Hamas-linked SameCoin campaign malware analysis

Identifier: TRR240201.

Summary

Following an X post by IntezerLab about an attack campaign that they dubbed “SameCoin”, we analyzed the samples they discovered and found a few identical variants. The infection vector appears to be an email impersonating the Israeli National Cyber Directorate, which tricks the reader into downloading malicious files which are presented as ‘security patches’.

Victims who download and execute linked files are infected with a wiper which, under certain circumstances, could also infect other hosts in the network. We assess that the campaign’s reach was limited, evidenced by the fact that the malware linked in the email was downloaded only a few dozen times.

Below we provide analysis of the discovered samples and discuss attribution.

Infection chain

According to a screen capture published by Intezer, the infection vector appears to be an email impersonating the Israeli National Cyber Directorate (INCD), sent on February 11, 2024. The email explains that *“The INCD has detected an imminent, major cyber attack sponsored by Iran, exploiting previously-unknown vulnerabilities in the personal computers and mobile phones of our citizens”*. It urges the reader to download “security patches” for macOS, iOS, Windows and Android, with the macOS and iOS links pointing at non-existing URLs under the legitimate INCD website. Victims who download and execute the Android or Windows applications links in the malicious emails are infected with a wiper.

While we could not retrieve any sample of such email, we could determine that some of the originally referenced malicious files (such as SHA-256 `556b5101e0e8aee004bed89f1686ce781a075fde5a8a86fa5409fe34a2d1b6d9`) were made available via the legitimate Gofile public files hosting service (see Fig. 1):

GoFile shortened link[1]	Distributed file	Metadata (at time of access)
`hxxps://gofile[.]io/d/WeFbpd`	`INCD-SecurityUpdate-FEB24.rar`, SHA-256 `82db3b82e49259ff9184b58c19e9107473d2eb40c586ffb85462e6a649db2051`	Uploaded on 2024-02-11 07:50:20, downloaded 13 times.
`hxxps://gofile[.]io/d/BnWjB6`	`INCD-SecurityUpdate-FEB24.rar`, SHA-256 `7e8caa1c3c1de1d8d8761e618408efdc875fb925bda31e0489234664642e33c3`	Uploaded on 2024-02-11 07:52:12, downloaded 8 times.
`hxxps://gofile[.]io/d/ikswEJ`	`INCD-SecurityUpdate-FEB24.apk`, SHA-256 `556b5101e0e8aee004bed89f1686ce781a075fde5a8a86fa5409fe34a2d1b6d9`	Uploaded on 2024-02-11 08:33:13, downloaded 11 times.
`hxxps://gofile[.]io/d/ssLPJv`	`INCD-SecurityUpdate-FEB24.apk`, SHA-256 `556b5101e0e8aee004bed89f1686ce781a075fde5a8a86fa5409fe34a2d1b6d9`	Uploaded on 2024-02-11 08:37:17, downloaded 1 time.

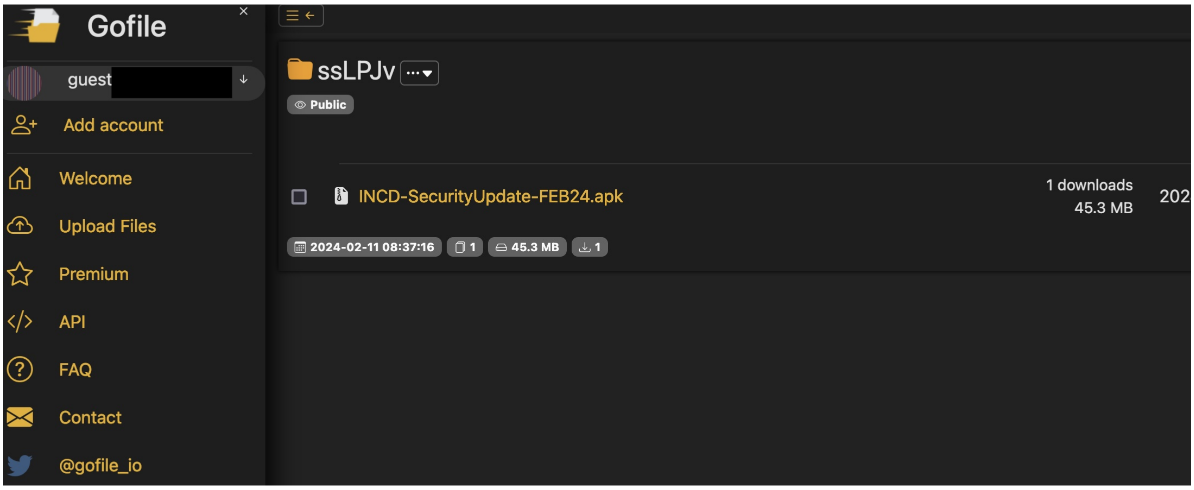


Figure 1 – Malicious file as published on Gofile

Both the names of the hosted files and their upload times are consistent with what has been reported on the email-based infection campaign. As a result, we believe with medium to high confidence that these links were embedded in the malicious emails, and were the source of distribution for the malicious files:

- `INCD-SecurityUpdate-FEB24.rar`: RAR archive which contains a SameCoin Loader payload, ultimately deploying the SameCoin Wiper for Windows;
- `INCD-SecurityUpdate-FEB24.apk`: malicious APK, a wiper for Android.

Loader

Filename	`INCD-SecurityUpdate-FEB24.exe`
Compilation time	2023-Oct-07 13:30:55
Hash (SHA256)	`cff976d15ba6c14c501150c63b69e6c06971c07f8fa048a9974ecf68ab88a5b6`

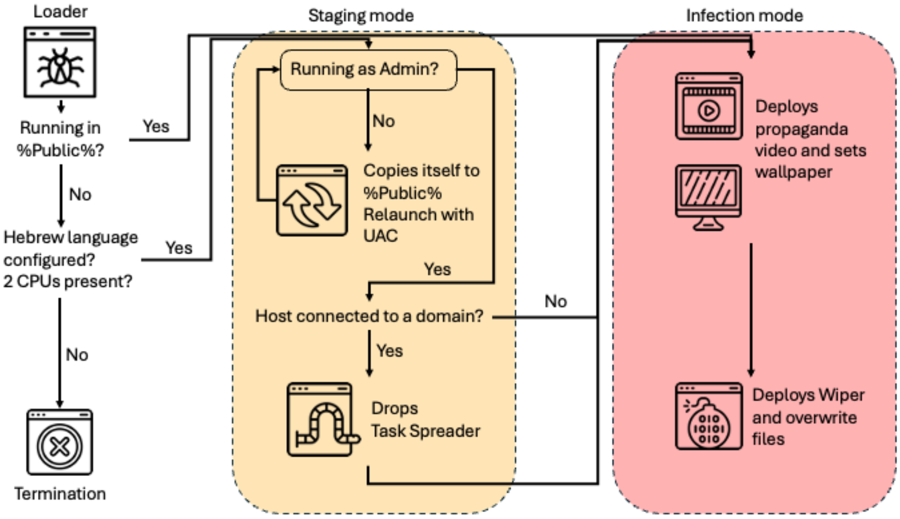

Loader
Running in %Public%?
Yes
No
Hebrew language configured? 2 CPUs present?
Yes
No
Termination
Staging mode
Running as Admin?
No
Copies itself to %Public% Relaunch with UAC
Yes
Host connected to a domain?
No
Yes
Drops Task Spreader
Infection mode
Deploys propaganda video and sets wallpaper
Deploys Wiper and overwrite files

Figure 2 – Loader logic

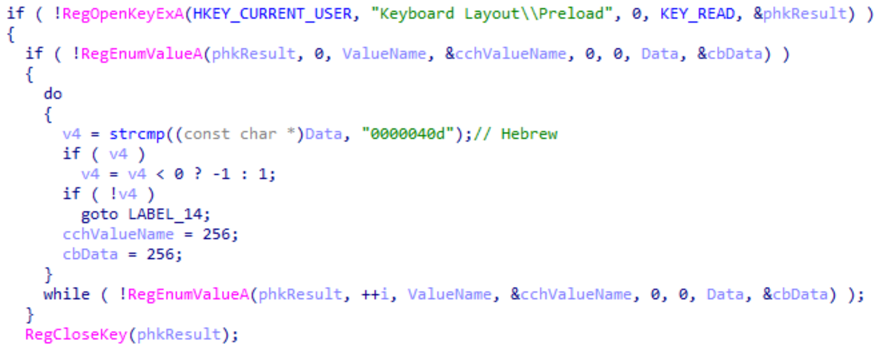
Staging mode

The loader operates in two modes (see Fig. 2). The first one consists in staging itself, and is triggered automatically when the program isn't located in `C:UsersPublic`. In that case, it performs a few checks to verify if the current machine is suitable for infection:

- Looking into the following registry keys if one of the configured keyboard layouts corresponds to Hebrew (`0x40d`, as shown in Figure 3):
    - `HKCUKeyboard LayoutPreload`
    - `HKLMSystemKeyboard LayoutPreload`
- Ensuring that the current machine has at least 2 CPUs, supposedly as a light anti-sandbox countermeasure.

If these conditions are met, a last check is performed: whether the program is running with administrator privileges. If so, the program moves on the the next mode of operation. Otherwise, it copies itself as `C:UsersPublicMicrosoft System Agent.exe` and invokes its new copy using the `runas` keyword to request elevated privileges to the user, and terminates execution. To do so, it dynamically resolves the `ShellExecuteEx` function by manually parsing the imports of `Shell32.dll`.

If `ShellExecuteEx` fails for any reason, the loader moves on to the next mode.



```
if ( !RegOpenKeyExA(HKEY_CURRENT_USER, "Keyboard Layout\\Preload", 0, KEY_READ, &phkResult) )
{
  if ( !RegEnumValueA(phkResult, 0, ValueName, &cchValueName, 0, 0, Data, &cbData) )
  {
    do
    {
      v4 = strcmp((const char *)Data, "0000040d");// Hebrew
      if ( v4 )
        v4 = v4 < 0 ? -1 : 1;
      if ( !v4 )
        goto LABEL_14;
      cchValueName = 256;
      cbData = 256;
    }
    while ( !RegEnumValueA(phkResult, ++i, ValueName, &cchValueName, 0, 0, Data, &cbData) );
  }
  RegCloseKey(phkResult);
```

Figure 3 – Code excerpt looking for traces of Hebrew language on the infected machine

Infection mode

In this second mode, the loader performs its malicious actions on the victim machines. First, it deploys a number of embedded components:

- A .NET sample (Tasks Spreader) as `C:UsersPublicWindows Defender Agent.exe`, if the machine is connected to a DC.
- A C Wiper as `C:UsersPublicMicrosoft System Manager.exe`.
- A JPG file, as `C:UsersPublicMicrosoft Connection Agent.jpg`.
- A propaganda Video, as `C:UsersPublicVideo.mp4`.

When all files are dropped, the loader runs all the deployed executables (manually resolving `CreateProcessA` by manually parsing the exports of `Kernel32.dll`). Finally, it sets the dropped JPG file as the Desktop wallpaper, turns the sound volume up by simulating roughly 50 presses on the corresponding keyboard key, and starts playing the propaganda video with the default player.

Among the three loader variants we discovered, the only difference we observed is between the compilation timestamps, set a few minutes apart. Those timestamps are set to October 7, 2023, the day of Hamas' attack.

Wiper

Filename	`Microsoft System Manager.exe`
Compilation time	2024-Feb-11 16:18:24
Hash (SHA256)	`e6d2f43622e3ecdce80939eec9fffb47e6eb7fc0b9aa036e9e4e07d7360f2b89`

This C executable is a simple wiper which crawls all drives on the system from `A:` to `Z:` in order to discover all writable files. Directories named `Program Files (x86)`, `Program Files`, `Program Data` and `Windows` are excluded (whether or not they are located at the root of a drive).

Next, the wiper creates 20 threads which all receive a portion of the generated filelist and iterate through it. Each file is opened and overwritten with 1111 random bytes obtained from the `rand()` function. This implies that the bytes located after offset 1111 in each file may still be present on the storage medium, and could potentially be recovered through forensics methods. The wiper excludes the files that were dropped by the Loader from being overwritten:

- `C:UsersPublicMicrosoft Connection Agent.jpg`
- `C:UsersPublicVideo.mp4`
- `C:UsersPublicMicrosoft System Agent.exe` (the Loader)
- `C:UsersPublicMicrosoft System Manager.exe` (the Wiper itself)
- `C:UsersPublicWindows Defender Agent.exe` (the Tasks Spreader)

The main logic of the wiper runs in a `while (1)` loop. In other words, the wiping process restarts as soon as it finishes.

Tasks Spreader

File name	`Windows Defender Agent.exe`
Compilation time	2091-Dec-05 00:39:31
Hash (SHA-256)	`b447ba4370d9becef9ad084e7cdf8e1395bafde1d15e82e23ca1b9808fef13a7`

"Tasks Spreader" is a .NET binary which is dropped and executed by the Loader when the infected computer is connected to an Active Directory (AD) domain. Its only goal is to spread the Loader across the domain, by copying it to other computers, then triggering its execution through a scheduled task.

The compilation time of the binary has been tampered with, but the standard “Legal Copyright” metadata property of the .NET assembly is set to `Copyright © 2024`, which indicates it may have been compiled in 2024.

The Tasks Spreader leverages the .NET `System.DirectoryServices` namespace and a LDAP query to browse all computers of all domains in the AD forest of the infected computer. It then copies the local Loader to all those remote computers (except for the one currently executing Tasks Spreader), from `C:UsersPublicMicrosoft System Agent.exe` to `<remote computer name>C$UsersPublicMicrosoft System Agent.exe`, using the `File.Copy` .NET method.

Tasks Spreader uses `System.DirectoryServices`‘s `Connect` method to remotely create a scheduled task on all computers where the Loader has been copied to. This task is aimed at executing the Loader, immediately and when a user of the `Domain users` default group logs in. The scheduled task is created using a `Schedule.Service` COM object, enabling universal operation of the Windows Task Scheduler Service without the need for language-specific objects, or reliance on `schtasks.exe`. This may effectively bypass the detection of scheduled tasks creation by some security products.

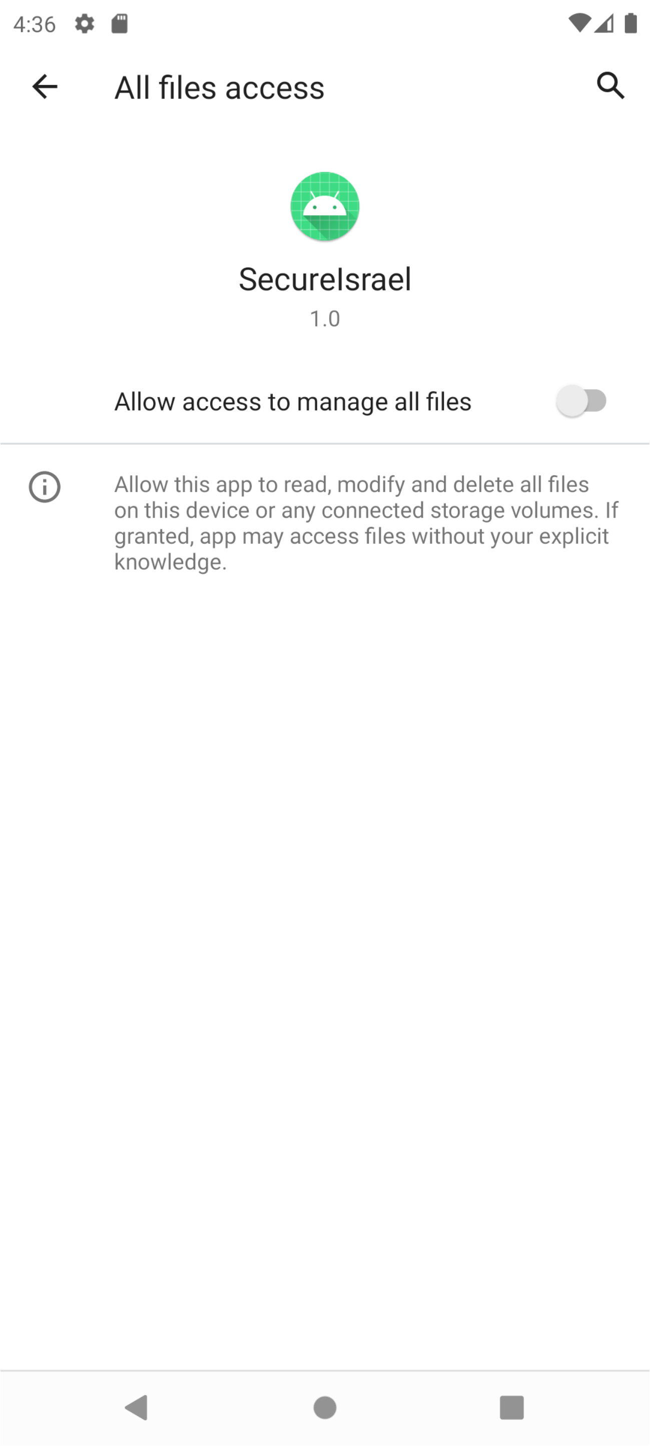
The created task is named `MicrosoftEdgeUpdateTaskMachinesCores`, set to execute the copied Loader binary at `<computer name>C$UsersPublicMicrosoft System Agent.exe`, and its description mimics the one from Microsoft Edge update tasks.

It should be noted that most actions that are attempted by the Tasks Spreader (most notably creating a file and scheduling a task on remote computers) requires proper privileges. The file copy operation to remote computers is attempted without any prior check for such privileges. Any failure (programatic exception) is caught and ignored, which means that if the AD forest can be browsed for computers, the file copy will be attempted to all found computers of all domains, making the operation extremely noisy (and in theory, every computer that executes the Loader in a domain will attempt the same).

APK

Filename	`INCD-SecurityUpdate-FEB24.apk`
Certificate start date	2024-Feb-09 15:16:26
Hash (SHA256)	`556b5101e0e8aee004bed89f1686ce781a075fde5a8a86fa5409fe34a2d1b6d9`

This APK (Android Package Kit), also distributed under the pretense of being a security update, starts by requesting the permission to access all files on the device (see Fig. 4 and 5). If the permissions are denied, an English message is shown (“Permission not granted!”).



4:36

All files access

SecureIsrael

1.0

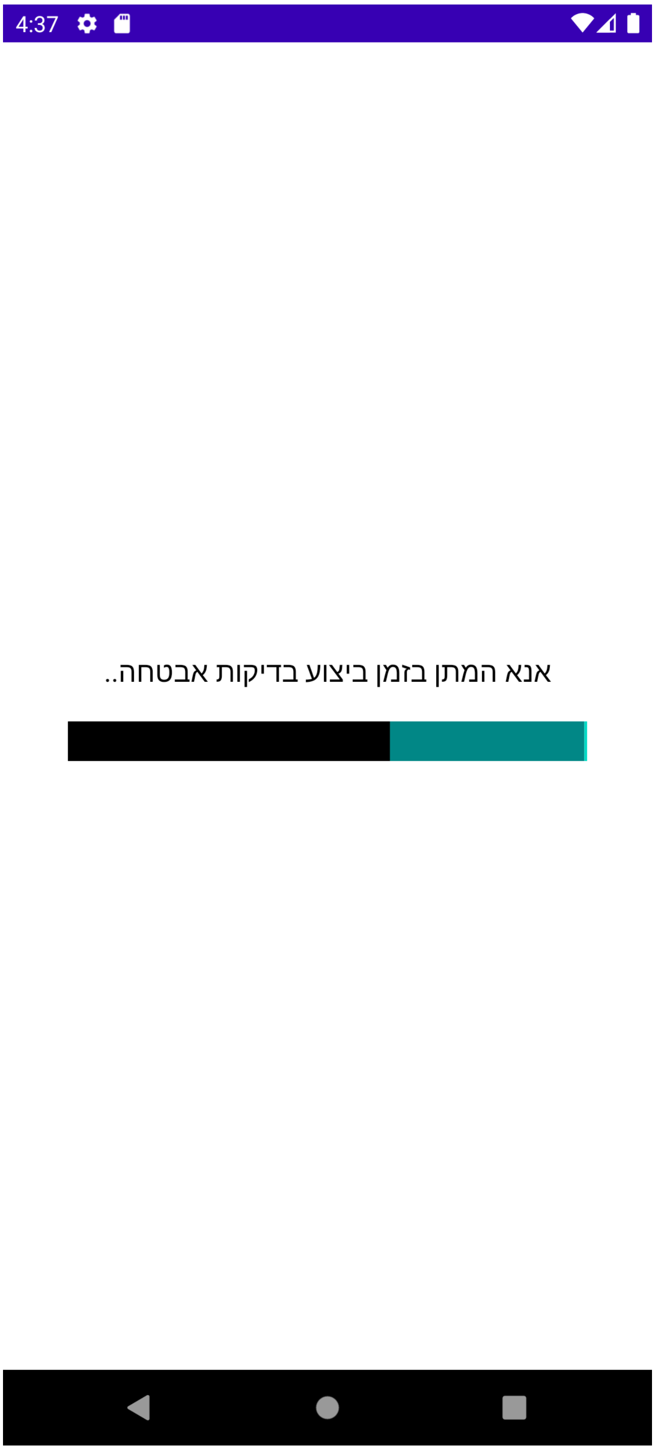
Allow access to manage all files

Allow this app to read, modify and delete all files on this device or any connected storage volumes. If granted, app may access files without your explicit knowledge.

Figure 4 – Request to manage the device files from the SecureIsrael application

Once filesystem access has been granted, control is transferred to a function called `deleteInCHunks()` (sic.) located in a native library (`libexampleone.so`) bundled with the APK, while the application starts playing the same propaganda video as the one contained in the Loader. This video (~45MB) represents most of the contents of the APK.

Even if file access permissions are not granted, the application eventually shows the video to the victim. This may be a programming error, as the app contains traces of being built from an example/tutorial project, or could be a conscious attempt at trying to scare the victim despite not being able to delete its data.



אנא המתן בזמן ביצוע בדיקות אבטחה..

Figure 5 – The progress bar displayed by the application until permissions are granted by the user. The text (in poorly-worded Hebrew) reads: “Please stand by while performing security check”

Native library

Filename	`libexampleone.so`
Hash (SHA256)	`248054658277e6971eb0b29e2f44d7c3c8d7c5abc7eafd16a3df6c4ca555e817`

This library is written in C++ and bundled with its parent APK in several variants (one for each possible CPU architecture). It crawls the `/storage/emulated/0/` folder (corresponding to the root directory of the emulated storage for the primary user profile) and builds a list of files to wipe, similarly to the Windows Wiper. During this process, the folders `/storage/emulated/0/Android/obb` and `/storage/emulated/0/Android/data` are ignored, as they contain binary blobs and application data which typically cannot be accessed by other applications.

Once the file list is created, the library creates threads which receive a “chunk” of the list. Each file is overwritten with zeroes and then removed from the filesystem. Finally, the wiper attempts to remove `/storage/emulated/0` and all the files it contains.

Attribution and propaganda contents

The wallpaper (JPG file, see Fig. 7) which is dropped by the Loader shows an Israeli “Namer” Armoured Personnel Carrier vehicle, currently utilized in the ongoing Israel-Hamas war. The image, originally sourced from Wikipedia[2], was altered by the attacker (see Fig. 6) to depict the vehicle as damaged:

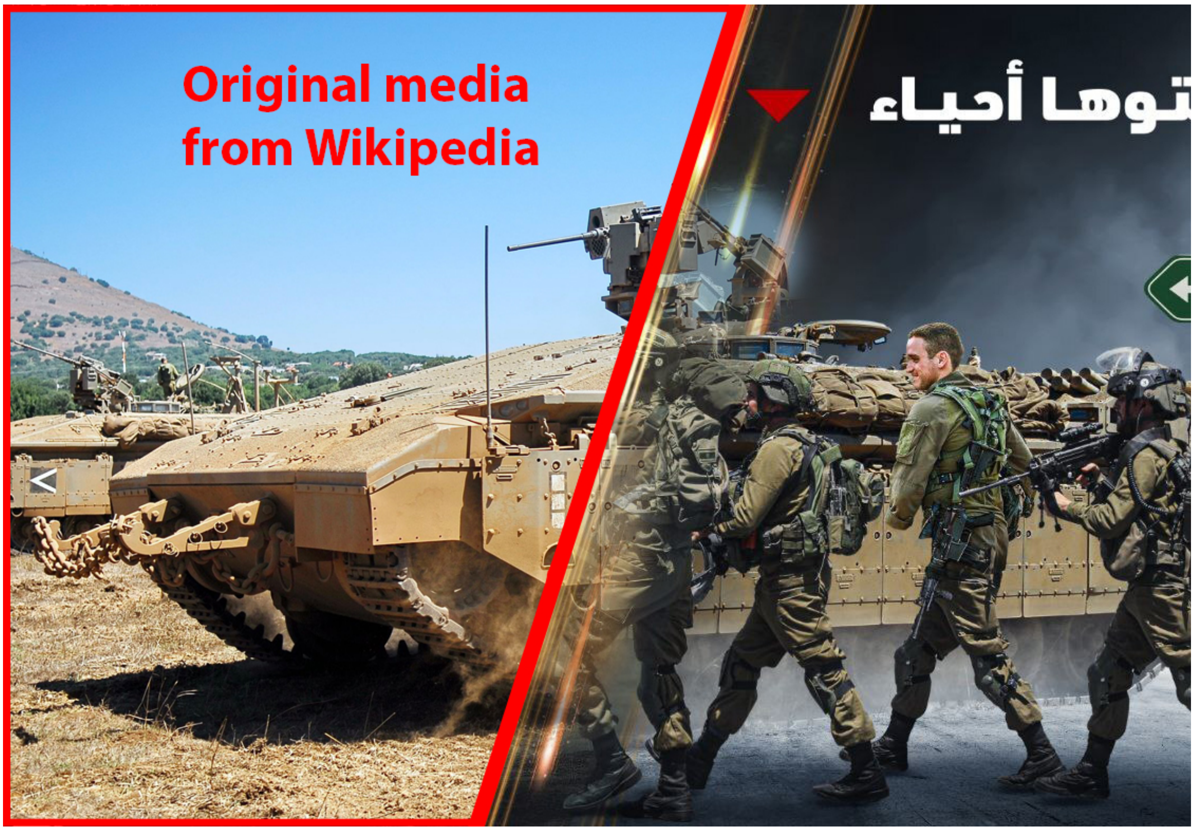


Figure 6 – Composition comparing the original image with the propaganda image, original image from Wikipedia (CC 3.0)

In the propaganda image (see Fig. 7), Israeli Defence Forces (IDF) are depicted moving into Gaza, transitioning from right to left, concluding in body bags. Notably, during this transition, the inverted red triangle emoji (🔻) is utilized – this symbol is used to *“represent Hamas itself and glorify its use of violence”*[3], often shown in Hamas’ propaganda videos upon attacking IDF. The text on the right reads “You entered it alive”, and above the body bags “You will leave it in pieces”:

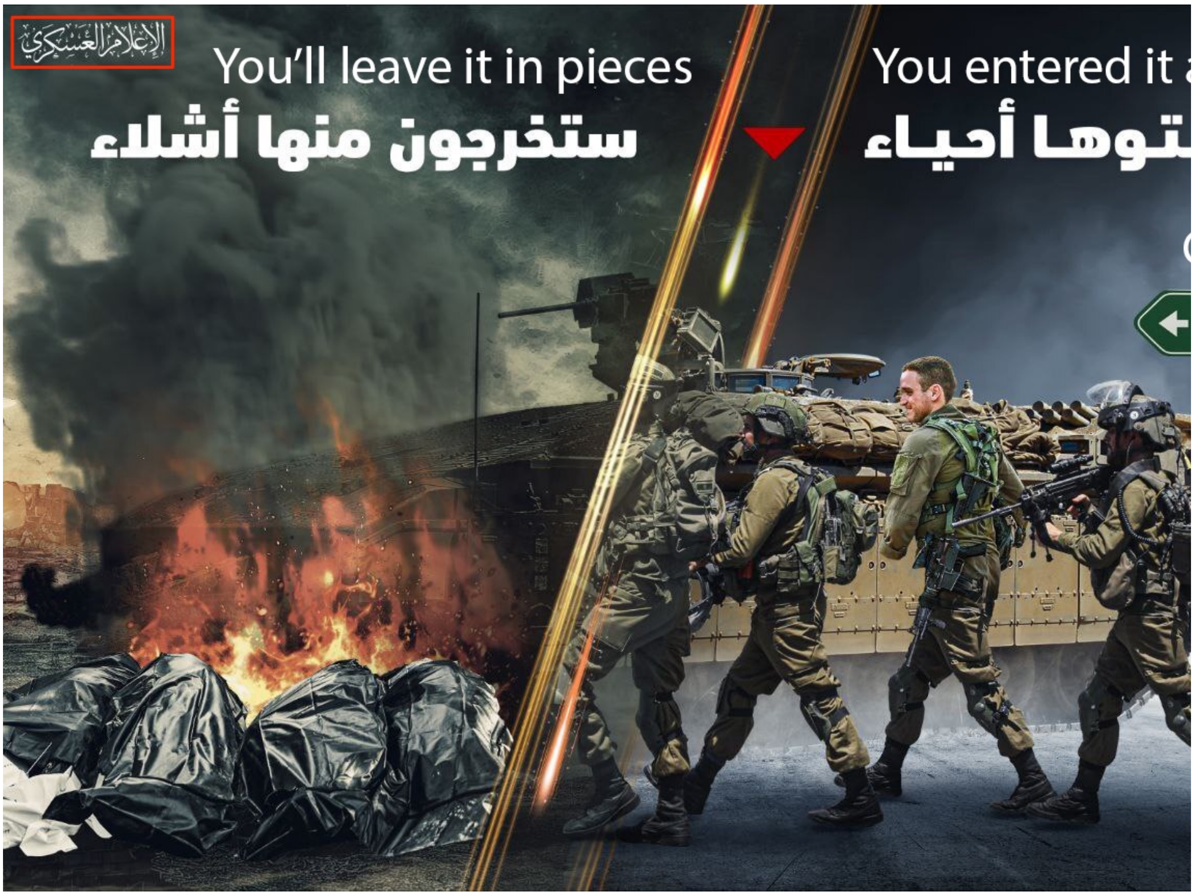


Figure 7 – Propaganda image that is dropped by the Loader

This image however, isn't new. It was posted as early as December 8, 2023 on X.

The video which is dropped by the Loader and embedded in the APK also fits Hamas' ongoing propaganda efforts in turning the hostages' families against Israel's prime minister Bibi Netanyahu.

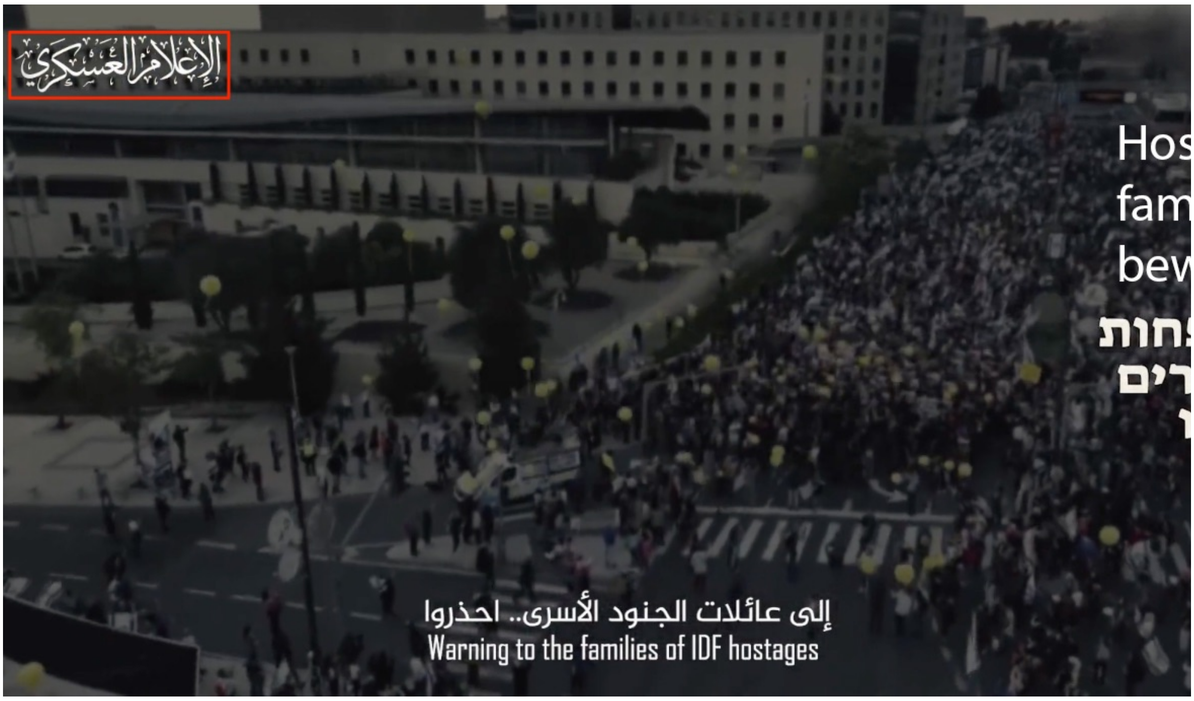

الإعلام العسكري
Hos
fam
bew
إلى عائلات الجنود الأسرى.. احذروا
Warning to the families of IDF hostages

Figure 8 – Screen capture of the propaganda video that is shown by the APK and dropped by the Loader

Both the wallpaper and the video use the same signature "الإعلام العسكري" ("Military Media", see Fig. 8, highlighted in red), which is tied with propaganda pieces generated by Hamas. This comes in addition to the aforementioed inverted red triangle emoji ("🔻").

Lastly, the timestamps of the Loader samples were modified to match 07 October 2023, the day when Hamas launched its attack on Israel.

Cumulatively, these elements lead us to conclude that a threat-actor who is associated with Hamas is at the helm of this campaign. While we could not identify any technical indicator overlap with other known malicious campaigns or actors, parts of the described activities and techniques fit the Arid Viper APT (APT-C-23, Desert Falcons), which is usually associated with Hamas. Namely, we see similarities in the targeting, the high-grade lure content that contrasts with the malwares' lack of sophistication, as well as the diversity in targeted systems (Android and Windows) and programming languages employed. Moreover, using native libraries alongside Android implants is a technique which has been leveraged by Arid Viper since 2021, as documented by Talos.

Appendix

Indicators of compromise (IOCs)

Associated IOCs are also available on our GitHub repository.

Hashes (SHA-256)

```
9b62af6b13b610f4f90810b2f5aef0a455a301a06c98c49a531384d90f90f921|SameCoin Malicious
Attachment (RAR)
ea2ff8f681fd1ab2a4d22f245e6475c68e7fcf9d7f6ec3a549776c4bbe279553|SameCoin Malicious
Attachment (RAR)
82db3b82e49259ff9184b58c19e9107473d2eb40c586ffb85462e6a649db2051|SameCoin Malicious
Attachment (RAR)
7e8caa1c3c1de1d8d8761e618408efdc875fb925bda31e0489234664642e33c3|SameCoin Malicious
Attachment (RAR)
cff976d15ba6c14c501150c63b69e6c06971c07f8fa048a9974ecf68ab88a5b6|SameCoin Loader
1624e5c9dd10c4ef21dee571cac3343cac1a6a94a847d85dc264786f4ef24f40|SameCoin Loader
598ed8a0a9a3b3c94bf8d8bfdd9f86882d7c97f9f3dc6c85e3e34ad77489186c|SameCoin Loader
4d28afa4d22ddae336de418380de21bb750231331ccdacfd4b7eff5ab6b1b693|SameCoin Loader
e6d2f43622e3ecdce80939eec9fffb47e6eb7fc0b9aa036e9e4e07d7360f2b89|SameCoin Wiper
b447ba4370d9becef9ad084e7cdf8e1395bafde1d15e82e23ca1b9808fef13a7|SameCoin Tasks
Spreader
556b5101e0e8aee004bed89f1686ce781a075fde5a8a86fa5409fe34a2d1b6d9|SameCoin APK
5a5eea6a56aebb2d8b939dc57967395b1b85cbfe7ca06b86a1916dfa31858e09|libexampleone.so
(arm64-v8a)
c3938b85ec97fe4f433102b050f89250236b7379994da55314c24c623fb469a9|libexampleone.so
(arm64-v7a)
248054658277e6971eb0b29e2f44d7c3c8d7c5abc7eafd16a3df6c4ca555e817|libexampleone.so
(x86)
18d6b9d09782c49162b9b57eaae077cbc37d25132253578fa4874eb2b7a46c50|libexampleone.so
(x86_64)
```

File paths

```
C:UsersPublicMicrosoft System Agent.exe|SameCoin Loader
C:UsersPublicWindows Defender Agent.exe|SameCoin Tasks Spreader
C:UsersPublicMicrosoft System Manager.exe|SameCoin Wiper
C:UsersPublicMicrosoft Connection Agent.jpg|SameCoin propaganda image
C:UsersPublicVideo.mp4|SameCoin propaganda video
```

URLs

```
hxxps://store9.gofile[.]io/download/76732040-c118-40cc-a33e-f7fb22f1c1aa/INCD-
SecurityUpdate-FEB24.rar|SameCoin Malicious Attachment (RAR)
hxxps://store27.gofile[.]io/download/15c1c6a0-2f5b-44ee-951d-a64778fed86d/INCD-
SecurityUpdate-FEB24.rar|SameCoin Malicious Attachment (RAR)
hxxps://store2.gofile[.]io/download/fab845cc-aba0-49ce-ab89-753d7685bd46/INCD-
SecurityUpdate-FEB24.apk|SameCoin APK
hxxps://store2.gofile[.]io/download/803df49c-d77d-44d0-ad2f-28818432a4ce/INCD-
SecurityUpdate-FEB24.apk|SameCoin APK
```

Yara rules

The provided Yara rules require Yara 3.2.0 (Nov 10, 2014) and up.

```
rule samecoin_campaign_loader {
    meta:
        description = "Matches the loader used in the SameCoin campaign"
        references = "TRR240201"
        hash = "cff976d15ba6c14c501150c63b69e6c06971c07f8fa048a9974ecf68ab88a5b6"
        date = "2024-02-13"
        author = "HarfangLab"
        context = "file"
    strings:
        $hebrew_layout = "0000040d" fullword ascii
        $runas = "runas" fullword ascii
        $jpg_magic = { FF D8 FF E0 00 10 4A 46 49 46 00 01 }
        $wl_1 = "C:\Users\Public\Microsoft Connection Agent.jpg" ascii
        $wl_2 = "C:\Users\Public\Video.mp4" ascii
        $wl_3 = "C:\Users\Public\Microsoft System Agent.exe" ascii
        $wl_4 = "C:\Users\Public\Microsoft System Manager.exe" ascii
        $wl_5 = "C:\Users\Public\Windows Defender Agent.exe"
    condition:
        uint16(0) == 0x5A4D and filesize > 5MB and filesize < 7MB and
        $hebrew_layout and $runas and $jpg_magic and 3 of ($wl_*)
}

rule samecoin_campaign_wiper {
    meta:
        description = "Matches the wiper used in the SameCoin campaign"
        references = "TRR240201"
        hash = "e6d2f43622e3ecdce80939eec9fffb47e6eb7fc0b9aa036e9e4e07d7360f2b89"
        date = "2024-02-13"
        author = "HarfangLab"
        context = "file"
    strings:
        $code = { 68 57 04 00 00 50 E8 } // push 1111; push eax; call
        $wl_1 = "C:\Users\Public\Microsoft Connection Agent.jpg" ascii
        $wl_2 = "C:\Users\Public\Video.mp4" ascii
        $wl_3 = "C:\Users\Public\Microsoft System Agent.exe" ascii
        $wl_4 = "C:\Users\Public\Microsoft System Manager.exe" ascii
        $wl_5 = "C:\Users\Public\Windows Defender Agent.exe" ascii
    condition:
        uint16(0) == 0x5A4D and filesize < 200KB and
        $code and 3 of ($wl_*)
}

rule samecoin_campaign_tasksspreader
{
    meta:
        description = "Detect .NET Task Scheduler that is dropper by SameCoin
Loader"
        references = "TRR240201"
        hash = "b447ba4370d9becef9ad084e7cdf8e1395bafde1d15e82e23ca1b9808fef13a7"
        date = "2024-02-13"
        author = "HarfangLab"
        context = "file"
    strings:
        $dotNet = ".NETFramework,Version" ascii fullword
        $a1 = "System.DirectoryServices.ActiveDirectory" ascii fullword
        $a2 = "GetTypeFromProgID" ascii fullword
        $a3 = "DirectorySearcher" ascii fullword
        $a4 = "SearchResultCollection" ascii fullword
        $a5 = "UnaryOperation" ascii fullword
        $b1 = "$dc1b29f0-9a87-4383-ad8b-01285614def1" ascii fullword
        $b2 = "Windows Defender Agent" ascii fullword
        $b3 = "Windows Defender Agent.exe" wide ascii fullword
        $b4 = /(\)?C(:|$)\Users\Public\Microsoft System Agent.exe/ wide fullword
        $b5 = "MicrosoftEdgeUpdateTaskMachinesCores" wide fullword
        $b6 = "WindowsUpdate" wide fullword
        $c1 = "RegisterTaskDefinition" wide fullword
        $c2 = "DisallowStartIfOnBatteries" wide fullword
        $c3 = "StopIfGoingOnBatteries" wide fullword
        $c4 = "Schedule.Service" wide fullword
        $c5 = "\Domain Users" wide fullword
        $c6 = "(objectClass=computer)" wide fullword
    condition:
        filesize > 8KB and filesize < 40KB
        and (uint16be(0) == 0x4D5A)
        and $dotNet
        and (4 of ($a*))
        and (
            ((any of ($b*)) and (any of ($c*)))
            or (all of ($c*))
        )
}

rule samecoin_campaign_nativewiper {
    meta:
        author = "HarfangLab"
        description = "Matches the native Android library used in the SameCoin
campaign"
        references = "TRR240201"
        last_modified = "2024-02-13"
        context = "file"
        hash = "248054658277e6971eb0b29e2f44d7c3c8d7c5abc7eafd16a3df6c4ca555e817"
    strings:
        $native_export = "Java_com_example_exampleone_MainActivity_deleteInCHunks"
ascii
        $f1 = "_Z9chunkMainv" ascii
        $f2 = "_Z18deleteFilesInChunkRKNSt6__" ascii
        $f3 = "_Z18overwriteWithZerosPKc" ascii
        $s1 = "/storage/emulated/0/" ascii
        $s2 = "FileLister" ascii
        $s3 = "Directory chunks deleted."
        $s4 = "Current Chunk Size is:  %dln" ascii
    condition:
        filesize < 500KB and uint32(0) == 0x464C457F and
        ($native_export or all of ($f*) or all of ($s*))
}
```

[1] Full links, as they were likely put in malicious emails, are provided in Appendix. ↩

[2] https://en.wikipedia.org/wiki/Namer ↩

[3] https://extremismterms.adl.org/glossary/inverted-red-triangle ↩

Published on 14 February, 2024 Last update on 15 February, 2024