

UNIT 42
BY PALO ALTO NETWORKS

Threat Research Center > Threat Research > Malware

MALWARE

ChromeLoader: New Stubborn Malware Campaign

🕒 19 min read

RELATED PRODUCTS

Advanced WildFire

Cortex XDR

By: Nadav Barak

Published: July 12, 2022

Categories: Malware , Threat Research

Tags: Adware , Browser hijacker , Choziosi Loader , ChromeBack , ChromeLoader , Infostealer , Malvertising

Download

Print

Share

This post is also available in: [日本語 \(Japanese\)](#)

Executive Summary

In January 2022, a new browser hijacker/adware campaign named **ChromeLoader** (also known as Choziosi Loader and ChromeBack) was discovered. Despite using simple malicious advertisements, the malware became widespread, potentially leaking data from thousands of users and organizations.

Instead of more traditional malware like a Windows executable (.exe) or Dynamic Link Library (.dll), the malware authors used a browser extension as their final payload. The browser extension serves as adware and an infostealer, leaking all of the user’s search engine queries. We discovered significant changes and additions of capabilities throughout this campaign's evolution, and we predict further changes as this campaign continues.

In this article, we examine the technical details of this malware, focus on the evolution between its different versions and describe changes in its infection process. This article also reviews new variants that have not yet been publicly reported.

Palo Alto Networks customers using **Cortex XDR** and **WildFire** receive protections against this newly discovered malware out of the box.

Names for malware discussed	ChromeLoader, Choziosi Loader, ChromeBack
-----------------------------	---

Introduction to ChromeLoader Malware

ChromeLoader is a multi-stage malware family. Each variant contains different stages throughout its infection chain, but the infection chain often looks quite similar among the different variants, including malicious browser extensions used in all variants.

The different payload extensions we tracked had a hardcoded version added by the attacker. This labeling routine contributed to the research process, linking the different versions to the same campaign – and by their correct chronological order.

The various extension versions are related to different variants of this malware. We differentiate the variants not only by the related extension version but also by the techniques used throughout their infection chain and the targeted operating systems.

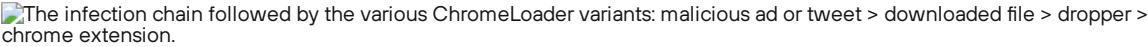
TABLE OF CONTENTS

Executive Summary
Introduction to ChromeLoader Malware
Infection Vector (Variant 1)
Deployment
Dropper Statistics
Payload
Infostealer and Adware
Version Control
Version 2.0 (First seen Jan. 4, 2022):
Version 3.0 (Jan. 6, 2022):
4.* versions (Jan. 7, 2022):
MacOS variant
Second Windows Variant (Variant 2)
The Real First Windows Variant – December 2021 (Variant 0)
Conclusion
Product Coverage
Indicators of Compromise
Variant 1 ISO hashes
Variant 1 executable hashes (aka CS_Installer.exe)
Variant 2 ISO hashes
Variant 2 executable hashes (aka Tone.exe)
Variant 0 executable hashes

Variant 1: Mentioned first (beginning in the “**Infection Vector**” section). It used versions 2.0–4.4 of the Chrome extension as its payload and a DotNet executable that launches obfuscated PowerShell as its dropper. It was mainly active in January.

Variant 2: Mentioned third (see the section “**Second Windows Variant**”). It uses the 6.0 version of the Chrome extension and uses an obfuscated executable as its initial dropper. It has been active since March.

MacOS Variant: Mentioned second (see the section “**MacOS Variant**”). This variant focuses on MacOS computers (while other variants target Windows users only). Uses the 6.0 version of the extension. Active since March.

The infection chain followed by the various ChromeLoader variants: malicious ad or tweet > downloaded file > dropper > chrome extension.

Additional Resources

RELATED ARTICLES

Jumpy Pisces Engages in Play Ransomware

Spoofed GlobalProtect Used to Deliver Unique WikiLoader Variant

Payload Trends in Malicious OneNote Samples



Figure 1. The infection chain of the different variants.

Infection Vector (Variant 1)

The first variant of ChromeLoader Malware (referred to in the Introduction as Variant 1) was first seen in January 2022.

The chain of events starts when a user is enticed to download a torrent or a cracked video game through malvertising campaigns on ad sites and social media platforms. The user scans a QR code on these web pages and is redirected to a compromised website that presents an ISO image (an optical disc image file, typically used with CD/DVD). The user downloads the ISO image, mounts it by double-clicking and executes content contained in the mounted ISO image.


The image shows what appears to be a QR code to download "The entire five nights at Freddy's."



Figure 2. An example of a QR code posted on Twitter.


The screenshot shows the "Five Nights at Freddy's" download, but the link to the QR code leads to a malicious ISO image as shown.



Figure 3. An example of a download link to the malicious ISO image from the QR code.

Deployment

The downloaded ISO image contains the following:

- **Microsoft.Win32.TaskScheduler.dll:** a legitimate .NET DLL signed by Microsoft, used by other .NET programs for integrating with the scheduled tasks mechanism.
- **Language folders:** contains a resource file used by the mentioned DLL.
- **CS_installer.exe and its config file:** malicious executable written by the malware authors (note that the name might change from one version to another). In some versions, the authors (probably accidentally) left its PDB file, containing its debug data, inside this folder as well
- **_meta.txt:** a text file found in advanced versions of this malware, containing **scrambled** ASCII letters.

Most files in this directory are hidden, and the ordinary user will not notice them when opening this directory using Windows File Explorer. The only non-hidden file is `CS_installer.exe`, which tempts the victim to double-click it to complete the software installation download.


Selecting "show hidden files" reveals a number of additional files in the mounted malicious ISO image – everything except `CS_installer.exe`



Figure 4. An example of a mounted malicious ISO image (after selecting "show hidden files").

The victim launches `CS_installer.exe` by double-clicking it. In most cases, the executable presents the message shown below in Figure 5, indicating that the program failed to execute. However, this is an attempt by the authors to mislead their targets.

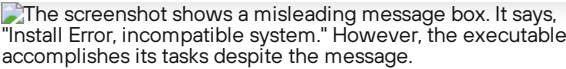
The screenshot shows a misleading message box. It says, "Install Error, incompatible system." However, the executable accomplishes its tasks despite the message.



Figure 5. Message box presented by the dropper, designed to deceive the user.

The executable is a non-obfuscated program written in .NET, so .NET reflectors can decompile it to read the source code. The code shown in Figure 6 is revealed when the executable is loaded into a reflector. This code creates a scheduled task configured to execute a malicious base64 encoded PowerShell command every ten minutes. The task name is constructed from the `Chrome` string concatenated with a random suffix from the `namesDict` array.

The decompiled CS_installer.exe source code shown is annotated to show where the code opens the message box, checks whether a malicious task already exists and creates a new malicious task named Chrome*



Figure 6. An example of decompiled CS_installer.exe source code.

The script content derives from the `_meta.txt` file, unscrambled by the following predefined function below in Figure 7, which applies simple character replacement.

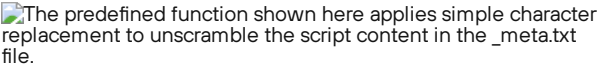
The predefined function shown here applies simple character replacement to unscramble the script content in the `_meta.txt` file.



Figure 7. An example of the unscramble function.

Some of the features mentioned were missing in earlier versions of this variant of the malware. For instance, in the version shown in Figure 8, which was discovered only one week before the version mentioned in Figures 6 and 7, the authors did not use a descramble function but simply hardcoded the encoded PowerShell script in the .NET executable and used the predefined `ChromeLoader` name for their task instead of generating a more randomized suffix.

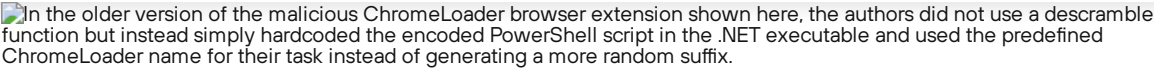
In the older version of the malicious ChromeLoader browser extension shown here, the authors did not use a descramble function but instead simply hardcoded the encoded PowerShell script in the .NET executable and used the predefined ChromeLoader name for their task instead of generating a more random suffix.



Figure 8. An example of source code from an older version of Variant 1.

The attacker uses the encoded PowerShell script for downloading and loading a malicious browser extension into the user’s Chrome browser.

Variable definitions in ChromeLoader’s PowerShell dropper include extPath, confPath, archiveName, taskName and domain.



Figure 9. An example of variable definition in the PowerShell dropper.

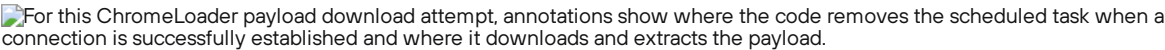
For this ChromeLoader payload download attempt, annotations show where the code removes the scheduled task when a connection is successfully established and where it downloads and extracts the payload.



Figure 10. An example of a payload download attempt.

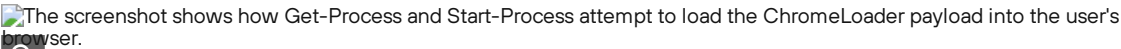
The screenshot shows how Get-Process and Start-Process attempt to load the ChromeLoader payload into the user’s browser.

Figure 11. An attempt to load the payload into the user’s browser.

Figure 11. An attempt to load the payload into the user’s browser.

The evolution from early versions of this malware to later ones is also seen in the encoded PowerShell script. Figure 12 shows PowerShell script executed by an earlier version of this variant, which is significantly shorter and contains less complicated code.


 PowerShell script executed by an earlier version of this variant, which is significantly shorter and contains less complicated code.



Figure 12. An example of an older version of this PowerShell dropper.

Dropper Statistics

ChromeLoader attacks on Palo Alto Networks Cortex XDR customers were blocked by our Behavioral Threat Protection module starting from the first day of this campaign. However, we were curious about the following stages of this attack. Consequently, we decided to continue our research, tracking down the attacker’s footprints and intentions.

The scheduled task is using the malware to download a malicious Chrome extension and installing it to the victim’s browser. The URL hosting the Chrome extension is hardcoded in the obfuscated PowerShell command and changes between the different versions.


 A list of Installation Servers (left) and the dates of the first connection attempt (right)



Figure 13. First infection attempt for installation servers.

ilsandothe 1.5%, yeconnected 3.1%, idwhitdoe 4.6%, rsonalrecom 3.1%, ithconsukultin 7.7%, etterismype 10.8%, yflexibilituky 38.5%, learnataloukt 12.3%, brokenna 15.4%



Figure 14. Installation server connection attempts distribution.

US 56, EU 8, CA 1 – blocked infections per region



Figure 15. Blocked infections per region.

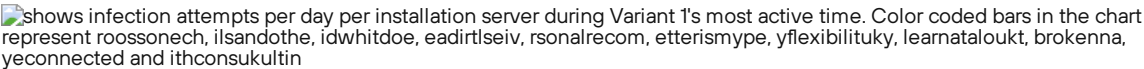
shows infection attempts per day per installation server during Variant 1's most active time. Color coded bars in the chart represent roosonech, ilsandothe, idwhitdoe, eadirtlseiv, rsonalrecom, etterismype, yflexibilituky, learnataloukt, brokenna, yeconnected and ithconsukultin



Figure 16. Infection attempts per day per installation server during the Variant 1's most active time.

Payload

The payload of the malware is a Chrome extension – every downloadable extension has the same format:

Figure 17. An example of the downloaded extension files.

Using some definitions in the `manifest` file, and using a known legitimate picture, the extension claims to be legitimate and harmless. However, the extension asks for elevated privileges. Requested privileges include accessing browser data, manipulating web requests and accessing every possible URL address, which legitimate browser extensions would not do.



Annotations call out where the downloaded extension pretends to be a legitimate extension named "settings" by changing its name, description and title, and also where is asks for high privilege permissions.



Figure 18. An example of a downloaded extension’s manifest file.

The Javascript file `conf.js` declares constant variables, which will use the main script `background.js` later. The C2 domain is stored in `_ExtDomNoSchema`.

The `conf.js` declares constant variables, which will use the main script `background.js` later.



Figure 19. An example of a downloaded extension’s `conf.js` file.

`background.js` is a one-line JavaScript file containing all of the extension’s functionality; it is heavily obfuscated but can be converted to readable JavaScript code in a short series of steps. However, any attempt to deobfuscate this code using known public JavaScript deobfuscation tools will fail due to reasons which will be detailed **later**.

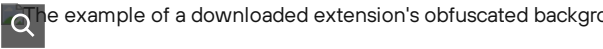
The example of a downloaded extension's obfuscated `background.js` file starts with `s0QQ.W3`

Figure 20. An example of a downloaded extension’s obfuscated `background.js` file.

This script uses various obfuscation techniques to hide its purpose and malicious code. One of the first functions executed is responsible for copying standard JavaScript functions and objects into new objects with scrambled names, which will later use the script for decoding the final payload, located in this script's last instructions.

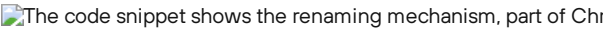
The code snippet shows the renaming mechanism, part of ChromeLoader's obfuscation techniques.



Figure 21. An example of the renaming mechanism. For instance, in this case, the String object is stored as `b4VV`.

During the entire execution of this script, the authors use switch-case-oriented programming to make their program harder for malware analysts to read and understand.


The malware authors use switch-case-oriented programming as shown to make their program harder for malware analysts to read and understand.



Figure 22. An example of switch-case-oriented programming.

The program loops using the `E3` variable shown above in Figure 22 and acts differently for each value. When the relevant flow in the switch case has ended, the program changes the value of `E3` to its next instruction. The program also uses the obfuscated object names mentioned previously. In Figure 22, we added the original object name in a comment below the relevant code line.

After understanding the obfuscated names and switch-case-oriented programming, we can better analyze the purpose of this code section. It uses a hardcoded four-sized array of integers, translating it to the associated ASCII characters and sorting it by randomized order. Later, this array will be joined to a string, and the program will search for a defined function in that name. The execution flow will start over if the function isn't found.

This stage reveals another obfuscation technique in the script. One of the key features used by standard deobfuscation tools is dropping unreferenced functions and objects. Often, it helps to shorten the code, leaving out complicated parts which will never actually run – removing functions whose whole purpose is to mislead a malware analyst. However, in this case, using deobfuscation tools drops an essential function, and the script will be stuck in an endless loop without it.

The function `h0QQ` is not directly referenced even once during the script execution. Yet, the previously mentioned code section using a randomized sort algorithm will eventually attempt to execute it, since `h0QQ` is a permutation of the `0hQQ` string. If `h0QQ` does not exist, the code simply tries to sort the characters and repeatedly looks for a function name.


An example of the unreferenced crucial function (`h0QQ`)



Figure 23. An example of the unreferenced crucial function.

This function returns a long scrambled string, XORed by a hardcoded key, and then splits into an array of strings.

An example of the malware usage of the deXORed array to decode its final payload. You can also see that the malware doesn't use integers as the array's indexes but strings combined with arithmetic operations.



Figure 25. An example of the malware usage of the deXORed array to decode its final payload. You can also see that the malware doesn't use integers as the array's indexes but strings combined with arithmetic operations.

We exported the mentioned list members after utilizing a debugger to execute the initialization code. Then we used a Python script to deobfuscate the remaining sections of the JavaScript code.

We used a Python script as shown to deobfuscate the remaining sections of the JavaScript code.



Figure 26. A script used for deobfuscating background.js.

Infostealer and Adware

For communicating with the malicious extension, the authors used command and control servers (C2s), which are different from the installation server used for installing the extension previously. The malware uses various extension features, giving it a strong foothold in the user's browser.

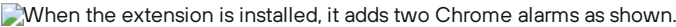
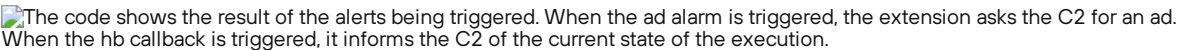
When the extension is installed, it adds two Chrome alarms as shown.



Figure 27. An example of alerts installed by the malware.

When the extension is installed, it adds two Chrome alarms (alarms allow the developer to install a callback / scheduled task that will be triggered periodically). When these alarms are triggered, two corresponding functions are being called:

- When the `ad` alarm is triggered, the extension asks the C2 for an advertisement and presents it in a new tab.
- The `hb` callback is triggering functions that communicate with the C2, informing it of the current state of the execution.

The code shows the result of the alerts being triggered. When the `ad` alarm is triggered, the extension asks the C2 for an ad. When the `hb` callback is triggered, it informs the C2 of the current state of the execution.

Bing. If it does, the extension will send the search details to the C2, leaking the victim’s thoughts and interests.

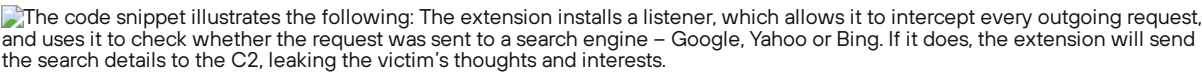
The code snippet illustrates the following: The extension installs a listener, which allows it to intercept every outgoing request, and uses it to check whether the request was sent to a search engine – Google, Yahoo or Bing. If it does, the extension will send the search details to the C2, leaking the victim’s thoughts and interests.



Figure 29. An example of browser hijacker capability.

In addition, the extension uses different mechanisms to verify that it executes properly. For example:

- A hard-coded header named `dd` for each outgoing packet to the C2. This could be used by the C2 to identify the different distribution channels/affiliates.

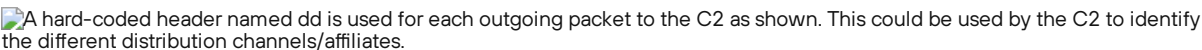
A hard-coded header named `dd` is used for each outgoing packet to the C2 as shown. This could be used by the C2 to identify the different distribution channels/affiliates.



Figure 30. An example of the added `dd` header.

- Cancelling search suggestions, probably in order to make sure that the search queries were intended by the user.
- Uninstalling existing Chrome extensions from the browser. It also sends the names of the extensions to the C2 and gets back an `allowlist.json`, in order to exclude chosen extensions from being removed.
- Disabling every attempt to access `chrome://extensions` and open `chrome://settings` instead to prevent the user from uninstalling this malicious extension.

Version Control

Most malicious extensions contained a file named `conf.js` alongside the main Javascript code stored at `background.json`. This `conf.js` (or `manifest.json`, or `background.js` file if `conf.json` is missing) file stores relevant configuration for the extension: the C2’s hostname (e.g., `krestinaful[.]com` and `tobepartou[.]com`), the verification header’s value of `dd`, the extension name, and its version. It seems like the version information is accurate – there are several differences between the versions we saw (2.0, 3.0, and 4(0.3,4)). (For our observations on the relationship between variants and versions, please see the **“Introduction”** section.)

Version 2.0 (First seen Jan. 4, 2022):

Missing functionality:

- No advertisements for victims.
- Search engine query gathering from Google only.
- No deletion of existing browser extensions.

Version 3.0 (Jan. 6, 2022):

Added functionality:

- Search engine queries are now gathered from Yahoo and Bing as well.
- `SetWithExpiry()` and `GetWithExpiry()` functions added and used for storing variables (i.e., query URL) and deleting existing extensions, respectively.
- Existing extensions deletion mechanism.

- Chrome advertising mechanism added.
- Changed the hardcoded C2 URL.
- Chrome alert mechanism.

MacOS variant

In March 2022, a new variant emerged targeting MacOS users. This variant remains active and uses similar techniques to install its payload and hide its actions. It uses the same infection method of directing victims to compromised pay-per-download websites to install its dropper.

In this case, the dropper is a disk image (DMG) file – the MacOS implementation for ISO files – containing several files, including one bash script. The bash script resembles the scheduled PowerShell script in multiple manners:

- Downloads the payload – a browser extension from a remote installation server.
- Loads the payload into the target’s browsers – Google Chrome and the built-in Safari browser.

An example of an early version of the MacOS installation script for ChromeLoader.



Figure 31. An example of an early version of the MacOS installation script.

In more advanced cases, instead of hardcoding the download execute portion in the bash script, the authors encoded these commands in a separate file, then decoded and executed by the bash script using OpenSSL.

An example of a later MacOS installation script for ChromeLoader.



Figure 32. An example of a later MacOS installation script.

The downloaded extension functions were similar to those used in the Windows OS versions. The MacOS variant uses the same obfuscation method to execute the same vital components – gather search engine queries and present advertisements. In addition, new C2 addresses were used in this version.

Based on the version number of the malicious extensions delivered by this variant, the attackers reference the MacOS variant as later than the Windows variants, which fits the timeline of infections in this campaign. In our research, the extensions found with this variant were labeled as the 6.0 version of this malware.

Second Windows Variant (Variant 2)

In March 2022, several weeks after the last known infection of Variant 1, we identified a new campaign with multiple similarities to the first one, which makes us believe that we are actually facing another variant of the same ChromeLoader malware, referred to in this blog as Variant 2.

The infection vector for this Variant 2 is identical to Variant 1. Users are enticed to download a torrent or cracked video game through malvertising campaigns on pay-per-install sites and social media.

ISO images used for Variant 2 contain new executables. Victims would only see a Windows shortcut, which they would double-click to install the desired software or watch the movie.

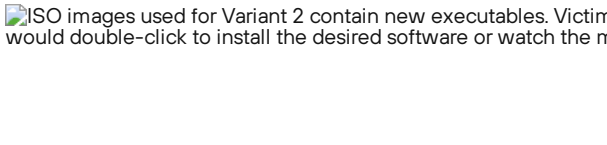
ISO images used for Variant 2 contain new executables. Victims would only see a Windows shortcut as shown, which they would double-click to install the desired software or watch the movie.



Figure 33. An example of a malicious mounted ISO.

However, the ISO image contains other hidden files executed when the victim launches the Windows shortcut (.lnk file). The

The .lnk file runs a batch script named resources.bat as shown.



Figure 34. An example of the LNK file configuration.

An example of resources.bat content. Note the presence of Tone.exe



Figure 35. An example of resources.bat content.

Like Variant 1, Variant 2 installed the same type of Chrome extension. The malware launched a `cmd.exe` process, which in turn executed `powershell.exe`. The PowerShell process executed WMI queries, used for installing a new scheduled task named `chrome *`, launching another encoded PowerShell command.

An example of a causality chain when the malware installs a scheduled task.



Figure 36. An example of a causality chain when the malware installs a scheduled task.

When analyzing the above-mentioned obfuscated PowerShell script, we were faced with a script used as a dropper. This script doesn't directly install a new Chrome extension, so it does not exactly match Variant 1's PowerShell script pattern. However, the structure and use of variables resembles the behavior of Variant 1.

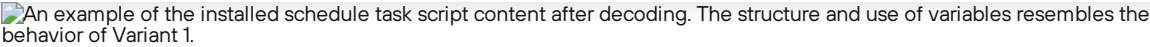
An example of the installed schedule task script content after decoding. The structure and use of variables resembles the behavior of Variant 1.



Figure 37. An example of the installed schedule task script content after decoding.

Using XQL queries, when the installation server is available, the PowerShell script creates and loads the familiar malicious Chrome extension (the 6.0 version, used in the latest MacOS variant).

An example of the files downloaded by the encoded PowerShell. These include manifest.json and background.js



Figure 38. An example of the files downloaded by the encoded PowerShell.

The Real First Windows Variant – December 2021 (Variant 0)

Due to its **multiple infection incidents**, this malware family has drawn worldwide attention in the cybersecurity community.

As mentioned earlier, we detected different versions of this malware during our investigation. Each version was labeled not only by us but also by the malware authors themselves. The earliest labeled version we detected was 2.0. Therefore, we were confident that this wasn’t the first time these attackers struck, and we were determined to expose the actual first version of this malware.

Due to the attackers’ history of frequent payload updates, we were convinced that the first infection case occurred relatively close to the currently reported infection case in January 2022.

Pivoting over the installation server domains used for the Variant 1 PowerShell dropper revealed that another piece of malware used some of these domains as its installation servers in December 2021.

This malware was an executable file written using AutoHotKey (AHK) – a framework used for scripting automation.

Using this tool, the programmer can write short, easy-to-understand scripts using the AHK syntax. Then, by the programmer’s definitions, the framework creates matching hooks that will cause the execution of these scripts.

When transforming AHK scripts into Windows executables, the original script source code is pasted into the end of the executable, making the investigation process for the researcher much more effortless compared to the other variants, which used heavy obfuscation. In this case, the hardcoded script contained the following source code, which looks quite similar to the PowerShell droppers we already analyzed:

An example of the AutoHotKey script content. Of note, it looks quite similar to the PowerShell droppers we already analyzed.

In short, this dropper downloads a payload from its installation server. We can assume that this payload is another browser extension by the variable name used for the downloaded payload (`Extension_Name`).

After a more thorough investigation, we found the downloaded extension. Unsurprisingly, it also contained features related to the ChromeLoader malware family – but more importantly, it was labeled version 1.0 (!)

These extensions were quite similar to the rest of the extensions related to this family, with one main difference – this time, the extension was not obfuscated. It even contained some of the author’s comments regarding different code sections.

An example of the extension downloaded by Variant 0 of ChromeLoader, without any changes from our side.



Figure 40. An example of the extension downloaded by this variant, without any changes from our side.

Conclusion

This blog documents different examples of a new malware family, ChromeLoader, spread using malicious advertisements. This malware demonstrates how determined cybercriminals and malware authors can be: In a short time period, the authors of ChromeLoader released multiple different code versions, used multiple programming frameworks, enhanced features, advanced obfuscators, fixed issues, and even adding cross-OS support targeting both Windows and MacOS.

This malware is used for hijacking victims’ browser searches and presenting advertisements – two actions that do not cause serious damage or leak highly sensitive data. However, based on the wide distribution the attackers gained in such a short time, they were able to inflict heavier damage than the damage inflicted by the two primary functions of the Chrome Extension.

Additionally, the authors were quite organized, labeling their different malware versions and using similar techniques throughout their attack routines. This probably made their lives easier while developing their attack framework and maintaining their attack chains, but unintentionally, this also made the investigation process significantly easier. In fact, it improved the research ability so much that we were able to detect two new versions of this malware – the first one and the latest, which have never been linked to this malware family before.

Finally, this attack chain demonstrates two rising trends among malware authors that security products and even common users should be aware of – the use of ISO (and DMG) files and the use of browser extensions.

Product Coverage

Palo Alto Networks customers using Cortex XDR Prevent or and Pro receive protections from such campaigns in different layers, including the Local Analysis Machine Learning module, Behavioral Threat Protection, BIOC and Analytics BIOC’s rules that identify the tactics and techniques that ChromeLoader uses at different stages of its execution.

Most rules are not customized for ChromeLoader and are based on unusual, rare behaviors – and therefore provide protection against many additional malware families and campaigns that use the same methods.

The following rules provide behavioral detections and preventions that block this malware at different stages for Cortex XDR customers:

Rule Name	Description
Power Empire - 2280642765	Power Empire post-exploitation framework
PowerShell Activity - 83290630	Suspicious PowerShell activity
PowerShell Activity - 1683698903	Suspicious PowerShell activity

Suspicious Scheduled Task Installed - 161058768	Potential malware granted persistency via scheduled task
Suspicious File Dropped - 1664970582	Potential malware dropped a suspicious payload executable
Suspicious File Dropped - 1833473256	Potential malware dropped a suspicious payload executable
Suspicious Chromium Extension - 4043645859	Potential malware tries to load malicious extension to victim's browser
Staged Malware Activity - 2903131508	Activity similar to ChromeLoader malware
Staged Malware Activity - 4059467241	Activity similar to ChromeLoader malware

In addition, you can use the following XQL queries to detect ChromeLoader variants during their different execution stages.

Variant 1 (January) – Installer

```
dataset = xdr_data
2 | filter event_type = ENUM_RPC_CALL
3 | filter actor_process_signature_status = ENUM_UNSIGNED
4 | filter action_rpc_interface_uuid = "(86D35949-83C9-4044-B424-DB363231FD0C)" and action_rpc_func_opnum = 1
5 | filter lowercase(action_rpc_func_str call fields) contains "chrome"
```

Variant 2 (March) – Scheduled Task Installer

```
1 dataset = xdr_data
2 | filter event_type = ENUM.RPC_CALL
3 | filter actor_process_image_name = "powershell.exe"
4 | filter action_rpc_interface_uuid = "{956DC99-828C-11CF-A37E-00AA003240C7}" and action_rpc_func_opnum = 25
5 | filter lowercase(action_rpc_func_str_call_fields) contains "powershell -windowstyle hidden -e" and action_rpc_func_str_call_fields contains "PS ScheduledTask"
```

Variant 2 (March) – Tone.exe Extraction

```
1 dataset = xdr_data
2 | filter action_process_image_name == "tar.exe"
3 | filter action_process_image_command_line contains "-xvf" and action_process_image_command_line contains "-C" and action_process_image_command_line contains "AppData\Roaming"
```

MacOS Variant – Extension Download Encrypted

```
1 dataset = xdr_data
2 | filter agent_os_type = ENUM.AGENT_OS_MAC
3 | filter event_type = ENUM.PROCESS
4 | filter action_process_image_command_line contains "sh -c echo* | base64 --decode | bash"
5 | dedup agent_hostname , action_process_image_command_line
6 | fields time , agent_hostname , action_process_image_command_line , actor_process_command_line
```

Suspicious Browser Extension Loaded

```
1 dataset = xdr_data
2 | filter action_process_image_name in ("chrome.exe", "safari", "chrome")
3 | filter action_process_image_command_line contains "-load-extension="
4 | filter actor_process_image_name in ("powershell.exe", "bash", "sh")
```

Palo Alto Networks customers using WildFire also receive protections from this threat.

Indicators of Compromise

Variant 1 ISO hashes

```
fa52844b5b7fcc0192d0822d0099ea52ed1497134a45a2f06670751ef5b33cd3
e1f9968481083fc826401f775a3fe2b5aa40644b797211f235f2adbcb0a0782f
860c1f6f3393014fd84bd29359b4200027274eb6d97ee1a49b61e038d3336372
0ecbe333ec31a169e3bce6e9f68b310e505dedfed50fe681cfd6a6a26d1f7f41
614e2c3540cc6b410445c316d2e35f20759dd091f2f878ddf09eda6ab449f7aa
2e006a8e9f697d8075ba68ab5c793670145ea56028c488f1a00b29738593edfb
bcc6cfc82a1dc277be84f28a3b3bb037aa9ef8be4d5695fcbfb24a1033174947
6d89c1cd593c2df03cdbc7cf3f58e2106fff210eeb6f60d5a4bf3b970989dee2e
edeec82c65adf5c44b52fbdcb4b7ff754c6bd391653bba1e0844f0cab906a5baf
6c54e1ea9c54e4d8ada1d15fcdcbf53e4ee7e4a677d33c0ea91f6203e02140788
a9670d746610c3be342728ff3ba8d8e0680b5ac40f4ae6e292a9a616a1b643c8
fb9cce7a3fed63c0722f8171e8167a5e7220d6f8d89456854c239976ce7bb5d6
1717de403bb77e49be41edfc398864cfa3e351d9843afc3d41a47e5d0172ca79
1b4786ecc9b34f30359b28f0f89c0af029c7efc04e52832ae8c1334ddd2b631e
486c966b6e2d24dd8373181faf565d85abfd39559d334765f5135e20af55542c
03b2f267de27dae24de14e2c258a18e6c6d11581e6caee3a6df2b7f42947d898
dd2da35d1b94513f124e8b27caff10a98e6318c553da7f50206b0bfdded3b52c9
```

Variant 1 executable hashes (aka CS_Installer.exe)

ded20df574b843aaa3c8e977c2040e1498ae17c12924a19868df5b12dee6dfdd
1dbe5c2fecae1706fafc6f767cc16427a2237ab05d95f94b84c287421ec97c224
9eca0cd45c00182736467ae18da21162d0715bd3d53b8df8d92a74a76a89c4a0
c56139ea4ccc766687b743ca7e2baa27b9c4c14940f63c7568fc064959214307
3b5a18d45ab6fcf85df51703ef6fac8226fc274ecd0a21c0a1f15f15f7d39e01
44464fb09d7b4242249bb159446b4cf4c884d3dd7a433a72184cdbdc2a83f5e5
2d4454d610ae48bf9ffbb7bafcf80140a286898a7ffda39113da1820575a892f
53347d3121764469e186d2fb243f5c33b1d768bf612cc923174cd54979314dd3
afc8a5f5f8016a5ce30e1d447c156bc9af5f438b7126203cd59d6b1621756d90
564e913a22cf90ede114c94db8a62457a86bc408bc834fa0e12e85146110c89b

Variant 2 ISO hashes

e72a42ad27c06ba0a9951705423a3650a0c4a1f8c18c5782ab98e2e72021bbb8
26bce62ea1456b3de70d7ac328f4ccc57fe213babce9e604d8919adf09342876
44f9680710ba7635bb3bfe025b087e85d51857d9618c5ffa5c247ccdc8bca3c3
5ee2b7ea46cc3f34b796ab4992e778938c057490695e9109f016fc7a1b308395
a0ff3b427c77594fa48d79ed52d372bd2a8baae54ee85b243d86d9dd493ffbc6
f3176bcd28b89e4ae7a4426c82c8b73ca22c62ecbc363296193c8f5becef973c
424347b6f5caca8174dlb0ac2e32867a4201a41176fed1af7b3e1a0716fc7e46
c67b87cb7420500e4b0bb6500f1875bc77a7d96997ed2850d8142dfd9636da29
8f2da6c721251edd251addb795552ed54d89fb53d2a470d8a7f807e77aac402c
e0d57152524e79a07e5b7d7b37831cb7596cd3afe651b4eecaf4123b1af1ffa6
606d49ae054e13461bad3e405cc5996462c14bd48e94fe8a63f923fbb7c14b71
7ef7bdf8ea2f8751f45482453bf7441d2b2f92d743324afdf1afc11ea248c56d
84c93f1f7bdc44e8e92be10bf5e566f3116c9962c35262643fe2084c3b8d1bb5
4673c1f8d307b70c4be837e842cfd5cce60c6bf793ae85a1bce07c9c15fe14d
0257dccfdeb1bc9683334d0d964c72ea0eedbfda33cba1f60a395cca8e516da
0d510dbcf8ed5c7b81206598886a7fbd86f11d36871612ba066d6ec85723fada
e920dbc4741114f747a631928e398ef671fe9133b6aab33991d18150b4fcd745
3d65f5a060f8ecc92de9f5e0754b8f6c129cb9a243bf1504a92143ac3bc5a197
11174dbaca376288fd59c66d1c00255ad6c034beff96a075e833897ef3a113cc
44e77ac27a8b7d9227d95feb87bad1cc2a4ed2172c85f5e16d335a4d62d385f4

Variant 2 executable hashes (aka Tone . exe)

00c07e354014c3fb21d932627c2d7f77bf9b4aeb9be6efb026afdbd0368c4b29
3c7acdce8a37e40672eb4fba092804f9e783f284e7d52cbcf8a9f9f3cf306af7
5fbf4d8d44b2e26450c1dd927c92b93f77550cebfbcb267c80ff9d224c5318b88
1bb6f2a9498a220ade34b64f3208287fca6699847a5fd61e0e5ed4ee56b19316
4e5001c698f9f1758874067c5fb6fb2911e1f948db2cc0f289d42c61f2e2fec1
747ba8be14e4d465f79a8211a26204230719ce19293725ca139f4386e57a7dff
fcc92f1736b5b4bd9fe503e7d6debeb7e69858fc582783c3f35e7cdece9d4feb
0b00a215a42739809a55f05b6028399843e305fb285028de6efc5544b949a1ef
66ababb8bd9f8b19193f56678568197350be6306f448ee9a01eeee21a487f765
ce129e2e14fb0de7bd0af27a8303686bde1c330c05449c1fff95591f364189e33
1a01be5f08943ce03811f398f7b77aba26313dc0d0681cfad89f37db59819bc2
c93fbf63d82b816cd32dfc7bb0eaf7053fb27cfb78433638248010e83636ae20
7f9d31d382cef81bf858b8e848897b41397c033ad5aa5c416277cf843d7218f5
6c87e496ba0595ac161be8abb4e6da359d5d44c7e5afbe7de8fd689e4bb88249
d3212f79f33c8ccf6ba27984ed18acc86ec2297fe9c3df8fad5a00878986f2e2
329e7494d516652e64c1181979fdf53b507b4a3ab23b4821823f0aef96abc6a4
b73becdb7ad8b130072622ac7b2f03d450d7d0f9aae28e67dcb6724e5727f96c
10bd1b5144d9a2582aaecd28eb0b80366a2675d0fd8a2f62407f8c108d367ec7
11ad9d3e25bee2275f4930818bd737df1e1d79b334f990970c61763078c532d0
061408f4e1f37feb0b89db3cafc496194941fade412c96ee03fc46e492df3d29
8bdaf2a1e5400df06ce4d47b5b302b20cfb62e662e778a657485c6599865e393

Variant 0 executable hashes

01c2516c2276c001f566500026610211c7577c000010c1d07801700651610c0c47

8ef4026b254dd0918bf3ace7741b26ff52a52ef024c721d8129c5ccfa4ccde24d2b1b9642884a6839f09204135944c02c7437f7e692d07bb0d0269c4ff8316bbd8d18baa934a4f1ad6777f2ca862be8d3b3a59a1fedb8d2a8e50f0a419793a15e4ab0e5ecbd6c87432f08398b7f7424a248f98ff780e0adb710edd0698bf543445510bf70bc9063392ac0514f4e26431b9c38631ed0e61b6847fe9385f5eb17ce4ab0e5ecbd6c87432f08398b7f7424a248f98ff780e0adb710edd0698bf5434f3727e372949d12ce9f214b0615c9d896dcf2ac0e09fcd40f4a85ff601ef01f0

MacOS DMG hashes

965a6729b89f432f61b65a7addbe376317e8fd4a188c05c6aae7f9e4a1a88fbb6f105daec2336658629042afa4f334f4949fc189404f66c09400fd2ca260eb0c267ab450a5965a525bda34deccd64bf22b5fb6cc04d811a3eec1d9289e28bc73a6c8cbbe502df8407861590b97e634f51b85e4fe176bf68f86f6088ce81baaac6845a4b37e51fbf01a9573330c81483d5a438dbb1c87cbe069f72896927b4dabfad5e680c181fd7415e8c03ee20735411d1259f4ae19ead0100f0929d48f3f5340232e0ffdb8fe925f9d4a1f10d5aeda208bb58d82390ac7d1952f9219770103fd9a89dc83d26994708a1d9661322df12d107693d4b483a89bf9b03c974f418cb65dc44a3288b1718657d2197b1e0b22aa97d0e33b05e2877320e838da0ccb262b24417ea8cb3271636e1747be0cc205af4bdc0d31686f024693259afdca259edffdad0ced320b9934019a75658b16cf8f6abb2e4af48cb73f66a761dfe723920c1700551ca47143590722ae60204f1a597040d5fa6afa966d4fc3c42d82d517060c0b17a2d6fc7fb3a7a866c2013891527f1cf4602c420bc186d55b1802e3821286ff043574dfffb0c0a677b102272d7ea858030dc48d6c50534dba19d95adb61adc521a448a3588c892c98e00c9e58ba30a453b0795286b79ff2f0eaf821d2590acb46c7964404cf22b7faad5910dfa97ae8d49b45808bd9f98bb61b7bc878ff0da9bf1fc8da212ae1bcb10339539f5127e62aae0ad5809c2ae855921d2ab96c0e50646addd20136befa520380e4d0f8915c0e0808fd8d393a386f5af87e6232612ee5c099d6115dcbed7247cc56838fdeeb2654ba365b1b00d6294e6981f228ea53e242e05e5da560ac9a4c286f707e888784d9c64c43ae307d78b296d258aa660f95f4649f7c1c4a48e1da45a622f3751ee826511167f3de726e2a03df05c

Zipped extension hashes

6c1f93e3e7d0af854a5da797273cb77c0121223485543c609c908052455f045d92dc59664ab3427fb4b0d2d4108f1729abb506a2567770f7c4406e64db9aafae79114e6392bb8fffee76738e71f47131b0a2c843efe3e14f1b5e6a6d2a94c1046667f5bb50318fe13ea11227f5e099ab4e21889d53478a8ee1677b0f105bdc70a34d21f3a543a69f34973c25bbaaedb5c8bc797d63da493cbac97bfbbedbe7206a950e93ab9b2c4d1771a52fbeb62a9f2f47dc20e9921b9d23d829b949ba187b548efaa1fdb9810705945c15e80939b0f8fe3e5646b4d4ebcace0c049d1a677896c1af2e5cf6d6ea68c7e017d279b432d5259358b81ea1c444dc20625805b95b90f5fb924eb5eb646ba6789db665545a08c0438e99e5a24f27c37bc0279b1a8a6a1005c22c2305781fbbce5552dcc095f9ef0237023d7041eace005542fcd3d817f2cd9ad91ddab408619d3c80eef614b91a727c35285ebd813bcd1636b2cb0307e3d97c3802cc8bc9524480170d78aa68a9de28e3a7f4ce35d103f77843a3d0cf940e948586d3148e28df3e35e5671e87bc7c49525606068ac6f00783409d7aa63c97409bb2a8b5026b459fff6c6dcc93dd12fdd8c0a4915e9298bd96dfdedb5c3b4c3c598b87a3c3b9590940b4e67861c6541316bac1e1c07a139b1892307c04a113128466145973de141c4e5c5199e5474050edd4d9225463d0527d68935ef0ef633a38fb49a81a30fe8977dff378bb9e89f849ceceb709cbcf76272f92c402cc01324cbefb6d79e3a7ea1031edb6256fb3d40832ea621913aadda70e08a3b93271eac4d9d20044a5fc27be6d0feece31791f3889dce2788f7ef4e201ffff4e8e74b6d667d7ddb7859687fd5c599f67b62b491087d1d926037effc7f7890b434556d3c5e6a3322fcb39da3ef5b36d541bab70fa2f68a12e52c3de41bef092a6181a15d583d1ba4ad42b09ab62f3ef401c8cc2103e7ea2717d0571864f5440fd a950e93ab9b2c4d1771a52fbeb62a9f2f47dc20e9921b9d23d829b949ba187b5308071d4e8298b4eba9f82ca7269ac58f8e39f64da515c0761406aacd110b731ddb1793220d75c7126eb8af9f0d35f22e7be6998bf8ede8199c2019119b265925b7dedcf0802547c8e18d46fbfe1a5daa91e77a6cf464c4b5f0cfc48fa235c1db8b8f57edbd70345e2134abd8917371a29e04aa37210b553879710f717b69ddd6b1db4f891aa9033b615978a3fcfef02f1904f4eba984ba756ff5cd755d6f0b4099c2d8c3c34a24f6ed3cbf5c4fff6b22312546f2c3881281b7cc66ebff899136786111125f095f687f56171141041651160001011475667700152115043111

309c87b34966daecd05c48b787c3094eed85b5f23ec93b20fc9cdbf8ff9b586
47c65ef4d6b0ffe7109c588e04575dcf05fdf3afe5796078b4f335cb94c438b7
502a8d1e95c21b5dc283ef4877ca2fe2ba41570bd813c47527fca2fb224d5380
5e6b5a9c0849db8ca0696a16c882d6945a62e419bd646f23d4d00533bbe9bca5
6e0cb7518874437bac717ba1888991cee48dfaca4c80a4cbbbe013a5fe7b01a6
83cf9d2244fa1fa2a35aee07093419ecc4c484bb398482eec061bcbfbf1f7fea
87f0416410ac5da6fd865c3398c3d9012e5488583b39edacd37f89bc9469d6a9
c6a68fac895c0b15d5cbbba63f208e5b0a6f3c1d2382b9465375d1794f447ac5
c7aedc8895e0b306c3a287995e071d7ff2aa09b6dac42b1f8e23a8f93eee8c7a
d374ef30aa17f8bad0fb88d0da47f4038669c340d4c7fc2ff6505b07c17fdf65
dfc90f64139b050cf3c72d833e1a7915af1bd689ece7222b9ac2c8426a0bfd0a
9a5be852afef127b5cbe3af23ef49055677b07bcaca1735cf4ad0ff1e8295ccb
7ba5e623ad2e09896f0e1d1167758bcf22a9092e4a65856f825a2b8740e748f6
edb21b3f6f52ab0d0e17aca7e658a6e3f9ce98002433810612562b8e6ab41920
0cf40fbce8a48bfc5068ac24ec1dd1f828af31fe3cff0342003d12b0ea561dcf
4a0ababa34024691dc1a9e6b050fe1e5629220af09875998917b1a79af4e2244
52c7bb3efafdd8f16af3f75ca7e6308b96e19ef462d5d4083297da1717db8b07
bcac3fee6182a64764e88b4ed4f78cc071f297c501746df6473b0e9e679b3b43
aa9b742267bba71507a644ea4ee52a0f118ee6d595bd7eac816a8e8ee0246427
55f240467cf2c0891484d97ded9e0c53b259a88814b6f1c78a8961bda58c9377
49006f7529453966d6796040bb1c0ab2d53a1337c039afe32aaa14a8cce4bf0e
08de8a1103ccd7980a9900e2ceccdef0fe4db6bd06184eb628bfbcf76a7ff997
2eb1056cc176747c1be4b115be90cc7ee26da11a597cff6631da54c517d1a15c
436dde0fb44f95371832a55e56ed9ee9cb22f5323ce0d2a4cdcd61cbab713503
c05dbec1aaa11703195c743433a4319d49180c7fbd9a962e162cacd6b605ddd9
b919fbd354654a7bf99db7206adf6a5fba9ce73ee3fedb6d08ed932ee527f301
bfead4ccc3c16dee5f205b78e12aaaa2b33bdedbc57e22a4dbc48724f13f6277
eddd3ce6d39909be6fd5a093c2798a0c9113769b8f0f24a038449b409232472a
22f4a87053769ae21efa8945a83e46df2f56e8f01a66f156cacf5ef6b6a8262a
a3631d6012b72a63b0f1b4a013d0971ea8505ee3db32d4a0b7b31cb9ba8dd309
1ad535854fe536fd17aa56ae82f74872d6fad18545e19950afa3863bcbcf34eb
9d46a0509291bf3365771f6ad53e213fffb58e4926f11365687f4a11fd0f03855

C2 and Installation Servers

Installation Servers

brokenna[.]work
etterismype[.]co
idwhitdoe[.]work
ithconsukultin[.]com
learnataloukt[.]xyz
rsonalrecom[.]co
yflexibilituky[.]co
yeconnected[.]com

ableawid[.]com
airplanegoobly[.]com
baganmalan[.]com
balljoobly[.]com
balokyalokd[.]com
boogilooki[.]com
bookimooki[.]com
carfunusme[.]com
carmoobly[.]com
chairtookli[.]com
chookiebooki[.]com
choopinookie[.]com
ckgrounda[.]com
computermookili[.]com
dubifunme[.]com
dudesurfbeachfun[.]com

madorjabl[.]com
malanbagam[.]com
mokkilooki[.]com
myeducatio[.]com
nakasulba[.]com
ndinterper[.]com
ndworldwi[.]com
nookiespooti[.]com
oempafnyfi[.]com
saveifmad[.]com
siwoulukdli[.]com
slootni[.]com
sonalskills[.]com
tabletoobly[.]com
toogimoogi[.]com
toukfarep[.]com
uiremukent[.]com
ukrawinrusyes[.]com
utfeablea[.]com
voobmijump[.]com
xoomitsleep[.]com
yalfnbagan[.]com
yalokmalos2[.]com
yescoolservermate[.]com
yourretyeq[.]com

tcaukthw[.]com
tooblycars[.]com
koooblycar[.]com
rooblimyooki[.]com
yooblygoobnku[.]com
playkooblni[.]com
rockslootni[.]com
muendakere[.]xyz
mpleeysihigh[.]xyz
adiingsinsp[.]xyz

ajorinryeso[.]xyz
ktyouexpec[.]xyz
learnataloukt[.]xyz
ngwitheaam[.]xyz
ptonnervent[.]xyz
ukmlastttype[.]xyz
ukseseem[.]xyz
withyourret[.]xyz

C2s

betasymbolic[.]com
krestinaful[.]com
tobepartou[.]com
tobedirectuke[.]com
eandworldw[.]com
etobepartou[.]com
kfareputfeabl[.]com
blesasmetot[.]com
siwoulukdlik[.]com
sforourcompa[.]com

Additional Resources

Back to top

TAGS

- Adware
- Browser hijacker
- Choziosi Loader
- ChromeBack
- ChromeLoader
- Infostealer
- Malvertising

<

Threat Research Center

Next: When Pentest Tools Go Brutal: Red-Teaming Tool Being Abused by Malicious Actors

>

Related Malware Resources

C THREAT RESEARCH

IC

November 1, 2024

TA Phone Home: EDR Evasion Testing Reveals Extortion Actor's Toolkit

Extortion

Data exfiltration

Read now →

C THREAT RESEARCH

IC

October 9, 2024

Contagious Interview: DPRK Threat Actors Lure Tech Industry Job Seekers to Inst...

North Korea

Social engineering

Python

C THREAT RESEARCH

IC

October 1, 2024

Detecting Vulnerability Scanning Traffic From Underground Tools Using...

Machine Learning

Read now →

C THREAT ACTOR GROUPS

IC

September 26, 2024

Unraveling Sparkling Pisces's Tool Set: KLogEXE and FPSpy

MITRE

Keylogger

North Korea

Read now →

UNIT 42

Get updates from Unit 42

Small Logo

Peace of mind comes from staying ahead of threats. Contact us today.

Your Email

Subscribe

Risk Awareness

Subscribe for email updates to all Unit 42 threat research.

By submitting this form, you agree to our [Terms of Use](#) and acknowledge our [Privacy Statement](#).

CLOUD DELIVERED SECURITY SERVICES	Prisma Cloud	Careers	Communities
Advanced Threat Prevention	Cloud-Native Application Protection Platform	Contact Us	Content Library
DNS Security		Corporate Responsibility	Cyberpedia
Data Loss Prevention		Customers	Event Center
IoT Security		Investor Relations	Manage Email Preferences
		Location	Products A-Z
		Newsroom	Product Certifications
Next-Generation Firewalls			Report a Vulnerability
Hardware Firewalls			Sitemap
Strata Cloud Manager			Tech Docs
			Unit 42
SECURE ACCESS SERVICE EDGE			Do Not Sell or Share My Personal Information
Prisma Access			
Prisma SD-WAN			
Autonomous Digital Experience Management			
Cloud Access Security Broker			
Zero Trust Network Access			
AI-Driven Security Operations Platform	Threat Intel and Incident Response Services		
Cortex XDR	Proactive Assessments		
Cortex XSOAR	Incident Response		
Cortex Xpanse	Transform Your Security Strategy		
Cortex XSIAM	Discover Threat Intelligence		
External Attack Surface Protection			
Security Automation			
Threat Prevention, Detection & Response			