3CX Supply Chain Compromise Leads to ICONIC Incident

By mindgrub

Published: 2023-03-30 · Archived: 2026-04-05 16:45:01 UTC

[Update: Following additional analysis of shellcode used in ICONIC, in conjunction with other observations from the wider security community, Volexity now attributes the activity described in this post to the Lazarus threat actor. Specifically, in addition to other claims of similarity, the shellcode sequence {E8 00 00 00 00 59 49 89 C8 48 81 C1 58 06 00 00} appears to have been only used in the ICONIC loader and the APPLEJEUS malware, which is known to be linked to Lazarus. The original post has been left as written.]

On Wednesday, March 29, 2023, Volexity became aware of a supply chain compromise by a suspected North Korean threat actor, which Volexity tracks as UTA0040*. Endpoints with the 3CX Desktop application installed received a malicious update of this software that was signed by 3CX and downloaded from their servers. This was part of the default automatic update process and would result in information-stealing malware being installed on the victim's host. It is possible that additional malicious activity may have taken place if the threat actor deemed the endpoint to be of sufficient interest.

3CX is a phone system company and claims to have more than 600,000 customers and 12 million users, including world-renowned brands. They have posted an update on their website acknowledging the compromise, though it should be noted the information in this post should not be deemed conclusive or entirely accurate based on Volexity's analysis.

In a public post on Reddit, CrowdStrike identified signed 3CX installation files as being malicious and reported that customers were seeing malicious activity emanating from the "3CXDesktopApp". Volexity further identified public

forum postings on 3CX's own website that stated various endpoint detection and response (EDR) and antivirus (AV) vendors began flagging malicious activity from updates as early as March 22, 2023. Volexity's analysis suggests the malicious activity likely began much earlier.

Volexity was able to obtain multiple malicious installers for Windows and macOS directly from 3CX download servers. Analysis of installers from both platforms allowed Volexity to identify several new indicators of compromise and gain further insight into how the malware functions.

This post details what Volexity discovered from its analysis of the malicious installers and the additional files it downloads. Highlights of the findings include the following:

- Both the macOS and Windows installers for 3CX are affected.
- Based on data recovered from GitHub, infrastructure used by the Windows variant was activated on December 7, 2022.
- Domains and web infrastructure used in the attacks were registered as early as November 2022.
- A reconnaissance payload was deployed far and wide to Windows users.
- The same functionality to download a payload was identified in the macOS sample, although Volexity could not confirm the final payload as the C2 was unresponsive at the time of analysis.

Any endpoint impacted by this malicious update should be isolated and investigated for further signs of compromise. Organizations should assess the potentially impacted information on these endpoints and look to cycle secrets to reduce the risk of additional future compromise.

ICONIC Analysis

Stage #1: Supply Chain Attacks – ICONIC

Volexity's analysis began with one of the installers tagged as malicious in public discourse:

Name(s)	3CXDesktopApp-18.12.416.msi
Size	97.8MB (102555648 Bytes)
File Type	Windows Installer
MD5	0eeb1c0133eb4d571178b2d9d14ce3e9
SHA1	bfecb8ce89a312d2ef4afc64a63847ae11c6f69e
SHA256	59e1edf4d82fae4978e97512b0331b7eb21dd4b838b850ba46794d9c7a2c0983

The installer contains a malicious version of *ffmpeg.dll*, an open-source video player library:

Name(s)	ffmpeg.dll
Size	2.7MB (2814976 Bytes)
File Type	Win32 DLL

MD5	74bc2d0b6680faa1a5a76b27e5479cbc
SHA1	bf939c9c261d27ee7bb92325cc588624fca75429
SHA256	7986bbaee8940da11ce089383521ab420c443ab7b15ed42aed91fd31ce833896

The library is loaded by *3CXDesktopApp.exe*, and it is used to decode and inject a payload into memory:

Name(s)	N/A
Size	288.0KB (294912 Bytes)
File Type	application/x-dosexec
MD5	11bc82a9bd8297bd0823bce5d6202082
SHA1	894e7d4ffd764bb458809c7f0643694b036ead30
SHA256	f79c3b0adb6ec7bcc8bc9ae955a1571aaed6755a28c8b17b1d7595ee86840952

The purpose of this malware, which Volexity will refer to as “ICONIC”, is as follows:

- Download various files that contain additional code, with names such as *icon[0-15].ico*, hosted at *https://github[.]com/IconStorages/images/*. (Note: the GitHub repository has since been taken down.)
- Parse these files to identify a “$” character followed by a base64-encoded string appended to the end of the ICO files.
- Decrypt the base64 string using the AES-GCM encryption algorithm. All values required to decrypt AES-GCM are derived from a complex function that third-party researchers have indicated is based on a publicly available gist.
- Once the string is decoded, it contains the URLs with which the DLL will then communicate to receive a next-stage payload.
- The next-stage payload is a JSON object that is then parsed and must further be decrypted (with the same AES-GCM decryption function). The next stage is expected to be a 64-bit PE that is reflectively loaded through a shellcode loader stored at the head of the file.

A script is provided with this post on GitHub that can be used to decrypt the base64 blobs appended to the ICO files. Volexity was able to clone the GitHub project, and through the commit history, was also able to retrieve files that had previously been deleted. The table below provides details of each file and the decoded URL from each one. Note that there are duplicate filenames due to deletions, and in some cases the files were identical.

Active Files

Filename	Hash (SHA1)	Decoded URL
icon0.ico	9c943baad621654cc0a0495262b6175276a0a9fb	https://www.3cx[.]com/blog/event-trainings/
icon1.ico	96910a3dbc194a7bf9a452afe8a35eceb904b6e4	https://msstorageazure[.]com/window

icon2.ico	ffccc3a29d1582989430e9b6c6d2bff1e3a3bb14	https://officestoragebox[.]com/api/session
icon3.ico	89827af650640c7042077be64dc643230d1f7482	https://visualstudiofactory[.]com/workload
icon4.ico	b5de30a83084d6f27d902b96dd12e15c77d1f90b	https://azuredeploystore[.]com/cloud/services
icon5.ico	3992dbe9e0b23e0d4ca487faffeb004bcfe9ecc8	https://msstorageboxes[.]com/office
icon6.ico	caa77bcd0a1a6629ba1f3ce8d1fc5451d83d0352	https://officeaddons[.]com/technologies
icon7.ico	57a9f3d5d1592a0769886493f566930d8f32a0fc	https://sourceslabs[.]com/downloads
icon8.ico	f533bea1c0558f73f6a3930343c16945fb75b20f	https://zacharryblogs[.]com/feed
icon9.ico	31d775ab577f3cc88991d90e9ae58501dbe1f0da	https://pbxcloudeservices[.]com/phonesystem
icon10.ico	0d890267ec8d6d2aaf43eaca727c1fbba6acd16e	https://akamaitechcloudservices[.]com/v2/storage
icon11.ico	0d890267ec8d6d2aaf43eaca727c1fbba6acd16e	https://akamaitechcloudservices[.]com/v2/storage
icon12.ico	b1dee3ebcffad01a51ff31ff495fef1d40fdfaa0	https://azureonlinestorage[.]com/azure/storage
icon13.ico	64ab912d0af35c01355430d85dd4181f25e88838	https://msedgepackageinfo[.]com/microsoft-edge
icon14.ico	8377fb40c76aa3ba3efae3d284fa51aa7748e010	https://glcloudservice[.]com/v1/console
icon15.ico	11ae67704ea0b930b2cc966e6d07f8b898f1a7d2	https://pbxsources[.]com/exchange

Deleted Files

Filename	Hash (SHA1)	Decoded URL
icon1.ico	ad37112b302c5193e60f6f6f49f4df668f5d3eb9	https://msedgeupdate[.]net/Windows
icon2.ico	ad37112b302c5193e60f6f6f49f4df668f5d3eb9	https://msedgeupdate[.]net/Windows
icon10.ico	3a2138cd38ff2cef246f122a97d3c8f85ab6fc94	https://pbxphonenetwork[.]com/voip
icon0.ico	3df119f322c5182bdbea4ab364eec8a0e23d888b	https://msstorageazure[.]com/window
icon1.ico	9c943baad621654cc0a0495262b6175276a0a9fb	https://www.3cx[.]com/blog/event-trainings/
icon0.ico	9c943baad621654cc0a0495262b6175276a0a9fb	https://www.3cx[.]com/blog/event-trainings/

A summary of the created, last modified, and domain registration times for each of these files is provided as an attachment to this post here.

Volexity believes the *www.3cx[.]com* entries were used for testing because, at the time of analysis, these URLs would not return a payload that could be parsed by the malware. Volexity was not able to retrieve payloads from the *msedgeupdate[.]net* or *pbxphonenetwork[.]com* domains, while the remainder of the URLs all provided the same valid second-stage payload.

The first commit to the GitHub page containing an ICO file with an encrypted 3cx[.]com URL was added on December 7, 2022, which suggests that the attacker had potentially initiated their own testing of the backdoor at this time.

Stage #2: ICONIC Stealer

Once a URL is decoded from an ICO file, a specially formatted request is made to download a second-stage payload. The format of the request is below:

```
accept: */*
accept-language: en-US,en;q=0.9
accept-encoding: gzip, deflate, br
content-type: text/plain
cookie: __tutma={MachineGuid}
```

The *MachineGuid* is derived from the system's registry via *SOFTWARE\Microsoft\ Cryptography*.

Volexity's analysis determined that the "cookie" header is the crucial component to retrieving the second-stage payload. If this value is not sent, no payload is returned to the user; the C2 responds with a *204 (No Content)* status code instead. Volexity also determined that the second-stage payload servers are forwarding requests upstream to central infrastructure. This assertion is based on the fact that any given *MachineGuid* sent in the cookie header will only work once, even when used with different C2s.

Below is a snippet of what the returned JSON looks like:

```
{"url":"","description":"","meta":"vyoAAL4D<snip>"}
```

Each of the live servers returned identical responses, consisting of shellcode followed by a 64-bit DLL, which Volexity refers to as "ICONICSTEALER". The DLL was compiled on March 16, 2023, and is designed to collect information about the system and browser using an embedded copy of the SQLite3 library. Details of the DLL are given below:

Name(s)	N/A
Size	1.1MB (1182208 Bytes)
File Type	application/x-dosexec
MD5	7faea2b01796b80d180399040bb69835
SHA1	3b3e778b647371262120a523eb873c20bb82beaf
SHA256	8ab3a5eaaf8c296080fadf56b265194681d7da5da7c02562953a4cb60e147423

The DLL retrieves the hostname, domain name, and OS version. Then, it will retrieve the browser history (title and URL) of the following browsers:

- Brave
- Chrome
- Edge
- Firefox

It limits the output to the first 500 entries, and it passes this data back to the ICONIC malware that then POSTs the data back to the C2. It is likely that the attacker then serves a further payload to victims of interest. Volexity has not been able to retrieve a further payload at this time.

macOS ICONIC Analysis

The macOS installer for 3CX was also compromised. The following table shows the details of this installer:

Name(s)	3CXDesktopApp-18.12.416.dmg|3CXDesktopApp-latest.dmg
Size	164.2MB (172150545 Bytes)
File Type	Macintosh Disk Image
MD5	d5101c3b86d973a848ab7ed79cd11e5a
SHA1	3dc840d32ce86cebf657b17cef62814646ba8e98
SHA256	e6bbc33815b9f20b0cf832d7401dd893fbc467c800728b5891336706da0dbcec

The backdoor component is *libffmpeg.dylib* located in */Contents/Frameworks/Electron Framework.framework/Versions/A/Libraries*. It is worth noting that this is the equivalent of the same library that was abused in the Windows binary.

Name(s)	libffmpeg.dylib
Size	4.7MB (4979136 Bytes)
File Type	Mach-O
MD5	660ea9b8205fbd2da59fefd26ae5115c
SHA1	769383fc65d1386dd141c960c9970114547da0c2
SHA256	a64fa9f1c76457ecc58402142a8728ce34ccba378c17318b3340083eeb7acc67

The macOS version does not use GitHub to retrieve its C2 server. Instead, a list of C2 servers is stored in the file encoded with a single byte XOR key, *0x7A*. Below is a list of the URLs it will attempt to contact. Note that the domains largely overlap with the Windows sample, but the URIs are different.

- msstorageazure[.]com/analysis
- officestoragebox[.]com/api/biosync
- visualstudiofactory[.]com/groupcore

- azuredeploystore[.]com/cloud/images
- msstorageboxes[.]com/xbox
- officeaddons[.]com/quality
- sourceslabs[.]com/status
- zacharryblogs[.]com/xmlquery
- pbxcloudeservices[.]com/network
- pbxphonenetwork[.]com/phone
- akamaitechcloudservices[.]com/v2/fileapi
- azureonlinestorage[.]com/google/storage
- msedgepackageinfo[.]com/ms-webview
- glcloudservice[.]com/v1/status
- pbxsources[.]com/queue
- www.3cx[.]com/blog/event-trainings/

It is interesting to note that IDA Pro is confused by the main malicious function used in the macOS malware. The decompiled pseudocode hides most of the features. This means analysts relying on this view may miss the malicious functionality. Figure 1 shows the pseudocode of the function in IDA Pro. Figure 2 shows the same function with Ghidra, with more than 800 lines of pseudocode.

```
1  void *__fastcall sub_48430(void *a1)
2  {
3    char *v1; // rax
4    char *v2; // r14
5    __int64 i; // rax
6    int v4; // eax
7    int v5; // ebx
8    __int128 v7; // [rsp+710h] [rbp-1CF8h]
9    __int64 v8; // [rsp+720h] [rbp-1CE8h]
10   char __big[16]; // [rsp+1330h] [rbp-10D8h] BYREF
11   time_t v10[2]; // [rsp+1B40h] [rbp-8C8h] BYREF
12   __int128 v11; // [rsp+1B50h] [rbp-8B8h]
13   __int128 v12; // [rsp+1B60h] [rbp-8A8h]
14   __int16 v13; // [rsp+1B70h] [rbp-898h]
15
16   v1 = getenv("HOME");
17   if ( v1 )
18   {
19     v2 = v1;
20     v12 = xmmword_20A150;
21     v11 = xmmword_20A140;
22     *(_OWORD *)v10 = xmmword_20A130;
23     v13 = 31241;
24     for ( i = 0LL; i != 50; ++i )
25       *((_BYTE *)v10 + i) ^= 0x7Au;
26     __bzero(__big, 512LL);
27     sprintf(__big, (const char *)v10, v2, ".session-lock");
28     v4 = open(__big, 514, 384LL);
29     if ( v4 != -1 )
30     {
31       v5 = v4;
32       v8 = 0x300000000LL;
33       v7 = 0LL;
34       fcntl(v4, 7);
35       close(v5);
36     }
37   }
38   return 0LL;
39 }
```

Figure 1. IDA Pro Pseudo Code

```
/* WARNING: Could not reconcile some variable overlaps */

undefined8 UndefinedFunction_00048430(void)

{


  lStack_38 = *(long *)__got::___stack_chk_guard;
  pcVar11 = __stubs::_getenv("HOME");
  if (pcVar11 == (char *)0x0) goto LAB_00048965;
  uStack_8a8 = 0x3e5a2239;
  uStack_8a4 = 0xe11091f;
  uStack_8a0 = 0x3b5a0a15;
  uStack_89c = 0x5f550a0a;
  uStack_8b8 = 0x130e1b19;
  uStack_8b4 = 0x295a1415;
  uStack_8b0 = 0x150a0a0f;
  uStack_8ac = 0x49550e08;
  uStack_8c8 = (undefined *)0x1b0818133655095f;
  uStack_8c0 = 0x3b550308;
  uStack_8bc = 0x13160a0a;
  uStack_898 = 0x7a09;
  lVar12 = 0;
  do {
    *(byte *)((long)&uStack_8c8 + lVar12) = *(byte *)((long)&uStack_8c8 + lVar12) ^ 0x7a;
    lVar12 = lVar12 + 1;
  } while (lVar12 != 0x32);
  __stubs::___bzero(&uStack_10d8,0x200);
  __stubs::_sprintf((char *)&uStack_10d8,(char *)&uStack_8c8,pcVar11,".session-lock");
  iVar7 = __stubs::_open((char *)&uStack_10d8,0x202,0x180);
  if (iVar7 == -1) goto LAB_00048965;
  uStack_1ce8 = 0x300000000;
  auStack_1cf8 = ZEXT816(0);
  iVar8 = __stubs::_fcntl(iVar7,7);
  if ((iVar8 != -1) && (uStack_1ce8._4_2_ == 2)) {
    __stubs::_flock(iVar7,2);
    __stubs::___bzero(acStack_12d8,0x200);
    auStack_12e8 = ZEXT816(0);
    auStack_12f8 = ZEXT816(0);
    __stubs::___bzero(auStack_14f8,0x200);
    __stubs::___bzero(auStack_1cf8,0x800);
    auStack_1d08 = ZEXT816(0);
[...redacted...]
    lVar12 = 0;
    do {
      *(byte *)((long)&uStack_2228 + lVar12) = *(byte *)((long)&uStack_2228 + lVar12) ^ 0x7a;
      lVar12 = lVar12 + 1;
    } while (lVar12 != 0x2f);
    lVar12 = 0;
    __stubs::_sprintf(auStack_1df8,(char *)&uStack_2228,pcVar11);
    __stubs::_sprintf(acStack_1ff8,"%s/.main_storage",auStack_1df8);
    __stubs::_sprintf(acStack_21f8,"%s/UpdateAgent",auStack_1df8);
    __stubs::_gethostname(acStack_12d8,0x200);
    __stubs::___bzero(&uStack_10d8,0x801);
    uStack_8a8 = 0x2c171f0e;
    uStack_8a4 = 0x1309081f;
    uStack_8a0 = 0xa541415;
    uStack_89c = 0xe091316;
    uStack_8b8 = 0x1f081539;
    uStack_8b4 = 0xc081f29;
    uStack_8b0 = 0x91f1913;
    uStack_8ac = 0x9032955;
    uStack_8c8 = (undefined *)0x55171f0e09032955;
    uStack_8c0 = 0x8181336;
    uStack_8bc = 0x5503081b;
    uStack_898 = CONCAT11(uStack_898._1_1_,0x7a);
    do {
      *(byte *)((long)&uStack_8c8 + lVar12) = *(byte *)((long)&uStack_8c8 + lVar12) ^ 0x7a;
      lVar12 = lVar12 + 1;
    } while (lVar12 != 0x31);
    pFVar13 = (FILE *)__stubs::_fopen$DARWIN_EXTSN(&uStack_8c8);
    if (pFVar13 != (FILE *)0x0) {
      __stubs::_fseek(pFVar13,0,2);
      sVar14 = __stubs::_ftell(pFVar13);
      __stubs::_rewind(pFVar13);
      sVar17 = 0x800;
      if ((long)sVar14 < 0x800) {
        sVar17 = sVar14;
      }
      __stubs::_fread(&uStack_10d8,sVar17,1,pFVar13);
      pcVar11 = __stubs::_strstr((char *)&uStack_10d8,"<key>ProductVersion</key>");
      if (pcVar11 != (char *)0x0) {
        pcVar15 = __stubs::_strstr(pcVar11,"<string>");
        pcVar11 = __stubs::_strstr(pcVar11,"</string>");
        if (((pcVar15 != (char *)0x0) && (pcVar11 != (char *)0x0)) && (pcVar15 + 8 < pcVar11)) {
          __stubs::_strncpy(auStack_12f8,pcVar15 + 8,(size_t)(pcVar11 + (-8 - (long)pcVar15)));
          __stubs::_fclose(pFVar13);
          _auStack_2248 = ZEXT816(0);
          _auStack_2258 = ZEXT816(0);
          _auStack_2268 = ZEXT816(0);
          puStack_2238 = (undefined *)0x0;
[...redacted...]
```

```
94                  } while ((long)uStack_8c8 - (long)uStack_10d8 < (long)uVar26);
95                }
96              }
97            }
98            pcStack_2350 = "stringWithCString:encoding:";
99            pcStack_2358 = "URLWithString:";
100           pcStack_2360 = "requestWithURL:";
101           pcStack_23c8 = "addValue:forHTTPHeaderField:";
102           pcStack_2368 = "setCachePolicy:";
103           pcStack_2370 = "setHTTPShouldHandleCookies:";
104           pcStack_2378 = "setTimeoutInterval:";
105           pcStack_2380 = "ephemeralSessionConfiguration";
106           pcStack_2388 = "setConnectionProxyDictionary:";
107           pcStack_2390 = "setWaitsForConnectivity:";
108           pcStack_2398 = "setTimeoutIntervalForResource:";
109           pcStack_23a0 = "sessionWithConfiguration:";
110           pcStack_23a8 = "dataTaskWithRequest:completionHandler:";
111           pcStack_23b0 = "resume";
112           pcStack_23b8 = "invalidateAndCancel";
113           do {
114             tVar16 = __stubs::_time((time_t *)0x0);
115             __stubs::_srand((uint)tVar16);
```

Figure 2. Ghidra pseudo code (totaling more than 800 lines)

The malware randomly picks one of the servers from the list to retrieve the next stage. As with the Windows version of the malware, a specially formatted cookie must be included in the web request to retrieve a further payload:

```
3cx_auth_id=%s;3cx_auth_token_content=%s;__tutma=true
```

The user-agent is also hardcoded and may be used by the attacker to filter valid requests:

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.128
```

Volexity was not able to retrieve the next stage from the C2 servers, as the upstream C2 infrastructure had stopped responding by the time it made requests.

Infrastructure & Attribution

In terms of attacker infrastructure, the domains used in these attacks are hosted on shared infrastructure and appear to simply proxy requests to an unknown upstream C2. Domains were registered with several providers, including NameCheap, Public Domain Registry, and NameSilo. Some of the domains were not registered using WHOIS protection, and each was registered using a unique email address. The following emails were observed in WHOIS records for related domains:

- cliego.garcia@proton[.]me
- remey.simpson@outlook[.]com
- jackiewcaudill@gmail[.]com
- philip.je@proton[.]me
- haroldjmarable@gmail[.]com

In terms of attribution, the original CrowdStrike post suggests the incident is related to LABYRINTH CHOLLIMA, which is related to the public Lazarus moniker (although Volexity does not have visibility of exactly which parts of Lazarus this maps to). Volexity cannot currently map the disclosed activity to any threat actor, so it will be tracked under UTA0040.

Conclusion & Mitigations

Volexity's analysis concludes that both the Windows and macOS installers for the 3CX desktop application had malicious code inserted into them before being provided to customers. This suggests that 3CX was itself compromised by the threat actor for a period of time prior to the infection, allowing the attacker to develop an understanding, access, and malicious code for the development-update process of the company.

The end result for victims of this campaign was that information-stealing malware was installed on endpoints that installed this update, and for selected victims, an additional arbitrary payload may also have been delivered.

Supply chain attacks are relatively rare due to the high level of technical and operational capability required for success. However, organizations with a large customer base, such as 3CX, are attractive targets due to the broad level of access these attacks can grant threat actors.

Volexity assesses that it is likely UTA0040 is a nation-state-backed threat actor based on the level of capabilities utilized in this campaign, combined with a perceived intent to gather information from victims for further targeted compromise. Crimeware-based threat groups who have historically conducted supply chain attacks typically push ransomware payloads immediately with their access, rather than try to conduct reconnaissance to filter victims of their true payload. While Volexity cannot attribute this cluster to any known group at this time, CrowdStrike has attributed this activity to LABYRINTH CHOLLIMA, a North Korean group.

Supply chain attacks are complex issues for defenders to defend against. This instance highlights how large code bases can be backdoored with minor additions to existing code and remain undetected by the software provider and the end user. However, information in the public domain highlights the value of endpoint and network detection capabilities, which provided valuable identification of anomalous behavior that may have prevented further impact for end users.

The infrastructure registration and public artifacts (notably the GitHub page) suggests that the attacker had access to the software provider at least as early December 2022, and perhaps as early as November 2022. It is not clear when the first malicious update described in this post was downloaded by victims of this campaign, but the public discussions around detections suggest this could be as early as March 22, 2023. This does not rule out other potentially malicious activities having occurred before this time related to this software.

To detect and investigate these attacks such as these, Volexity recommends the following:

- Use the YARA rules provided here to detect related activity.
- Use the provided Suricata rules here to detect related activity. It should be noted that these requests take place over HTTPs, meaning they are only effective if this traffic is being decrypted prior to matching.
- Block the IOCs provided here.

Appendix A – Third-Party Reporting

There is a great deal of third-party reporting on this subject covering various aspects of the campaign. A list of resources is provided below, note that this list was compiled on March 30, 2023, and inevitably more resources will become available after publication.

Social Media Posts

- https://twitter.com/cyb3rops/status/1641130326830333984 (https://github.com/SigmaHQ/sigma/pull/4151/files, https://github.com/Neo23x0/signature-base/blob/master/yara/gen_mal_3cx_compromise_mar23.yar)
- https://twitter.com/cyb3rops/status/1641339448053858304
- https://twitter.com/patrickwardle/status/1641294247877021696
- https://twitter.com/fr0gger_/status/1641325932760948737
- https://twitter.com/donnymaasland/status/1641349104113524736
- https://twitter.com/jamesspi/status/1641262032870686721
- https://twitter.com/dez_/status/1641204732445478912
- https://twitter.com/vxunderground/status/1641261800594210817

CrowdStrike

- https://www.crowdstrike.com/blog/crowdstrike-detects-and-prevents-active-intrusion-campaign-targeting-3cxdesktopapp-customers/
- https://www.reddit.com/r/crowdstrike/comments/125r3uu/20230329_situational_awareness_crowdstrike/

Sophos

- https://news.sophos.com/en-us/2023/03/29/3cx-dll-sideloading-attack/

SentinelOne

- https://www.sentinelone.com/blog/smoothoperator-ongoing-campaign-trojanizes-3cx-software-in-software-supply-chain-attack/

Symantec

- https://twitter.com/threatintel/status/1641339467398017024

3CX

- https://www.3cx.com/blog/news/desktopapp-security-alert/

Objective-See

- https://objective-see.org/blog/blog_0x73.html

Bleeping Computer

- https://www.bleepingcomputer.com/news/security/hackers-compromise-3cx-desktop-app-in-a-supply-chain-attack/

Huntress

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

---

** Beginning in December 2022, Volexity began to use the following schema to refer to smaller unclassified clusters of threat activity not significant enough to warrant their own name: UTAXXXX. In this schema, "UTA" refers to "Unclassified Threat Actor", and the numbers that follow are a unique identifier for that group of activity.*
