



Global Threat Report

2024 —

Table of Contents

1	Introduction	03
2	Generative AI	04
	Threat overview	04
	Augmenting defenders	05
3	Malware Detections	06
	Distribution by operation system	06
	Malware categories	07
4	Endpoint Behaviors	11
	Distribution by operating system	11
	Distribution by tactic	12
5	Cloud Security	36
	Distribution by cloud service provider	37
	Benchmarking cloud security posture	47
6	Threat Profiles	57
	REF5961 — BLOODALCHEMY, RUDEBIRD, EAGERBEE, DOWNTOWN	58
	REF8207 — GHOSTPULSE	61
	REF4578 — GHOSTENGINE	64
	REF7001 — KANDYKORN	66
	REF6127 — WARMCOOKIE	69
7	Responding to 2023 Forecasts	72
8	Forecasts and Recommendations	75
9	Conclusion	79

Introduction

With the best technologies, the most widespread information distribution, and the greatest public awareness of threats all in motion, the security environment is stronger than ever. Yet, almost in spite of these things, threat ecosystems are thriving like never before.

Truthfully, the threat landscape is dynamic and reactive — a new technique empowers a previously unknown threat group, vendors swarm to mitigate that threat and create new technologies in the process, operators on both sides seek out new techniques or tools, and so it goes. This landscape is shaped daily by the efforts of attackers and defenders alike. This report is one of the ways that [Elastic Security Labs](#) holds itself accountable as a force of change for good in a dynamic environment — by sharing what we're seeing, as well as what we think needs further visibility.

The intention with our Global Threat Report is the same as it's been over the past couple years: to democratize knowledge, uphold transparent principles, and identify impacts. We are leveraging Elastic's powerful global technology to provide unique insights that inform our priorities for the

Elastic Security solution and serve the security community at large.

This report would not be possible without the security commonwealth — our observations are based on billions of security events voluntarily shared by our users, enriched with open sources, representing tens of thousands of distinct entities in practically every industry. The valued partnership we have with our users enables us to discover previously unknown threats in their data and anonymously share those discoveries with the broader security industry. This collaboration has led to hundreds of new protections, freely available to the public. We'd like to extend a huge thank-you to our users — the learnings surfaced from this shared data benefit all of us throughout the security community.

As a research group, we power and are empowered by Elastic technologies. In many ways, we also demonstrate what can be achieved with the [Elastic Security](#) solution. The same visibility and capabilities our users have provided are powering these insights, and we're excited to see the other positive contributions that they'll make to the threat landscape at large.



Generative AI

The rise of [generative artificial intelligence](#) (generative AI) brought tremendous excitement, with the hope that this technology will soon touch almost every aspect of life in one way or another. This excitement also brought about understandable hesitation alongside a desire to understand the ways this new technology could be abused.

Elastic is no stranger to generative AI. While many of our colleagues are building [incredible capabilities](#) into our products, Elastic Security Labs and our partners in the InfoSec house have also devoted considerable time to understanding the technology and the associated risks.

are difficult to distinguish from legitimate communications. While this technique is still maturing, examples of these incidents are already being [studied](#).

At a wider scale, deepfake-based scams have interfered with political elections and some extortion cases ([ABC](#), for example). The use of generative AI tools to create deepfakes — video or audio of a person that has been manipulated to spread misinformation — will continue to be a threat in the coming years. It's important to note, however, that deepfakes have been limited in terms of cybersecurity incidents.

Both of these threats will continue to accelerate, which reaffirms the importance of training users to identify AI creations. Organization-wide cybersecurity training programs have been crucial for stopping regular phishing attempts, so it's important for CISOs and other security leaders to implement AI-focused training into their programs as well.

Threat overview

Augmented phishing and social engineering

Like many, we anticipated that one of the first threat use cases for generative AI would be the creation of more sophisticated phishing campaigns. Suddenly, threat actors can scale the creation of personalized documents that

Malware development

There has been [research](#) highlighting how AI has been used to create more adaptive malware, but this technology has yet to see wide adoption. The full extent of AI-driven malware availability continues to

be a concern, and as models and services become more accessible, this will continue to develop. Like regular malware development, security teams should remain up to date on threats and trends. Additionally, maintaining a robust [protections library](#) with consistently updated and [tuned rules](#) will be crucial.

Augmenting defenders

Like any technology, generative AI can be abused; however, that doesn't mean it isn't an incredibly powerful tool in the hands of defenders.

Enhanced cybersecurity tooling

One of the many exciting prospects of building generative AI capabilities specifically for defenders was the idea of advanced threat detection — specifically a way to automatically synthesize alerts and distill them into the highest-priority attacks. Automating repetitive manual triage tasks empowers practitioners to remove the monotony and focus their efforts on strategic initiatives instead. We're excited that Elastic has built this function into our Security solution, and it exists to aid defenders in their day to day with [Attack Discovery](#).

Another exciting use for defending teams comes in the form of automated security testing, which has seen increased adoption in recent years. This has not replaced traditional testing methodologies fully — and we don't anticipate them doing so — but they have become valuable in more effectively identifying vulnerabilities and

should continue to improve.

Going even further, generative AI provides enhanced cybersecurity training for InfoSec teams, with simulations becoming more realistic. While this has much potential, it is still in the early stages and will need some refinement before it sees widespread adoption.

Governance and ethics

Generative AI is powerful, and it will remain an important tool for Security teams and practitioners as time goes on. But utilizing powerful technology requires a firm, ethical hand. Thankfully, organizations and governments around the world are racing to develop safety guidelines and responsible frameworks.

Some examples of these include:

- [NIST AI Risk Management Framework](#): From the National Institute of Standards and Technology, this framework is focused on the governance, risk mapping, measuring, and management of AI.
- [FAIR-AIR](#): This framework from the FAIR Institute helps teams identify AI-related loss exposure and make risk based decisions.
- [AI TRiSM](#): Gartner's framework ensures reliable, safe, and compliant AI implementations with emphasis across the entire lifecycle.

Elastic Security Labs is excited to keep an eye on this emerging technology and will continue to report on its evolution. A more in-depth look at threats and protections to generative AI applications — specifically large language models — can be found in our [LLM Safety Assessment](#).

Malware Detections

Elastic Security provides mechanisms to detect and mitigate malware on all major desktop operating systems (OS). For these purposes, malware is any software developed to facilitate adversary actions, disrupt legitimate activities, or otherwise cause harm to a computer or network.

In this section, readers can find details about the distribution of malware by OS, category, and family, as well as a dedicated breakdown of ransomware observations.

This year, Elastic Security Labs reviewed malware and memory threat protection alerts from Elastic Security. To improve the accuracy of our malware observations, this subsection only includes YARA signature events for named malware families. YARA signatures are a powerful component of Elastic Security, providing the ability to mitigate malware using string- or byte-sequences found in

executables. These signatures, which apply to both file system and memory-resident software, are available to the public through [Elastic's Protections Artifacts repository](#) as part of our ongoing commitment to free and open principles.

The materials presented in this section of the Elastic Global Threat Report include the most significant threat capabilities and phenomena observed in Elastic's telemetry over the past year. Elastic users voluntarily share these alerts and other data with us, providing Elastic Security Labs with a powerful tool for discovering, diminishing, and disrupting threats. This telemetry consists of data from Elastic Security, Elastic Agent, and a diverse range of third-party instrumentation.

Distribution by operating system

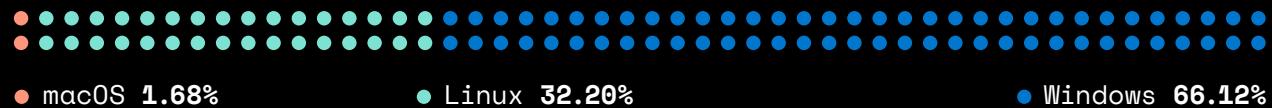


Figure 1: Malware infections by operating system

Windows

The distribution of YARA events across each OS suggests some disparity in the prevalence of threats. Windows hosts accounted for the majority of detections with 66.12% of all detected cases. This isn't entirely unexpected given the widespread use of Windows in enterprise environments and its susceptibility — both historically and in contemporary terms — to various techniques employed by malware. While it would be deceptive to state that any one OS is "most likely" to be infected, techniques like [Bring Your Own Vulnerable Driver \(BYOVD\)](#) represent architectural conditions that threat actors frequently target to achieve their goals.

Linux

Linux hosts still represent a significant portion of infections at 32.20%. This may suggest that adversaries are increasingly [targeting Linux systems](#), likely due to their prevalence in server environments and critical infrastructure. Linux containers, which are often intended to exist for minutes or hours versus weeks or months, may contain unpatched vulnerabilities that even nascent threats can exploit.

Readers should note that last year we reported 92% of malware infections were on Linux endpoints due in part to the inclusion of weakly attributable events. This shift has occurred in part from the changes our team made to the way we process Elastic's telemetry — providing a more accurate representation of common endpoint malware observations, as well as the broader categories of malware described.

macOS

macOS hosts represent the smallest proportion of endpoints from which Elastic receives telemetry. With that in mind, macOS also represents the fewest malware observations at 1.68%. Elastic Security Labs does not conclude that macOS is more secure, less widely present in enterprises, or less likely to be targeted. In actuality, research that we conducted earlier this year around [pirated macOS applications](#) found that several straightforward methods of infecting these systems were widely used.

Malware categories

Malware subcategorization will be necessarily subjective, defined from the perspective of each vendor or reporting entity. The categories presented here are aligned to collections of YARA signatures, and are explained in more detail below.

Malware category	SUM
Trojan	82.03%
Generic	8.03%
Cryptominer	4.39%
Ransomware	2.10%
Backdoor	1.01%
Other	2.44%

Table 1: Categories of malware observed

Trojans account for 82.03% of all malware types observed, a proportion attributed to the useful nature of masquerading as legitimate software. Once executed, Trojans often deploy additional malicious payloads like [infostealers](#), effectively serving as a delivery mechanism for various types of malware.

Last year, we reported that Trojans accounted for around 61% of all malware types we saw. The 21% increase is attributed to a small but broadly-distributed number of trojanized applications. While many of the organizations who chose to share data employ application-level controls, few adopted block or allow lists in ways that constrain the execution of unknown software.

The “Generic” category comprises 8.03% of infections, representing broadly identified threats that do not fit specific malware classifications but still pose significant risks. For example, generic malware might be developed by an aspiring malware developer new to the ecosystem. We observed a significant change in cryptominer identifications this year, decreasing from 21.80% to 4.39%. Cryptomining software takes advantage of resource scalability to calculate cryptocurrencies, and has played increasing roles in financially-motivated scenarios; families like [GHOSTENGINE](#), which was discovered in May 2024, contain a cryptomining application that can be installed during intrusions. More information on GHOSTENGINE can be found in the *Threat Profiles* section of this report.

YARA-based ransomware observations represented 2.10% of detections and further reinforces the fact that ransomware remains a critical threat due primarily to the resulting impacts of extortion, theft, and reputational loss. YARA works as a complement to ransomware and other protections — one with a shorter distribution interval ideal for the rapid response to emerging threats.

Backdoors account for 1.01% of observed malware and will be familiar to many readers, often used like Trojans to provide mechanisms for intrusion. Not all readers may be aware that remote monitoring and management (RMM) software — less than half of the 1% — is the tool of choice for many scammers who use urgency alongside other social engineering approaches to convince users to self-infect.

Malware families

We can examine the distribution of YARA signature matches to identify that a small number of malware families make regular appearances: Cobalt Strike, Metasploit, Sliver, DONUTLOADER, and Meterpreter represent about two-thirds of all malware we saw last year. Reflecting on the distribution of malware by category (figure 2), there was significant overlap between Trojanized software and the presence of these malware families. We’re presenting the information in this way (excluding ransomware families) because those were most commonly seen during later stages of intrusions, after many of these prevalent malware families were already present in the environment. Enterprises need to prioritize malware during all stages of the intrusion lifecycle in order to de-risk bad outcomes.

Offensive security tools (OSTs) are a common topic of heated debate right now, due largely to the frequency with which they are abused by malicious actors. The most commonly seen malware families correlated primarily to OSTs – a significant increase since last year. However, it is essential that readers understand that the offensive security community exists for their benefit.

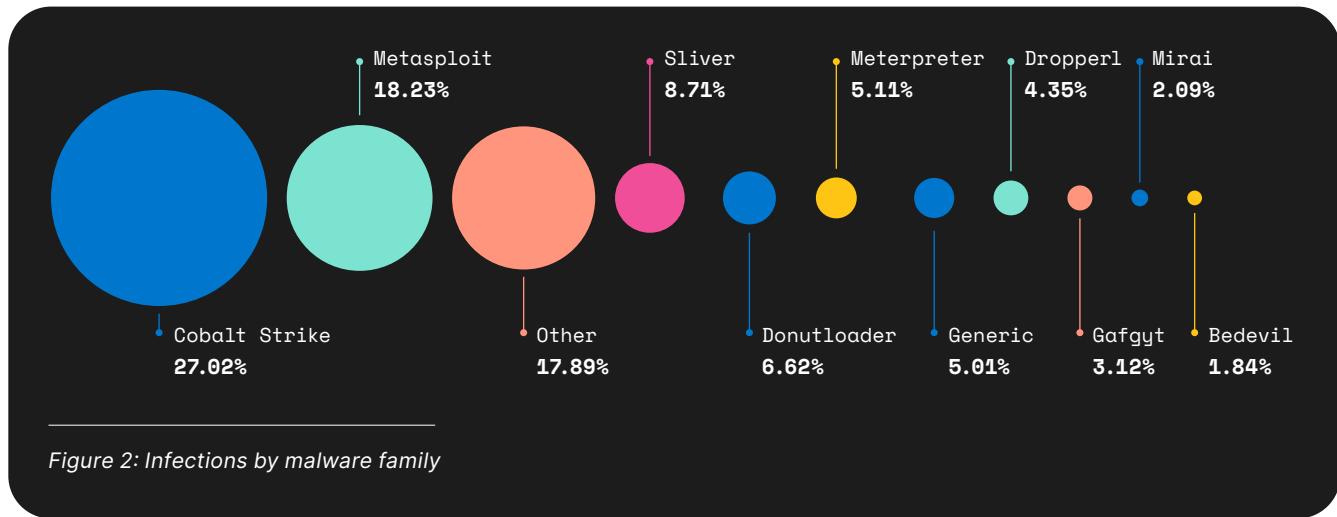


Figure 2: Infections by malware family

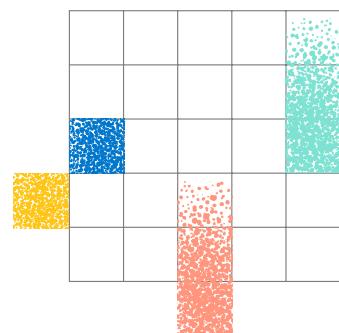
The most prevalent malware family we observed this year was [Cobalt Strike](#), accounting for 27.02% of infections. Cobalt Strike is a very mature commercial post-exploitation framework with an experienced research and development team. It is so effective that [threat actors](#) frequently steal and weaponize this product to further their malicious objectives, rather than the benign purpose it was intended for. Metasploit variants represented 18.2% of YARA matches and are another example of OSTs abused by threat actors. [Meterpreter](#), the reverse shell bundled with Metasploit, also appeared in the top 10 common families, accounting for 5.1% of signals. Elastic Security Labs [maintains visibility](#) of these families, which regularly appear together in the wild. [Sliver](#), with 8.71% of infections, is an OST designed for adversary simulation. Its use in post-exploitation activities demonstrates the power of its capabilities.

Sightings of Sliver increased significantly from last year. [DONUTLOADER](#) amounted to 6.62% of infections, and serves as a loader to execute additional malicious payloads in memory, avoiding disk-based detection mechanisms.

Malware families such as Gafgyt (3.12%), Mirai (2.09%), and Bedevil (1.84%) appeared less often than in prior years, which may be a reflection of attempts to neutralize botnets from propagating. These malware families are typically distributed to Internet of Things (IoT) devices like residential broadband routers using hardcoded credentials or unpatched vulnerabilities, and are used to launch distributed denial-of-service (DDoS) attacks and to hijack advertising or DNS networks.

The distribution of malware families reiterates the importance of endpoint instrumentation capable of identifying and mitigating malicious software, and enterprises that rely on visibility over capability may experience worse outcomes than those who don't.

The use of OSTs by threats is a strong indicator that technical innovations, procedural maturity, adoption of least privilege, and information sharing is impacting malicious actors.





Endpoint Behaviors

Security practitioners often define a threat in terms of tactics, techniques, and procedures (TTPs). In other words, what are the goal-aligned methods used by threats? This section describes the most commonly observed tactics, techniques, and procedures (TTPs) on Windows, macOS, and Linux.

Distribution by operating system

- Linux 3.30% ● macOS 3.97%

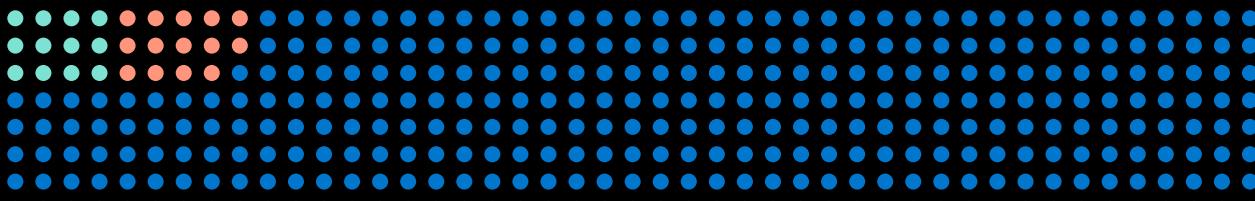


Figure 3: Endpoint behavior alerts by operating system

Windows

The majority of endpoint behaviors we observed were seen on Windows hosts, accounting for 92.73% of all sightings. This is a marginal decrease from 94.2% last year, a reduction of about 1.5%. These statistics reflect the proportions of systems sharing telemetry with us, populations that

fluctuate regularly to small degrees.

Linux and macOS

Linux events increased from 2.80% last year to 3.30%, a fluctuation of a half-percent. At 3.97%, macOS events increased not quite 1% since last year. The increase in macOS detections can be

attributed to a growing interest from adversaries in exploiting macOS vulnerabilities, as highlighted in our [research](#) earlier this year.

Because relatively rare phenomena can be artificially amplified in small populations of systems, readers should be aware that macOS and Linux statistics are not significant and may not resemble what you are seeing. These have been included as data points and do not influence recommendations.

achieving them. Each prebuilt behavior protection Elastic develops is aligned to [MITRE ATT&CK®](#), the most widely-accepted taxonomy of TTPs, and this section will reflect that.

At a high level:

- *Persistence* hits increased by nearly 8%
- *Defense Evasion* events decreased to 38%, a nearly 6% difference from last year
- *Execution* remained prevalent, representing around 16% of detected tactics

The *Persistence*, *Defense Evasion*, and *Execution* tactics are commonly employed in some combination by adversaries during intrusions. These three make up almost 70% of all behaviors seen over the prior year, a reduction from last year when they amounted to more than 81%.

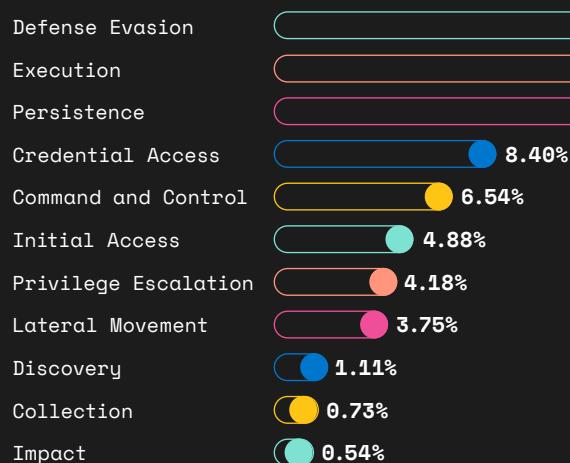


Figure 4: EDR behavior alerts by tactic

Surprisingly, we observed a nearly 2% decrease in detections related to *Privilege Escalation* when compared to the previous year – the growing threat of information stealers and the widespread distribution of stolen credentials may have reduced the necessity of this tactic. For instance, there was a slight 3% increase in *Credential Access* detections as [described](#) in our analysis earlier this year.

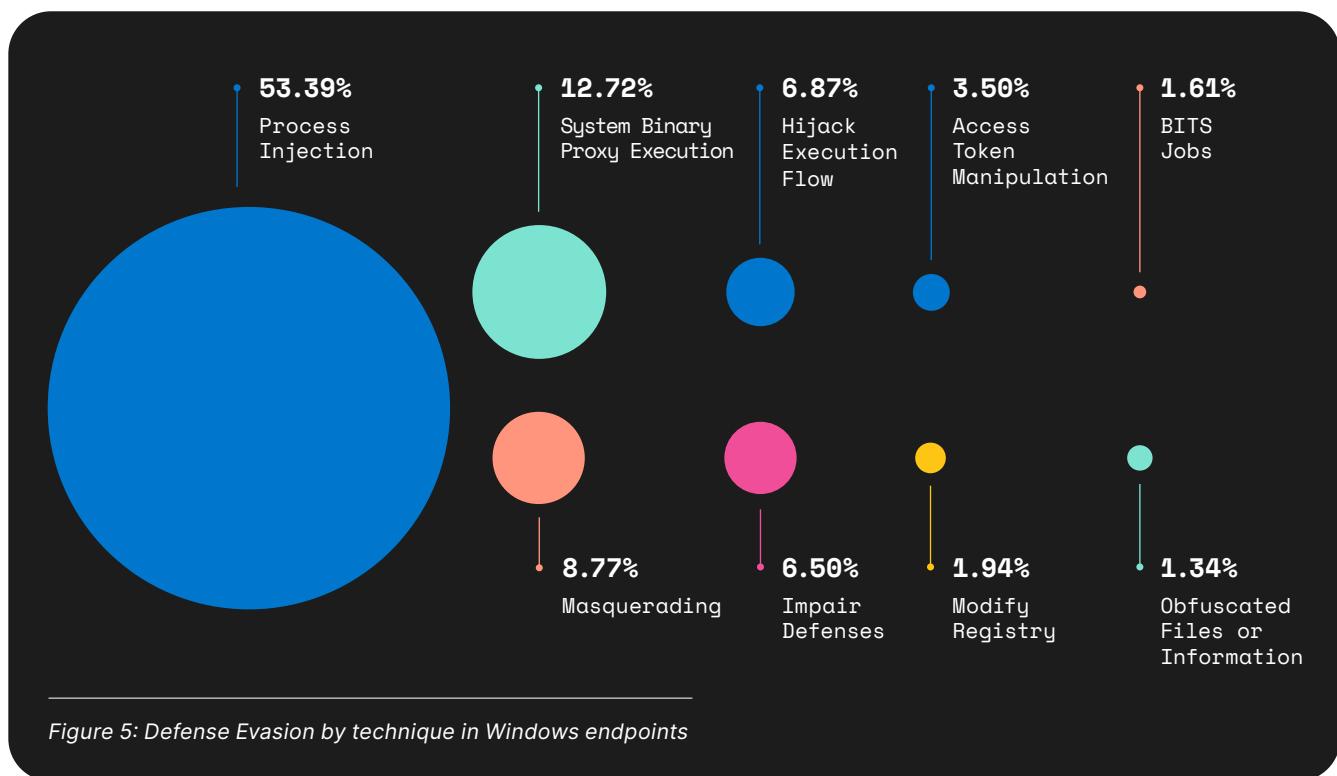
Defense Evasion

Defense Evasion remains the prominent tactic among adversaries. Accounting for approximately 38% of all detections, these are techniques used to bypass or blind security capabilities.

Windows

The most common techniques in this category — *Process Injection*, *System Binary Proxy Execution*, and *Impair Defenses* — collectively describe entire attack chains: a threat actor exploits a vulnerability in a client application, injects malicious code into

a privileged process, and spawns `rundll32.exe` to execute an adversary-controlled dynamic link library (DLL) that writes a vulnerable driver to disk that is used to disable Endpoint Detection & Response (EDR) sensors.



This year, *Process Injection* detections increased by approximately 31%, now accounting for 53.39% of all *Defense Evasion* detections. *Process Injection* is a commonly used method used to inject malicious code into other processes, making

it harder to detect and potentially allowing them to bypass security controls. *System Binary Proxy Execution*, which made up nearly 47% of detections last year, has decreased about 35% to 12%. This technique involves using

trusted system binaries to proxy the execution of malicious payloads, a method that has become less prevalent as detection capabilities have improved. The reduction in this technique suggests that adversaries are adapting their strategies in

response to enhanced security measures. The *Impair Defenses* technique increased marginally to 9.56%. We most commonly associate this with disabling security instrumentation or data sources.

rule_name	Percentage
Network Module Loaded from Suspicious Unbacked Memory	13.67%
Suspicious Memory Write to a Remote Process	6.05%
Potential Masquerading as Windows Error Manager	5.07%
Remote Thread Context Manipulation	4.22%
Potential Injection via an Exception Handler	3.88%
Potential Remote Code Injection	3.87%
Potential Evasion via Sleep Obfuscation	3.63%
Suspicious Remote Memory Allocation	3.30%
Microsoft Common Language Runtime Loaded from Suspicious Memory	3.23%
Suspicious Windows API Call via ROP Gadgets	3.07%

Table 2: Top 10 Process Injection by rules in Windows endpoints

Why the significant increase in *Process Injection* techniques? Breaking down process injections by prebuilt Elastic Security behavior rules, a majority were caused by suspicious memory intrusion events originating from Windows exception handlers and accounted for 9% of detections. This detection logic focuses on a specific set of Windows API behaviors and call stack routines where symbols related to exception handlers are present. Elastic Security Labs has analyzed similar behavior with the shellcode-based downloader, [GULADER](#).

We also observed a significant amount of suspicious endpoint behavior related to detecting unbacked memory regions in process address space — nearly 14% of *Process Injection* detections were unbacked code, meaning they were not directly linked to any executable file on disk. Malicious actors frequently use unbacked

memory to inject or execute malicious code into existing processes. This can be accomplished via *Process Injection* or *Shellcode Execution*. Elastic Security's Windows-based agents have visibility into API calls and call stacks, enabling detection based on native DLLs loaded where call stack analysis shows frames indicating

memory regions not backed by any known executable image on the file system. Sliver, one of the prevalent *Command and Control* (C2) frameworks described previously in this report, uses *Process Injection* and unbacked code as described in [Hunting in Memory](#) and [Upping the Ante](#).

System Binary Proxy Execution remains prevalent for adversaries, well distributed across different techniques to accomplish this. From Elastic Security detections, it's evident that nearly 20%

of all this activity relates to abusing [rundll32.exe](#). Elastic Security Labs took note of this while analyzing [LATRODECTUS](#). Specifically, similar malware families will download a DLL from respective C2 servers, write to disk with a randomly generated file name, and simply execute the malicious code in the DLL using [rundll32.exe](#). Alternatively, in Elastic Security Labs' discovery of [WARMCOOKIE](#), a PowerShell script invoked the Background Intelligent Transfer Service (BITS) utility to download and execute a malicious DLL.

rule_name	Percentage
Script Execution via Microsoft HTML Application	13.71%
Unusual DLL Extension Loaded by Rundll32 or Regsvr32	11.55%
Suspicious Execution via DCOM	10.68%
Binary Proxy Execution via RunDLL32	8.30%
Suspicious MsiExec Child Process	8.04%
Regsvr32 with Unusual Arguments	7.75%
Execution via Renamed Signed Binary Proxy	7.01%
RunDLL32 with Unusual Arguments	6.85%
Suspicious Execution via DotNet Remoting	2.90%
Potential Evasion via DotNet Framework Installation Utility	2.64%

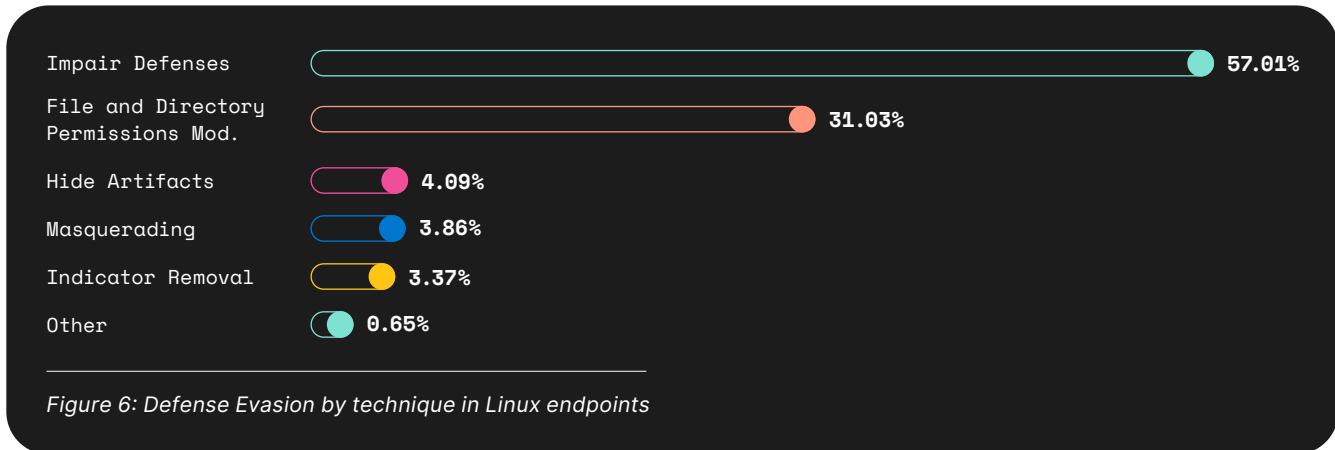
Table 3: Top 10 System Binary Proxy Execution by rules in Windows

Execution via Microsoft HTML applications was also common, decreasing about 4% to 14% of total *System Binary Proxy Execution* detections. This identifies the execution of scripts via HTML applications using either [rundll32.exe](#) or [mshta.exe](#). Adversaries may bypass process and/or signature-based defenses by proxying execution of malicious content with these signed

Microsoft binaries, present on all Windows systems by default. Notably, the abuse of [mshta.exe](#) was more prevalent than [rundll32.exe](#). Researchers noticed malicious code being hosted in files with [.hta](#) extensions whereas with [rundll32.exe](#), Windows Script Host (WSH) interpreters were often used to run malicious scripts.

Linux

Although *Defense Evasion* was the most common tactic observed for Windows, in Linux environments it accounted for 10.67% of all tactics.



When breaking down *Defense Evasion* by techniques in Linux, we observed that 57.01% related to *Impair Defenses*. *File and Directory Permission Modifications* accounted for 31.03% and *Indicator Removal* accounted for 3.37%. Regarding table 4, adversaries most often

attempted to disable iptables and/or firewall services via `ufw` or `iptables` native tools. Cryptocurrency miners and botnet variants commonly flush/add `iptablesrules` one-by-one, triggering these alerts multiple times during an infection.

Impair Defenses — Linux	Percentage
Attempt to Disable IPTTables or Firewall	59.98%
Kernel Module Removal	19.71%
Elastic Agent Service Terminated	6.24%
Potential Disabling of SELinux	6.21%
Attempt to Disable Linux Security and Logging Controls	4.34%
Attempt to Disable Syslog Service	3.51%
Other	0.02%

Table 4: Impair Defenses techniques in Linux endpoints

Manipulating Linux kernel modules was relatively common, leveraging `modprobe`

and/or `rmmmod` to remove specific modules via the command line. It was also very common for

threats to attempt to modify file permissions in writable directories, which accounted for 26.74% of all *Defense Evasion*. More specifically, adversaries would modify file permissions in common writable directories by a non-root user on Linux.

Adversaries often attempt to drop files or malicious payloads into a writable directory and change permissions prior to *Execution*. Common commands leveraged to achieve this are `chattr`, `chgrp`, `chmod`, and `chown` that point to working directories `/dev/shm` or `/var/tmp`.

rule_name	Percentage
File Deletion via Shred	34.16%
System Log File Deletion	33.07%
Tampering of Bash Command-Line History	20.43%
Tampering of Shell Command-Line History	8.98%
WebServer Access Logs Deleted	3.16%
Other	0.20%

Table 5: Indicator Removal by rule in Linux endpoints

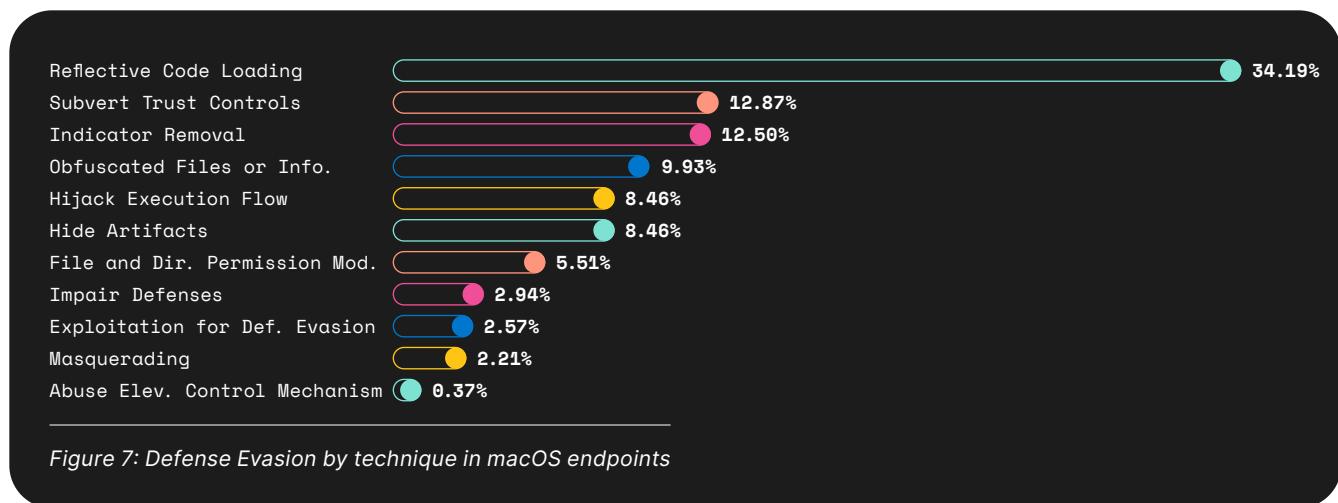
Elastic Security Labs researchers observed *Indicator Removal* on Linux hosts that involved file deletion, command-line history tampering,

and access log deletion. Organizations should monitor for log deletion events, especially by unexpected accounts.

macOS

Within macOS, *Defense Evasion* accounted for 27.59% of all tactics. These alerts were spread across several techniques with *Reflective Code*

Loading at 34.19%, *Subverting Trust Controls* at 12.87%, and *Indicator Removal* encompassing nearly 60% of all *Defense Evasion*.



Breaking this down further, Reflective Code Loading, specifically Reflective Dynamic Library (Dylib) Loading, has been observed being used by advanced threats in order to load additional payloads into previously compromised or malicious processes. This technique was observed during

Elastic Security Labs' discovery of [KANDYKORN](#) malware, which was attributed to nation-state Democratic People's Republic of Korea (DPRK) activity in October 2023 and continues to be a stealthy approach for *Defense Evasion* on macOS.

Execution

Execution, as expected, continues to be a common tactic in adversary playbooks via both malicious binaries and toolkits. In the most general of terms, all the techniques organized in this category involve methods of executing adversary code, either directly or indirectly. Even when deploying an otherwise legitimate tool for *Remote Access*, threats are employing *Execution techniques*.

Windows

While decreasing about 13% since 2023, *Command and Scripting Interpreters* remained the most common *Execution* category technique with around 58% of all *Execution*. Windows Management Instrumentation (WMI) makes up

around 24% — the significance here is the surge in WMI abuse to execute malicious commands and payloads, along with new techniques not reported on in previous years like *Native API*.

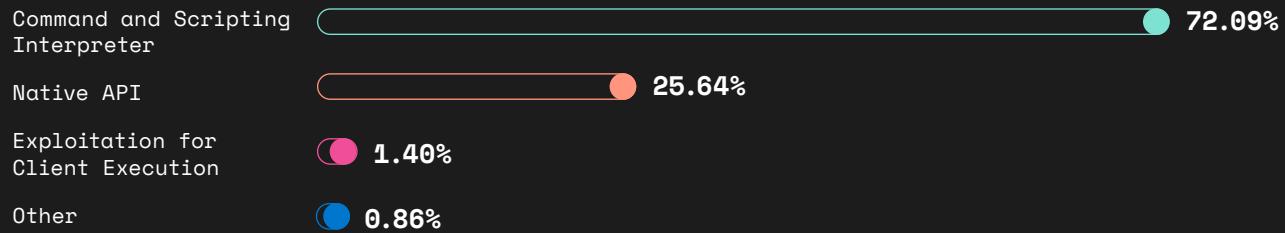
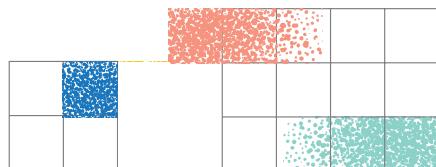


Figure 8: Execution by technique in Windows endpoints

It's important to note that *Execution* tactics displayed a wide distribution of rules compared to other tactics, highlighting the diverse methods adversaries use to achieve *Execution* objectives. This also underscores the multifaceted approaches attackers employ.

EDR Behavior Execution Command and Scripting Interpreter by Rule	SUM of Percentages
Suspicious PowerShell Execution	20.20%
Execution of a Windows Script File Written by a Suspicious Process	12.61%
Suspicious Windows Command Shell Execution	10.12%
Execution of a File Written by Windows Script Host	8.33%
Suspicious Execution from a Windows Script	7.45%
Suspicious PowerShell Execution via Windows Scripts	6.03%
Suspicious Windows Script Interpreter Child Process	5.13%
Windows Script Execution from Archive File	4.32%
PowerShell Engine Loaded via Injection	3.58%
Potential Command and Control via Windows Scripts	3.09%
Unusual PowerShell Engine ImageLoad	2.83%
Suspicious JavaScript Execution via Node.js	2.78%
Command Shell Activity Started via RunDLL32	2.13%
Suspicious Windows Script Process Execution	1.90%
Suspicious Batch File Execution from a Mounted Device	1.37%
Command and Scripting Interpreter from Suspicious Parent	1.25%
Execution of a Windows Script with Unusual File Extension	1.18%
Other	5.71%

Table 6: Command and Scripting Interpreter alerts by rule in Windows endpoints



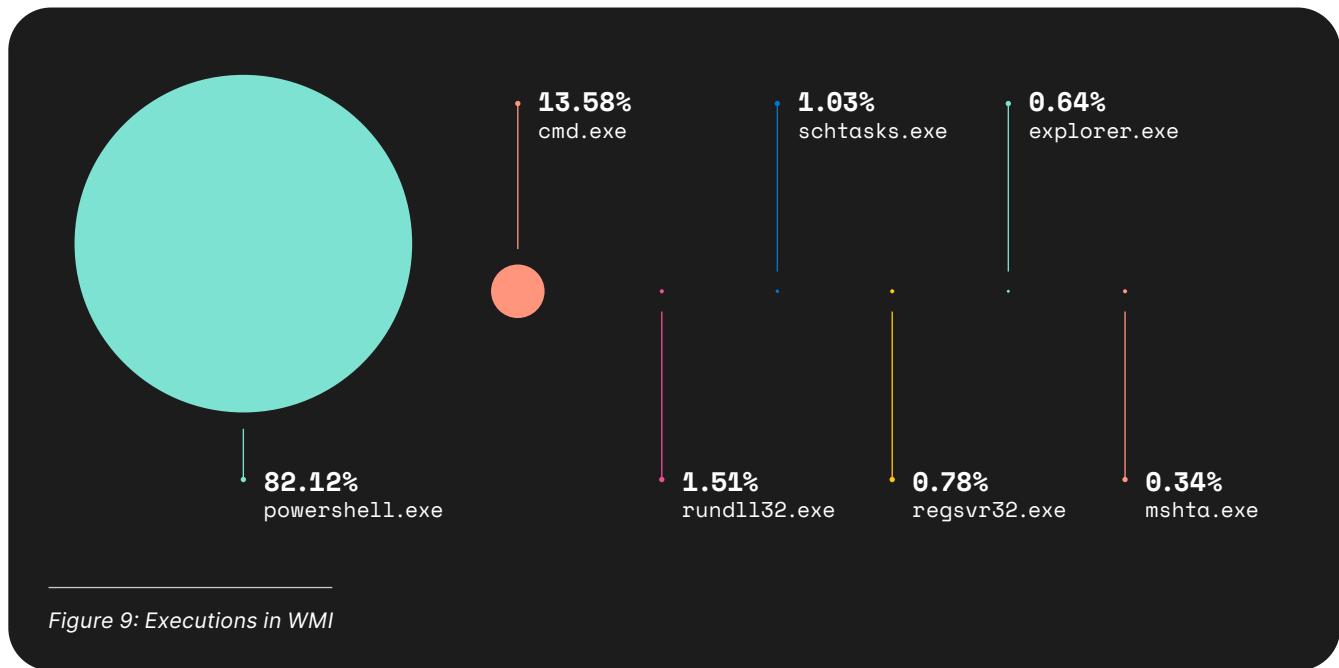
While there are many other Elastic Security behavioral rule results analyzed in this report, we will focus on the top 10 in terms of volume. *Suspicious PowerShell Execution* ranks #1, accounting for 20% of all *Command and Scripting Interpreter* activity. PowerShell abuse is a well-known technique among adversaries, so this should come as no surprise.

Based on our analysis of command line execution, the following summaries highlight key malicious activities (in no particular order):

- **Data gathering via WMI:** Querying various namespaces and classes related to clustering, virtualization, and system information.
- **Malicious reflective DLL loading and method invocation:** Loading DLLs into memory and invoking their methods.
- **Downloading and executing remote scripts:** Leveraging the `Net.WebClient` cmdlet to download and execute remote scripts.
- **Obfuscation techniques:** Using mixed cases, special characters, and other methods to obfuscate commands.
- **Memory-based DLL loading and execution:** Utilizing `[Reflection.Assembly]::Load([System.IO.File]::ReadAllBytes())` to load and execute specific DLLs directly in memory.
- **Privilege Escalation attempts:** Employing `runas` arguments and scheduled tasks to gain elevated privileges.
- **Registry and system configuration access:** Leveraging the `Get-ItemProperty` cmdlet to access and manipulate registry settings, particularly targeting the `HKEY_LOCAL_MACHINE` (HKLM) hive.

Earlier in 2024, Elastic Security Labs noted similar behavior from our [GHOSTENGINE](#) discovery and code analysis where a PowerShell script orchestrated the entire execution flow of that intrusion.

Regarding the execution of Windows Script Files written by suspicious processes, a notable amount of this activity involved `WScript.exe` execution targeting Visual Basic Scripting Edition (VBScript) files, typically stored in user `%AppData%` directories. This is indicative of script-based malware or malicious macros leveraging VBScript for execution. When Elastic described the discovery of [GrimResource](#), a novel technique for *Initial Access and Defense Evasion*, researchers provided a WSH example. Although uncommon in our data set, `mshta.exe` targeting HTML Application (HTA) files was also observed. HTA files are often used for executing scripts within an HTML framework, which adversaries often abuse for malicious purposes. While `mshta.exe` execution can be benign in certain circumstances, the scripts identified in our analysis were dropped by processes that were profiled to be suspicious. These profiles excluded activities from processes associated with the Windows User Security Identifier (SID) S-1-5-18, known trusted and signed processes, and native executable locations. This filtering helps in isolating potentially malicious behavior from legitimate activities.



WMI abuse has increased significantly, to the point that it should be both expected and well-understood by security teams. Our analysis of parent and child process relationships revealed that the WMI Provider Host process (`\WmiPrvSE.exe`) spawns many recognizable child processes such as PowerShell, CMD, RunDLL32, Scheduled Tasks, and more, as shown in the following visualization.

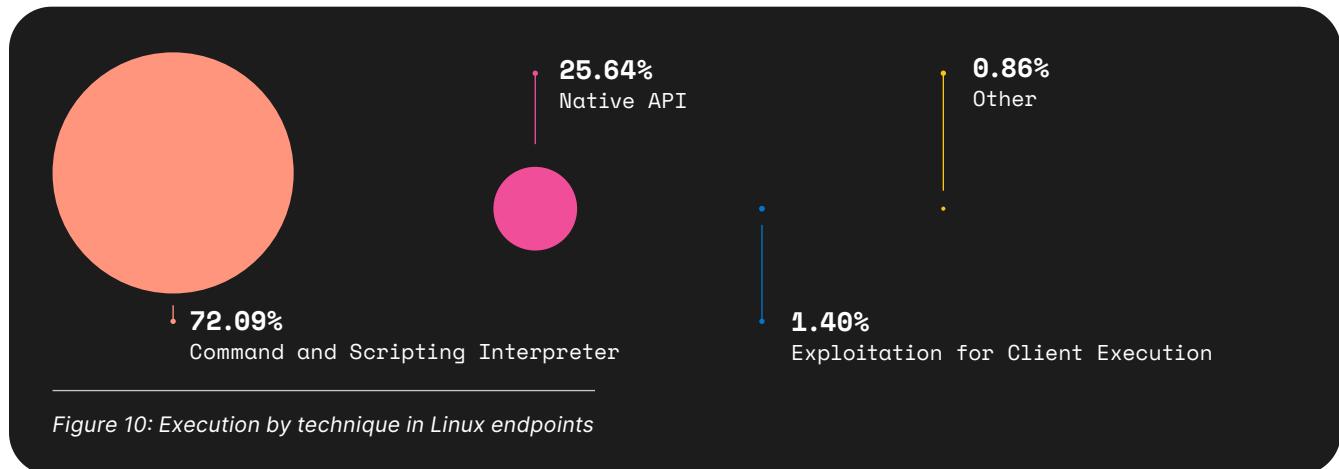
Each of these child processes provides insight into specific malicious actions taken via WMI, such as.

- **PowerShell execution:** `powershell.exe` commands were observed with actions like executing scripts from remote locations, adding exclusions to Windows Defender and enabling PowerShell remoting.
- **CMD execution:** `cmd.exe` commands observed included running PowerShell scripts, mapping network drives and executing batch files.

- **RunDLL32 execution:** `rundll32.exe` was used to execute DLLs, often for loading and running malicious payloads, indicating potential code injection and persistence.
- **Scheduled tasks creation:** `sctasks.exe` was used to create scheduled tasks, likely to ensure recurring execution of malicious scripts and/or binaries.
- **Registry modification:** `regsvr32.exe` was used to register or unregister DLLs, which can be leveraged for Persistence.
- **HTML application execution:** `mshta.exe` was used to run HTA files, which can execute scripts within an HTML framework, often exploited for Initial Access or to run complex scripts.
- **Network communication:** `curl.exe` and `bitsadmin.exe` were used to download files or transfer data during exfiltration or payload delivery stages.

Linux

Execution for Linux ranked as the third most common tactic observed and comprised nearly 14.56% of all alerts.



Command and Scripting Interpreters accounted for 72.09% of all techniques related to *Execution* on Linux, followed by *Native API* at 25.64%. *Command and Scripting Interpreter* was also the #1 ranked technique for *Execution* on Windows as well; however, *Native API* was not seen as often on Windows as it was on Linux endpoints.

rule_name	Percentage
Restricted Shell Breakout via Linux Binary(s)	42.65%
Interactive Terminal Spawning via Python	20.15%
Suspicious System Commands Executed by Previously Unknown Executable	9.96%
Potential Reverse Shell via Suspicious Child Process	7.50%
Linux Restricted Shell Breakout via Linux Binary(s)	3.92%

Table 7: Command and Scripting Interpreter alerts by rule in Linux endpoints

When analyzing the top five *Command and Scripting Interpreter* by alert rules, 42.65% related to *Restricted Shell Breakout Attempts via Linux Binaries*, followed by *Interactive Terminals Spawning via Python* parent processes. Restricted shell breakouts occur when adversaries attempt to abuse a native Linux binary to break out of a restricted shell or environment by spawning an

interactive system shell. Spawning these shells from a binary are typically not common behavior for a user or system administrators and are often attributed to tools like `bash`, `dash`, `ash`, `zsh` and more but can also be accomplished via `ftp`, `zip`, `tar`, and `strace` as well where process arguments include `exec`. Using these types of living-off-the-land binaries (LOLBins) are

also useful for breaking detections, by proxying commands and changing the execution chain. We observed suspicious terminals like `tty` being spawned by Python, often attributed to simple reverse shells being upgraded to fully interactive `tty` after obtaining *Initial Access* to the host, highlighting the eagerness to do so after access is achieved.

These observables often show Python as a parent process, followed by common processes such as `bash`, `dash`, `ash`, `zsh`, and others. To accomplish this, adversaries can simply invoke the Python interpreter on the Linux endpoint, import the `pty` library and spawn an interactive shell from any shell program.

macOS

Execution ranked #1 with nearly 32.66% of all tactics for macOS.

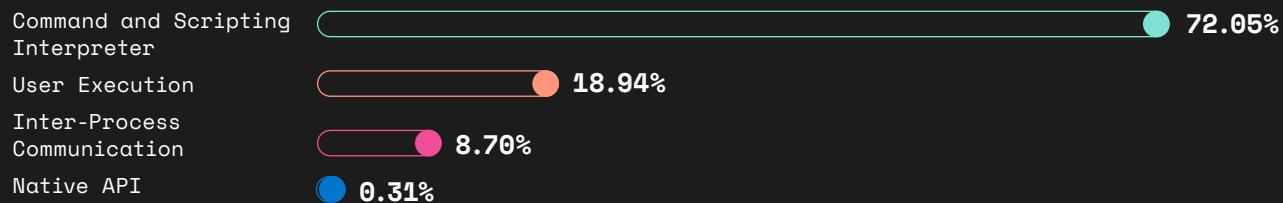


Figure 11: Execution by technique in macOS endpoints

More specifically, *Command and Scripting Interpreter* accounted for 72.05%, followed by user execution at 18.94%. Combined, these account for nearly 91% of all macOS *Execution*-related alerts.

rule_name	Percentage
Suspicious Child Process Execution via Interactive Shell	14.22%
Suspicious Automator Application Execution	14.22%
Executable File Extracted to Temporary Directory	7.33%
Suspicious Nohup Execution	5.60%
Initial Access via OSA Shell Script Piped to Python Interpreter	5.60%
Curl Download and Execution of JavaScript Payload	5.17%
Nohup Execution followed by Outbound Network Connection	4.74%
User Discovery Command Execution from Volume Mount	3.88%
PowerShell Outbound Network Connection	3.88%
PowerShell Encoded Command	3.45%
Potential Reverse Shell Activity via Terminal	3.45%

Table 8: Command and Scripting Interpreter by rule in macOS endpoints

Diving deeper, we observed that *Suspicious Child Process Execution via an Interactive Shell* were commonly observed, often where `bash`, `zsh`, and `sh` were commonly used as a shell to then spawn osascript processes. Often, osascript is abused by adversaries to run malicious AppleScript code. Additionally, we noticed a considerable amount of *Suspicious Automator Application Execution* via XNU Inter-process Communication (XPC) with 14.22% of

all *Command and Scripting Interpreter* alerts. The “Application Stub” binary on macOS is associated with Automator, a tool that allows users to create automation scripts without needing to write code. Adversaries commonly target this as it is a native automation framework on macOS and thus allows not only for adversaries to execute malicious code but also remain stealthy.

Persistence

Maintaining access to victim environments has always been a top priority for threats, and those environments provide numerous opportunities via OS-specific features, misconfigurations, and malware capabilities. In this section, we’ll describe the most common *Persistence* mechanisms we observed.

Windows

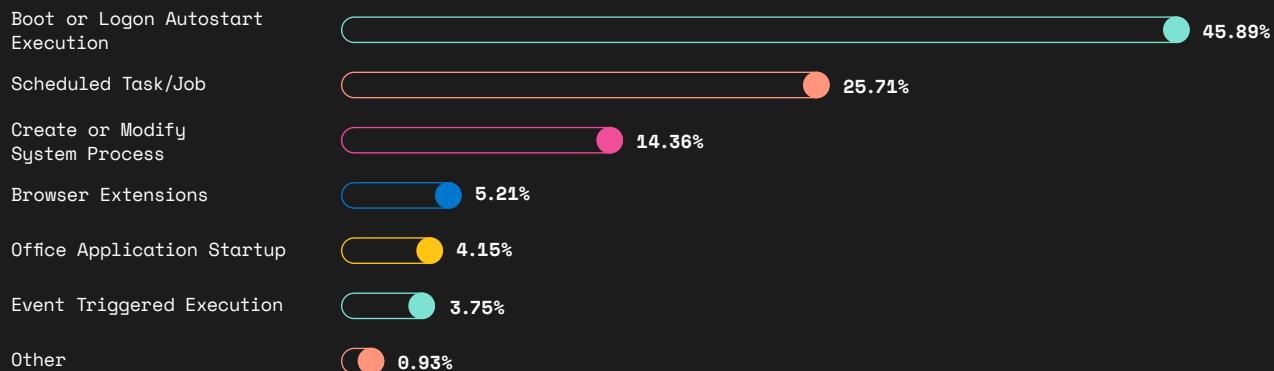


Figure 12: Persistence by technique in Windows endpoints

As *Persistence* remains a top priority, we continue to see methodologies for implementation focused heavily on *Boot or Logon Autostart Execution* (with nearly 46% of all *Persistence* techniques), followed

by *Scheduled Tasks* at 26%, and then *Modification of System Processes* (14%). Note that *Persistence*, in these instances, is heavily dependent on Windows distributions.

This year we also saw a shift in the popularity of some *Persistence* techniques, with *Scheduled Task/Job* and *Boot Logon Autostart Execution* swapping places. *Boot Logon Autostart Execution* saw a 29% increase in usage, highlighting

its growing preference among adversaries. When reviewing the specific detections behind *Persistence* mechanisms involving *Boot* or *Logon Autostart Execution*, we saw a lot of variety without any single dominant behavior pattern.

rule_name	Percentage
Startup Persistence via Windows Script Interpreter	15.69%
Unusual File Written or Modified in Startup Folder	12.60%
Suspicious String Value Written to Registry Run Key	12.07%
Startup Persistence from a Browser or Compression Utility Descendant	8.83%
Registry Run Key Modified by Unusual Process	6.18%
Suspicious Shortcut Modification	6.03%
Suspicious Launch Service Property List File Creation	5.89%
Persistence via a Process from a Removable or Mounted ISO Device	5.46%
Uncommon Persistence via Registry Modification	4.27%

Table 9: Boot or Logon Autostart Execution by rule for Windows endpoints

The *Startup Persistence via Windows Script Interpreter* rule accounts for the highest percentage at 15.69%, indicating a prevalent use of scripting interpreters like PowerShell and `mshta.exe` by adversaries to maintain *Persistence*. Additionally, the abuse of `reg.exe` was commonly noted in these instances. Typically, complex script file types such as VBS, LNK, PS, HTA, and occasionally EXE files are identified during these detections. These scripts become versatile tools for adversaries to achieve *Persistence* with minimal effort. A significant portion of these detections involved directly modifying the `HKEY_CURRENT_USER` (HKCU) and HKLM registry hives to establish *Persistence* with scripts. Adversaries leverage

these registry locations to ensure malicious payloads execute during system boot or user logon. The analysis underscores the importance of monitoring and securing startup locations and registry keys to prevent unauthorized *Persistence* mechanisms.

Regarding the rule *Unusual File Written or Modified in Startup Folder*, Elastic Security Labs observed various types of files, including DLLs, LNK, EXE, TXT, HTA, and PowerShell scripts being written to the startup folder by LOLBins. This activity highlights how adversaries leverage legitimate system tools to establish *Persistence*.

rule_name	Percentage
Scheduled Task Creation by an Unusual Process	25.69%
Suspicious Windows Schedule Child Process	21.51%
Scheduled Task Creation via Unsigned Parent	20.14%
Suspicious Scheduled Task Creation	14.63%
Scheduled Task from a Removable or Mounted ISO Device	5.32%
Scheduled Task from a Browser or Compression Utility Descendant	3.87%
Scheduled Task by a Low Reputation Process	3.44%
Scheduled Task Creation from Suspicious Parent	3.25%

Table 10: Scheduled Task/Job by rule for Windows endpoints

When analyzing *Persistence* mechanisms involving scheduled tasks and jobs by rule, we observe a significant distribution across several rules. The rules *Scheduled Task Creation by an Unusual Process*, *Suspicious Scheduled Child Process*, and *Scheduled Task Creation via Unsigned Parent* account for 26%, 22%, and 20% of detections, respectively. This distribution indicates that the detection logic effectively focuses on the relationships between parent and child processes as well as the signatures of the involved processes.

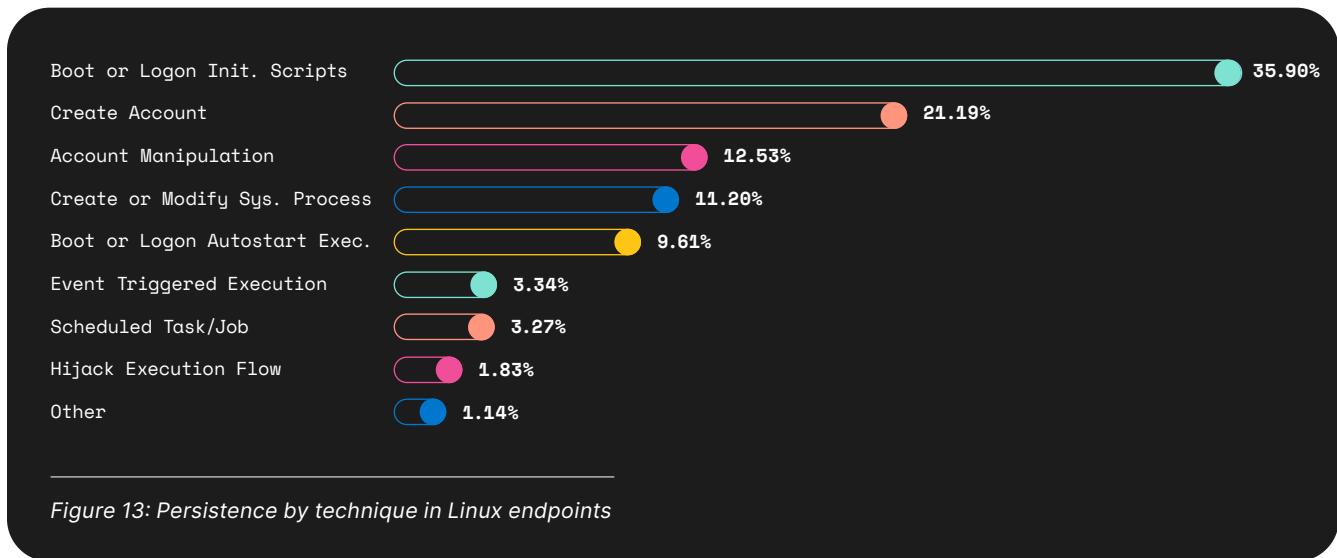
Adversaries often leverage WScript, MSHTA, RunDLL32, WMIC, and CScript to spawn [svchost](#). These parent processes can be used to create or modify scheduled tasks that point to executables located in commonly abused directories like [%AppData%\Local\Temp](#), [%Users%\Public](#), and [%Windows%\Microsoft.NET](#). These directories are frequently used by adversaries because they are often overlooked and provide a convenient location to store malicious payloads.

By creating scheduled tasks that point to these executables, adversaries ensure that their malicious code is executed at specified intervals or system events, thereby maintaining *Persistence* on the compromised system. This method is particularly effective because it leverages legitimate Windows functionality to execute malicious code, making it harder to detect and remediate.

The high fidelity of detection logic in identifying unusual parent-child process relationships and unsigned processes highlights the importance of monitoring scheduled tasks and their associated processes. Similar behavior has been highlighted by Elastic Security Labs in our analysis of [LATRODECTUS](#). This analysis underscores the necessity for robust monitoring and defense strategies to identify and neutralize such threats effectively.

Linux

While *Persistence* is a common goal among all adversary campaigns and playbooks, it was the #1 Linux tactic with 24.26% of alerts.



We observed that *Boot or Logon Initialization Scripts*, *Account Creation*, and *Account Manipulation* were often the most observed techniques, collectively accounting for more than 36% of all *Persistence* mechanisms. After this, there is a nearly even distribution for modifying system processes and event-triggered execution.

rule_name	Percentage
Suspicious File Creation in /etc for Persistence	54.93%
Suspicious Process Spawned from MOTD Detected	21.57%
Chkconfig Service Add	15.47%
Potential Persistence Through Run Control Detected	2.77%

Table 11: Boot or Logon Initialization Scripts by rule in Linux endpoints

The `/etc/` directory is a common target for Linux *Persistence* mechanisms because it houses critical system configuration files and scripts, such as:

- `/etc/rc.local` for startup commands
- `/etc/update-motd.d/` for message of the day scripts
- `/etc/sudoers` for privileged user configurations

- `/etc/profile` for shell environment settings
- `/etc/systemd` for service management
- `/etc/cron` for scheduled tasks and the dynamic linker configuration

All of these can be manipulated to maintain unauthorized access or execute malicious code at system startup or during specific events.

rule_name	SUM
Linux Group Creation	46.70%
Linux User Account Creation	46.56%
Linux User Added to Privileged Group	6.29%
Other	0.45%

Table 12: Account Creation in Linux endpoints

As shown by table 13, Linux user and group creation were often achieved by adversaries with a similar distribution, which is expected given that user and group creations happen in parallel.

rule_name	Percentage
New Systemd Service Created by Previously Unknown Process	43.15%
Modification of Standard Authentication Module or Configuration	34.50%
Potential Execution via XZBackdoor	12.17%
Modification of OpenSSH Binaries	8.69%
Systemd Service Created	1.37%
Other	0.11%

Table 13: Create or Modify System Process by rule in Linux endpoints

Creating or Modifying System Processes also accounted for nearly 11.20% of all *Persistence* on Linux endpoints. While this is not a large distribution, the specific activity observed is rather important upon analysis. Specifically, 43.15% of all suspicious process creations flagged as related to the `systemd` service were created by an unknown process. A large portion of these

were related to the creation or renaming of new `systemd` files in common service locations for both root and regular users. This is often done by adversaries because they can leverage `systemd` service files to achieve *Persistence* by creating or modifying services to execute malicious commands or payloads during system startup or at predefined intervals via a `systemd` timer.

rule_name	Percentage
Tainted Kernel Module Load	29.02%
Tainted Out-Of-Tree Kernel Module Load	27.76%
Kernel Module Load via insmod	20.58%
Persistence via KDE AutoStart Script or Desktop File Modification	13.96%

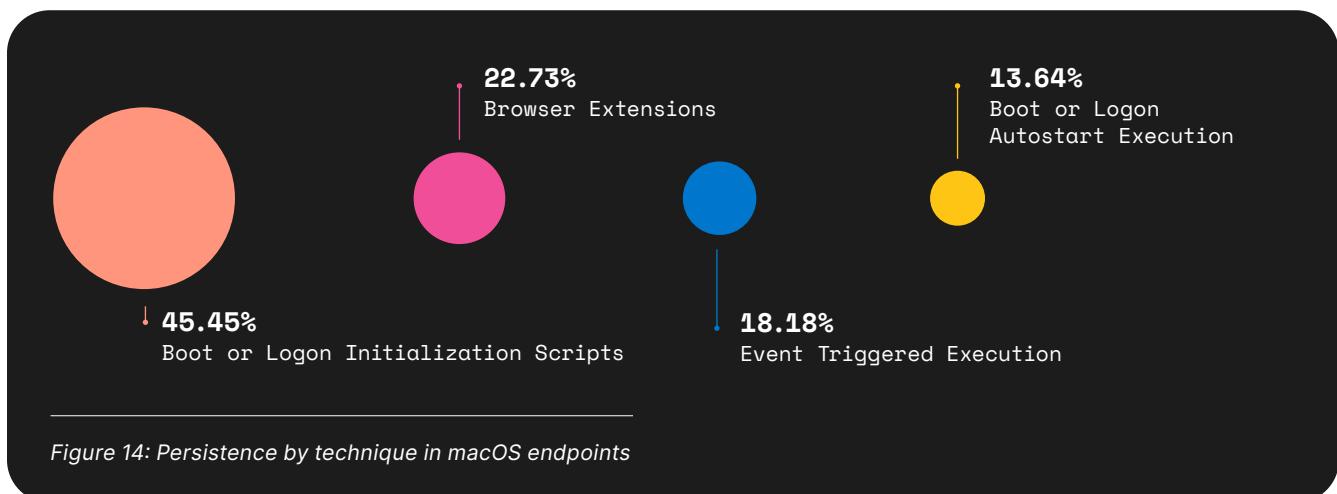
Table 14: Boot or Logon Autostart by rule in Linux endpoints

The term “tainted kernel module” is defined as a Linux kernel that is in an unsupported state because functionality cannot be guaranteed — often because the kernel contains unsigned, non-standard (out-of-tree), or other types of modules. These are similar to unsigned DLLs in Windows, which are commonly loaded through *System Binary Proxy Execution* via native processes such as RunDLL32.

Combined, kernel modules either tainted or suspiciously loaded accounted for nearly 76% of all observed boot or logon autostart alerts, which is significant since rootkits often leverage these kernel modules for *Persistence* and *Defense Evasion*. Adversaries often accomplish this with the help of loadable kernel module (LKM) rootkits, where modules are added as an extension for the kernel but do not contain valid signatures or belong to the standard kernel build tree.

macOS

Launch daemons and agents were the most popular *Persistence* mechanisms on macOS. As shown below, *Logon Initialization Scripts* accounted for 45.45% of all *Persistence* on macOS, followed by browser extensions at 22.73%.



Nearly 60% of all Persistence-related alerts tie directly to suspicious property list (plist) files, which are searched for by `launchd`. These are then launched on-demand from these plist files.

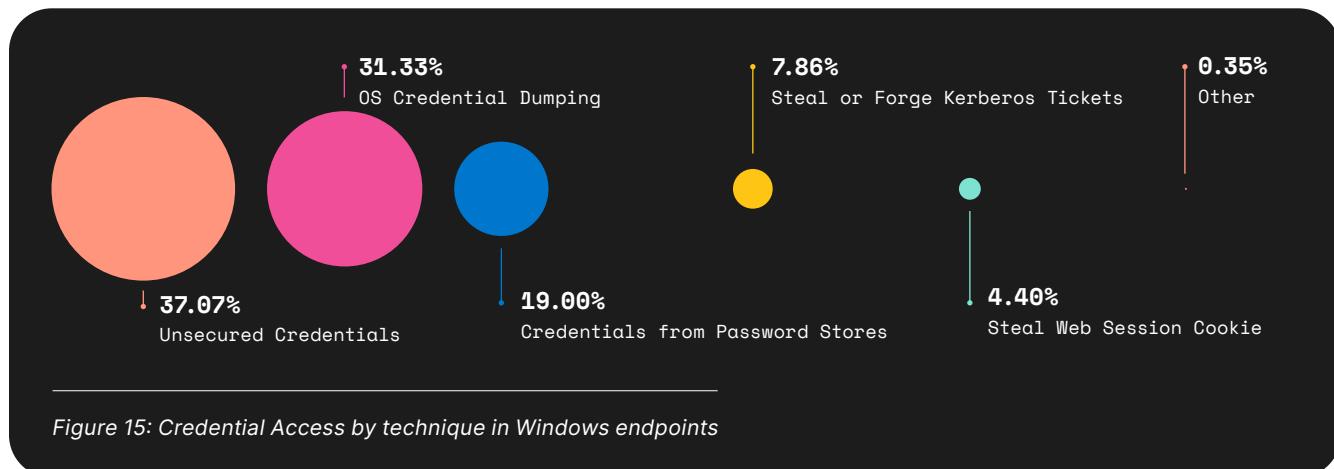
rule_name	Percentage
Suspicious StartupItem plist Creation or Modification	45.45%
Manual Loading of a Suspicious Chromium Extension	22.73%
Initial Access Staging via Installer Package	9.09%
Persistence via a Masqueraded plist Filename	6.06%
Persistence via a Hidden plist Filename	4.55%
Untrusted or Unsigned Binary Executed via Launch Service	3.03%
Suspicious File Creation via Pkg Install Script	3.03%
Suspicious Property List File Creation or Modification	1.52%
Suspicious Apple Mail Rule plist Creation or Modification	1.52%
Potential Persistence via Emond	1.52%

Table 15: plist alerts by rule in macOS endpoints

Credential Access

Elastic Security Labs has observed a significant trend in information stealers leading to *Credential Access*, with stolen credentials often sold or used by other adversaries in subsequent campaigns.

Windows



This activity has contributed to many recent public breaches, where attackers used stolen credentials to log in to valid accounts — both local and cloud-based — to continue their intrusion. Although

Credential Access accounts for only about 9% of behavioral detections overall, understanding the techniques and methodologies employed by adversaries is crucial.

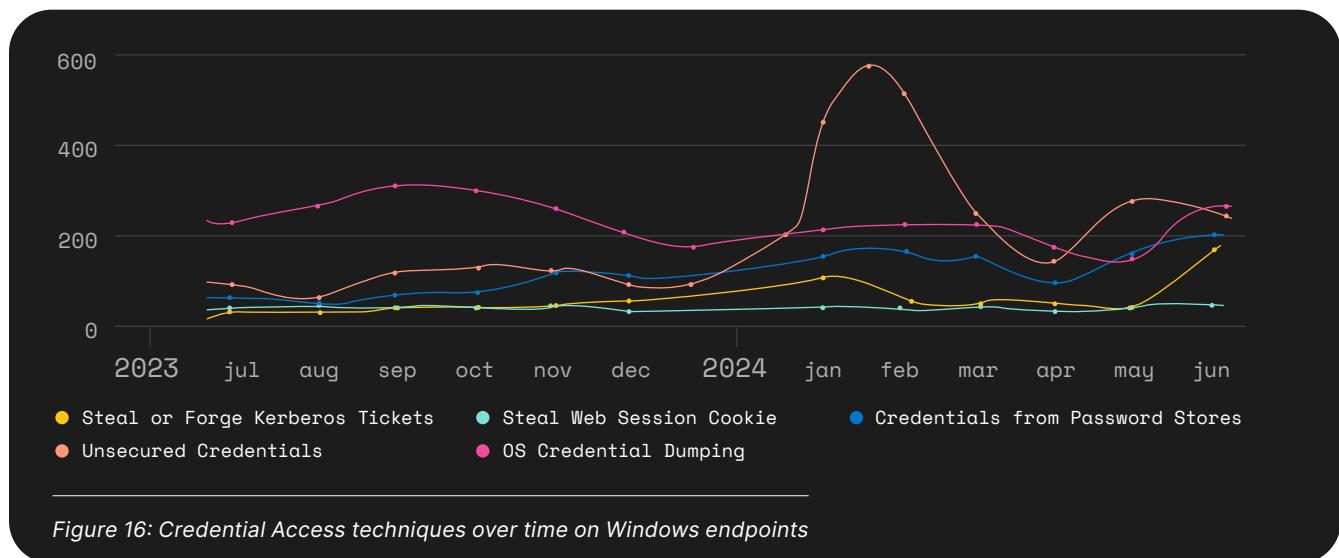


Figure 16: Credential Access techniques over time on Windows endpoints

Unsecured credentials represent nearly 37% of all *Credential Access* techniques, marking a notable increase of nearly 31% compared to last year. This indicates a growing preference among adversaries for exploiting unsecured credentials. *OS Credential Dumping*, typically associated with extracting credentials from protected memory

regions of specific processes on Windows, has decreased to 35% of *Credential Access* but remains significant. Additionally, targeting *Credentials from Password Stores* has seen a 4% increase compared to last year, highlighting its continued popularity among adversaries.

Readers should be aware that vast networks of access brokers – persons or organizations who monetize stolen credentials sold to criminal and espionage ecosystems – exist, leading to everything from identity to intellectual property theft. However, it may be incredibly challenging to conclusively tie intrusion activity to stolen or exposed credentials.

As organizations increasingly leverage continuous integration and continuous delivery (CI/CD) pipelines, hybrid cloud environments, and software as a service (SaaS) solutions, the potential to exploit *Unsecured Credentials* only

grows. Adversaries are likely shifting their focus to these abundant and often inadequately secured credentials, emphasizing the need for robust security measures to protect sensitive information across various environments.

rule_name	Percentage
Access to Browser Credentials from Suspicious Memory	49.43%
AutoLogons Access Attempt via Registry	16.21%
Sensitive File Access - SSH Saved Keys	11.63%
Failed Attempts to Access Sensitive Files	8.62%
Failed Access Attempt to Web Browser Files	7.00%
Other	7.10%

Table 16: Unsecured Credentials by rule in Windows endpoints

It's important to note that adversaries target unsecured credentials via tools, manual efforts, and malware. Browser credentials in memory accounted for 49.43% of all unsecured *Credential Access* attempts. Elastic Security's behavioral detection logic is designed to identify attempts to access web browser-stored credentials from processes exhibiting suspicious memory properties. By monitoring for processes that access specific files like login

data, `logins.json`, `cert.db`, `key.db`, and SQLite databases for sign-ons and cookies from unusual memory regions, Elastic Security captures adversary behavior related to unsecured credentials that may evade strictly file-based monitoring mechanisms. Elastic Security Labs noted similar behavior from our research into [Globally Distributed Stealers](#), specifically with families such as REDLINE Stealer.

rule_name	Percentage
Credential Access via Known Utilities	23.15%
LSASS Memory Dump via MiniDumpWriteDump	16.73%
Suspicious Access to LSA Secrets Registry	12.24%
LSASS Access Attempt from Unbacked Memory	10.61%
Suspicious Registry Hive Dump	9.50%
Security Account Manager (SAM) File Access	7.58%
Security Account Manager (SAM) Registry Access	7.41%
Potential Credential Access via Mimikatz	4.55%
LSASS Access Attempt from an Unsigned Executable	3.44%
Other	4.78%

Table 17: OS Credential Dumping by rule in Windows endpoints

Credential Access via Known Utilities accounted for 23% of all OS *Credential Dumping*, mainly for Windows systems. This logic is designed to identify the execution of known Windows utilities frequently abused by adversaries to dump Local Security Authority Subsystem Service (LSASS) memory or the Active Directory database ([NTDS.dit](#)). The rule focuses on detecting utilities like [procdump](#), [esentutl.exe](#), [diskshadow.exe](#), [rundll32.exe](#), and [reg.exe](#), which are often used in credential dumping activities. These tools, when executed with specific arguments, can create memory dumps or export registry hives containing sensitive credential information, making them prime targets for malicious actors seeking to escalate privileges or pivot within a network.

Analysis of the process execution data reveals common patterns and techniques used by adversaries. For instance, [procdump64.exe](#) is frequently observed creating dumps of the [lsass.exe](#) process, which is a typical method for extracting password hashes and other credential material stored in memory. Similarly, [reg.exe](#) commands are used to save critical registry hives such as [HKLM\SAM](#), [HKLM\SYSTEM](#), and [HKLM\SECURITY](#), which can be exploited to retrieve stored credentials. The execution of [rundll32.exe](#) with [comsvcs.dll](#) to create minidumps and the use of [esentutl.exe](#) to copy the [NTDS.dit](#) file further highlights the diverse strategies employed by attackers to gain access to credential stores.

The most common *Credential Access* tool observed in these detections is [powershell.exe](#), which was identified in several instances. PowerShell is often misused by attackers due to its powerful scripting capabilities and deep integration with Windows, making it an ideal tool for accessing and manipulating system resources, including the Credential Vault. Other processes like [mscorsvw.exe](#), [notepad.exe](#), and [rundll32.exe](#) were also identified loading [vaultcli.dll](#) – suggesting that adversaries are leveraging these processes to perform credential theft. By using these processes, attackers can blend their malicious activities with legitimate system operations, reducing the likelihood of detection.

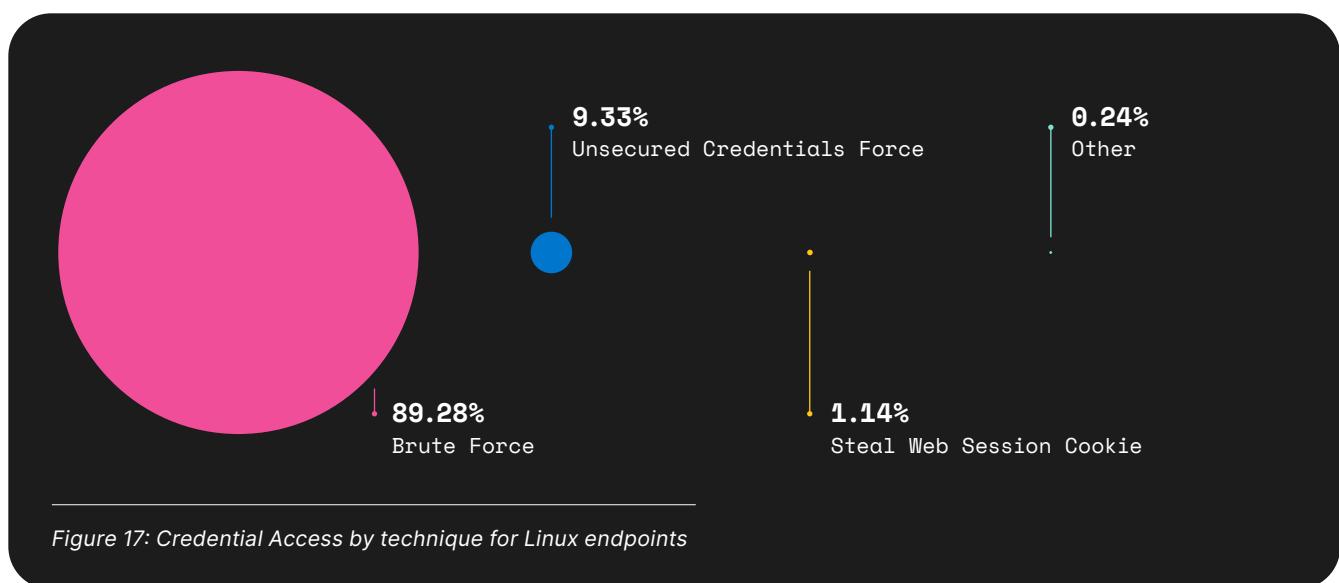
The targeted processes typically access [vaultcli.dll](#) from unusual locations or with unexpected arguments, which is indicative of suspicious behavior. For instance, the use of [rundll32.exe](#) with specific arguments targeting user AppData directories, or [powershell.exe](#) with encoded commands, suggests attempts to hide malicious intentions and evade security controls. Additionally, the inclusion of other processes like [mshta.exe](#), [cvtres.exe](#), and various .NET executables indicates a broader strategy to use a

diverse set of tools to achieve their goals. Adversaries exploit these methods to extract credentials from the Windows Credential Manager, allowing them to escalate privileges, move laterally across the network, and gain unauthorized access to sensitive systems and data. By monitoring for these unusual process activities and the loading of the [vaultcli.dll](#), security solutions can identify and thwart these attempts, protecting the integrity of user credentials and maintaining the security of the overall environment.

Linux

When analyzing *Credential Access* for Linux, it's important to remember that the storage of credentials is different across platforms. Often, organizations with users behind endpoints rely on identity access management (IAM) solutions or identity providers (IdPs) from third-party services

such as Okta and Entra ID to handle authentication and password storage and retrieval. However, on Linux, authentication with users and groups expands to services like SSH where authentication is based on private and public keys.



89.28% of all *Credential Access* signals related to *Brute Force*, with only a small amount related to *Unsecured Credentials* at 9.33% respectively.

rule_name	Percentage
Potential Internal Linux SSH Brute Force Detected	50.03%
Potential Linux SSH Brute Force Detected	20.90%
Potential External Linux SSH Brute Force Detected	18.24%
Potential Successful SSH Brute Force Attack	4.44%
Potential SSH Password Guessing	4.10%
Potential Linux Local Account Brute Force Detected	2.29%

Table 18: Brute Force by rule for Linux endpoints

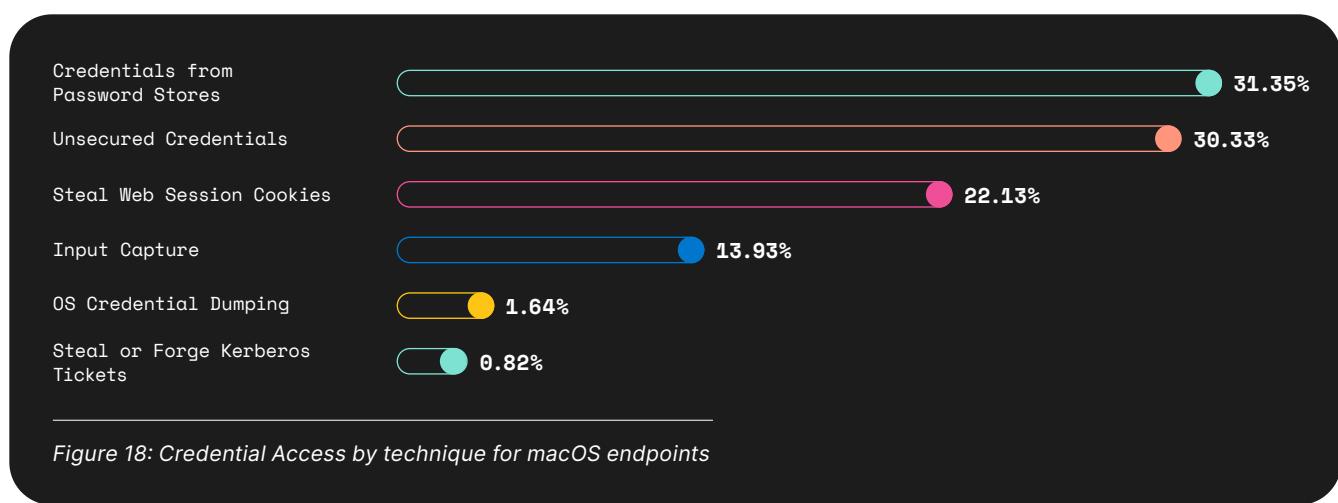
Brute Force, while not the most complex attack, often has different sub-techniques such as password spraying and credential stuffing that are more prevalent than ever for adversaries at the time of this report. In nearly 97% of all brute forcing alerts, the main culprit was consecutive

login failures targeting user accounts from the same external source address in short time intervals. This also suggests these Linux endpoints were public-facing as well, where logins captured were typically SSH-based.

macOS

Researchers identified that *Credentials from Password Stores*, *Unsecured Credentials*, *Steal Web Session Cookies*, and *Input Capture* were the most common techniques at 31.35%, 30.33%, 22.13%, and 13.93% respectively. Compared to other operating systems, input capture is an

anomaly and unique to macOS due to the abuse of osascript by adversaries. It should also be noted that regardless of technique or rule, the prevalent targets by adversaries are often keychains, crypto wallets, web browsers, and capturing input from users.



More than ever, information stealers have become a popular tool in an adversary's arsenal when targeting potential macOS endpoints, specifically for *Credential Access* attempts against SaaS platforms and other cloud-hosted applications.

Cloud Security



Few organizations operate entirely outside of cloud-hosted environments, expanding the enterprise attack surface from self-managed resources to “someone else’s computer” in the cloud. Elastic customers voluntarily provided the alert telemetry used in this section, helping researchers discover new threats and engineering functions to improve security capabilities. These alerts are generated based on out-of-the-box (OOTB) detection rules, which utilize data from Elastic integrations specific to each cloud service provider (CSP).

It’s important to note that the nature of detecting potentially malicious activity within CSPs — especially when involving valid accounts and

legitimate activities — results in these alerts often being of lower fidelity compared to those from EDR systems. We treat these alerts more like potential signals of threat activity rather than confirmed evidence of threats, and highlight this distinction for readers who may be inclined to draw more definitive conclusions. This year, Elastic Security Labs combined Microsoft 365 with Microsoft Azure data and Google Workspace with Google Cloud data rather than separate them into a SaaS category. Since entities and services are closely related — often using the same APIs and resources — we believe this gives a better holistic view of each provider.

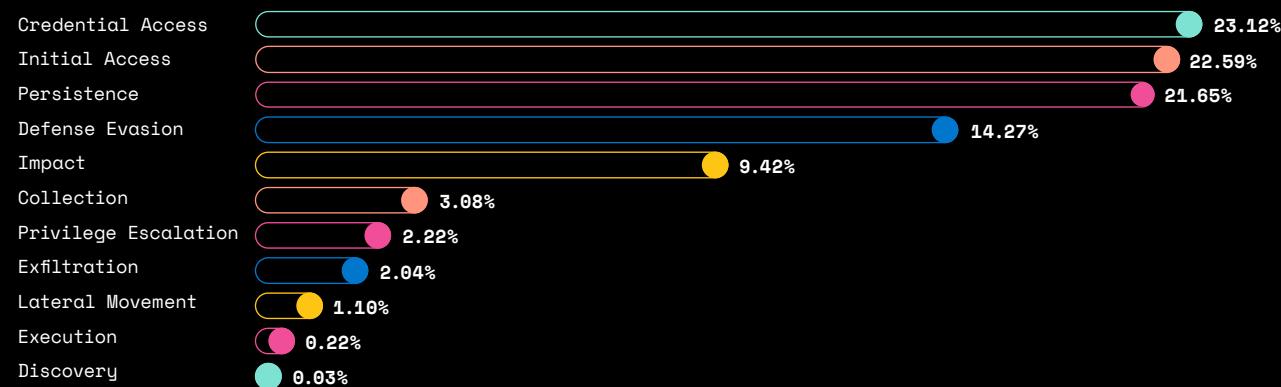


Figure 19: MITRE ATT&CK tactics observed in Cloud environments

Overall, *Credential Access* represented a little more than 23% of all activity, followed by *Initial Access*, *Impact*, and *Defense Evasion* at 22%, 21%, and 14%, respectively. At the heart of cloud security is IAM, which is the technology that

shapes *Credential Access* attempts. We expect that techniques in this category will account for the greatest proportion of behaviors we observe targeting cloud platforms.

Distribution by cloud service provider

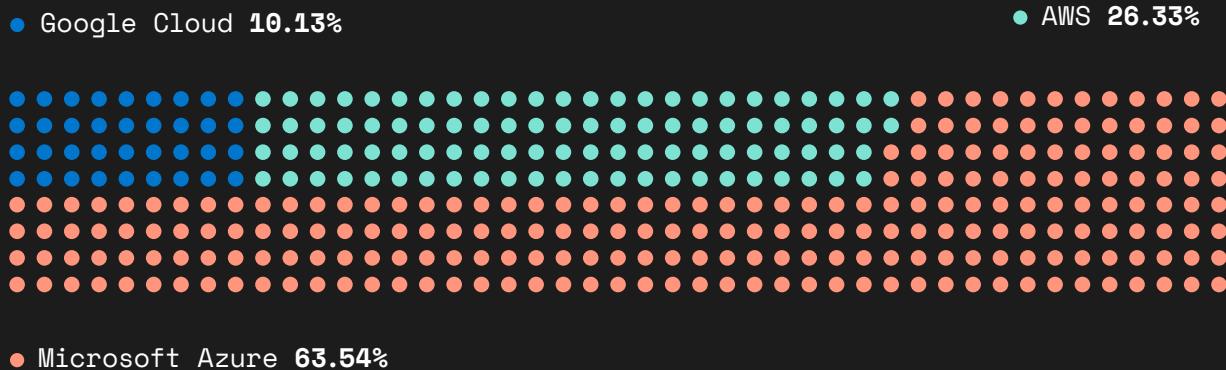


Figure 20: Signals by service providers

In our analysis of signal distribution by CSP, we found that Microsoft Azure was the most common environment for anomalous signals, accounting for 64% of the total. This marks a shift from previous years, where AWS has had the greatest number of signals. Combining Microsoft 365 data, which includes *Credential Access* and *Phishing* attempts, with Microsoft Azure is the reason for this change — readers are discouraged from concluding that this reflects any form of targeting preference or threat trend.

Notably, Microsoft Azure includes Entra ID, the Microsoft default IAM solution. Despite AWS holding the [largest market share](#) among CSPs,

Microsoft data sources provided the largest number of events. With the popularity of hybrid deployments, the adoption of Entra ID over third-party identity providers is becoming increasingly common. Google Cloud, along with Google Workspace, only accounted for roughly 10% of all signals from CSPs.

In the following subsections, we'll provide details on a per-CSP basis for Microsoft, Amazon, and Google platforms. The final section will include a cloud posture overview based on [Center for Internet Security \(CIS\)](#) benchmarks.

Microsoft Azure

Microsoft Azure has evolved beyond Windows infrastructure-as-a-service (IaaS) to include robust platform-as-a-service (PaaS) capabilities, web hosting, and a variety of identity management features. Entra ID is Microsoft's integrated identity

management capability. Microsoft Azure accounts often are high-value for adversaries who may want to move laterally into physical endpoints or employee mailboxes in Office 365 to either further their intrusions or complete actions-on-objectives.

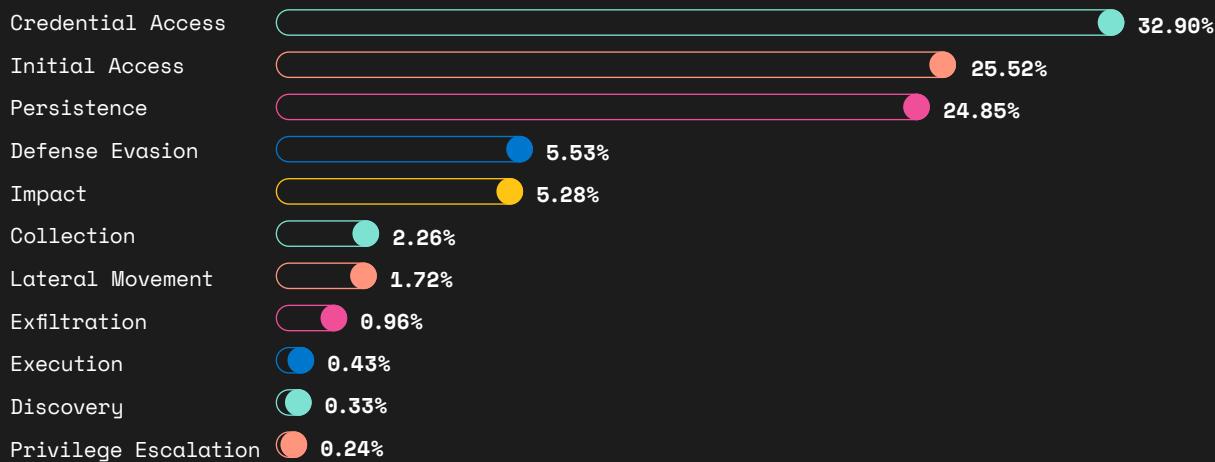


Figure 21: Microsoft Azure Signals by tactic

Credential Access

Credential Access accounts for about 32.90% of Microsoft Azure signals — an increase of more than 26% over last year — and is often achieved with infostealer malware and fraudulent portals that masquerade as legitimate. Large economies exist to monetize stolen credentials to threats of all kinds, enabling low-maturity bad actors to succeed against otherwise hardened environments. Access brokers — individuals and groups who sell stolen credentials and previously-infected systems — are active innovators who commonly combine social engineering, unpatched

vulnerabilities, and permissive environments to succeed.

We attributed 98% of *Credential Access* attempts for Microsoft Azure to *Brute Force* techniques, as shown in table 20. These are often a combination of techniques like password guessing, hashed password cracking, password spraying, and credential stuffing. Enterprises should anticipate the risks of brute forcing and take them under consideration when deploying any public-facing infrastructure.

Brute Force attempts previously represented about 86% of *Credential Access* signals within Microsoft Azure. This number has increased by 12%.

technique_name	Percentage
Brute Force	97.74%
Unsecured Credentials	2.05%
Steal Application Access Token	0.17%
Network Sniffing	0.04%

Table 19: Credential Access by technique in Microsoft Azure

Initial Access

Initial Access, 25.52% of Microsoft Azure signals, describes techniques used to achieve a foothold. We see two primary techniques employed: abusing *Valid Account* credentials, and *Phishing* users. *Valid Accounts* represented almost 57% of all *Initial Access* methods in Microsoft Azure, a strong sign that the access broker ecosystem is thriving right now. *Phishing* accounted for 43% of signals, a significant increase from last year due largely to the way data has been combined for this year's report.

A detailed analysis of our detection rules revealed that 62% of the *Phishing* incidents observed involved Microsoft 365 emails flagged as malware or phishing by users. The remaining 38% were linked to consent grant attacks via Microsoft Azure-registered applications. These attacks exploit OAuth 2.0 permissions by tricking users into granting consent to malicious applications, thereby gaining unauthorized access to their data.

Persistence

The techniques in this category provide adversaries with persistent or on-demand access to victim environments, systems, or data. In our analysis of *Persistence*, which accounted for 25% of all Microsoft Azure-related anomalous signals, we observed nearly all signals were directly linked to *Account Manipulation* in one way or another. Unlike endpoints, where *Persistence* is achieved via applications, settings, the file system, the registry, or misconfiguration, *Persistence* in the cloud environment (excluding compute instances) is frequently tied to an IAM account.

technique_name	Percentage
Account Manipulation	98.02%
Valid Accounts	1.03%
Create Cloud Instance	0.83%
Create or Modify Cloud Account	0.07%
Modify Authentication Process	0.06%

Table 20: Persistence by technique in Microsoft Azure

Once a valid account is compromised and credentials remain unrotated, adversaries can log in and establish *Persistence* by modifying existing principal policies, altering authentication workflows, and exploiting unnecessary permissions and privileges.

Our analysis revealed that Microsoft 365 Exchange mailbox permissions are common targets. Because both registered applications and compromised valid accounts frequently add permissions such as Full Access, Send As, or Send on Behalf, adversaries hide their signals in the noise. From a trusted Microsoft Azure tenant, threats can more successfully phish other victims.

Amazon Web Services

According to [Statista](#), AWS stands as the most widely adopted cloud service provider this year by market share, though only about 26% of cloud signals came from AWS. This can be explained by our decision to combine Microsoft Azure and Microsoft 365 events this year.

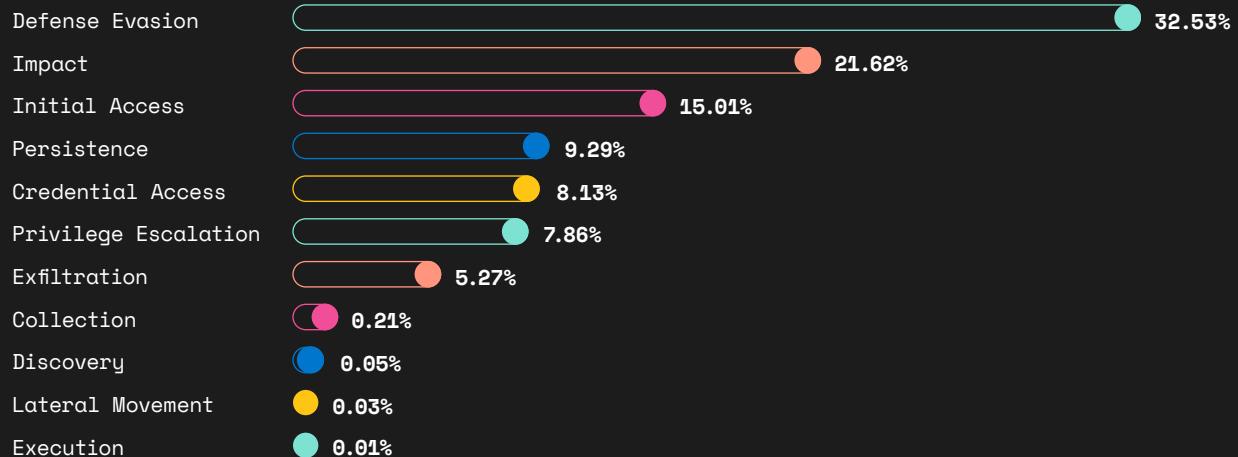


Figure 22: AWS Signals by tactic

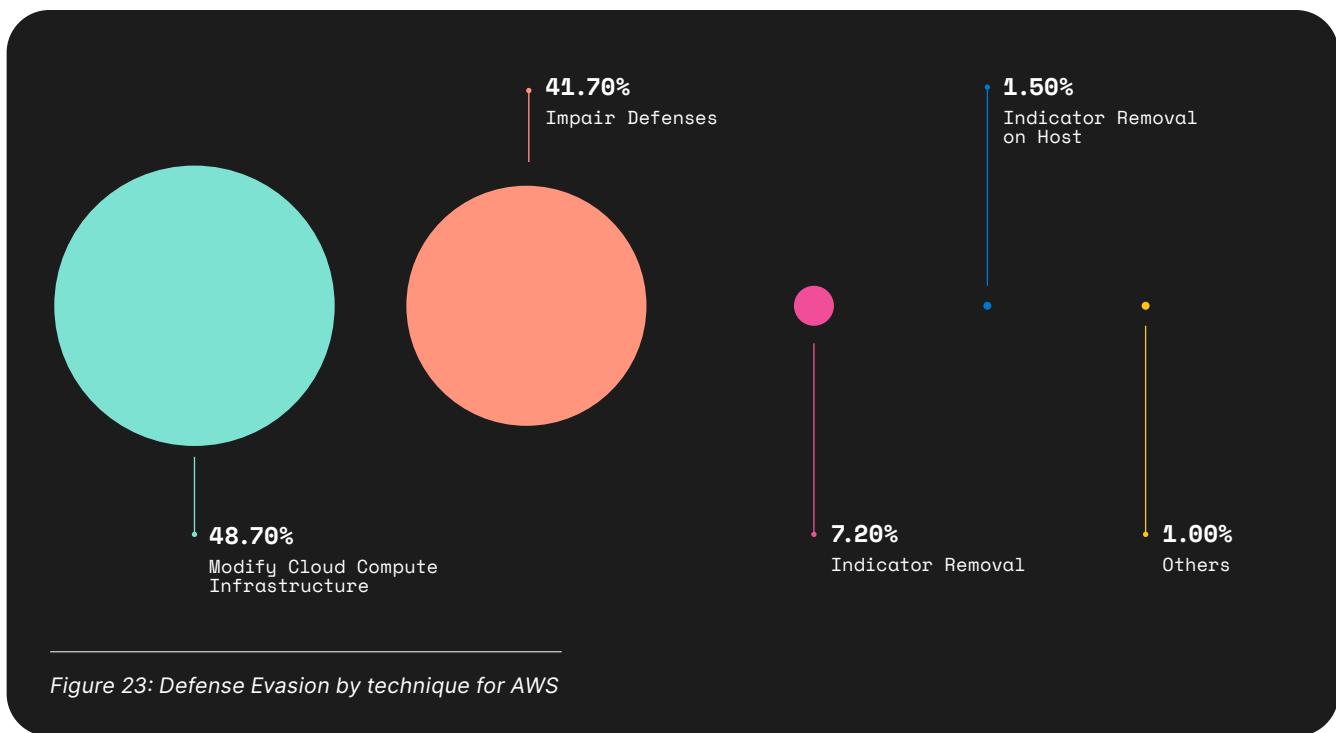
During our analysis, 32.53% of all anomalous signals in AWS were mapped directly to potential *Defense Evasion* techniques. Last year this value was 38% — a slight decrease. *Impact* techniques demonstrated one of the more significant increases, growing in volume by more than 20%.

Initial Access also increased from less than one tenth of a percent last year to more than 15%, and *Persistence* methods were observed significantly from less than 1% last year to more than 9% in 2024.

Defense Evasion

Defense Evasion techniques were the most significant observation in AWS data, and we assess that they will continue to be important for enterprises to maintain visibility over. The most prevalent events involved modification of cloud compute infrastructure, accounting for 48.70% of signals. Last year, signals related to this event were observed less than 1% of the time. Efforts to *Impair*

Defenses, which were the majority of AWS signals last year at 37%, modestly increased to 41.70%. Indicator removal rose from 1% last year to more than 7%. The most frequently seen techniques for attempting to *Modify Cloud Compute Infrastructure* identified potential changes in security group configurations and restoring RDS snapshots, 83.47% and 13.90% respectively.



The high distribution of signals related to modifying cloud infrastructure can be attributed to the extensive reliance on AWS services for deploying and managing infrastructure, particularly for applications, CI/CD pipelines, and other critical functions. In an Agile development environment, frequent and dynamic adjustments

to infrastructure are common. This constant flux makes it challenging for security analysts to distinguish between benign and malicious activities. As a result, attackers benefit from this complexity to mask their actions, making modification of cloud infrastructure a prominent tactic for evading detection.

Impair Defenses

It should be assumed that adversaries are aware of security visibility and capabilities, and strategize to ensure these don't interfere with their goals. The techniques and procedures in this category were developed for that express purpose.

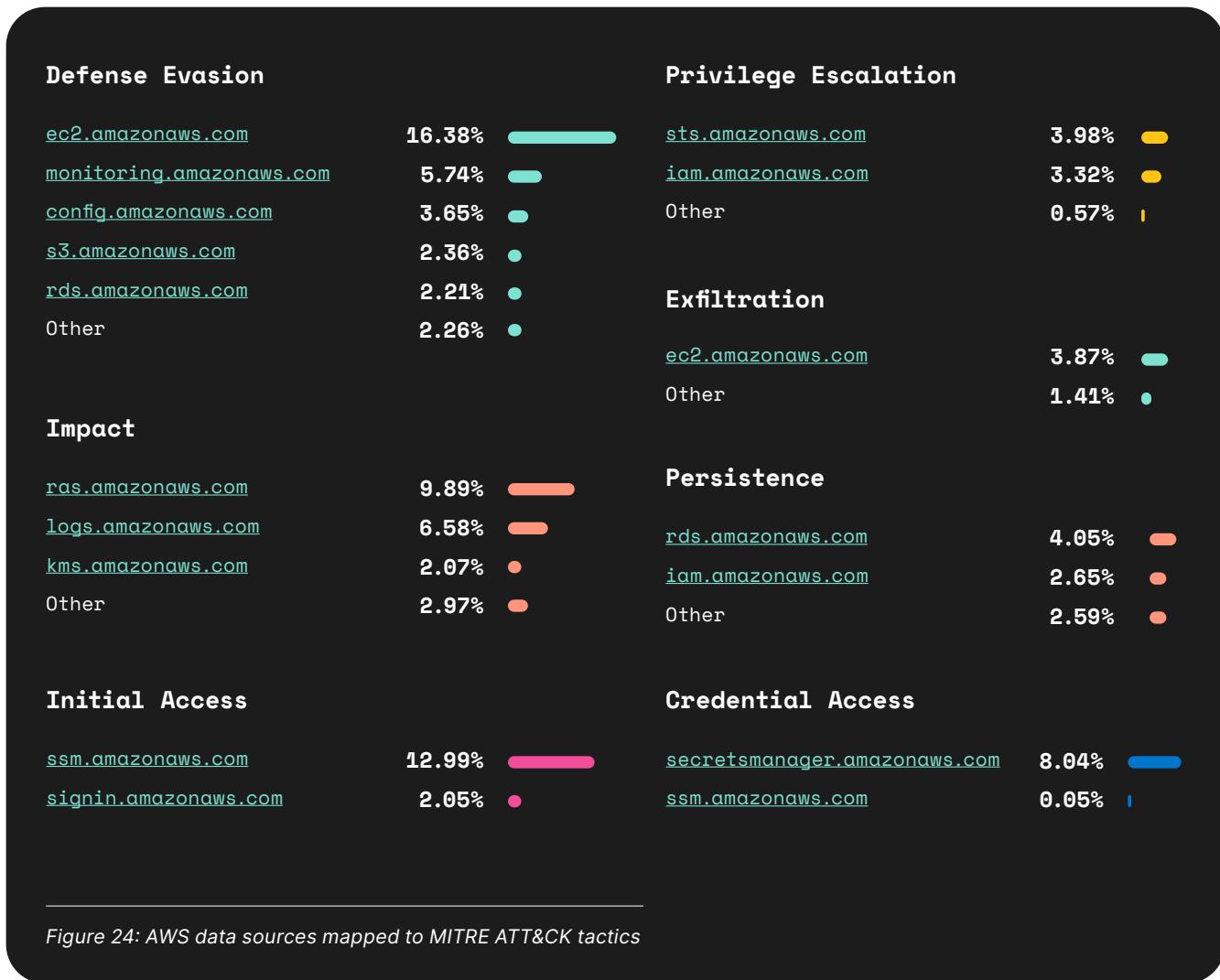
kibana_alert_rule_name	Percentage
AWS CloudWatch Alarm Deletion	42.16%
AWS Config Resource Deletion	25.72%
AWS EC2 Network Access Control List Deletion	10.79%
AWS VPC Flow Logs Deletion	5.54%
AWS WAF Access Control List Deletion	4.76%
AWS CloudTrail Log Deleted	3.80%
AWS WAF Rule or Rule Group Deletion	3.18%
Other	4.04%

Table 21: Impair Defenses by alert name in AWS

Adversaries can employ any one of several approaches to disable or otherwise tamper with security instrumentation:

- By deleting AWS CloudWatch alarms (42.16%) and AWS Config resources (25.72%), attackers can disable critical monitoring and alerting systems.
- Deleting network access control lists (ACLs) (10.79%) and Virtual Private Cloud (VPC) Flow Logs (5.54%) further obscures their activities by removing visibility into network traffic patterns and access attempts.
- Tampering with CloudTrail logs directly, either by deletion (3.80%) or suspension (1.13%), can interfere with detection and response functions.

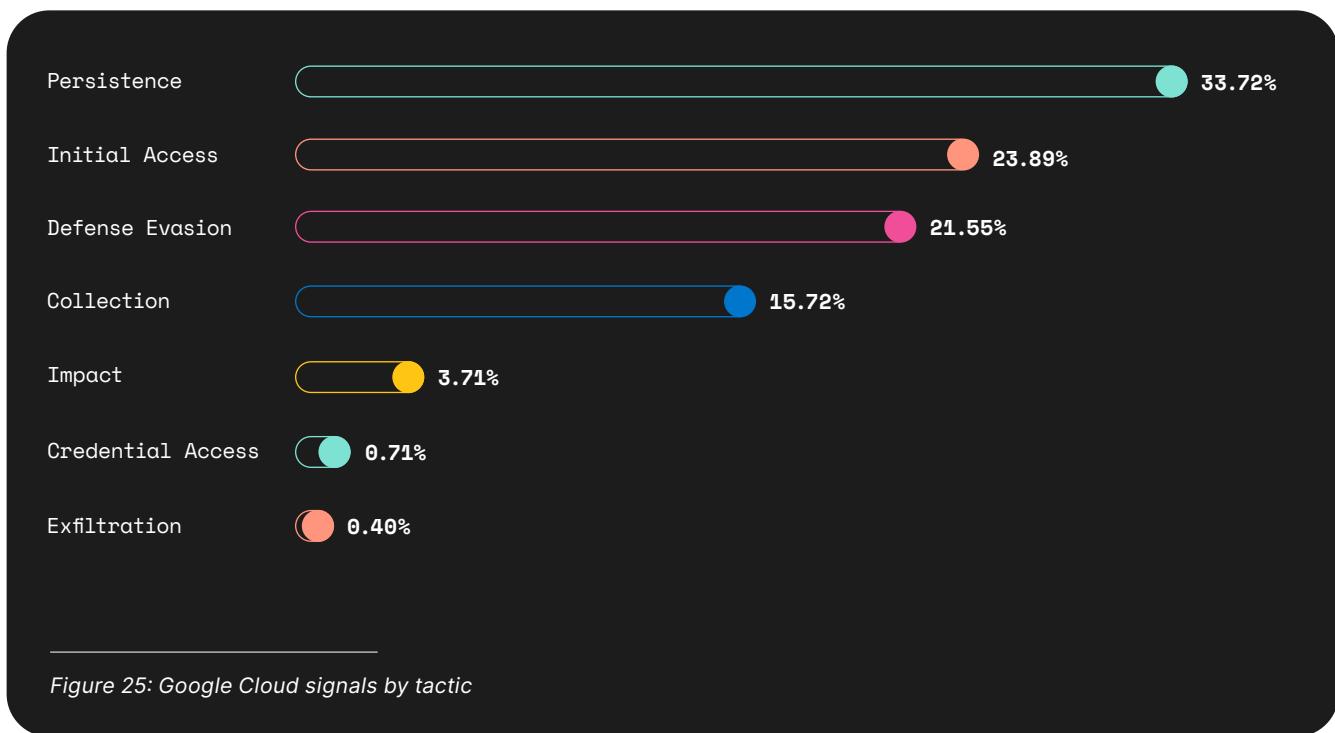
The techniques in this category are designed to create or expand blind spots, and monitoring for evidence of them should be a priority. The most common method of indicator removal involved deleting Simple Storage Service (S3) bucket configurations at 98.48% that could alert administrators to unauthorized changes or data theft. By removing or altering these indicators, attackers can prevent detection. AWS data sources contain a wealth of information, as evidenced by the following depiction that shows the frequency with which each appeared in telemetry data, aligned to MITRE ATT&CK tactic categories. *Defense Evasion*, as the most commonly observed phenomena, depends heavily on Elastic Compute Cloud (EC2) eventing.



Google Cloud

Google Cloud is another widely adopted cloud service provider, popular for reasons similar to Microsoft Azure and AWS, including its robust service offerings, global infrastructure, and advanced analytics capabilities. Organizations frequently choose Google Cloud, especially when they already leverage Google Workspace, due to the seamless integration of Google's

IAM, services, and tools. This close integration also makes Google Cloud a prime target for adversaries. The extensive use of Google Workspace within organizations means that compromising a single set of credentials can potentially grant attackers access to a broad range of services and data.



Persistence increased significantly from 1.5% of signals last year to 33.72%, *Initial Access* methods were observed almost 24% of the time compared to a statistically insignificant value last year, and *Defense Evasion* events decreased from about 85% to 21.55% this year. *Collection*, which was the second most common tactic last year at 10.74%, slightly increased to 15.72%.

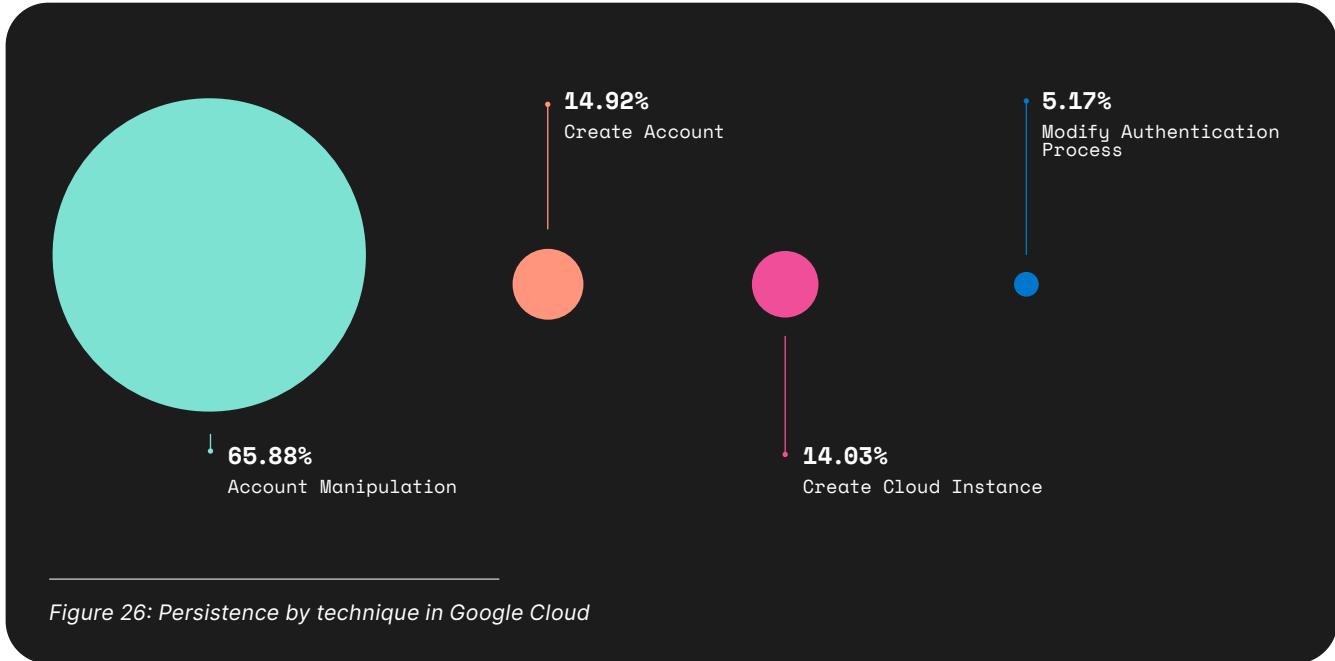
Similar to Microsoft Azure and AWS, *Persistence*,

Initial Access, and *Defense Evasion* rank high for anomalous signals on Google Cloud — however, *Credential Access* is considerably lower. One major difference between CSPs in this regard is data — the most important factor in detecting threat activity in cloud environments. While we suspect brute force login attempts are very common for Google Cloud, the data wasn't sufficient to conclude that impact.

Persistence

Breaking down *Persistence* by technique, we note that 65.88% of all *Persistence* techniques in Google Cloud relate to *Account Manipulation*, similar to the Microsoft Azure portion of this section. However, in Google Cloud, IAM often includes user and service accounts that are high-value targets for adversaries.

Compromised credentials for valid accounts allow adversaries to login and manipulate account authentication, permissions, and more. Organizations should monitor both successful and failed authentication events, and work to identify behavioral anomalies.



Threats with the appropriate level of access commonly create accounts, potentially after *Privilege Escalation* or admin account compromise has already succeeded. Adversaries can use rogue accounts for on-demand access, using otherwise

benign productivity tools to quickly search and exfiltrate. IAM auditing along with contextual data about logins should reveal evidence of account changes, new accounts, and new tenants.

kibana_alert_rule_name	Percentage
Google Workspace API Access Granted via Domain-Wide Delegation of Authority	31.58%
Google Workspace Admin Role Assigned to a User	24.21%
Google Cloud Service Account Key Creation	20.87%
Google Cloud IAM Service Account Key Deletion	9.28%
Google Workspace Custom Admin Role Created	7.46%
Google Workspace Role Modified	5.12%
Google Workspace Password Policy Modified	0.95%
Suspended User Made Active	0.52%

Table 22: Account manipulation by rule name in Google Cloud

Account Manipulation remains a predominant *Persistence* technique within Google Cloud environments, with a significant portion of these activities centered around the misuse of administrative privileges and service accounts. 31.58% of the anomalous signals related to *Account Manipulation* were due to Google Workspace API access being granted via *Domain-Wide Delegation of Authority*. Such delegation of authority can be misused by

adversaries to maintain persistent access to sensitive data and services, bypassing normal authentication mechanisms security teams may be more likely to monitor.

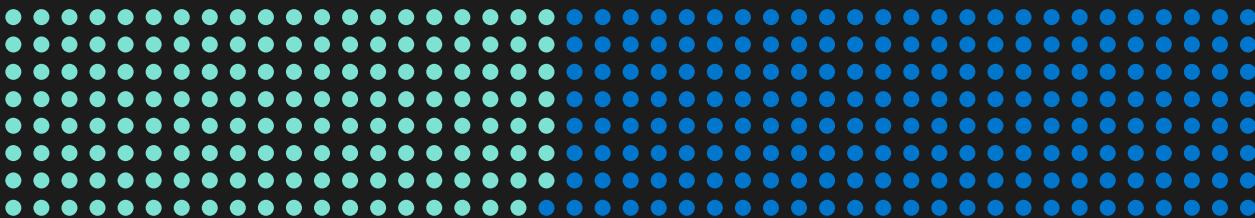
The assignment of Google Workspace Admin roles to users accounts for 24.21% of the *Account Manipulation* signals. Similarly, the creation and deletion of Google Cloud IAM service account keys, which represent 20.87% and 9.28% of alerts respectively, highlight critical security events.

Other notable *Account Manipulation* activities include the creation of custom admin roles (7.46%) and modifications to existing roles (5.12%) within Google Workspace. Custom admin roles can be tailored to grant specific privileges that facilitate ongoing malicious activities while evading detection. Additionally, even less frequent actions, such as modifying password policies (0.95%) or reactivating suspended users (0.52%), can significantly impact security by weakening authentication controls or restoring access to previously compromised accounts.

Initial Access

In the analysis of *Initial Access* techniques within Google Cloud environments, *Valid Accounts* and *Phishing* stand out as the primary methods adversaries use to gain entry. For many security teams, these two methods will be very familiar.

- Phishing **43.80%**



- Valid Accounts **56.20%**

Figure 27: Initial Access by technique

Valid Accounts represented 56.20% of *Initial Access* signals, making it difficult for security teams to detect malicious actions. The high prevalence of valid account usage underscores the critical need for strong password policies, MFA, and continuous monitoring for unusual

login patterns and behaviors. *Phishing*, responsible for 43.80% of *Initial Access* signals, remains a significant threat vector in Google Cloud environments. Once obtained, hijacked credentials can be used to access sensitive data and services within Google Cloud.

kibana_alert_rule_name	Percentage
Google Workspace Object Copied from External Drive and Access Granted to Custom Application	41.44%
User Reported Phishing	29.52%
User Reported Spam Spike	14.98%
Gmail Potential Employee Spoofing	6.63%
Phishing Reclassification	6.09%
User Suspended (Spam)	1.33%

Table 23: Phishing alerts by rule name in Google Cloud

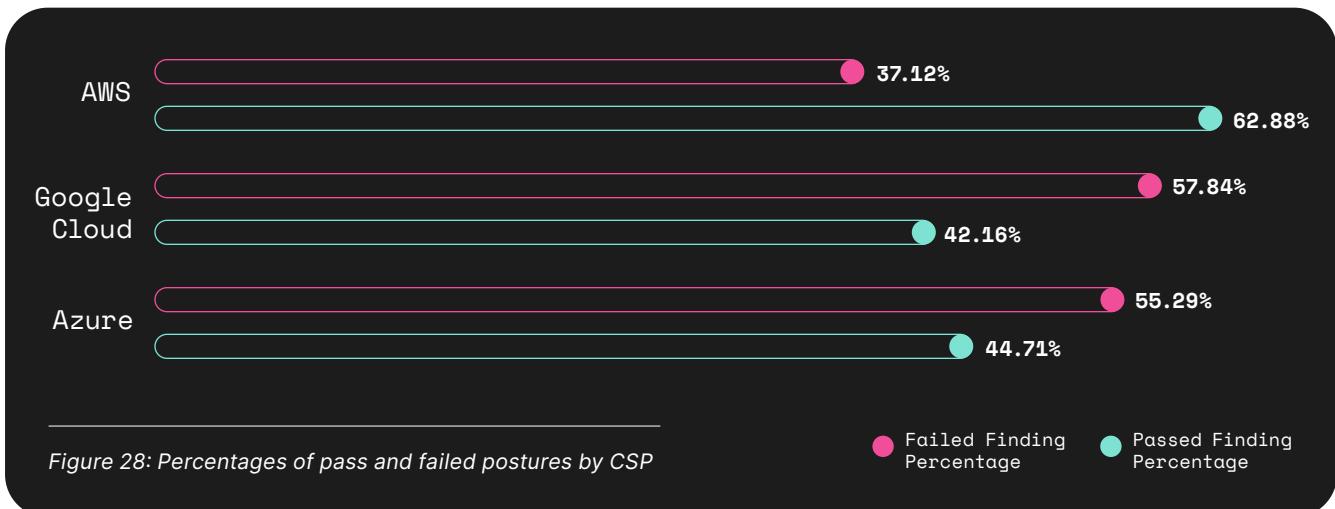
The most common *Phishing* technique involves Google Workspace objects being copied from external drives and access being granted to custom applications, accounting for 41.44% of signals. User reported phishing accounts for 29.52% of the *Phishing* signals, indicating that

end-users are a crucial line of defense in identifying and reporting suspicious activities. User reported spam, distinct from phishing attempts, constituted 14.98% of the signals.

Benchmarking cloud security posture

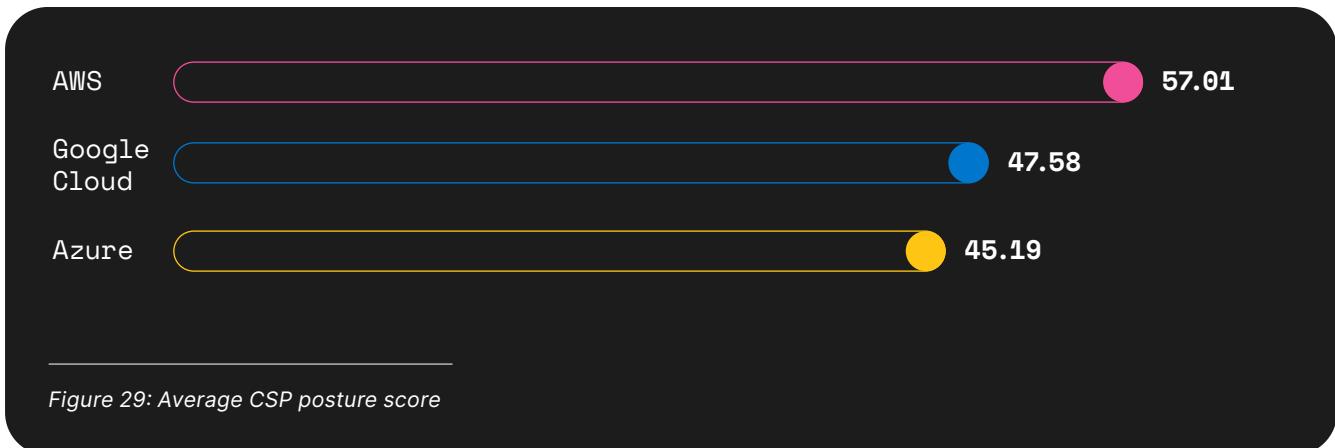
New this year, the Elastic Global Threat Report includes analysis on findings generated by Elastic Security's [cloud security posture management](#) (CSPM) capability. CSPM discovers and evaluates instances of cloud services — such as storage, compute, IAM, and more — against secure

configuration guidelines defined by the [CIS](#). This evaluation helps organizations identify configuration risks. These misconfigurations are described as findings, and figure 27 depicts the proportions of results (passing and failing) gathered from telemetry on a per-CSP basis.



Readers should note that ultimately enterprises are responsible for securing their CSP-hosted infrastructure, and these statistics are not reflections of CSP security but instead the general security of various user populations.

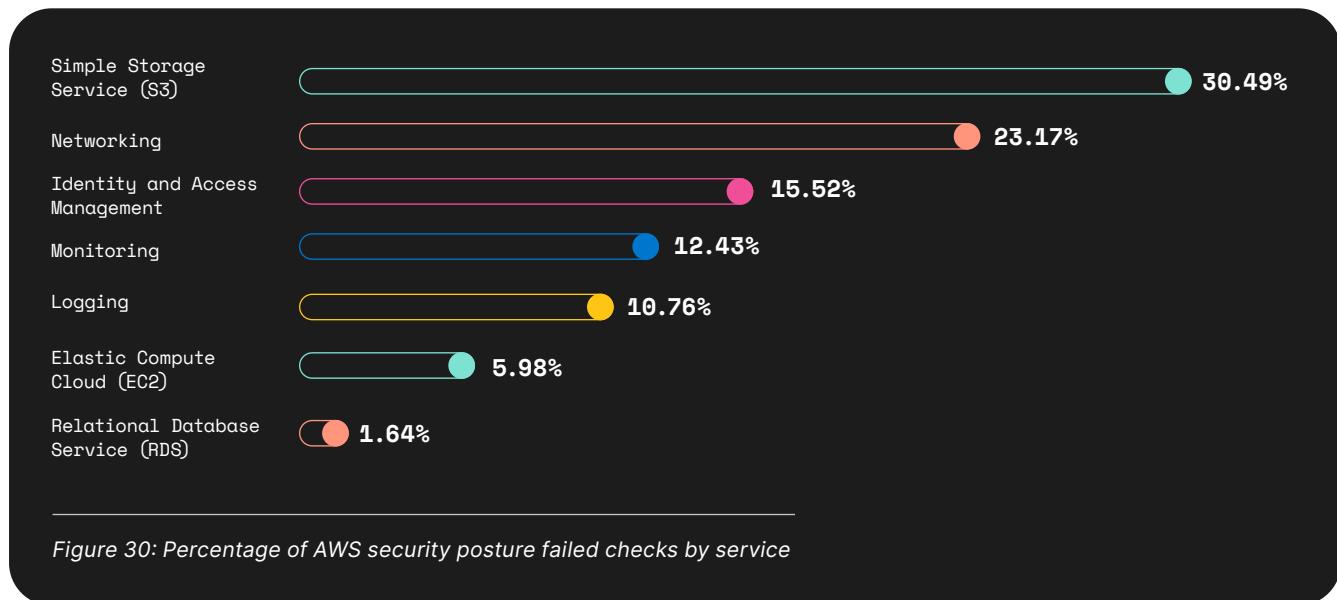
The CIS benchmark score of each CSP, referred to as posture score in figure 28, is scored out of 100 and measures compliance to the secure configuration guidelines outlined in CIS Benchmarks. On average, AWS has a posture score of 57 with Google Cloud and Microsoft Azure at 48 and 45 respectively.



The CIS posture score is an intentional abstraction. To make this actionable, we need to decompose the results of specific CSPs. We will explore the details of each CSP in the following subsections.

Amazon Web Services

Breaking down the failed posture checks by AWS, we observed that 30% of all failed posture checks relate to S3. S3 is an object storage service from AWS that allows users to store and retrieve data. Oftentimes, this service is used to store both sensitive and non-sensitive data by organizations, which is why AWS offers several critical [security features](#) native to S3.



AWS networking, IAM, and monitoring findings reveal an average failing checks percentage of 23.17%, 15.52% and 12.43%, respectively. Taking it a step further, we analyzed failed posture checks for AWS cross-referenced by the CIS benchmark rule names. Nearly 53% of all failed posture checks were related to MFA Delete permissions being

enabled in S3 buckets. The MFA Delete permission allows those with access to S3 buckets to delete an object version or the versioning state of that bucket. In addition to this, 24% of S3 buckets were not configured to block public access, exposing them and their potentially sensitive objects to unauthorized clients.

rule_name	Failed Findings Percentage
Ensure MFA Delete is enabled on S3 buckets	53.23%
Ensure that S3 Buckets are configured with 'Block public access (bucket settings)'	24.00%
Ensure S3 Bucket Policy is set to deny HTTP requests	22.77%

Table 24: Percentage of simple storage service (S3) posture failed checks by rule

When cross-referencing this with the anomalous security information and event management (SIEM) signals analyzed earlier in this report, we observed that a significant portion of these signals were related to S3 bucket configuration changes, objects being deleted, and object access attempts from unknown sources. This correlation underscores the critical importance of enforcing MFA on S3 buckets and objects. For those who seek to increase security posture, we recommend reviewing AWS's [best security practices](#) for S3. Table 26 expands on network security posture

findings by rule. Networking issues accounted for the second-highest number of security posture failures for AWS, with 23% of all checks. Upon further examination, we found that 33% of these failed checks were related to ingress access to networks hosted in AWS. Specifically, policies attached to resources allowed traffic from any IP address or port, whether administrative or otherwise. This misconfiguration leaves numerous VPC networks and potentially EC2 instances vulnerable to access from anyone, including threats.

Benchmark_rule	Failed Findings Percentage
Ensure no Network ACLs allow ingress from 0.0.0.0/0 to remote server administration ports	33.19%
Ensure no security groups allow ingress from 0.0.0.0/0 to remote server administration ports	32.56%
Ensure the default security group of every VPC restricts all traffic	30.62%
Ensure no security groups allow ingress from ::/0 to remote server administration ports	3.63%

Table 25: Percentage of network security posture failed checks by rule

Allowing access from anywhere, anytime, enables threat actors to conduct vulnerability scans, fingerprint web servers, and attempt Remote

Access using protocols such as RDP and SSH. Such a permissive posture undermines controls available at the network perimeter.

Raising your CIS score for AWS

Addressing these misconfigurations in security group rules and network ACLs to restrict ingress traffic to only trusted IP addresses and necessary ports is a fundamental step in improving security posture and raising CIS benchmark scores. Don't allow traffic from any IP address, apply the principle of least privilege by granting only the necessary access needed for services and users to function correctly. Additionally, isolate critical resources and services with network segments. (cont.)

Regular audits of network configurations and continuous monitoring for any changes or anomalies are essential practices. Automated tools and managed services like [AWS Config](#) on the [Elastic Security CSPM dashboard](#) can help maintain compliance with security best practices. Enhanced logging and alerting for unusual or unauthorized access attempts, and using services such as CloudTrail and VPC Flow Logs, provide visibility into network traffic and access patterns.

Microsoft Azure

Microsoft Azure's benchmarks include a significant focus on IAM and collaboration services through Microsoft 365. In analyzing the security posture of Microsoft Azure, a significant concern emerges with storage accounts, which account for 46.68% of all failed security checks. This corresponds

to misconfigurations in Microsoft Azure storage service accounts. A worst-case scenario tabletop exercise might ask: what happens if a ransomware affiliate using stolen credentials purchased from an access broker decides to delete the contents of Microsoft Azure storage?

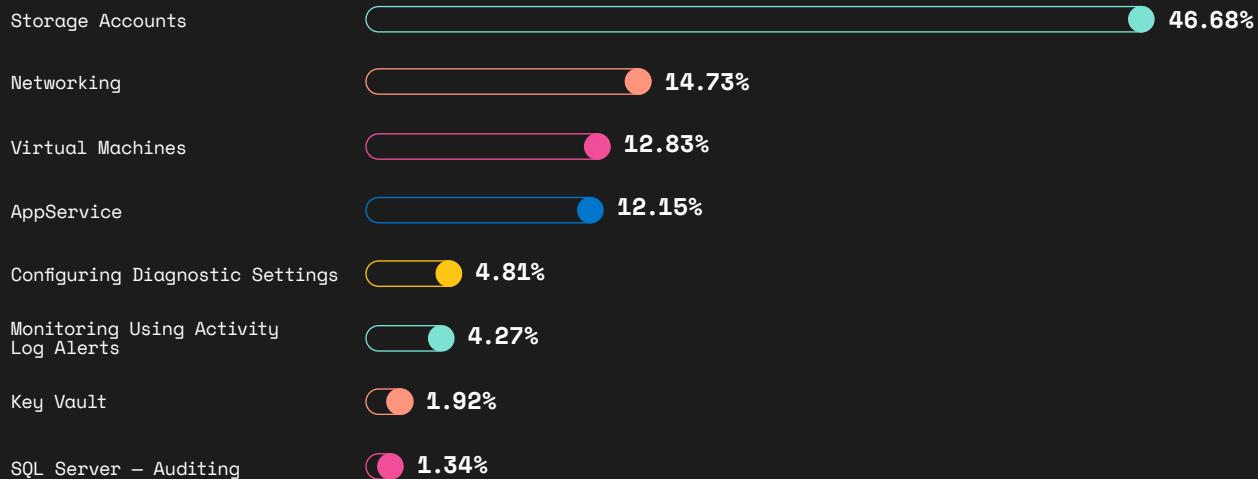


Figure 31: Percentage of Microsoft Azure security posture failed checks by service

Networking, virtual machines (VMs), and AppService represent additional areas of concern, with failed check percentages of 14.73%, 12.83%, and 12.15%, respectively. Networking

misconfigurations can expose internal resources to external threats, while vulnerabilities in VMs and AppService can provide adversaries with footholds to launch further attacks. Ensuring robust security

configurations and continuous monitoring for these services is essential to mitigate risks and protect the overall cloud infrastructure.

Microsoft Azure storage accounts are critical components in many cloud environments,

providing scalable and secure storage solutions for a wide range of data. However, further analysis reveals several common misconfigurations that impact security posture.

Benchmark_rule	Failed Findings Percentage
Ensure that 'Enable Infrastructure Encryption' for Each Storage Account in Microsoft Azure Storage is Set to 'enabled'	15.89%
Ensure Private Endpoints are used to access Storage Accounts	15.52%
Ensure Default Network Access Rule for Storage Accounts is Set to Deny	14.26%
Ensure 'Allow Microsoft Azure services on the trusted services list to access this storage account' is Enabled for Storage Account Access	14.26%
Ensure Storage Logging is Enabled for Queue Service for 'Read', 'Write', and 'Delete' requests	7.67%
Ensure Storage Logging is Enabled for Table Service for 'Read', 'Write', and 'Delete' Requests	7.62%
Ensure Storage logging is Enabled for Blob Service for 'Read', 'Write', and 'Delete' requests	7.57%
Ensure the "Minimum TLS version" for storage accounts is set to "Version 1.2"	6.56%
Ensure Soft Delete is Enabled for Microsoft Azure Containers and Blob Storage	5.14%
Ensure that 'Public access level' is disabled for storage accounts with blob containers	4.00%
Ensure that 'Secure transfer required' is set to 'Enabled'	1.50%

Table 26: Percentage of Microsoft Azure Storage Account security posture failed checks by rule

15.89% of failed checks were due to the lack of infrastructure encryption ('Enable Infrastructure Encryption' was not set to 'enabled'). [Infrastructure encryption](#) adds an extra layer of security by encrypting data at rest using a second layer of encryption to protect sensitive information from potential breaches. This last line of defense against intrusion and theft becomes more

vulnerable as other forms of security are disabled or tampered with. Private endpoints reside in Microsoft Azure virtual networks (VNets) and are used for interacting with Microsoft Azure Storage accounts — a key mechanism of abstraction that resulted in about 16% of failures. Using a private endpoint limits exposure of account credentials and data.

Several service checks were frequently misconfigured, thereby offering easy access to storage accounts for monitoring access patterns to detect suspicious activities:

- Allowing Microsoft Azure services on the trusted services list to access storage accounts — 14.26%

- Setting the default network access rule to deny — 14.26%
- Enabling storage logging for Blob, Queue, and Table services — 22.86% combined

Enabling logging for read, write, and delete requests is particularly important for forensic analysis and auditing purposes.

MFA was not enabled for all IAM users with a console password in 7% of the cases. MFA provides an additional layer of security beyond just a password, making it significantly harder for attackers to gain unauthorized access. Enabling MFA is a crucial security measure, especially for accounts with high privileges. This number should be 0% – the significance of MFA cannot be overstated when discussing IAM security controls.

Google Cloud

Google Cloud places a strong emphasis on its sophisticated data analytics and machine learning capabilities, which broadens the threat landscape. This necessitates robust security measures to protect against unauthorized access and potential breaches, highlighting the importance of maintaining a secure cloud posture in Google Cloud environments.

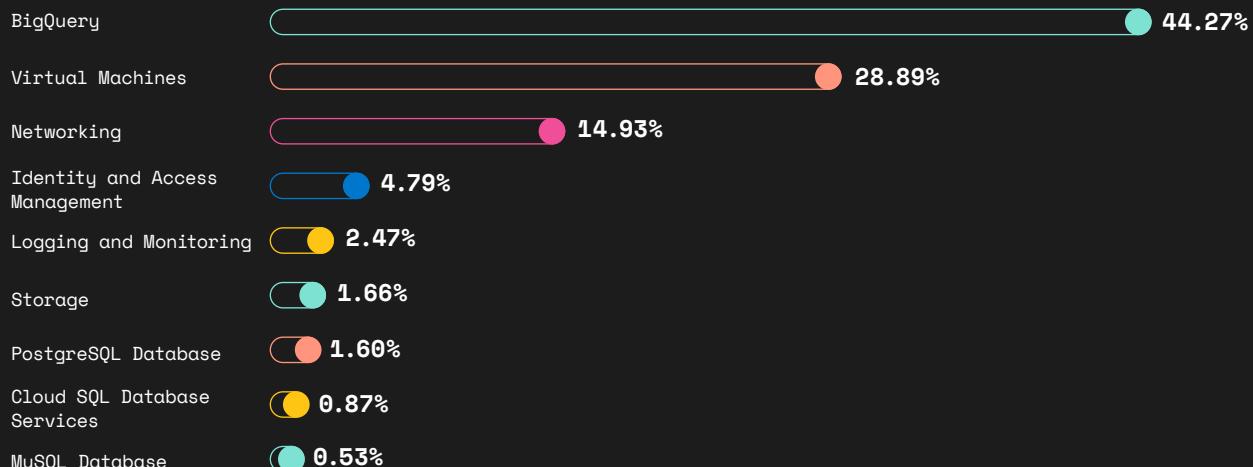


Figure 32: Percentage of Google Cloud security posture failed checks by service

According to our data, Google Cloud's BigQuery is the service with the highest percentage of failed security posture checks, accounting for nearly 44% of the total. BigQuery is a powerful data warehouse solution used for analyzing large data sets, and its extensive use in many organizations makes it a prime target for misconfigurations and security vulnerabilities.

Networking issues account for 15% of the

failed checks, highlighting another critical area where security practices need improvement. Misconfigured network settings, such as overly permissive firewall rules or improper segmentation, can expose cloud resources to external threats and unauthorized access. A significant insight from the security posture analysis of Google Cloud's BigQuery service is the overwhelming prevalence of encryption-related misconfigurations.

rule_name	Failed Finding Percentage
Ensure that All BigQuery Tables Are Encrypted with Customer-Managed Encryption Key (CMEK)	95.75%
Ensure that a Default Customer-Managed Encryption Key (CMEK) Is Specified for All BigQuery Data Sets	4.25%

Table 27: Percentage of Google Cloud BigQuery security posture failed checks by rule

An astonishing 96% of failed checks were due to users utilizing BigQuery tables without encrypting with CMEK. CMEK provides a higher level of control and security over data encryption by allowing organizations to manage their encryption keys. Some of the most critical workloads at any organization may be processed through BigQuery – users must ensure that all BigQuery tables are encrypted with CMEK.

4.25% of the failed checks pertain to the lack of a user-defined default CMEK specified for all BigQuery data sets. This configuration ensures that any new tables created within a data set inherit the specified encryption policy, streamlining security management and reducing the risk of unencrypted data. No failed checks were recorded for anonymous or public BigQuery data sets, which suggests that organizations are vigilant about access control policies. The recurring theme of encryption-

related misconfigurations across different CSPs, including AWS and Microsoft Azure, suggests the need for organizations to adopt more stringent encryption policies and conduct regular audits to protect their data assets in the cloud. Virtual Machines in Google Cloud also present significant security challenges, representing 29% of the failed checks. Misconfigurations in virtual machine settings can expose critical workloads to threats, making them vulnerable to attacks such as unauthorized access, malware infections, and

data breaches. Properly securing virtual machines involves implementing stringent access controls, ensuring up-to-date patching and updates, and using advanced security features like shielded

VMs and disk encryption.

The security posture analysis of Google Cloud VMs reveals significant vulnerabilities related to encryption and SSH key management.

Benchmark_rule	Failed Finding Percentage
Ensure VM Disks for Critical VMs Are Encrypted with Customer-Supplied Encryption Keys (CSEK)	51.24%
Ensure “Block Project-Wide SSH Keys” Is Enabled for VM Instances	20.06%
Ensure Oslogin Is Enabled for a Project	11.10%
Ensure that Compute Instances Have Confidential Computing Enabled	6.71%
Ensure that Compute Instances Do Not Have Public IP Addresses	4.19%
Ensure that Instances Are Not Configured to Use the Default Service Account	3.49%
Ensure that IP Forwarding Is Not Enabled on Instances	1.24%
Ensure Compute Instances Are Launched with Shielded VM Enabled	0.90%
Ensure that Instances Are Not Configured to Use the Default Service Account with Full Access to All Cloud APIs	0.85%
Ensure ‘Enable Connecting to Serial Ports’ Is Not Enabled for VM Instance	0.23%

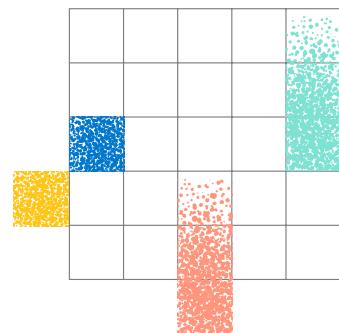
Table 28: Percentage of Google Cloud VM security posture failed checks by rule

More than 51% of the failed checks were due to the lack of CSEK for VM disks, which is crucial for ensuring that organizations retain control over their encryption keys and, consequently, their data security.

Another notable finding is that 20.06% of failed checks were due to the absence of project-wide SSH key blocking. When “Block Project-Wide

SSH Keys” is not enabled, it allows SSH keys that are added at the project level to be used across all VM instances in that project, potentially leading to unauthorized access if any of these keys are compromised. Enabling this setting ensures that only instance-specific SSH keys are permitted, reducing the attack surface and preventing unauthorized access to VM instances through SSH.

There is no CSP capable of protecting you from yourself, and we assess that the most common findings were the result of users or administrators weakening security – not flaws or inherent weaknesses. Readers should consider their own use of these CSPs and how close they can get to a perfect CIS benchmark with the fewest number of accepted risks.



Threat Profiles

Over the course of this year, Elastic Security Labs has tracked dozens of threats observed in Elastic telemetry. Researchers and engineers with expertise as intelligence analysts, malware reverse engineers, and detection scientists analyze these threats to help discover the most effective methods to mitigate them. For the Global Threat Report, we have described five major threat profiles developed during the past year as a part of Elastic Security Labs' dedication to democratizing access to the threat landscape. These profiles were chosen based on threats in-the-wild that were observed in our unique telemetry and which demonstrate new or novel approaches that security teams may not have seen. The activity groups included in this year's Global Threat Report are:

- **REF5961:** Three novel malware families targeting an Association of Southeast Asian Nations (ASEAN) foreign affairs ministry.
- **GHOSTPULSE:** Novel malware using Microsoft Installer for Windows 10 (MSIX) application packages to gain *Initial Access*.
- **GHOSTENGINE:** An undocumented backdoor used to establish *Persistence* and execute a cryptominer.
- **KANDYKORN:** Novel intrusion targeting blockchain engineers of a cryptocurrency exchange platform, an operation used by an isolated state to evade sanctions imposed by the international community.
- **WARMCOOKIE:** Novel backdoor utilized for espionage.

Threat naming

Elastic Security Labs uses a reference tracking system that clusters activity groups, attack patterns, and intrusion sets. These correlate to specific malware, attack logic, techniques, and sometimes, victimology. They are assigned a four-digit number with the prefix "REF" (example: REF1234). Readers should not confuse these for

cryptonyms, which Elastic Security Labs withholds. When the discovery is a previously-undocumented malware family like EAGERBEE or an intrusion technique like GrimResource, we use distinct naming conventions to avoid confusion; all capitals for malware and camel casing for techniques.

The diamond model

Each threat profile contains a conventional diagram called the Diamond Model for each group listed with a REF identifier. To improve readability, we have omitted overlaps with named groups tracked by other vendors, but readers should note that this doesn't indicate agreement or disagreement with those vendors.

We utilize the diamond model to describe

high-level relationships between adversaries, capabilities, infrastructure, and intrusion victims with an emphasis on the actionable technical axis (capabilities:infrastructure). This model is often used in an intrusion-centric way, but here we employ it with an adversary focus to highlight observations over many incidents.

Terminology

We use a subset of industry terms to describe threat activity and related outputs, many of which are nested within or derived from each other. An activity group can consist of one or more attack patterns, and be summarized as one or more intrusion sets. The following are listed by degree of information, from greatest to least:

- **Activity group:** A collection of activities attributed to individuals or organizations believed to be operating with malicious intent.

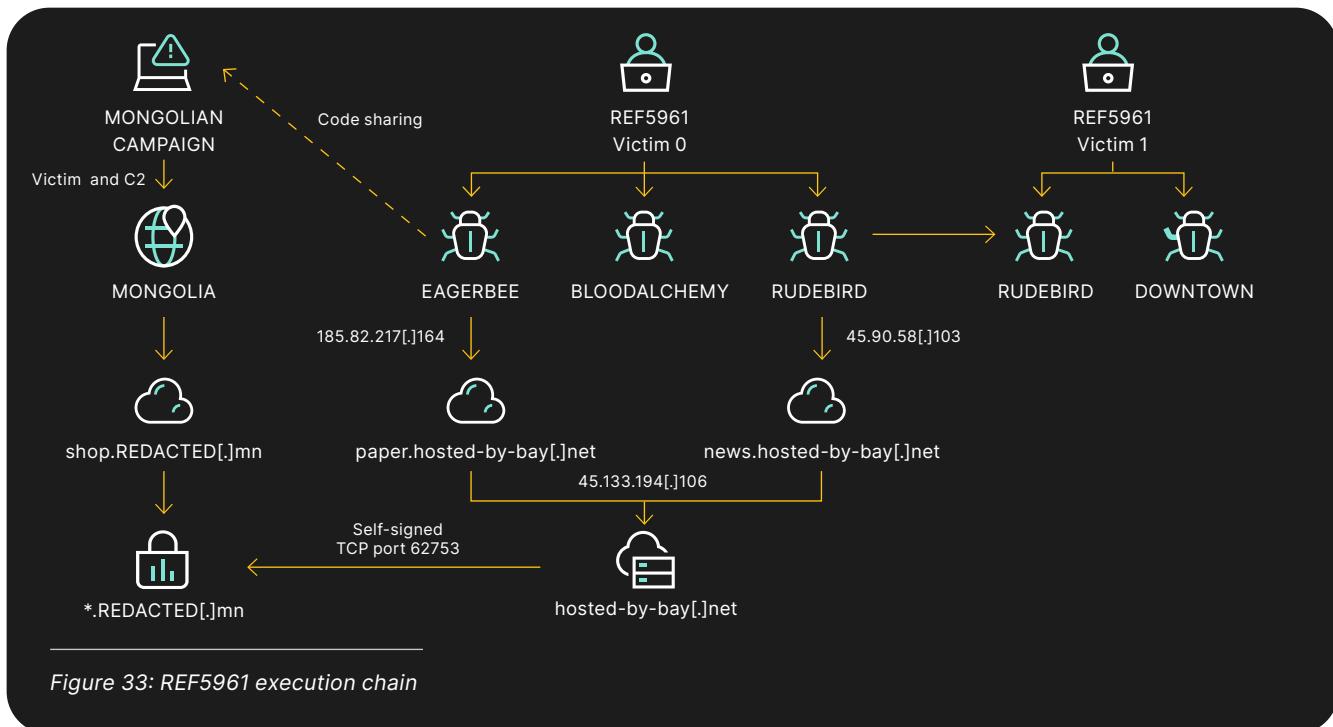
- **Attack pattern:** A detailed description of how adversaries attempted to compromise specific targets; more commonly referred to as TTPs.
- **Intrusion set:** A summary of adversarial behaviors and resources with common properties believed to be orchestrated by a single organizationobservations over many incidents.

REF5961

BLOODALCHEMY, RUDEBIRD, EAGERBEE, DOWNTOWN

REF5961 is an intrusion set that includes three novel malware families discovered and analyzed by Elastic Security Labs. These malware families were co-resident with families discovered in the REF2924 intrusion set and targeted a member of

the ASEAN Foreign Affairs Ministry. Figure 31 describes how REF5961 deployed the different malware, their associated infrastructure, and lateral movement.



What is the threat?

The REF5961 intrusion set is activity that aligns with state-sponsored and/or espionage-motivated threat actor behaviors. Further, the correlation of execution flows, tooling, infrastructure, and victimology between multiple campaigns we're tracking, as well as consensus with third-party intelligence peers, supports the hypothesis that this represents a China-nexus actor.

EAGERBEE is a newly identified backdoor that loads additional capabilities using remotely downloaded portable executable (PE) files hosted on adversary-controlled infrastructure. However, its implementation and coding practices rely on conventionally straightforward techniques. During our EAGERBEE analysis, we also observed two (previously unnamed) samples involved in a targeted campaign focused on the Mongolian

government. These samples were bundled with other shared files and coding metadata.

RUDEBIRD is a lightweight Windows backdoor that communicates over HTTPS and contains capabilities to perform reconnaissance and execute code.

DOWNTOWN is a modular implant that shares plugin architecture, has code similarities, and aligns with the victimology described in relation to the publicly reported malware SMANGER/PHANTOMNET.

BLOODALCHEMY is an x86 backdoor written in C and found as shellcode injected into a signed benign process. It is still in active development and includes multiple running modes, Persistence mechanisms, and communication options.

What is the impact?

The REF5961 victim's status as a government diplomatic agency would make it an ideal stepping-off point for other targets within and outside the agency's national borders. Additionally, multiple

entities within the foreign national intelligence apparatus of regional powers would have collection requirements that could be satisfied by this victim directly.

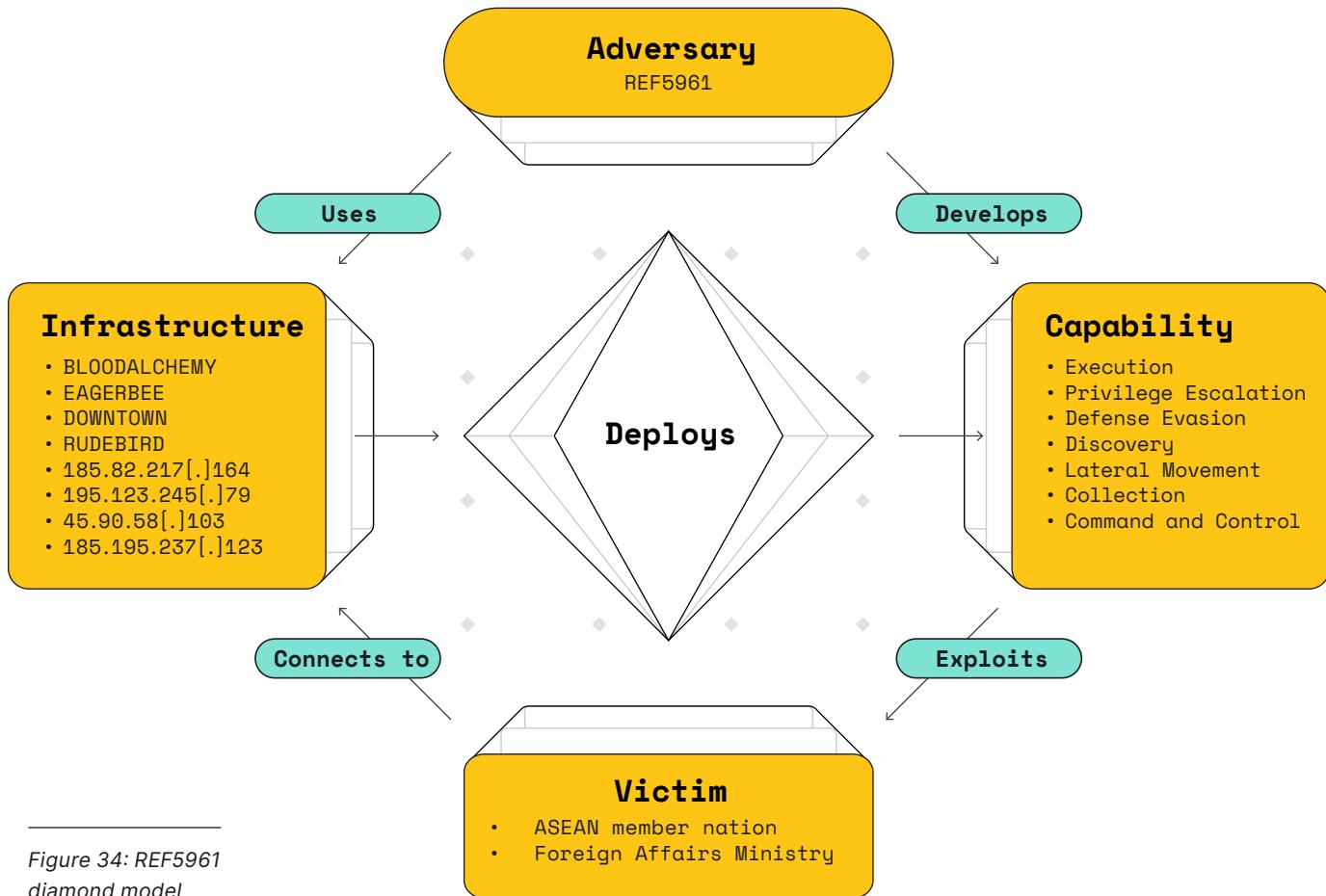


Figure 34: REF5961 diamond model

What was Elastic's response?

Elastic provides out-of-the-box detections and preventions for REF5961 in the Elastic Security solution. Additionally, Elastic publicly released YARA rules, a detailed campaign, and malware

analysis. Regular industry reporting, including Elastic's research publication, helps mitigate this threat. Here is an example of the YARA rule released specifically for RUDEBIRD:

```
RUDEBIRD
rule Windows_Trojan_RudeBird {
    meta:
        author = "Elastic Security"
        creation_date = "2023-05-09"
        last_modified = "2023-06-13"
        threat_name = "Windows.Trojan.RudeBird"
        license = "Elastic License v2"
        os = "windows"

    strings:
        $a1 = { 40 53 48 83 EC 20 48 8B D9 B9 D8 00 00 00 E8 FD C1 FF FF 48 8B C8 33 C0 48 85
C9 74 05 E8 3A F2 }

    condition:
        all of them
}
```

Learn More

↳

Elastic Security Labs articles:

- [Introducing the REF5961 intrusion set](#)
- [Disclosing the BLOODALCHEMY backdoor](#)
- [SiestaGraph: New implant uncovered in ASEAN member foreign ministry](#)
- [Update to the REF2924 intrusion set and related campaigns](#)

Malpedia entries:

- [BLOODALCHEMY](#)
- [EAGERBEE](#)

REF8207

GHOSTPULSE

In October 2023, Elastic Security Labs observed a campaign to compromise users with signed MSIX application packages. The campaign leveraged a newly-discovered loader we named GHOSTPULSE, which decrypts and injects its final payload to evade detection.

MSIX is a Windows app package format that developers can leverage to package, distribute,

and install their applications to Windows users. With App Installer, MSIX packages can be installed with a double-click. This makes them a potential target for adversaries looking to compromise unsuspecting victims. However, MSIX requires access to purchased or stolen code signing certificates, making them viable for groups that can obtain those resources.

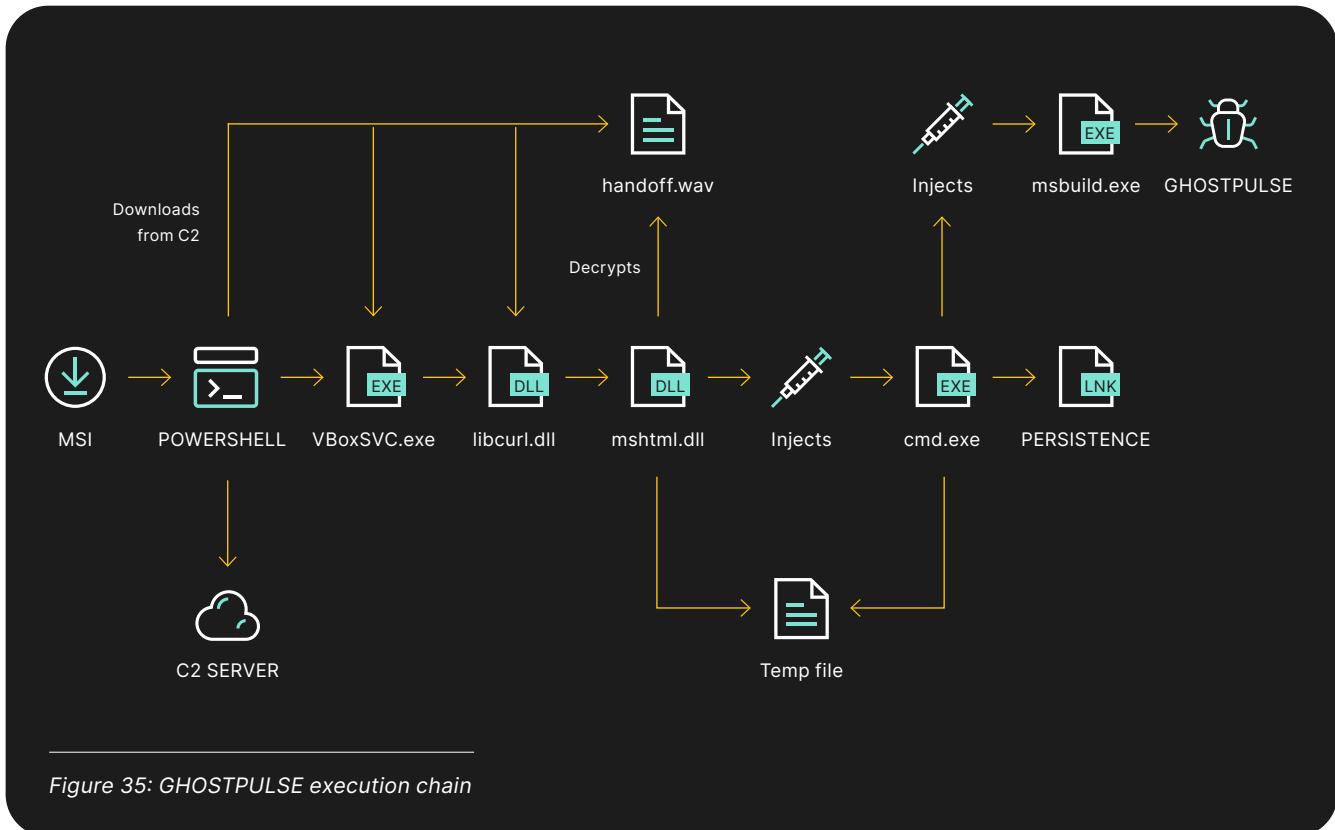


Figure 35: GHOSTPULSE execution chain

What is the threat?

GHOSTPULSE is a multi-stage malware implant used to gain an initial foothold, establish Persistence, collect information about the host, and then deploy additional malware. In a common attack scenario, users are directed to download malicious MSIX packages through compromised websites, search engine optimization (SEO) techniques, or malvertising.

The masquerading themes we've observed include installers for Chrome, Brave, Edge, Grammarly, and WebEx. From the user's perspective, the "Install" button functions as intended. No pop-ups or warnings are presented. However, a PowerShell script is covertly used to download, decrypt, and execute GHOSTPULSE on the system.

What is the impact?

Once GHOSTPULSE had completed its execution flow, information stealers such as SECTOPRAT, RHADAMANTHYS, VIDAR, LUMMA, and NETSUPPORT were observed loading.

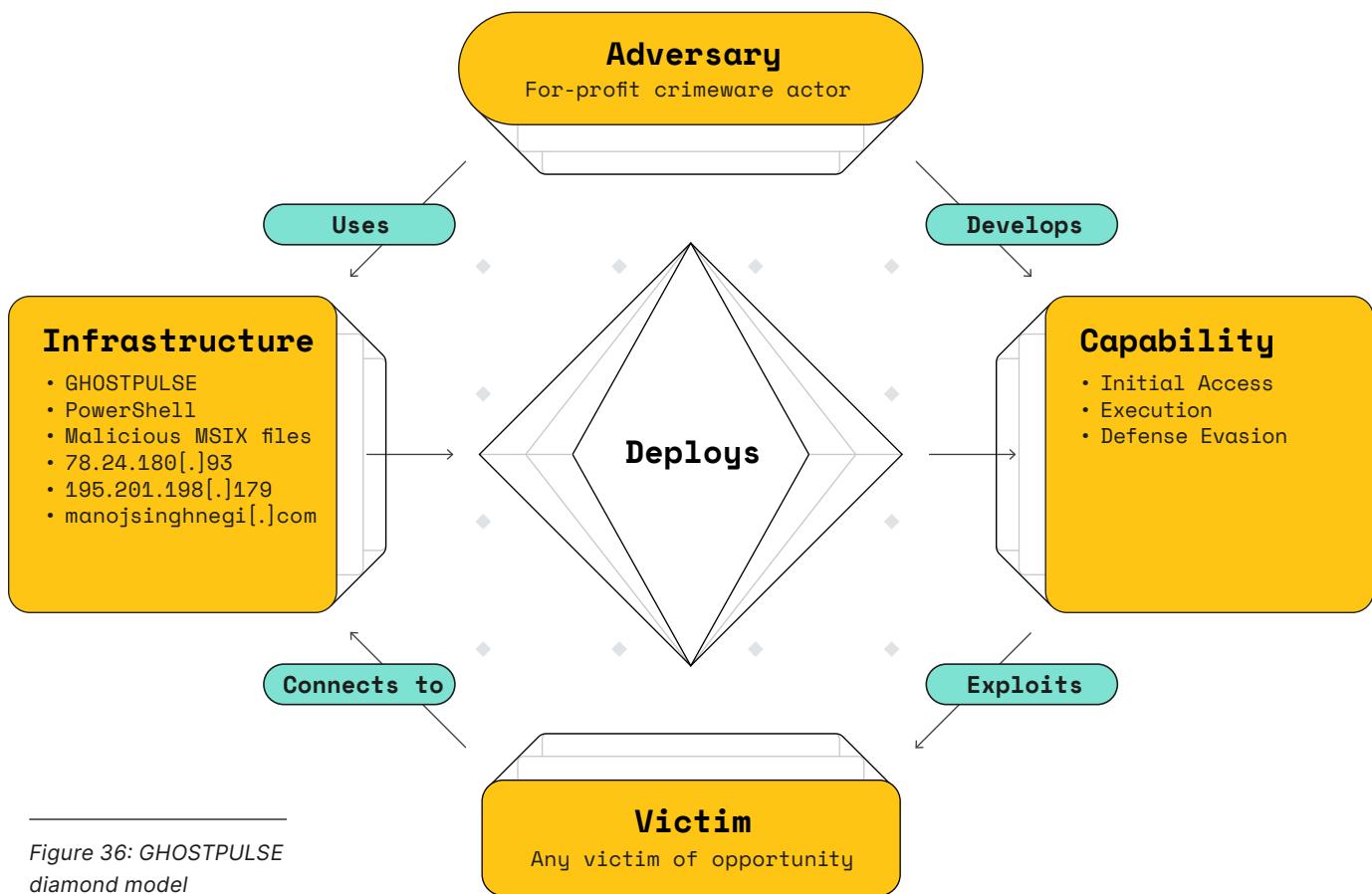


Figure 36: GHOSTPULSE diamond model

What was Elastic's response?

The Elastic Security Labs team detailed the malware architecture and execution, its phases, and the tactics and techniques observed. To further minimize impact, detection and prevention

capabilities were published and configuration extractor tools were shared, alongside file and network indicators of compromise (IOCs).

Learn More



Elastic Security Labs article:

- [GHOSTPULSE haunts victims using defense evasion bag o' tricks](#)

Malpedia entry:

- [GHOSTPULSE](#)

REF4578

GHOSTENGINE

In May 2024, Elastic Security Labs identified REF4578, an intrusion set describing several malicious modules and leveraging vulnerable drivers to disable known security solutions (EDRs)

for cryptomining. In this research, we shared details about GHOSTENGINE, a previously undocumented backdoor used to establish Persistence and execute a cryptominer.

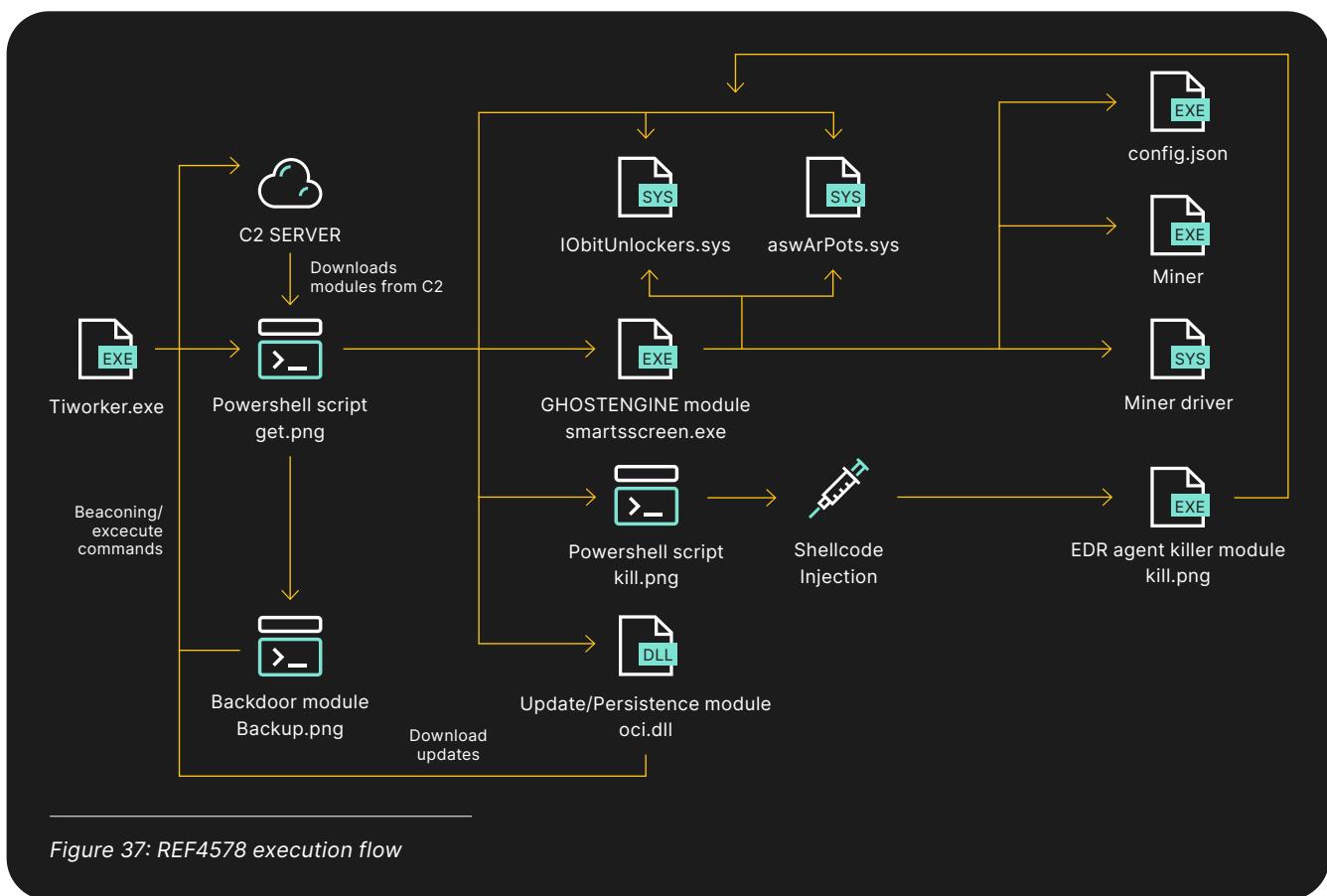


Figure 37: REF4578 execution flow

What is the threat?

REF4578 is an intrusion set which describes a new backdoor named GHOSTENGINE. The campaign owners incorporated many contingency and backup mechanisms, leveraged vulnerable drivers (BYOVD) to terminate and remove known

EDR agents, and executed the campaign with uncommon complexity. These elements suggest intentionality from an operator with a preference for reliable miners.

What is the impact?

The threat actor was able to disable any active EDR agents, load cryptominers (the XMRig client mining program was observed), and maintain

Persistence and Remote Access Software by using the Microsoft Distributed Transaction Service to load a phantom DLL to execute a PowerShell script.

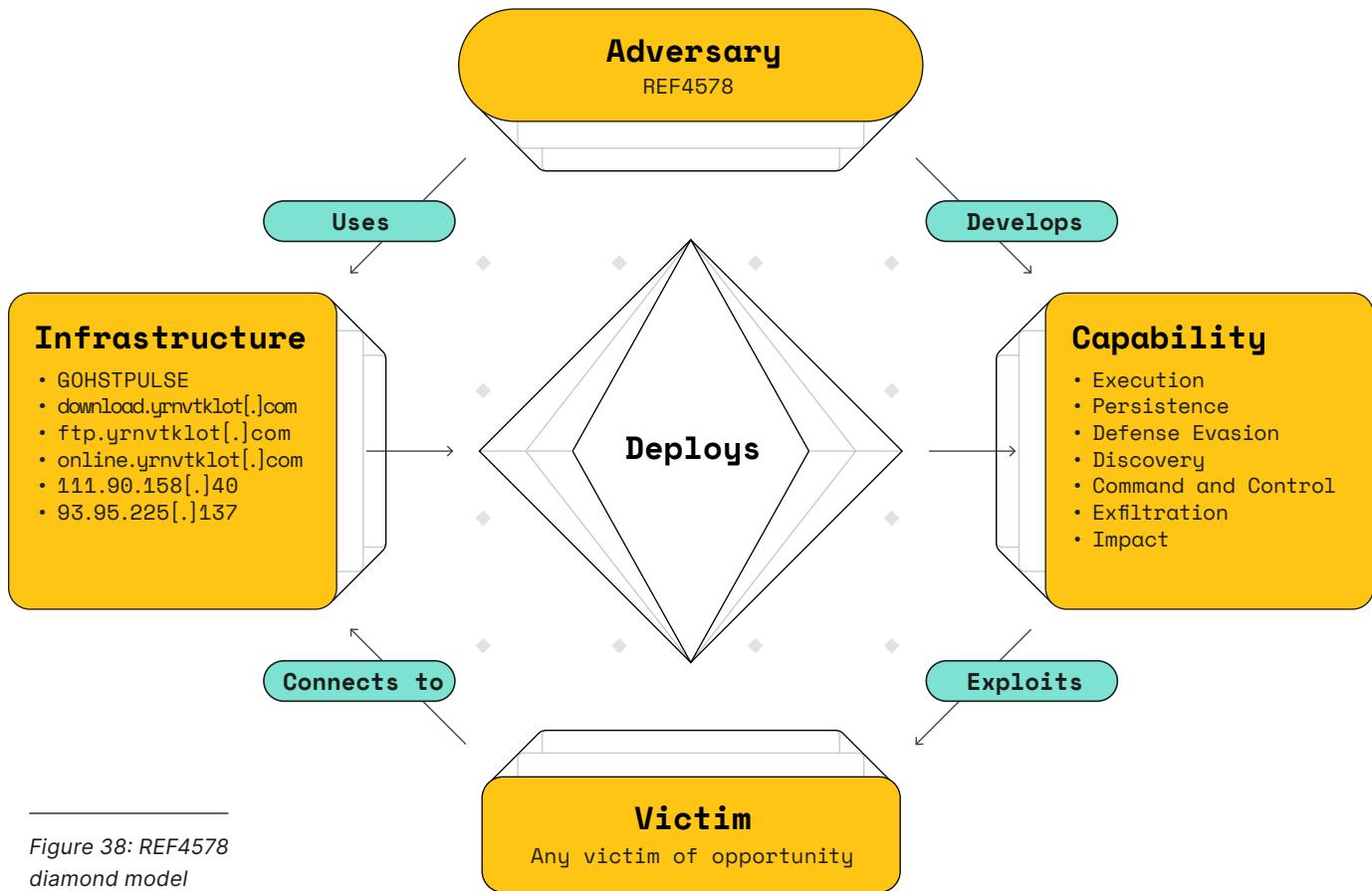


Figure 38: REF4578
diamond model

What was Elastic's response?

Elastic Security Labs released an analysis product highlighting the execution flow of REF4578 and the EDR controller, *Persistence*, and *Remote Access* mechanisms leveraged by the REF4578 intrusion set and mapping these observations to the MITRE ATT&CK framework.

To understand the campaign's scale, the XMRig configuration, Monero wallet identifiers, and wallet balance were shared. Also released were YARA rules for the malware observed in the intrusion set — namely, endpoint prevention rules and host and network atomic indicators.

```
rule Windows_Trojan_GhostEngine {
    meta:
        author = "Elastic Security"
        creation_date = "2024-05-07"
        last_modified = "2024-05-13"
        threat_name = "Windows.Trojan.GhostEngine"
        license = "Elastic License v2"
        os = "windows"

    strings:
        $str0 = "\\\.\IOBitUnlockerDevice"
        $str1 = "C:\Windows\Fonts\taskhostw.exe"
        $str2 = "C:\Windows\Fonts\config.json"
        $str3 = "/drives/kill.png"
        $str4 = "C:\Windows\Fonts\WinRing0x64.sys"
        $str5 = "C:\Windows\Fonts\smartsscreen.exe"
        $binary0 = { 89 C2 C1 E8 1F C1 E0 1F 85 C0 0F 84 74 01 00 00 D1 E2 89 CB C1 E9 1F 09 D1
D1 E3 C1 EB 1F 89 CA D1 E1 09 D9 89 CB 81 C1 80 7F B1 D7 C1 EA 1F 81 C3 80 7F B1 D7 83 D2 0D 81
C1 00 09 6E 88 89 4C 24 20 83 D2 F1 89 54 24 24 }
        $binary1 = { 83 F9 06 0F ?? ?? ?? ?? ?? 8B 10 81 FA 78 38 36 5F 0F 85 ?? ?? ?? ?? ?? 0F B7
50 04 66 81 FA 36 34 74 ?? E9 ?? ?? 00 00 C7 04 24 00 E4 0B 54 C7 44 24 04 02 00 00 00 }

    condition:
        3 of ($str*) or 1 of ($binary*)
}
```

**Learn
More**



Elastic Security Labs article:

- Initial Research exposing JOKERSPY

Malpedia entry:

- ## • JOKERSPY

REF7001

KANDYKORN

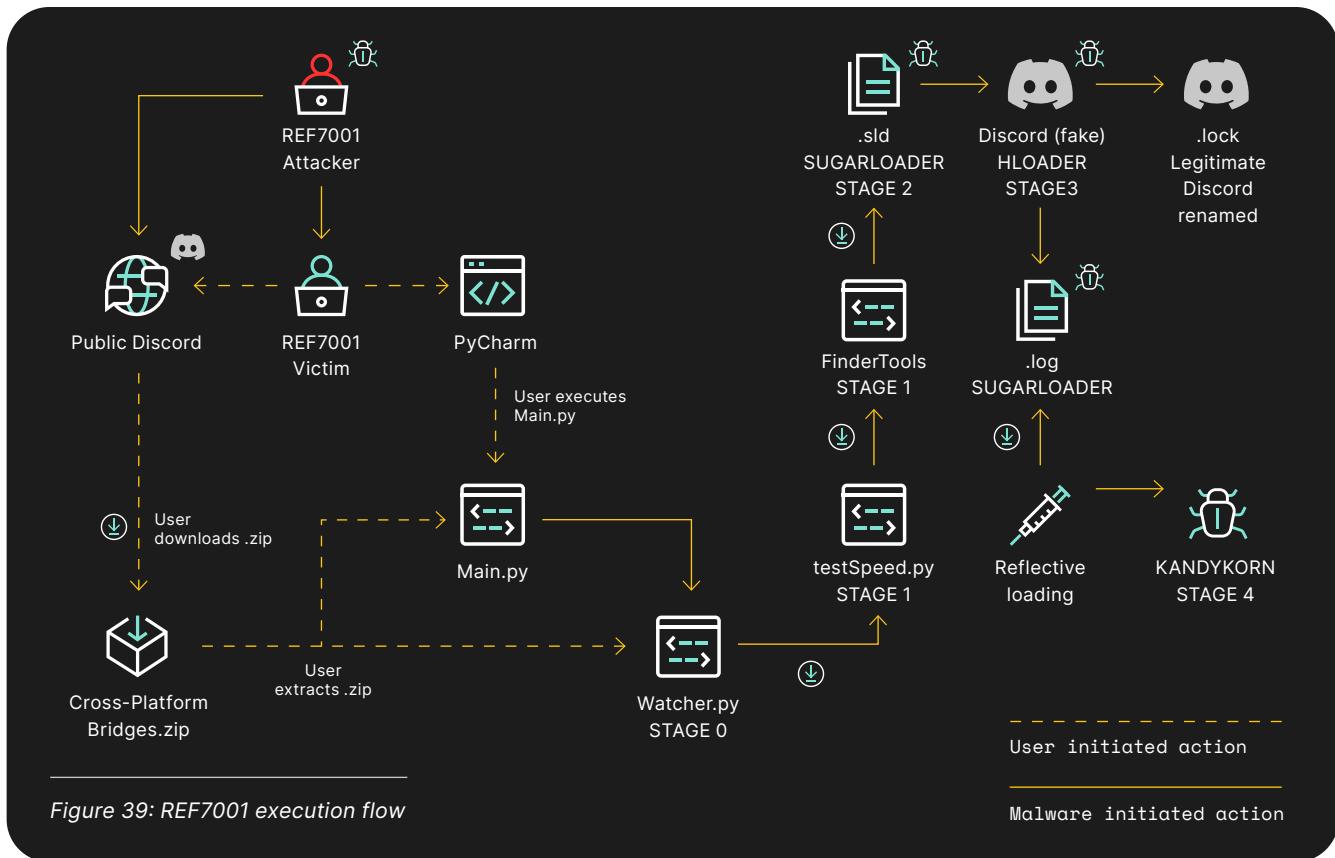
In October 2023, Elastic Security Labs disclosed a novel intrusion targeting blockchain engineers of a crypto exchange platform. The intrusion leveraged custom and open source capabilities for *Initial*

Access and post-exploitation.

The intrusion was discovered when analyzing attempts to reflectively load a binary into memory on a macOS endpoint. The intrusion was traced

to a Python application posing as a cryptocurrency arbitrage bot delivered via a direct message on a public blockchain Discord server.

We attribute this activity to the Democratic People's Republic of Korea (DPRK) and recognize overlaps with public reporting.



What is the threat?

Threat actors lured blockchain engineers with a Python application to gain *Initial Access* to the environment. This intrusion involved multiple complex stages that each employed deliberate *Defense Evasion* countermeasures. The intrusion

set was observed on a macOS system where an adversary attempted to load binaries into memory, which is atypical of macOS intrusions. The final stage involved loading KANDYKORN, which is a full-featured *Remote Access* and *Exfiltration* tool.

What is the impact?

REF7001 exposed an operational and tactical technique of social engineering and loading binaries directly into memory on macOS systems.

The DPRK commonly uses social engineering, targeting human resources and now engineers, to achieve *Initial Access* into a contested

environment. Coupling this with novel macOS malware connected the threat actors to hosts of engineers that could provide access to sensitive cryptocurrency exchange data, intellectual property, or supply chains. DPRK has demonstrated

creative solutions to sanctions that include cryptocurrency theft, fraudulent IT worker scams, and Society for Worldwide Interbank Financial Telecommunication (SWIFT) transaction fraud.

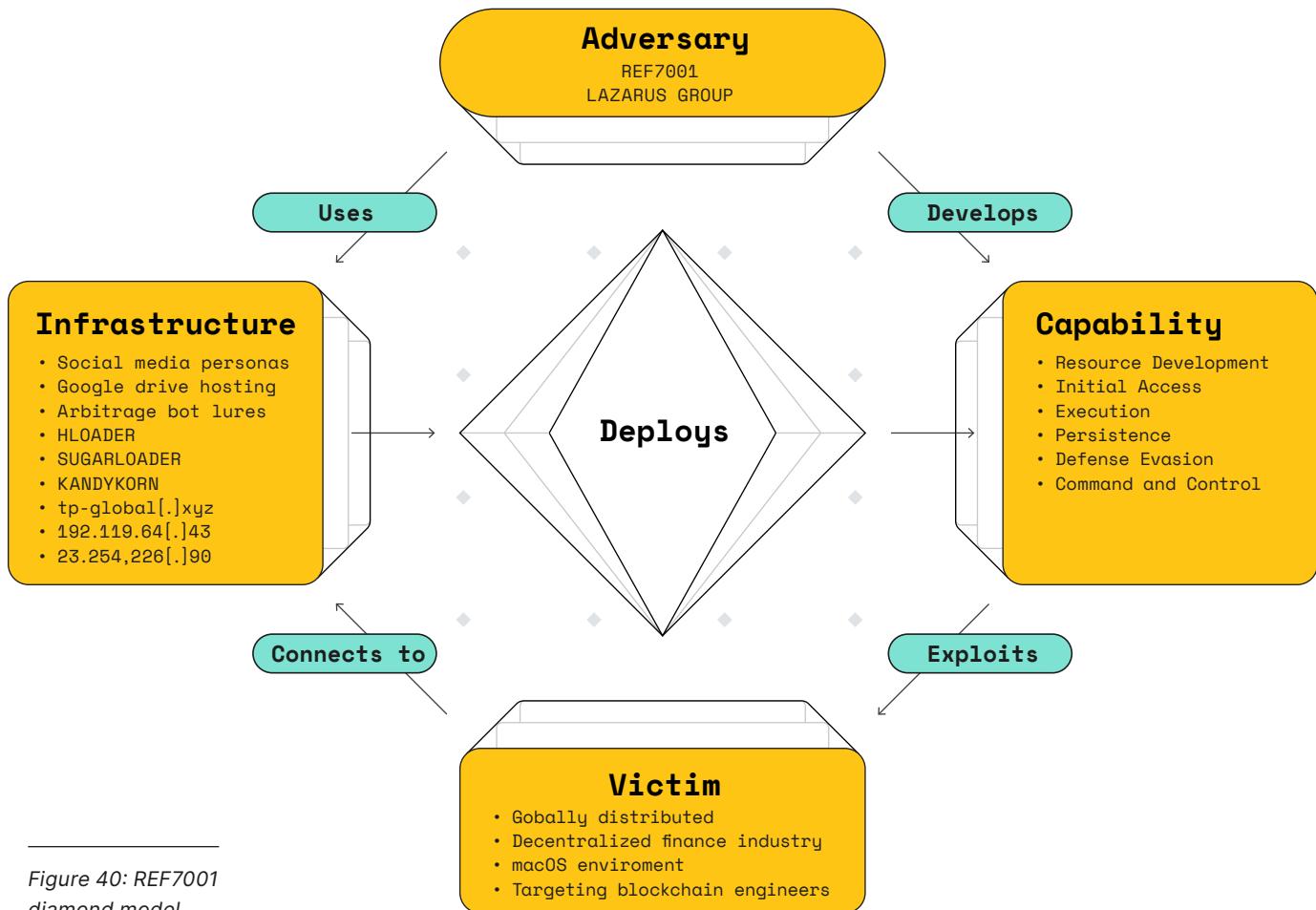


Figure 40: REF7001 diamond model

What was Elastic's response?

Elastic has publicly released detailed malware and campaign analysis, YARA signatures, and endpoint protections to detect and prevent malware in this intrusion set. Additionally, we released all atomic indicators observed in this intrusion.

Our research into REF7001 has resulted in three

YARA rules focused on identifying the different malware observed in this intrusion set alongside eight [Event Query Language](#) (EQL) hunting queries that identify behavioral and technical elements of this campaign.

The following is an example of an EQL query created for KANDYKORN; specifically, this can be used to identify when a hidden executable creates and then immediately deletes a file within a temporary directory:

```
sequence by process.entity_id, file.path with maxspan=30s
[file where event.action != "deletion" and process.name :".*" and
file.path : ("private/tmp/*", "/tmp/*", "/var/tmp/*")]
[file where event.action == "deletion" and process.name: ".*" and
file.path : ("private/tmp/*", "/tmp/*", "/var/tmp/*")]
```

Learn More

Elastic Security Labs article:

- [Elastic catches DPRK passing out KANDYKORN](#)

Malpedia entries:

- [KANDYKORN](#)
- [HLOADER](#)
- [SUGARLOADER](#)

REF6127 WARMCOOKIE

In June 2024, Elastic Security Labs observed a wave of recruitment-themed email campaigns targeting environments by deploying a new backdoor we named WARMCOOKIE. While some features are similar to previously reported

variants, such as the implementation of string obfuscation, WARMCOOKIE contains differing functionality. Our team has seen this threat distributed daily with the use of recruiting and job themes targeting individuals.

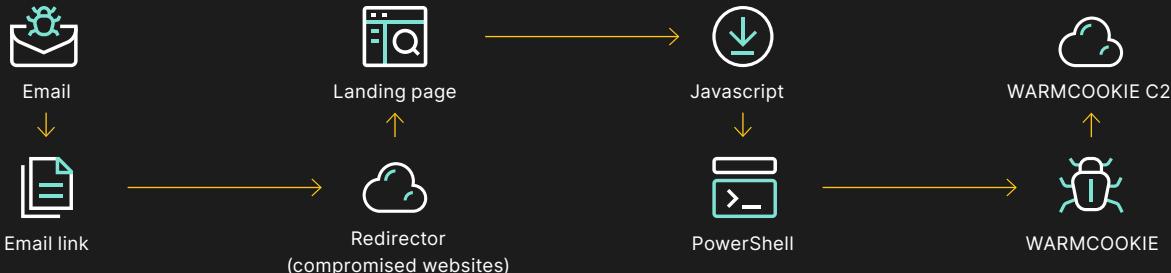


Figure 41: REF6127 execution flow

What is the threat?

WARMCOOKIE is an initial backdoor tool used to scout out victim networks and deploy additional payloads. Each sample is compiled with a hard-coded C2 IP address and RC4 key.

What is the impact?

WARMCOOKIE provides seven command handlers for threat actors to invoke different functions, including retrieving victim information, recording screenshots, launching additional payloads, and

more. The provided functionality is relatively straightforward, allowing threat groups that need a lightweight backdoor to monitor victims and deploy further damaging payloads such as ransomware.

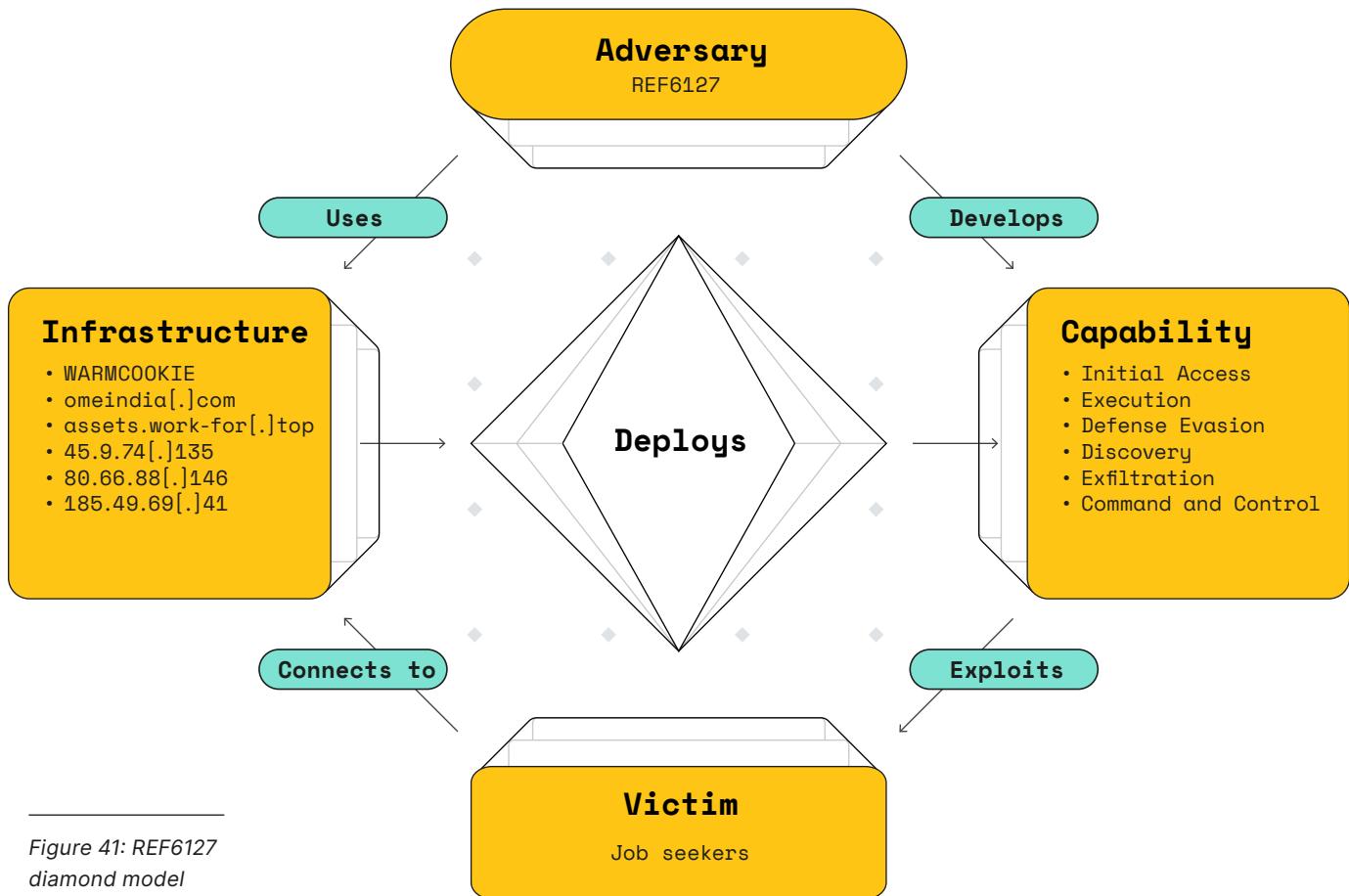


Figure 41: REF6127 diamond model

What was Elastic's response?

Elastic Security Labs has publicly released detailed malware and campaign analysis, a YARA signature,

and endpoint protections to detect and prevent malware in this intrusion set. We mapped this

intrusion set to the MITRE ATT&CK framework and released all atomic indicators observed in this

intrusion. The following is the YARA signature to detect WARMCOOKIE:

```
rule Windows_Trojan_WarmCookie_7d32fa90 {
    meta:
        author = "Elastic Security"
        creation_date = "2024-04-29"
        last_modified = "2024-05-08"
        os = "Windows"
        arch = "x86"
        threat_name = "Windows.Trojan.WarmCookie"
        license = "Elastic License v2"

    strings:
        $seq_checksum = { 45 8D 5D ?? 45 33 C0 41 83 E3 ?? 49 8D 4E ?? 44 03 DB 41 8D 53 ?? }
        $seq_string_decrypt = { 8B 69 04 48 8D 79 08 8B 31 89 6C 24 ?? 48 8D 4E ?? E8 }
        $seq_filesearch = { 48 81 EC 58 02 00 00 48 8B 05 82 0A 02 00 48 33 C4 48 89 84 24 40 02
00 00 45 33 C9 48 8D 44 24 30 45 33 C0 48 89 44 24 20 33 C9 41 8D 51 1A FF 15 83 4D 01 00 85 C0
78 22 48 8D 4C 24 30 E8 1D }
        $seq_registry = { 48 81 EC 80 02 00 00 48 8B 05 F7 09 02 00 48 33 C4 48 89 84 24 70 02
00 00 4C 89 B4 24 98 02 00 00 48 8D 0D 4D CA 01 00 45 33 F6 41 8B FE E8 02 4F 00 00 48 8B E8 41
B9 08 01 00 00 48 8D 44 24 }
        $plain_str1 = "release.dll" ascii fullword
        $plain_str2 = "\Main Invoked.\\" ascii fullword
        $plain_str3 = "\Main Returned.\\" ascii fullword
        $decrypt_str1 = "ERROR: Cannot write file" wide fullword
        $decrypt_str2 = "OK (No output data)" wide fullword
        $decrypt_str3 = "OK (See 'Files' tab)" wide fullword
        $decrypt_str4 = "cmd.exe /c %ls" wide fullword
        $decrypt_str5 = "Cookie:" wide fullword
        $decrypt_str6 = "%ls\%.*" wide fullword
    condition:
        (3 of ($plain*)) or (2 of ($seq*)) or 4 of ($decrypt*)
}
```

Learn More



Elastic Security Labs article:

- [Dipping into Danger: The WARMCOOKIE backdoor](#)

Malpedia entry:

- [WARMCOOKIE](#)



Responding to 2023 Forecasts

Each year, Elastic Security Labs offers several forecasts for the upcoming year based on trends, correlations, and our visibility into the dynamic global threat landscape. Aligned with Elastic's broader commitment to transparency, we would like to spend some time reflecting on last year's forecasts.

Forecast 1

Defense Evasion is going to remain the top investment, and Tampering will supersede Masquerading

Verdict: We were correct.

Our analysis of this year's data revealed that the number-one tactic category for endpoint was *Defense Evasion*, and *Tampering* beat *Masquerading* by about 1%. *Defense Evasion* played a significant role for CSPs as well, though to a lesser degree. We also noted that *Defense Evasion* capabilities are increasingly common in both targeted and nontargeted malware, as well as being a clear priority for offensive security researchers. More broadly, enterprises should acknowledge two phenomena:

- Adversaries are both more aware of and more likely to rely on *Defense Evasion*, including tampering.

- Security instrumentation — particularly sensors with the ability to mitigate behavior — is effective enough that evasions are no longer optional but necessary.

The following Elastic Security Labs articles describe *Defense Evasion* identified in the wild:

- [GHOSTPULSE haunts victims with defense evasion bag 'o tricks](#)
- [GrimResource - Microsoft Management Console for initial access and evasion](#)
- [Dismantling Smart App Control](#)

Forecast 2

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

Verdict: We were mostly correct.

In particular, changes in the eCrime ecosystem have motivated threat groups to abstract themselves from intrusions and the government interest this produces. As a result, there's been an explosion of no- to low-experience threats running tools and playbooks as proxies. This lowers the barrier to entry to a degree, though enterprises should consider that proxies without the skills and adaptability of mature threats may be easier to

impact than those they are representing. However, it should also be noted that this dramatically interferes with attribution — and focusing the powers-that-be on crime-busting coalitions.

The following Elastic Security Labs articles describe malware families used in a MaaS context:

- [PIKABOT, I choose you!](#)
- [Globally Distributed Stealers](#)

Forecast 3

Adversaries will become more reliant on open source communities for implants, tools, and infrastructure

Verdict: We were correct.

In nearly every intrusion Elastic Security Labs observed this year, open source tools and capabilities played a central role. Network tunnelers and proxies, credential harvesting tools, *Privilege Escalation* capabilities, scripts, webshells — it is a very safe bet going forward, and we don't expect that to change. However, it's also not practical for enterprises to focus too much on monitoring public repositories for new kinds of tools or malware.

For most of the organizations impacted by these intrusions, environmental awareness and

control would have made the greatest difference: segmenting networks, monitoring processes running unbacked code, regulating access to sensitive systems, and monitoring for new Persistence mechanisms.

The following Elastic Security Labs articles mention open-source capabilities used in intrusions:

- [Unmasking a Financial Services Intrusion: REF0657](#)
- [Elastic catches the DPRK passing out KANDYKORN](#)

Forecast 4

Cloud credential exposure will be a primary source of data exposure incidents

Verdict: **We were correct.**

For every CSP we received data from, *Credential Access* was the number-one tactic category. As previously stated, this is a foregone conclusion because *Credential Access* is the gateway to all non-anonymous CSP access. With stolen valid credentials, adversaries can easily access services like email and cloud-hosted storage.

We observed this activity regularly and identified some key posture recommendations for next year.

The following Elastic Security Labs article describes several ways stolen credentials can be obtained for CSP compromise:

- [Protecting your devices from information theft](#)

Forecast 5

Excessively privileged Kubernetes pods will compound the damage of container vulnerabilities

Verdict: **We were incorrect.**

We anticipated a much clearer relationship between Kubernetes, excessive privileges, and existing unpatched vulnerabilities. Due to the use of valid credentials, exploitation was a much less common phenomena. It was also much harder

to distinguish, given that process space for many containers is consistent over the lifetime of the container while the user access groups may change monthly or even weekly. *Credential Access* issues are likely to continue.

Forecasts and Recommendations

Researchers reviewed the data in Elastic's global telemetry in an attempt to forecast what we might observe in the coming year. In each case, we try to associate a specific and actionable recommendation so that organizations who share our assessments can make decisions.

Forecast 1

Adversaries will triple-down on *Defense Evasion*, especially techniques that hinder sensor visibility

The most common *Defense Evasion* signals were seen on Windows systems, generally involving a trio of techniques: *Process Injection*, *System Binary Proxy Execution*, and *Impair Defenses*. Collectively, these techniques can be used to gain an initial foothold with sufficient privileges to tamper or blind instrumentation before data can be sent to SIEM-like backends.

Recommendation:

No one solution exists for this complex methodology, but a group of them have shown to be effective:

- Monitor unbacked code injected into privileged processes

- Monitor and restrict the use of built-in binary proxies (mshta, RunDLL, etc.)
- Monitor for changes in endpoint visibility; diagnostic rules are one option

Importantly, none of these can be sufficiently achieved without interactive endpoint agents deployed prior to the discovery of threat activity, which won't be effective if they're misconfigured. Researchers frequently observed enterprises where administrators failed to enable licensed mitigations, resulting in undesirable outcomes.

Forecast 2

Log deletion will continue to be a low-effort method of interfering with the visibility of container and server infrastructure

Many enterprises still rely on visibility-centric approaches to detecting threats, contrasted with the interactive capabilities previously recommended. For those organizations, loss of logs can be a challenging obstacle to overcome. We regularly observed log deletion in container and server environments where that was the primary or only source of threat data. Due to the latencies involved in collecting, analyzing, and responding to that dynamic, organizations were unable to control the environment effectively.

This forecast was observed consistently across enterprise systems and not limited to workstations, servers, containers, or storage systems.

Recommendation:

Ideally, organizations will prevent unauthorized persons from accessing systems with the ability to delete these necessary logs. Enterprises should be aware that each operating system exposes different capabilities for this, and some may be more robust than others.

Forecast 3

Exposed credentials will increasingly result in data exposure or unauthorized access, due largely to access brokers and the infostealer ecosystem

During several high-tempo intrusions this year, researchers observed that adversaries brought stolen credentials sourced from the environment. In the majority of those cases, the environment also contained evidence of prior infostealers or artifacts of backdoors. It can be very difficult to determine which credentials have been compromised after time has passed, though a good rule of thumb is to rotate the credentials of all system users when a system has been compromised.

Recommendation:

Rotate exposed account credentials, and invest in quick workflows that support breach response objectives like account resets. User and entity behavior analytics (UEBA) is one class of technologies that can help identify compromised accounts, and monitoring the accounts used in *Brute Force* attacks (significantly common for CSP targeting) can help in cases where evidence has rolled or been deleted.

Forecast 4

Permissive posture of CSP resources will contribute to future data exposure or harm

We observed that these posture settings were consistently misconfigured across all providers, so it's difficult to state a single root cause. In one form or another, users misconfigured the same capabilities of all CSPs:

- Permissive access policies allowed logins from anywhere
- Permissive storage policies allowed file operations from accounts of all kinds
- Relaxed data handling policies or encryption requirements

Enterprises balancing usability and the overhead of securing critical resources may struggle to prioritize an aggressive posture, or prioritize it consistently. In many cases, audits and guidance

are well understood and widely available at no cost. For this reason, it seems more likely to be cultural or rooted in perception than any specific technical issue.

Recommendation:

Consider using the CIS benchmark process to identify which settings need more attention in your environment. Once the CIS posture score reaches 100, make sure your team is well-versed in the most common cloud-based intrusion techniques. Monitoring from this baselined state should help improve the speed of threat detection while hardening the environment against future threats.

Forecast 5

The adoption and innovation of generative AI will result in new forms of telemetry collection and identify new in-the-wild threats

From authenticating reproduction works of art to analyzing the malicious properties of a ZIP archive, generative AI technologies are likely to have an enduring impact in how businesses operate. However, vulnerabilities in how these models are implemented may lead to data exposure or system exploitation, or poisoning — especially in ways

that may be challenging to discover. Adversaries might discover a new way to extract privileged medical information from a healthcare prompt, or instruct a hosted model to take a disruptive action, and are likely researching methods to do so. We don't know exactly what these new threats will be, but we do know that it'll be easier to tell with

more data — one reason Elastic is continuing to invest in the telemetry instrumentation of these models.

Recommendation:

FAIR-based risk models are one way enterprises can determine if a generative AI capability

increases or decreases organizational risk. Frameworks like the [OWASP Top 10 LLM](#) can also help organizations have informed risk-based conversations with LLM providers.

Forecast 6

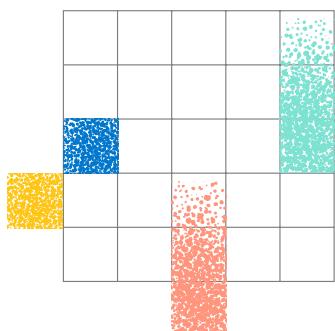
Cloud service providers will improve security default settings

The widespread adoption of cloud benchmarking based on security standards has highlighted a risk area: insecure default security controls implemented by cloud providers. When examining the most common threats to cloud-hosted platforms, these insecure defaults are an important factor in risk reduction.

One example is mandating MFA to neutralize the risk of credential theft. Another approach would be to segment storage access, restricting it to accounts with MFA enabled within specific groups, further limiting the potential impact of such threats. We anticipate cloud vendors will respond to these risks, implementing better standard controls for users as they adopt new capabilities.

Recommendation:

Ensure that benchmarking and risk assessments are part of a broader hardening and minimization strategy for cloud environments under management. Implement benchmarking frameworks, such as those provided by CIS, with a focus on enforcing least privilege. Users should advocate for more secure defaults related to authentication, data access, and configuration changes. Additionally, ensure that best practices are followed for each CSP environment under management, and take sufficient care to monitor for updates to default policies and standards.



Conclusion

Although it doesn't always feel like it, security efforts are making a difference. The tremendous attention paid to threats is proof enough that challenges are mounting for adversaries, and that's not a coincidence. However, mature threat actors are learning how to overcome obstacles — like leveraging inherent vulnerabilities in privileged device drivers for Windows to disable EDR sensors, injecting into privileged processes to delete critical security logs, or unloading security components to prevent security ingest from occurring.

For all our progress, we still have room to improve. This year we saw adversaries of all kinds leverage stolen credentials to gain *Initial Access*, facilitated by a massive marketplace of stolen data brokers. Enterprises need to work harder to constrain public-facing systems, enforce MFA, minimize their attack surface, and protect data needed to detect threats. For enterprises that are already paying for powerful mitigations, the most important step is enabling them. A detect-only approach doesn't work — you must prevent the things you can prevent. And let's not ignore the topic widely expected to make a big difference this year: AI. Artificial intelligence capabilities didn't transform the landscape for better or worse — it didn't lead to an explosion of new threats and it didn't create such

an advantage where all threats were eliminated. We think the ramifications of this technology class haven't yet been realized.

There are no guarantees in security, which is why security research remains a critical component for navigating the threat landscape. Attackers and defenders alike are pushing the boundaries of what's possible every day — understanding these dynamics is our mission at Elastic Security Labs. To remove the "occult" status from attackers' actions; to expose, contextualize, and mitigate. We hope that you'll join us in this mission. While overwhelming, the threat landscape is not insurmountable. Every action tilts the balance closer to our side, and we hope our efforts are seen in our commitment to democratizing knowledge, releasing powerful tools, and sharing Elastic's incredible visibility.

You can do this. And we're right here with you.

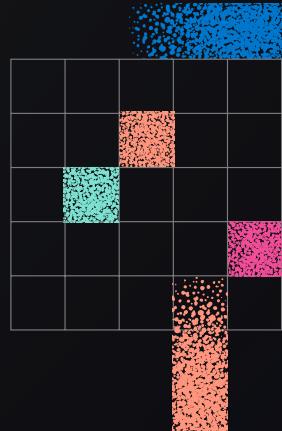
Learn about [Elastic Security](#) and protect against the threats covered in this report (and other vulnerabilities) by visiting [Elastic Security Labs](#). You can also [follow us on X](#) to see when we release breaking threat research.

The 2024 Elastic Global Threat Report features insights and expertise from across the Elastic organization. We'd like to thank the following Elasticians for their contributions:

- ◆ Mika Ayenson
- ◆ Samir Bousseaden
- ◆ Terrance DeJesus
- ◆ Chris Donaher
- ◆ Tinsae Erkailo
- ◆ Ayoub Faouzi
- ◆ Eric Forte
- ◆ Ruben Groenewoud
- ◆ Justin Ibarra
- ◆ Devon Kerr
- ◆ Jake King
- ◆ Shashank Suryanarayana
- ◆ Mark Mager
- ◆ Asuka Nakajima
- ◆ Andrew Pease
- ◆ John Uhlmann
- ◆ Alyssa VanNice
- ◆ Colson Wilhoit

Global Threat Report

2024



© 2024. Elasticsearch B.V. All Rights Reserved.

Elastic, Elasticsearch and other related marks are trademarks, logos or registered trademarks of Elasticsearch B.V. in the United States and other countries. Microsoft, Azure, Windows and other related marks are trademarks of the Microsoft group of companies. Amazon Web Services, AWS, and other related marks are trademarks of Amazon.com, Inc. or its affiliates. All other brand names, product names, or trademarks belong to their respective owners.