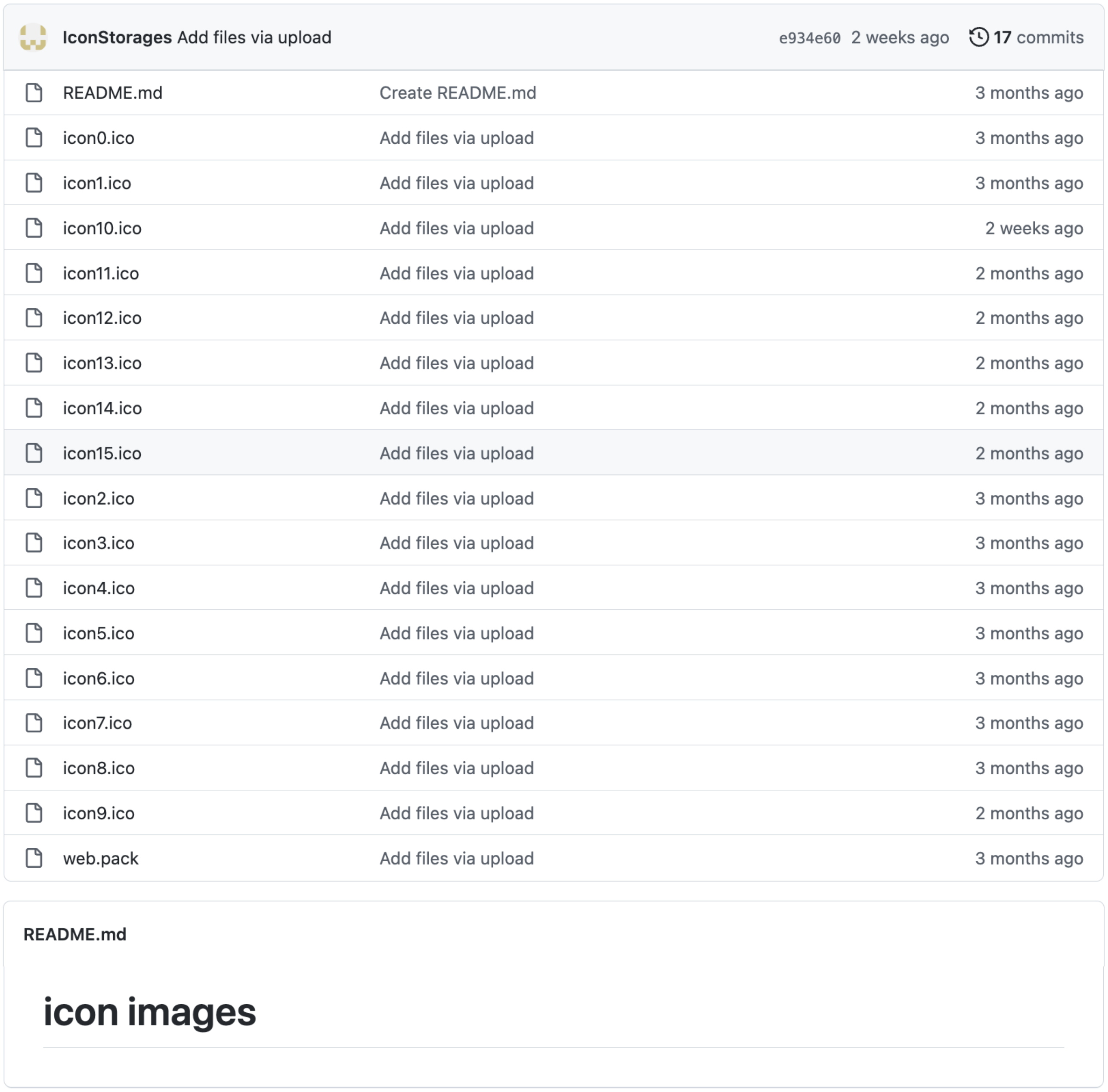
3CX Supply Chain Attack

research.openanalysis.net/3cx/northkorea/apt/triage/2023/03/30/3cx-malware.html

OALABS Research March 30, 2023



IconStorages Add files via upload e934e60 2 weeks ago **17** commits

README.md	Create README.md	3 months ago
icon0.ico	Add files via upload	3 months ago
icon1.ico	Add files via upload	3 months ago
icon10.ico	Add files via upload	2 weeks ago
icon11.ico	Add files via upload	2 months ago
icon12.ico	Add files via upload	2 months ago
icon13.ico	Add files via upload	2 months ago
icon14.ico	Add files via upload	2 months ago
icon15.ico	Add files via upload	2 months ago
icon2.ico	Add files via upload	3 months ago
icon3.ico	Add files via upload	3 months ago
icon4.ico	Add files via upload	3 months ago
icon5.ico	Add files via upload	3 months ago
icon6.ico	Add files via upload	3 months ago
icon7.ico	Add files via upload	3 months ago
icon8.ico	Add files via upload	3 months ago
icon9.ico	Add files via upload	2 months ago
web.pack	Add files via upload	3 months ago

README.md

icon images

Overview

From the Volexity post

> CrowdStrike identified signed 3CX installation files as being malicious and reported that customers were seeing malicious activity emanating from the “3CXDesktopApp”.

3CX is client software for VOIP phones, that was delivered to targets with a backdoor. The backdoored application was delivered in an MSI `3CXDesktopApp-18.12.416.msi` which is signed by a valid certificate belonging to 3Cx Ltd.

References

Samples

- `3CXDesktopApp-18.12.416.msi`
  59e1edf4d82fae4978e97512b0331b7eb21dd4b838b850ba46794d9c7a2c0983
- `icon15.ico`
  f47c883f59a4802514c57680de3f41f690871e26f250c6e890651ba71027e4d3

Analysis

Let's take a look at the `.msi` and see what is in there, we can just use 7zip to unzip it. Inside the `.msi` we have a backdoored file `ffmpeg.dll`

Stage 1 `ffmpeg.dll`

Artifacts

- `ffmpeg.dll`
  7986bbaee8940da11ce089383521ab420c443ab7b15ed42aed91fd31ce833896
- `d3dcompiler_47.dll`
  11be1803e2e307b647a8a7e02d128335c448ff741bf06bf52b332e0bbf423b03

Functionality

- Uses `CreateEventW` with the string `AVMonitorRefreshEvent` like a mutex to ensure it is only running once
- Gets its process path (file location) to locate `d3dcompiler_47.dll` which it expects to be in the same directory
- Scans `d3dcompiler_47.dll` for the magic hex bytes `0xFEEDFACE`
- The magic bytes `0xFEEDFACE` occur twice in a row
- All the file data following the magic bytes is decrypted with RC4 using the hard coded key `3jB(2bsG#@c7`
- Once decrypted the data contains shellcode followed by an embedded PE file (Stage 2) which is loaded into memory and executed

Signed DLL

The `d3dcompiler_47.dll` DLL is signed by Microsoft. The `0xFEEDFACE` magic bytes suggest that the open source tool SigFlip was used to patch the authenticode signed PE file without breaking the signature.

Stage 2

Artifacts

Shellcode with stage 2 PE attached
b56279136d816a11cf4db9fc1b249da04b3fa3aef4ba709b20cdfbe572394812

Functionality

- Creates a file called `manifest` in the directory from which the process was launched
- The `manifest` file is used to maintain a delay timer value for the malware
- The delay is calculated by adding 7 days to a randomly generated value between 0 days and 20 days, 20 hours for a total potential delay of between 7 days, and 20 days 20 hours
- When the malware executes this value is read from the `manifest` file and checked against the system time, if the time has not expired the malware will simply sleep
- The `MachineGuid` key value is read from the registry key `Software\\Microsoft\\Cryptography` then transformed into the following "cookie" value to be used in future C2 requests

```
_tutma=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx
```

A random number generator is used to build a variation of the following URL with an icon file between `icon1.ico` and `icon16.ico` (either I'm not reading the code right or this is an off-by-one error as the icon files are number 0-15?)

```
https[:]//raw.githubusercontent[.]com/IconStorages/images/main/icon%d.ico
```

- The icon file is downloaded from GitHub and parsed to extract encoded data that is appended to the file
- The appended data is preceded by a `$` which the malware uses as a marker to identify it
- The following is an example of the bas64 encoded data in `icon15.ico`

`KQAAAGVhV4u+Eo4SGUuZypP8kNOkwQWzha6sxQrtzFo3oPSejc470WC47cKqv12+CshijG0HCfex40WinKat
68EHqq8i6lHiifZpsxN3lxBRabtJ`

- The data is then base64 decoded and passed through an unidentified generator used to create a key for the data
- The key is then used to decrypt the remaining data using AES

- Once decrypted the data reveals the stage2 C2 URL `https[:]//pbxsources[.]com/exchange`, each icon file contains a different URL
- A request is then sent to the C2 using the `_tutma` cookie described above and stage 3 is downloaded

Stage 3 was not recovered