the endpoint level, we leverage the FIM integration. FIM monitors files in real time via a subscription with the OS and sends events when a change (create, update, delete) to those files occurs. This was a vital part of our detection strategy for Linux Persistence mechanisms. The rule *Potential Persistence via File Modification* uses FIM to detect modifications to a select group of files that are commonly abused for Persistence, like cron jobs, systemd services, message-of-the-day (MOTD), and SSH configurations. While Auditd can technically monitor files for changes as well, FIM is more optimized for this and is often the better choice for file monitoring.

These examples represent just a few of the many unique ways we leverage our broad selection of data sources to expand visibility and ultimately improve our endpoint detection capabilities. While we can use these integrations in isolation, they complement one another well in unique detection use-cases. For example, Elastic Defend might detect a process gaining specific capabilities, while Auditd Manager would then monitor for any unauthorized file access attempts using those capabilities. By combining these data sources, we can broaden our visibility and identify high-risk sequences of events indicative of malicious behavior.

By engineering deeper kernel-level visibility on Windows, pioneering novel dylib load and DNS telemetry on macOS, and using various Linux integrations creatively, our endpoint detection efforts have significantly expanded the scope of advanced threat coverage and early threat detection. These enhancements illustrate how strategic collaboration with endpoint developers and creative rule design can illuminate even the most elusive attacker behaviors.

Part 3

Internal metrics and evaluation

Detection engineering success at Elastic is measured at two distinct levels: immediate operational performance and long-term strategic impact. Some metrics are tactical, focusing on day-to-day rule performance such as detection efficiency, false positive rates, and query execution speed. Others are strategic, evaluating our contribution to broader goals like organizational objectives and key results (OKRs).

3.1 Operational performance analysis

Our Endpoint Behavior rules are high-confidence and prevention-focused, designed to require minimal tuning and limited false positives. In contrast, our SIEM Detection rules provide broader threat coverage by leveraging all available data sources, tuning is expected as many of our low severity rules are meant to be used as signals and noisy by design — allowing for greater compatibility and control by users to fit to their environments.

We continuously refine these rules to adapt to the unique environments of our users. This ongoing effort does not imply that our rules are flawed from the start; rather, it demonstrates our commitment to ensuring broad applicability and effectiveness across diverse environments. By avoiding overfitting our rules to any single organization, we maintain their relevance and reliability for all our users.

Performance metrics measure day-to-day rule efficacy and noise levels by validating rule logic and guiding our tuning efforts, especially for our Endpoint Protections. By minimizing unintended noise and keeping our rules relevant as new threats emerge, we ensure our users can trust the alerts, respond swiftly to real incidents, and ultimately maintain a stronger security posture.

Protections malware feed efficacy

The protections malware feed efficacy score is a key metric used to validate the effectiveness of our Endpoint Protections against real-world threat behavior.

This score assesses our ability to detect and block 99% of malware samples using our multilayered detection features:

- Behavioral protection rules
- Malware detection
- Ransomware prevention
- Memory threat detection
- Malicious behavior monitoring
- YARA signatures

To calculate this score, we ingest malware samples using our internal sandbox tool, Detonate. We process more than 500 per day, filtering out benign samples, and then for each malicious sample we analyze whether any Protection rules trigger. Any undetected malware is immediately triaged for coverage gaps, ensuring that new or evolving threats are accounted for. Our goal is to maintain a detection rate at or above 99%, and when coverage falls below this threshold, we investigate, improve detection mechanisms, and reassess protections. This metric validates the daily work of our detection engineers by ensuring broad malware coverage and identifying areas requiring improvement.

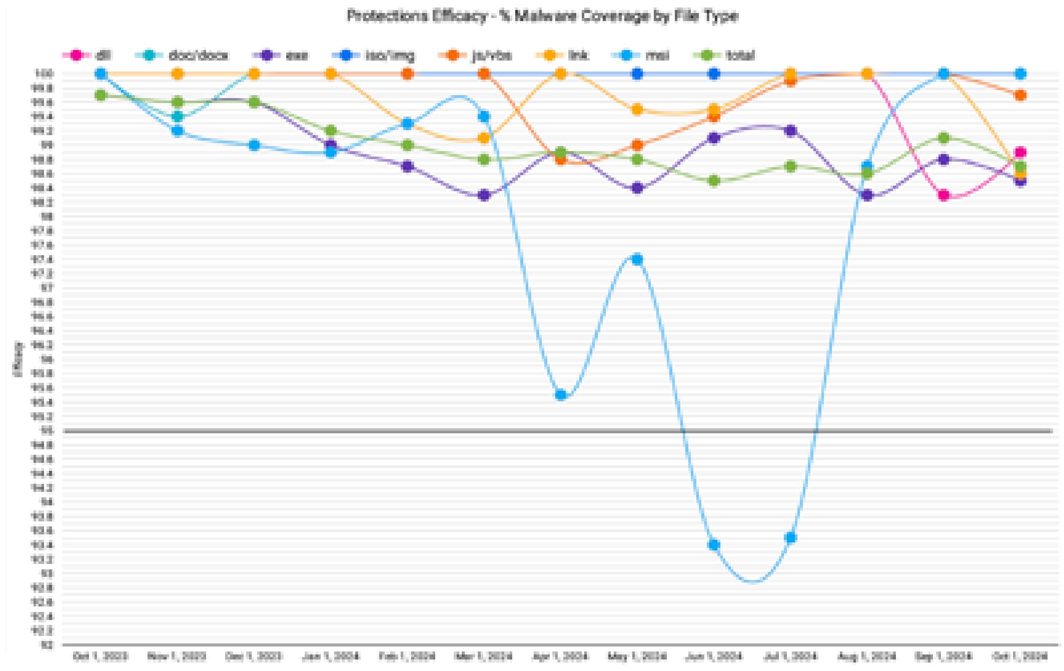


Figure 5: Coverage rate for Windows malware by file type from Oct 2023 - Oct 2024

Readers can view how we fared in the AV-Comparatives Malware Protection Test — an independent evaluation designed to simulate real-world attack scenarios against antivirus and endpoint security solutions.

Addressing false negatives and rule tuning

In addition to tracking malware coverage, we monitor telemetry for false negatives. If a rule is found to be underperforming, we collect additional telemetry and refine our detections. Fine-tuning rules is a crucial but selective process due to the scale of our coverage — spanning over 1,000 Endpoint Protection rules, each tailored to different attack techniques (file, process, API-based detections, etc.). Tuning decisions requires understanding the rule logic, determining whether to exclude specific conditions (e.g., `file.path`, `command.line`), and analyzing agent coverage trends.

One key threshold we monitor is sudden increases in alerting across agent environments. If a rule queries across 10,000 agents, and suddenly 1,000 agents begin generating alerts for it, it's likely false positives. The acceptable tuning threshold varies by environment size; for example, in a 1,000-agent deployment, a smaller alert spike may warrant tuning, whereas in a large-scale deployment, stricter thresholds are applied before changes are made.

We also assess whether or not there is room for additional protection types when an alert is triggered. For example, if a sample triggers a protection rule in behavior mode only but lacks corresponding memory or signature-based alerts, we look for ways we can further reinforce our detection of this sample using memory or signature-based rules.

Automated threshold-based prioritization for rule tuning

To systematically flag rules for review, we apply automated alert thresholds based on a 12-hour monitoring window. A rule is flagged for tuning based on the number of:

- Total detections
- Unique process names triggering the rule
- Unique agents from a single cluster triggering the rule
- Unique clusters affected, exceeding a total number of detections
- Unique agents triggering the rule in 24 hours

When a rule exceeds these thresholds, automated alerts are sent to our dedicated Slack channel and distributed via email, prompting investigation. Additionally, a separate alert prioritizes “Top Clusters,” ensuring that the most widespread detection issues receive immediate attention. Since some endpoints run older detection artifacts, we also account for delayed rule rollouts. The process is fully automated until investigation begins, at which point we determine whether to refine or exclude specific rule conditions based on false positive analysis.

Rule variability with relative alert magnitude

Relative alert magnitude acts as another false positive indicator based on alert distribution, serving as a standardized measurement of randomness for alerts. Using our alert telemetry data, we take the volume of alerts generated across all hosts per month to find the mean for each rule. We then take the standard deviation (std) for each rule and come up with a std-to-mean ratio for each rule. This value represents how consistently a rule is triggering alerts.

High variability means a rule is triggering more sporadically across clusters. This is more indicative of true positive alerts because threat behavior often impacts our users in a random way. Low variability means a rule is triggering more consistently across clusters, which is more indicative of false positive alerts because threat behavior very rarely impacts our users in a consistent way. Instead, this may be a poorly tuned rule generating noise across clusters.

We use this variability score to help us determine where to prioritize our detection engineering efforts. The goal is to maintain effective coverage while minimizing false positives, ensuring that variability reflects well-tuned rules aligned with detection engineering objectives. On a range of 0–n where 0 = truly consistent and n = variability, we determined a threshold value of .2 as a good indicator of a rule triggering too consistently and in need of potential tuning.

This threshold was based on our own independent knowledge of false positive alert data — we wanted to make sure the threshold we chose captured known noisy rules, while leaving room to include those we may not notice. Additionally, since this is only one of the metrics we use to determine rule efficacy, we felt comfortable keeping the scope a bit broad as we can use other indicators for further filtering.

Why mean alone is insufficient

While mean alert volume is a useful indicator of noise levels, it is not always the best measure of rule efficacy. A high mean could suggest an excess of false positives, but it could also reflect frequent legitimate alerting behavior or be skewed by older rule versions still in use. For example, the rule *Suspicious Network Activity to the Internet by Previously Unknown Executable* showed its highest alert volumes between October 2023 and February 2024, with mean values ranging from 969 to 4,660 alerts per month. At first glance, this spike might indicate excessive false positives; however, examining other variables paints a different picture.

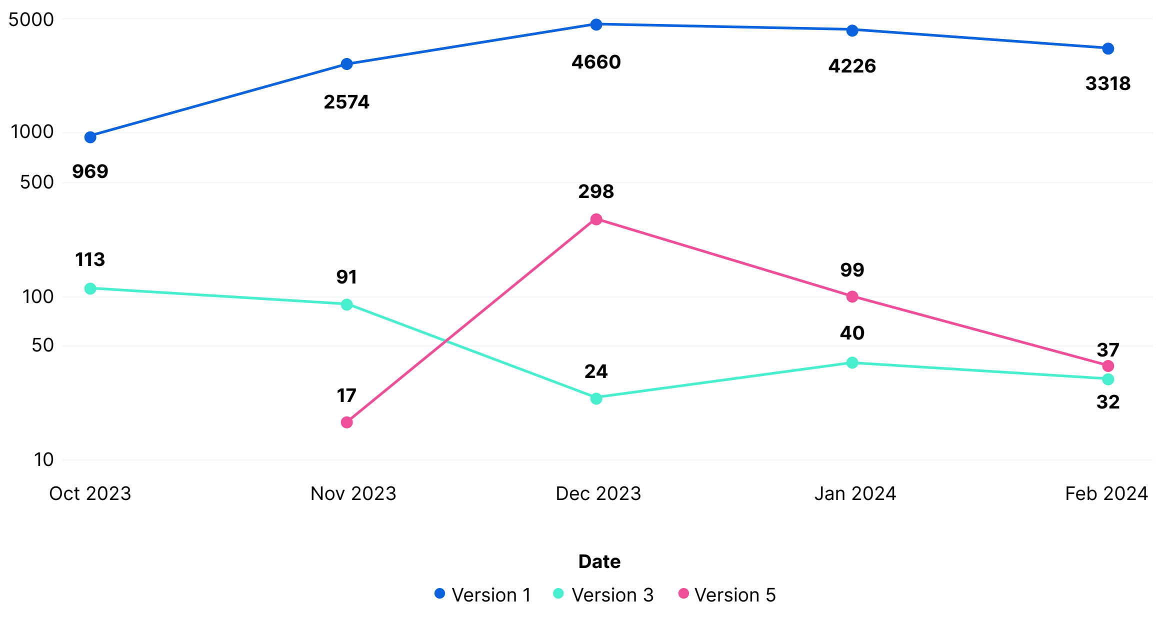


Figure 6: Mean rule alert volume timeline for *Suspicious Network Activity to the Internet by Previously Unknown Executable*, differentiated by rule version

For context, this is a New Terms rule, meaning it only alerts on previously unseen values of a specific field within each environment — in this case, process.executable. This design inherently limits false positives, suggesting that the high alert volume may be expected given its broad detection logic. To validate this further, we examined rule versions. By February 2024, seven different rule versions were in production, and the highest mean values were all associated with version 1, meaning users running older rule packages were driving the alert volume.

A major rule tuning update was released with version 5 in November 2023. By December 2023, after allowing time for users to adopt the update, the alert volume dropped significantly, with a mean value of 298 compared to 4,660 from version 1. Some might still consider 298 too high, prompting further tuning consideration; however, instead of relying solely on mean values, we can assess the rule's variability score to gain a more comprehensive view. The December 2023 version 5 std-to-mean ratio was 5.62, well above our .2 threshold, indicating high variability — which strongly correlates with true positive detections rather than persistent false positives.

Given these insights, we determine that this rule was performing as intended, and our detection engineering efforts would be better spent elsewhere. This example highlights why mean alert volume alone is insufficient — without additional context like variability scoring and rule version analysis, tuning efforts may be misguided, leading to unnecessary changes that could weaken detection efficacy.

Operational performance metrics drive continuous refinement of detection coverage, ensuring our detection engineering efforts remain data-driven and adaptive. By automating rule tuning thresholds and analyzing false negatives, we proactively close detection gaps and optimize rule fidelity. While we highlighted key examples — such as rule variability, malware efficacy, and alert distribution trends — these represent just a subset of the broader set of metrics we use to assess and refine detection performance. The success described in these metrics is a direct result of our detection engineering processes, including how we address false positives and our methods of prioritization for rule tuning. These measurements collectively provide a structured approach to improving detection accuracy and our users' experience by reducing unnecessary alert volume.

3.2 Strategic business objectives and results

We track strategic objectives to ensure our work not only improves day-to-day detection performance but also aligns with broader business goals. By measuring OKRs, we assess the user impact, operational efficiency, and efficacy of our detection engineering efforts. There is overlap between what our day-to-day performance metrics and OKRs measure. However, our OKRs are measured alongside the work of other teams as an indicator of our collective progress toward company-wide strategies; the primary strategy being to build a security product that improves the day-to-day value for our users.

Keeping endpoint alert volume below 1%

One of the main pain points that security teams tackle with prebuilt security content is that broad detection logic can become too noisy in certain environments, leading to alert fatigue. More importantly, Endpoint Protection alerts can cause direct disruptions by terminating processes, quarantining files, etc. With this in mind, one of our core objectives is ensuring that fewer than 1% of hosts generate Endpoint alerts. This serves as a key false-positive indicator and measures impact on our users' day-to-day experience with alert volume.

To ensure accuracy, we calculate this daily and average it monthly, reducing data skew from short-lived hosts. Breaking it down by protection type (malware, memory, behavior, ransomware) and operating system allows us to pinpoint areas needing improvement.

A high percentage of affected hosts may signal overly broad detection logic, excessive noise, or misconfigurations. Keeping this number below 1% ensures high-fidelity detections. This threshold was decided by our subject matter experts as a realistic expectation for the number of true positive alert instances at any given time. Over the last year, we've stayed well below this threshold each month across each OS. Our day-to-day processes discussed in section 3.1 allow us to maintain these results and ultimately support our users' experience by reducing unnecessary alerts.

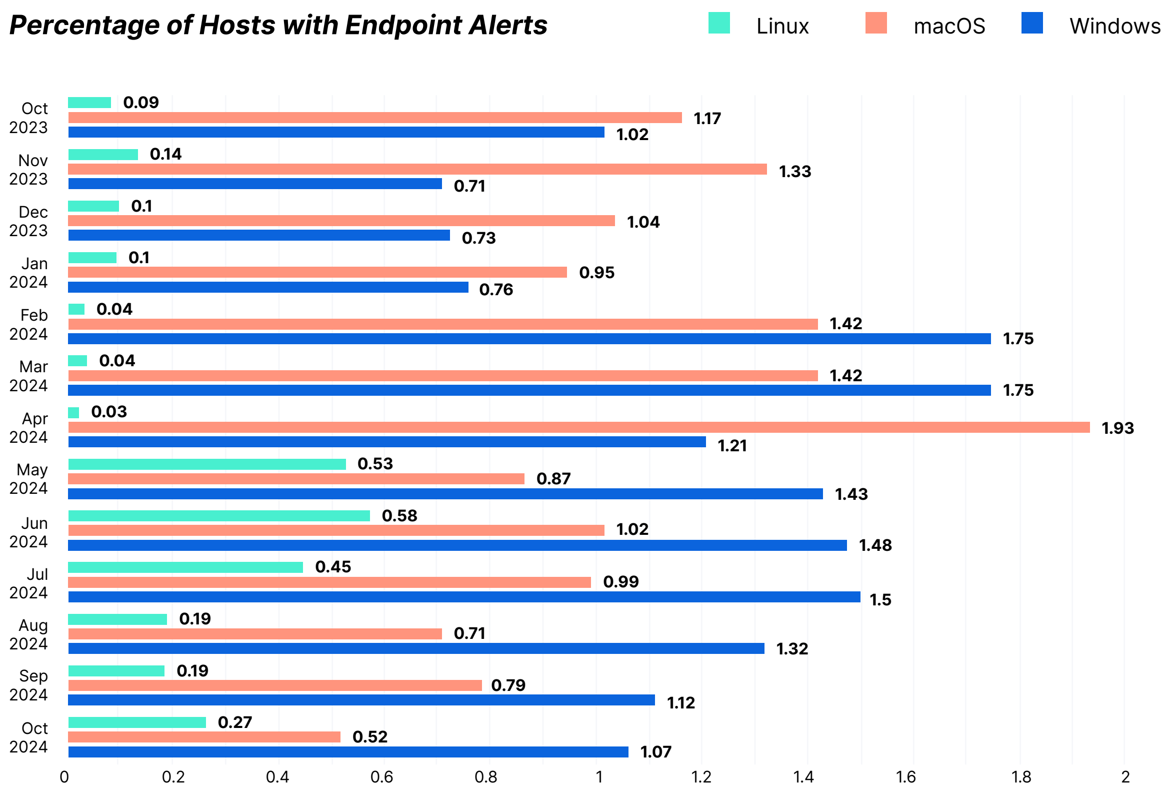


Figure 7: Monthly percentage of hosts with Endpoint Behavior alerts, Oct 2023 - Oct 2024

Maintaining a 14-day rule release cadence

Releasing both SIEM Detection and Endpoint Protection rules within a 14-day cycle is crucial for delivering timely and relevant protections against evolving threats we tackle day-to-day. This goal is important to us because it ensures our customers can quickly adopt new rules to safeguard their systems. To achieve and sustain this cadence, we focus on:

Scalable rule validation to maintain detection quality and minimize false positives

Streamlined CI pipelines to automate testing and deployment

Efficient threat intelligence integration to shorten research-to-detection timelines

Maintaining this cadence is challenging, but it is essential for keeping our users protected and we're delighted to consistently share our rules with the community at large through our Detection Rules and Protections Artifacts repositories.

Maintaining 100% RTA coverage of endpoint behavior rules

One of the ways we automate testing our ruleset is through Red Team Automations (RTAs). RTAs are Python scripts that either reference a malware sample hash that exhibits threat behavior we aim to detect or it emulates the attacker behavior through code. RTAs provide a simple way to verify that detection rules are generating the expected alerts for our SIEM and Endpoint Protection rules.

Due to the difference in design, we hold a different standard for testing each ruleset. For Endpoint Protection rules, our goal is to have 100% RTA coverage. Meaning when a new rule is created, it is created alongside an RTA that provides test data to verify the rule triggers on expected behavior. This additional testing ensures protection rules are valid and remain high fidelity across multiple user environments.

RTAs serve a larger purpose beyond new rule development. We use them to regression test rules to validate new features added to the SIEM or endpoint agent and any modifications based on rule tuning, as well as for maintenance. This process can become time-consuming with hundreds of rules to test across multiple Stack versions, but RTAs help to automate the process. Additionally, sometimes we get requests for sample data, or methods to generate suspicious events to baseline configurations. RTAs are a quick and easy way to provide this data in support of other initiatives at Elastic.

In addition to providing these tests for internal use, we have migrated all our RTAs to Cortado (Consolidated RTAs), a public repo we share with the community as an easy way to test our ruleset!

While we highlighted key examples, these represent just a subset of the OKRs we monitor. Together with our operational metrics, these OKRs provide a comprehensive framework for evaluating our detection engineering efforts and measure how effectively we provide value to our customers.

Part 4

Considering threat reports and looking ahead

Creating an efficient product requires strategic planning, so we spend a lot of time discussing our detection engineering plans. There are two key spaces that affect these discussions: the past and the future. Threat reports like Elastic's annual Global Threat Report are crucial for us as a strategic validation tool, highlighting which threats are most prevalent across Elastic Security's global telemetry.

4.1 Partnering with the Elastic Global Threat Report

The Elastic Global Threat Report (GTR) and our detection engineering efforts are deeply interconnected, each reinforcing the other. The alert telemetry that powers the GTR is influenced by the detection and protection rules we create, capturing real-world adversary behaviors across cloud, endpoint, and network environments. In turn, the threat trends surfaced in the GTR provide us with a broad, data-driven perspective, validating existing detection coverage and helping to direct our efforts.

One of our core responsibilities as detection engineers is to respond rapidly to in-the-wild threats. We continuously track emerging attack techniques, and much of our work involves adapting to real-world adversary behaviors as they happen. The data from the GTR helps us identify gaps, fine-tune detections, and prioritize research areas that will have the greatest impact on our customers.

The 2023 Elastic Global Threat Report

While not exhaustive, this section provides a retrospective on our responses to several of the key threat trends identified in the 2023 GTR.

Forecast: "Defense Evasion is going to remain the top investment, and tampering will supersede masquerading"

Adversaries continue to prioritize Defense Evasion, employing methods such as System Binary Proxy Execution and Masquerading techniques. In response, we have significantly enhanced detections across all operating systems. Out of 336 new EDR rules, 152 were focused on Defense Evasion, accounting for approximately 45% of detection efforts related to Endpoint Protection Rules. Out of 256 new SIEM rules, 67 were focused on Defense Evasion, representing around 26% of detection engineering efforts for SIEM Detection Rules.

These rules covered a broad range of evasion techniques beyond Masquerading, with the top being:

- Process Injection
- System Binary Proxy Execution
- Highjack Execution Flow
- Impair Defenses

The importance of Defense Evasion

As detailed in Unveiling Malware Behavior Trends, our large-scale analysis of behavior trends from over 100,000 Windows malware samples identified Defense Evasion as the most frequently observed adversary tactic. This tactic triggered 189 distinct detection rules, accounting for nearly 40% of all Windows rules in our library. The high use of Defense Evasion techniques in real-world attack patterns across various malware makes it a top priority in our detection engineering efforts for Windows.

> The primary techniques observed in this breakdown included Code Injection, Defense Tampering, Masquerading, and System Binary Proxy Execution.

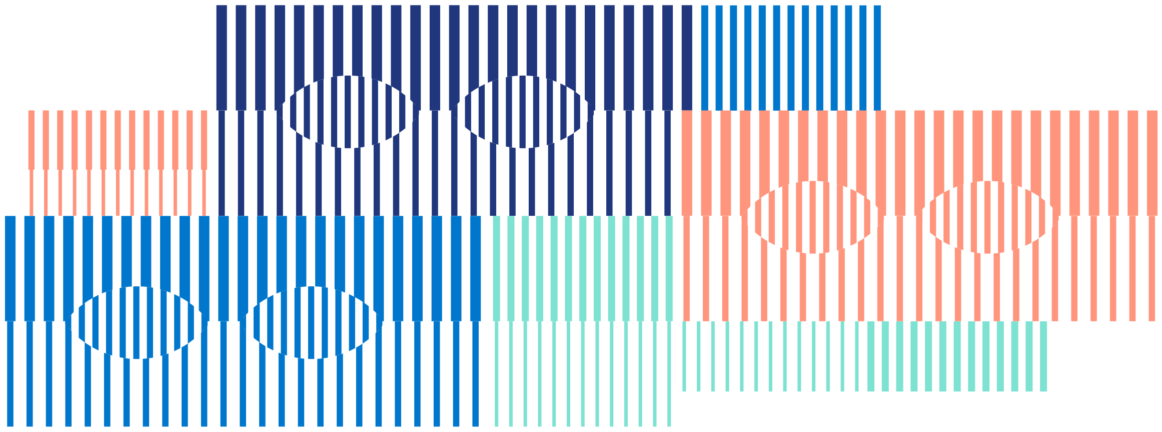
Specific coverage efforts: Linux Defense Evasion research and rule development

We analyzed numerous techniques for evasion on Linux such as process/pid hiding, encoding and decoding payloads, and the use of GTFOBins.

The outcome of this research included increased detection coverage for techniques within the following categories:

- GTFOBin SO loading
- GTFOBin reverse & bind shells
- GTFOBin proxy execution
- Busybox evasion techniques
- Dynamic Linker modifications
- Curl/Wget downloads
- SSL & CA certificates
- Disabling/removing protections (e.g., firewalls, AppArmor, Auditd, SELinux) and more.

This was a broadly scoped effort but is still not exhaustive. It helped us to identify gaps that will need to be addressed in the near future like rootkits, Process Injection techniques, and web shell usage.



Forecast: "The malware-as-a-service (MaaS) model will become more popular"

The increasing popularity of the Malware-as-a-Service (MaaS) model has abstracted the complexities of cyber intrusions, enabling less experienced threat actors to launch more sophisticated attacks. As mentioned in previous sections, malware analysis and malware feed detection efficacy is baked into our day-to-day detection engineering efforts, making this a focus area across all endpoint platforms. Additionally, we've devoted significant effort toward detecting commonly used scripts, tools, and malware behaviors in support of our broader effort to tackle less experienced actors using MaaS platforms.

Specific coverage efforts: macOS infostealer YARA research and rule development

We conducted an in-depth analysis of macOS malware samples collected from the macOS Malware Collection by Objective-See and various stealer variants (e.g., MacStealer, MetaStealer, RealstStealer) discussed in blogs from SentinelOne and Kandji. Our analysis focused on identifying unique characteristics and hardcoded behaviors of each sample, which could be used to create YARA rules for future detection.

Key findings and actions included:

- **Commonality detection:** We identified that many stealer samples encoded AppleScript in various ways (Hex, XOR, Base32, etc.). We developed a YARA rule to detect these encoding methods, achieving a 100% detection rate on the stealer samples with zero false positives. This rule was run as a live hunt for over a week, yielding only true positive detections and uncovering previously undetected stealer samples.
- **Specific rules for variants:** For MetaStealer and RealstStealer, we created specific YARA rules.
- **General rule for other samples:** For samples that did not encode AppleScript or were not Rust or Go-based, we created a rule to detect the presence of six or more crypto wallet extension IDs. This rule also ran as a live hunt for several weeks, resulting in zero false positives and numerous true positives, including the detection of previously undetected samples reported on social media.

These rules have provided a robust foundation for detecting the majority of macOS stealers and will be continuously updated to account for new variations or samples.

Forecast: "Cloud credential exposure will be a primary source of data exposure incidents"

As highlighted in section 2.1, we have significantly increased our detection engineering efforts for Cloud and SaaS platforms over the past year. Given the increasing reliance on these platforms and the prevalence of credential abuse, this area has become a critical focus for us. We developed 23 new SIEM rules related to Cloud Credential Access techniques, spanning platforms like AWS, Okta, Microsoft 365, and endpoint environments to account for cross-platform threat behavior.

Specific coverage efforts: Microsoft 365

Our efforts related to Microsoft 365 Credential Access threats focus on several key areas to identify and mitigate potential attack vectors.

- **Credential dumping and abuse:** We implement rules to detect attempts to access and dump credentials from sensitive processes and files, such as monitoring for the creation of Kerberos ticket dump files and unauthorized access to the LSASS process.
- **Unauthorized access and privilege escalation:** Our monitoring includes identifying activities that suggest attempts to gain unauthorized access or escalate privileges, which can lead to credential theft. This involves detecting suspicious processes and bypass attempts of security controls like User Account Control (UAC).
- **Persistence mechanisms:** We track various Persistence mechanisms that attackers might use to maintain access to compromised systems, which can facilitate ongoing credential access. This includes monitoring for registry modifications and the creation of malicious Persistence files.
- **Exploitation of legitimate tools:** We keep an eye on the misuse of legitimate tools and applications that can be leveraged to access credentials. This involves detecting unusual or suspicious usage patterns of tools like PowerShell and HTML applications.

By focusing on these areas, we ensure a comprehensive approach to detecting and mitigating credential access threats within Microsoft 365 environments.

The 2024 Elastic Global Threat Report

While the data sourced for this SDEE is only reflective of our response to the 2023 GTR, there are some interesting trends in our most recent Global Threat Report that we've started to address and will continue to focus on in the coming year.

For example, the 2024 Global Threat Report reported the following:

- Windows #1 tactic → Defense Evasion
- Linux #1 tactic → Persistence
- macOS #1 tactic → Execution
- Cloud #1 tactic → Credential Access

This distribution reinforces the need to differentiate our focus areas across platforms in order to address their unique threat landscapes.

New SIEM and EDR Rules

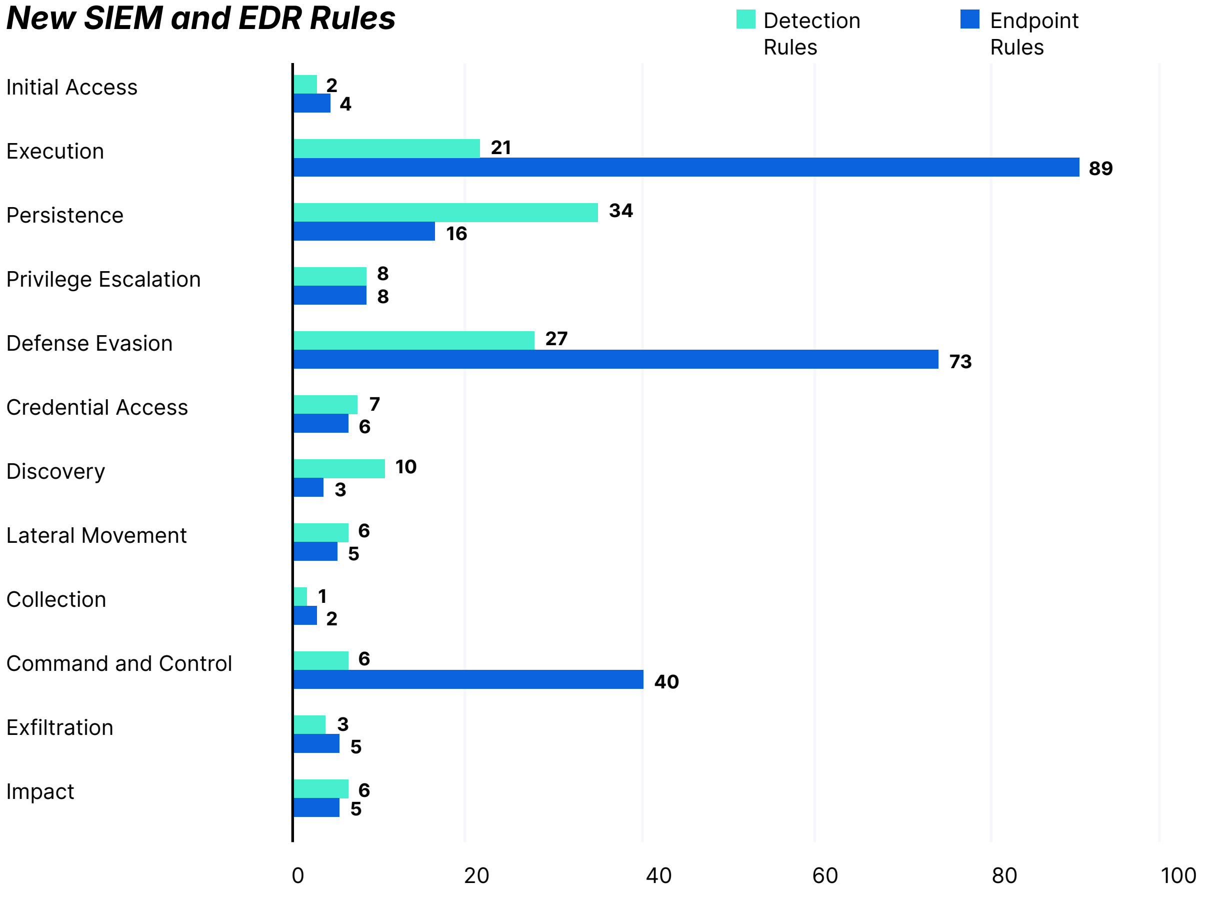


Figure 8: Distribution of new rules by tactic from October 2024 to March 2025

While our approach to SIEM rules allows for broader tactic scope, our EDR rules have heavily focused on high-confidence tactics that are strong indicators of an active threat. Primarily Execution, Defense Evasion, C2, and Persistence.

Additionally, the team is focused on the evolution of the generative AI landscape. Attackers are utilizing this technology in various ways and our team will continue to monitor these emerging techniques — especially sophisticated phishing campaigns. However, we also intend to watch how generative AI is benefiting defenders.

These benefits have been seen internally, with both Elastic's foundation in Search AI as well as our use of generative AI to produce investigation guides for our SIEM rules — an important yet time-consuming task. By using generative AI to analyze individual rule contexts alongside standardized guidance, we can create investigation guides more quickly. This enhances our ability to provide user value while freeing us to conduct more threat research.

Both attackers and defenders are continuously pushing the limits of what is achievable, and it is our mission at Elastic to comprehend these ever-shifting dynamics. We recognize that the threat landscape is not static; it is a fluid and adaptive environment where new threats and vulnerabilities can appear at any time. Therefore, we remain agile and responsive to emerging trends, constantly refining our detection capabilities to ensure that our users are well-protected against the full spectrum of threats, both present and future.

4.2 Looking forward

As the cybersecurity threat landscape continues to grow in complexity, our detection engineering team has undertaken a comprehensive review to outline our goals for the upcoming year. We consolidated insights from team discussions, focusing on priority threats, detection gaps, process improvements, and the incorporation of AI.

Priority threats and adversary behaviors

With regard to our detection engineering efforts, there are several areas that we are keeping a close eye on as the year unfolds. Our intention with this section is to show you how we plan to adapt to recent threat trends. Maybe you can beat us to it?

Artificial intelligence (AI)

The rapid advancement and adoption of AI technologies have introduced new vectors for adversarial attacks, making it a significant area of concern. Over the next year we will enhance our AI threat detection coverage by focusing on the following:

- **Mapping AI threats using frameworks such as MITRE ATLAS:** Helps us understand the TTPs used by adversaries.
- **Developing robust detections for adversary use of AI:** Identifying and mitigating AI-enabled attacks (e.g., the use of GenAI to create sophisticated phishing emails, prompt-based injections, and jailbreaking).
- **Monitoring identity access to LLM APIs and analyzing LLM responses:** Monitoring cloud-based LLM model API calls for suspicious access attempts and malicious outputs.

- **Developing heuristics for identifying suspicious prompt structures:** Analyzing AI-generated content for signs of manipulation.
- **Capturing endpoint activity related to LLM interactions:** Monitoring local hosts for unusual activities related to LLMs.

To address these growing concerns, we will build upon existing Prompt Injection detections and enhance the logging of AI model activity across cloud platforms.

Endpoint security

Endpoint security will remain a priority for our team over the next year. While some focus areas will overlap across operating systems, we will differentiate our efforts to address the unique threat landscape of each OS and the differing maturity levels of each ruleset.

- **Windows:** Adversaries often exploit the openness of our detection logic to evade us. For example, we observed some C2 frameworks attempting to bypass our existing detections by mimicking false positive exclusions for our rules. Some have also been observed using advanced TTPs (e.g., memory guard protection) to prevent memory scanning. We'll attempt to counter this by:
  - **Duplicating crucial Endpoint Protection rules as SIEM Detections with broader conditions:** With fewer exclusions, this will help catch more sophisticated evasion techniques.
  - **Make existing detections more aggressive or resilient:** Ensure that existing detections are robust enough to identify and mitigate advanced attempts to bypass defenses. Key areas of focus include Process Injection, Credential Access, and Initial Access.
- **Linux:** Continue to address Persistence mechanisms, prioritize Evasion techniques, enhance YARA signature development, and improve detections for common attack vectors. More specifically:
  - **Address evasion via exposed rule exclusions:** Similar to Windows, many Linux EDR rules rely on excessive exception lists, which attackers can exploit to evade detection. We will leverage SIEM Detection rules to create similar rules with fewer exceptions to combat this.

  - **Enhance malware signature detection via YARA signatures:** While behavior-based detection methods can account for a broad range of malware behaviors, there is still an important place for signature based detections that provide another layer of defense against known malware. We will develop new YARA signatures specifically for Linux malware and adversary tooling.
  - **Improve detections for brute-force attacks and public-facing service exploitation:** These remain common attack vectors for botnets, leading to deployments of crypto miners, use of compromised systems for pivoting, and other threats. DNS and reputation-based lookup services will be leveraged to help close this gap.
- **macOS:** For macOS we will shift focus to enhancing our SIEM ruleset, as most of our efforts thus far have been for Endpoint Protection rules. However, there are some considerations in doing this that we want to explore.
  - **Supplement threat research with more telemetry:** A large segment of the community utilizes our SIEM rules without installing our Endpoint Protection rules. By converting some of our macOS protections to SIEM, we will enable a larger community of researchers and funnel back more telemetry for our own continued research. This comes with an increased need to balance alerts across SIEM and Endpoint, and our team is committed to minimizing duplicate alerts.

By addressing these areas, combined with our ongoing advocacy for enhanced Elastic capabilities, we will significantly bolster the maturity and effectiveness of our endpoint rulesets.

Cloud and SaaS security

Adversaries are increasingly targeting cloud environments and, as mentioned in part 2, we increased our focus on cloud and SaaS over the last year with a particular focus on AWS and Okta. Moving forward, we plan to expand to other cloud and SaaS platforms and continue to concentrate our efforts on key areas within these core attack surfaces:

- **Securing compute and containerized workloads:** Tracking access to compute and containerized workloads, identifying vulnerabilities and misconfigurations, and detecting container breakouts and cryptominer distribution by monitoring logs from both local and cloud-managed instances.

- **Securing APIs and data workflows:** Monitoring API and network flow logs for anomalies, and improving detections for unauthorized data storage access by identifying anomalous CRUD (Create, Read, Update, and Delete) operations and potential exfiltration.
- **Detecting identity and access management (IAM) and credential abuse:** Monitoring for suspicious IAM activities and use of compromised credentials by incorporating entity analytics and contextual awareness into our detection methods. Identifying anomalies in authentication and authorization workflows (e.g., OAuth, SAML, OIDC).
- **Building context-aware detections:** Enhancing detection accuracy by correlating cloud-based threats with endpoint footprints and using context-aware capabilities (e.g., CSPM, CNVM, asset inventory, and risk scores) to fine-tune detection logic.
- **Researching cloud-focused threat actors:** Tracking and researching adversarial behaviors and footprints from groups targeting cloud and SaaS environments.

In order to achieve these goals, we must improve containerization auditing, enhance our emulation and threat modeling processes, and fine-tune existing detections for core API services and data workflows.

Innovations in rule development enablement

We are constantly searching for ways to optimize our rule development lifecycle. In this section, we want to share some of the key areas of innovation we plan to address as the year unfolds.

Process improvements

There are many ways we can improve our rule development process to the benefit of both our detection engineers and our users, including:

- **Automating RTA coverage using known/recent hashes:** Right now the RTA development process is completely manual. By using known malware hashes to trigger certain rules, we will greatly reduce rule developer burden while maintaining our testing standards.

- **Adjusting detection rule severity scores to make alerts more meaningful:** We will explore the use of factors like asset criticality, user behavior history, and AI-driven risk analysis to determine rule severity scores either globally or per-environment.
- **Tracking tuning efforts over an individual rule's lifetime:** This will help us determine the value of extensive rule tuning efforts when compared to a rule's ability to capture real-world threats like VirusTotal malware samples and ensure our efforts are placed on the most impactful rules for users.

Contextual data and entity analytics

Similar to our goals in cloud, integrating Entity Analytics for targeted data collection will enable context-aware detections across endpoint platforms. These represent ideas for capabilities that we will advocate for and hope to explore over the next year, like:

- **Collecting data on installed patches, user attributes, and hardening status:** This will help us utilize the security posture of an environment as part of detection criteria.
- **Specifying detection conditions based on user-specific technologies and attributes:** This will improve the relevance and accuracy of our detections, making them more context-aware.

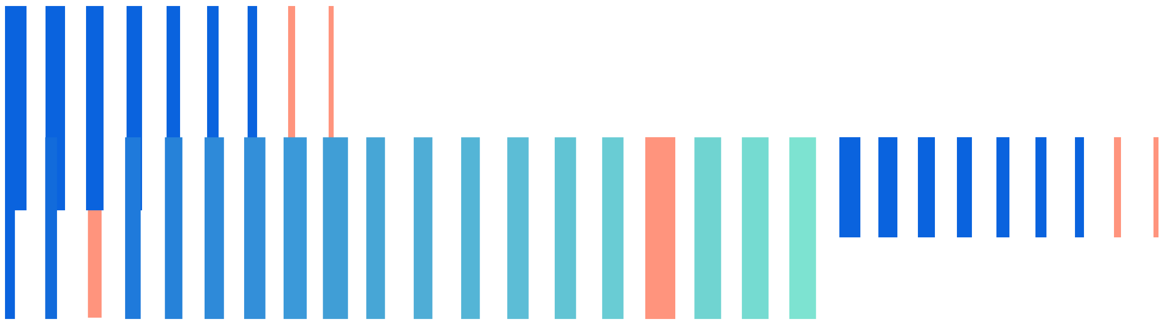
AI and machine learning assistance

Our team found the discussion around the potential for AI and ML to enhance our rule development process particularly engaging. While specific implementations are still exploratory and do not reflect Elastic's official product roadmap, we are eager to investigate the possibilities of AI and ML-driven advancements in areas such as dynamic threshold adjustment, false positive reduction, and rule tuning over the coming year and beyond, including:

- **Using AI to dynamically adjust detection thresholds and suggest exceptions:** Reducing false positives and improving the accuracy of our detections.

- **Generating False Positive (FP) reports:** Quickly identifying and addressing issues with our detection logic, and potentially providing us with recommended rule ideas.
- **Leveraging Elastic AI Assistant for Security:** Applying rule tunings to filter out user-suggested false positives, checking against telemetry and malware hashes, and creating drafts for rule changes.
- **AI for malware analysis:** Utilizing AI for static binary analysis and potentially dynamic malware analysis through a cloud sandbox running extensive ML models on binaries that seem to be suspicious.
- **Rescanning suspicious malware samples:** Using AI models to rescan suspicious low true positive malware samples in VirusTotal and other malware databases to find novel malware.
- **GenAI for telemetry analysis:** Implementing workflows for saving interesting alert telemetry, parsing it daily with an LLM, and sending Slack messages for notable events.
- **Automated alert grouping and summarization:** Utilizing GenAI to automatically group and summarize similar alerts within the UI, provide explanations for noisy patterns, and suggest tuning options.

We're incredibly enthusiastic about the strategic direction we've outlined for our detection engineering efforts in the coming year. Our roadmap prioritizes proactively addressing emerging threats, bridging detection gaps, and harnessing the transformative potential of AI and automation to optimize our rule development. We're also particularly excited to champion innovative ideas for Elastic capabilities and push the boundaries of what's possible.



Conclusion

The challenges of today's threat landscape are immense — far too big for any one company to solve. Elastic approaches these problems by fostering community, whether it's encouraging discussion or running our detection bug bounty program. These efforts create positive changes to the threat landscape; but even more importantly, they continue the conversation.

In order to demystify the threat landscape, it must be discussed thoroughly. How we approach it, break it down, create protections for it. Yes, this is exploring threats and actor behavior, but it is also exploring our own efforts and calling the community to hold us accountable. We're continuing the conversation on Elastic Security Labs, and we'll keep making Elastic Security the best it can be.

Some companies create products to make revenue, we're tuning a security tool to make it the most effective version possible. To give our users one less thing they have to put resources into.

There's no secret to delivering robust and reliable security capabilities that meet our user's needs, it's just a lot of hard work and planning. Seeing what the threat landscape throws at us and adapting. Those adaptations create improvements, and those benefit everyone.

So whether you're a veteran detection engineer, a security tool creator, or you downloaded this report to take a break, we're glad that you came by. We're glad that you participated in the ongoing demystification, that you chose to spend some of your limited time with us. We're excited to collaborate with you, directly or indirectly, on making the threat landscape easier to navigate.

The 2025 State of Detection Engineering report by Elastic incorporates insights and expertise from various departments within the Elastic organization. We extend our sincere gratitude to the following Elastic employees for their valuable contributions and dedication to producing this publication:

Isai Anthony
Mika Ayenson
Samir Bousseaden
Terrance DeJesus
Eric Forte
Ruben Groenewoud
Alyssa VanNice
Colson Wilhoit

The success of Elastic Security's threat research and detection engineering is thanks to the continuous and iterative efforts of many different teams at Elastic, including:

- Georgii Gorbachev, Marshall Main, and the Detections and Response team
- John Uhlman and the Endpoint Protections team
- Devon Kerr and the Malware Analysis team
- Chris Donaher and the Security Data Analytics
- Joe Desimone and Justin Ibarra of Security Intelligence
- Dan Kortschak, Gabe Landau, Norrie Taylor, and Ricardo Ungureanu of the Security Integrations team
- The Security Machine Learning team
- Kseniia Ignatovych, James Spiteri, and the rest of the Security Product Management team
- The Threat Data Services team
- Sergey Polzunov, Jonhnathan Ribeiro, and Shashank Suryanarayana of the Threat Research and Detection Engineering team