Threat Actors abuse signed ConnectWise application as malware builder

gdatasoftware.com/blog/2025/06/38218-connectwise-abuse-malware

Techblog

Since March 2025 there has been a noticeable increase in infections and fake applications using validly signed ConnectWise samples. We reveal how bad signing practices allow threat actors to abuse this legitimate software to build and distribute their own signed malware and what security vendors can do to detect them.

Analysis by Lance Go and Karsten Hahn

ConnectWise abuse 2024-2025

This isn't the first time that ConnectWise has been used by threat actors. Back in February 2024, we saw a spike in ransomware activity tied to two ConnectWise vulnerabilities: CVE-2024-1708 and CVE-2024-1709.

Around March 2025, a new wave of ConnectWise abuse started showing up, now being tracked under the name "EvilConwi".

When people suspect an infection, they often turn to the Internet for help. "UNITE against malware" forums (such as BleepingComputer.com) provide disinfection assistance in such cases. Several threads on BleepingComputer's forums (link1, link2) show unwanted ConnectWise clients as the culprit of the infection, usually with phishing emails as the starting point. The existence of several posts like these indicates a failure of security programs to prevent the threat. Even in May 2025 most antivirus products did not detect maliciously used ConnectWise samples as malware.

In one BleepingComputer case the origin of infection is a phishing email with a OneDrive link that promises to show a large document. The link redirects to a Canva page with a "View PDF" button which downloads and runs a ConnectWise installer. The user describes "fake Windows Update screens" and their mouse "moving on its own randomly". Aside from those indicators, there were no other visible signs for the active remote connection (sample[1]).

Reddit users have also reported similar incidents, for example in one case, a maliciously crafted ConnectWise sample[2] originated from a website offering an AI-based image converter. According to the original poster, the site had been advertised on Facebook.

Sample comparison

To figure out detection opportunities and settings location, we analyze the difference between two ConnectWise samples.

The images below show PortexAnalyzer reports for two ConnectWise samples[6][7] which we compare with Meld.

data directory	rva	-> offset
import table	0x129c4	0x10fc4
resource table	0x16000	0x12000
certificate table	0x546200	0x542200
base relocation table	0x54a000	0x545200
debug	0x11f20	0x10520
load config table	0x11e60	0x10460
IAT	0xd000	0xb600

data directory	rva	-> offset
import table	0x129c4	0x10fc4
resource table	0x16000	0x12000
certificate table	0x546200	0x542200
base relocation table	0x54a000	0x545200
debug	0x11f20	0x10520
load config table	0x11e60	0x10460
IAT	0xd000	0xb600

Figure1: The data directory comparison shows that the size of the certificate table is different in both samples

```
MD5:     dcf7aef6177e82202d4a9db07392b622
SHA256:  6d9442ae6ba5a9f34a47e234b6047f61d8ac129e269
ImpHash: 9771ee6344923fa220489ab01239bdfd
Rich:    a498773e40e07b36d0f4a2dab1633f09
RichPV:  81a811df476a4793b76f670729b49596

Section      Type      Hash Value
-------------------------------------------------
1. .text     MD5       d9fa6da0baf4b869720be8332234
             SHA256    eaba38650152f8688eed3ed2c4383
2. .rdata    MD5       8b45a1035c0de72f910a75db7749f
             SHA256    8d80004988f9a0ec5e1d00c2f0d11
3. .data     MD5       1f4cc86b6735a74429c9d1feb93e2
             SHA256    84a7f490102ace5e46c847381c8d5
4. .rsrc     MD5       d813d73373778ed5b0a4b71b25237
             SHA256    3062dfe0b1f9c3d0b4f3564a8c3a3
5. .reloc    MD5       a93b0f39998e1e69e5944da8c5ff0
             SHA256    e98540b66036ea262721678c35947

--- end of report ---
```

```
MD5:     f729d82c4d34ce7c1c8e821294900ac0
SHA256:  277ef6c0dcaf0e76291fbde0199dda1ca521c03e77d
ImpHash: 9771ee6344923fa220489ab01239bdfd
Rich:    a498773e40e07b36d0f4a2dab1633f09
RichPV:  81a811df476a4793b76f670729b49596

Section      Type      Hash Value
-------------------------------------------------
1. .text     MD5       d9fa6da0baf4b869720be8332234
             SHA256    eaba38650152f8688eed3ed2c4383
2. .rdata    MD5       8b45a1035c0de72f910a75db7749f
             SHA256    8d80004988f9a0ec5e1d00c2f0d11
3. .data     MD5       1f4cc86b6735a74429c9d1feb93e2
             SHA256    84a7f490102ace5e46c847381c8d5
4. .rsrc     MD5       d813d73373778ed5b0a4b71b25237
             SHA256    3062dfe0b1f9c3d0b4f3564a8c3a3
5. .reloc    MD5       a93b0f39998e1e69e5944da8c5ff0
             SHA256    e98540b66036ea262721678c35947

--- end of report ---
```

Figure2: This comparison shows the same hashes for every section but different hashes for the whole file

Aside from the certificate table in the overlay, the section contents have the same hashes. We confirmed with a binary diffing tool that the only substantial differences reside in the certificate table.

Thus, any customization that we could use to distinguish ConnectWise installers from each other must reside in the certificate table. At this point we suspect Authenticode stuffing.

Authenticode stuffing

Authenticode stuffing is deliberate misuse of the certificate structure that allows modifications to an executable without invalidating its signature. Developers use this technique to avoid re-signing their applications for minor changes. It's a relatively common practice, applications like Dropbox use it. Some installers[3], for example, track installation statistics by saving user agents, referrers, campaign IDs or similar data from the browser's cookies in the certificate shortly before the file is downloaded. In such cases, the Authenticode stuffing is harmless because it does not influence the sample's behavior.

There are various ways to abuse authenticode signing. To figure out which method ConnectWise uses, we run an authenticode linter on both samples.

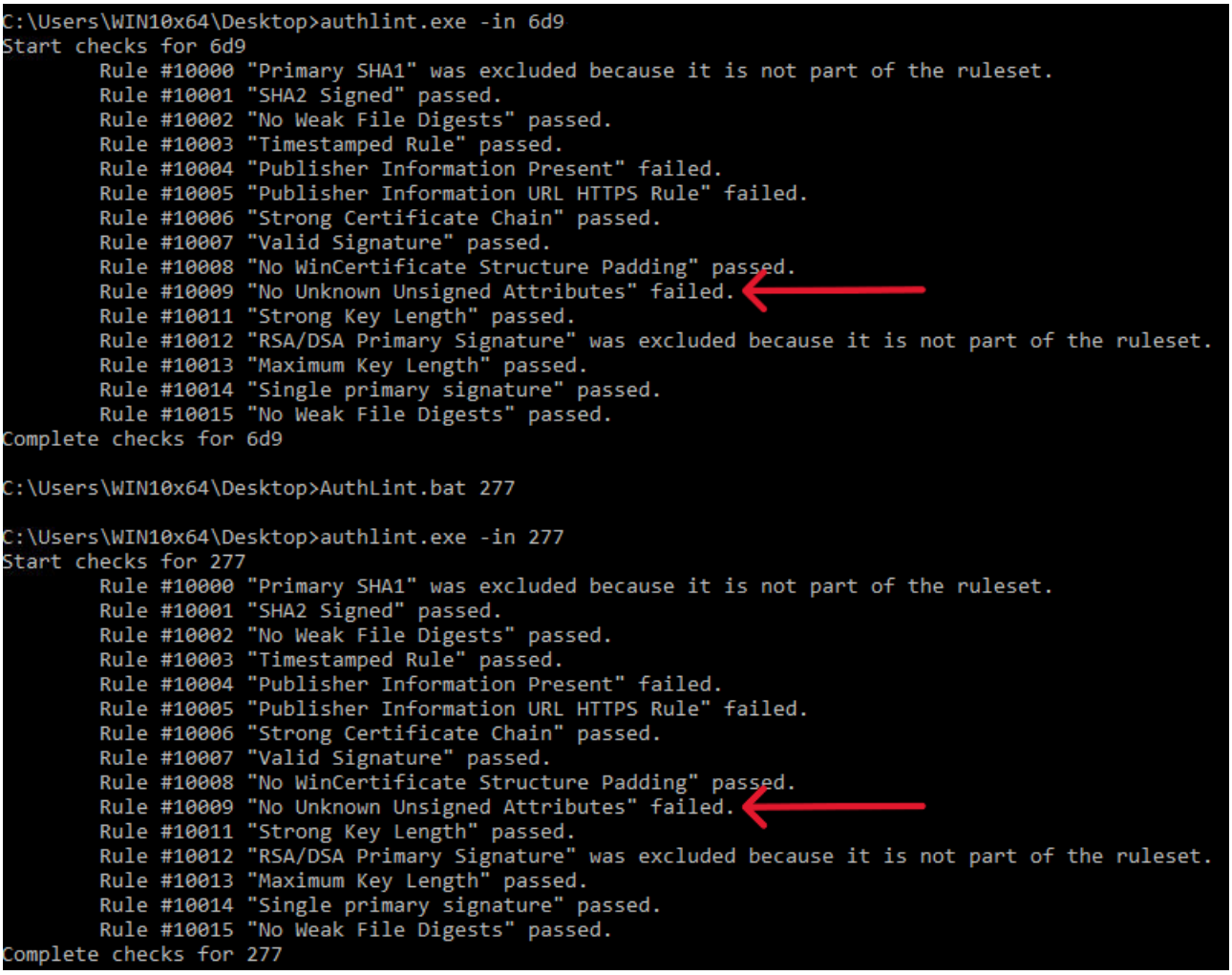

Figure3: output of AuthenticodeLint tool

The linter output shows that the “No Unknown Unsigned Attributes” check fails for both samples. That means ConnectWise has unauthenticated attributes which should not be there.

The following image shows the structure of a signed Portable Executable file, and where the unauthenticated attributes reside in the certificate table. The original image is from Microsoft’s official Windows Authenticode PE Signature Format document.

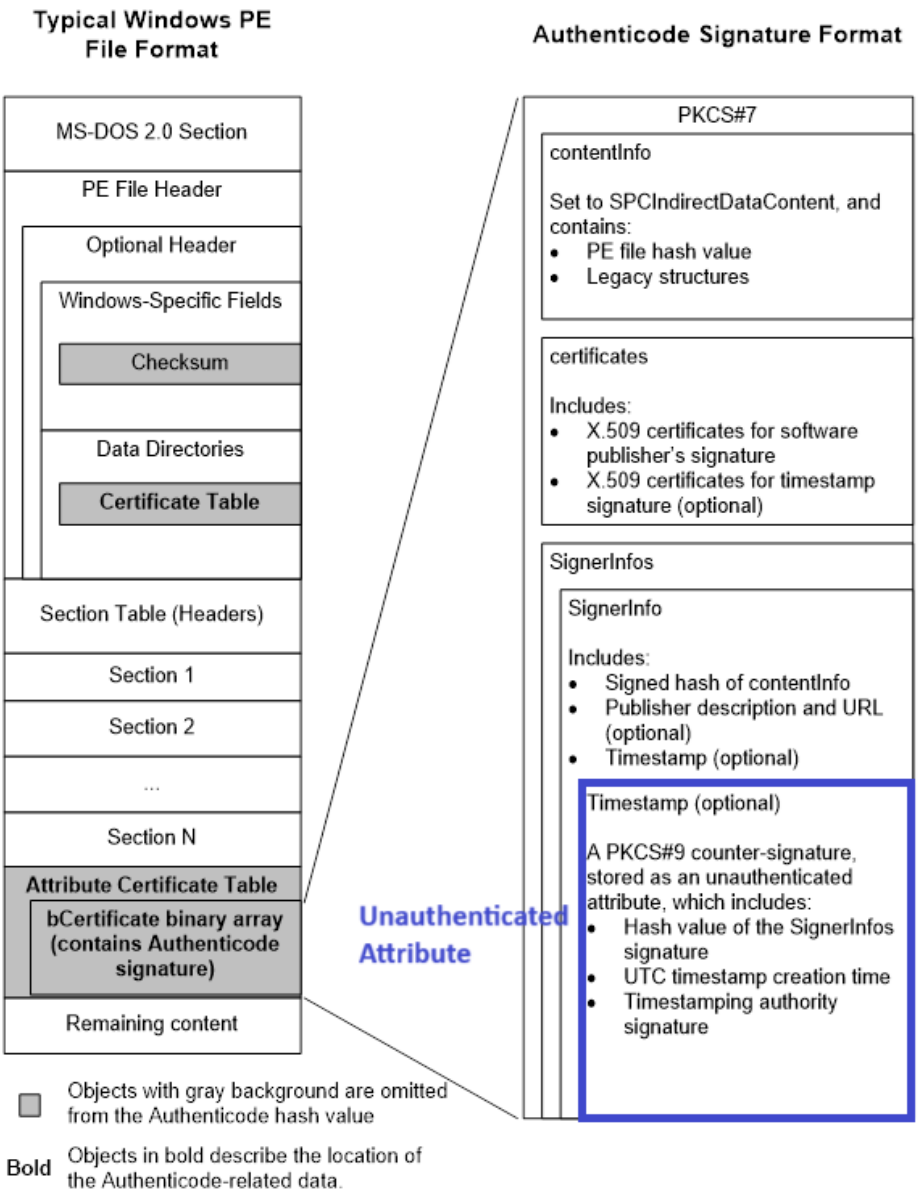

Figure 4: Windows Authenticode PE signature format

To verify the certificate of a Portable Executable file, Windows compares the authentihash in the certificate with the actual hash of the file. If these hashes are different, the verification fails, and the file is not validly signed anymore.

Windows calculates the authentihash on the file's contents except for the grayed-out areas on the left side of the image. That means the checksum in the Optional Header, the certificate table entry in the Optional Header, and the certificate table itself are omitted for the authentihash calculation. This includes unauthenticated attributes. They won't impact the validity of the certificate.

The right side of the image shows the certificate table structure. Unauthenticated attributes usually save timestamps but can also save arbitrary data.

Because we assume that ConnectWise uses unauthenticated attributes for Authenticode stuffing, we create a Python script to extract unauthenticated attributes from PE files.

```
import pefile
from asn1crypto import cms
import sys
import os

def dump_certificate_table(file_path, output_dir):
    pe = pefile.PE(file_path, fast_load=True)

    security_dir_entry = pe.OPTIONAL_HEADER.DATA_DIRECTORY[
        pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_SECURITY']
    ]
    if security_dir_entry.VirtualAddress == 0:
        print("No certificate table found.")
        return

    cert_table_offset = security_dir_entry.VirtualAddress
    cert_table_size = security_dir_entry.Size

    cert_data = pe.__data__[cert_table_offset: cert_table_offset + cert_table_size]

    offset = 0
    attr_counter = 0

    os.makedirs(output_dir, exist_ok=True)

    while offset < len(cert_data):
        length = int.from_bytes(cert_data[offset:offset+4], 'little')
        revision = int.from_bytes(cert_data[offset+4:offset+6], 'little')
        cert_type = int.from_bytes(cert_data[offset+6:offset+8], 'little')

        cert_blob = cert_data[offset+8: offset+length]

        try:
            content_info = cms.ContentInfo.load(cert_blob)

            if content_info['content_type'].native != 'signed_data':
                print("Not a SignedData structure, skipping.")
                offset += ((length + 7) & ~7)
                continue

            signed_data = content_info['content']

            for signer_info in signed_data['signer_infos']:
                unsigned_attrs = signer_info['unsigned_attrs']
                if unsigned_attrs is not None:
                    print("\nUnauthenticated Attributes")
                    for attr in unsigned_attrs:
                        print("\nNext Attribute")
                        print(f"Attribute Type OID: {attr['type'].dotted}")
                        print(f"Attribute Type Name: {attr['type'].native}")

                        for value in attr['values']:
                            attr_counter += 1
                            filename = os.path.join(output_dir, f"attr_{attr_counter}.bin")
                            with open(filename, "wb") as f:
                                # Dump raw bytes
                                f.write(value.dump())
                            print(f"Value dumped to: {filename}")
                else:
                    print("No unauthenticated attributes found in this signer.")

        except Exception as e:
            print(f"Failed to parse certificate blob: {e}")

        offset += ((length + 7) & ~7)
```

```
if __name__ == "__main__":
    if len(sys.argv) != 3:
        print(f"Usage: {sys.argv[0]} <input_pe_file> <output_directory>")
        sys.exit(1)

    input_file = sys.argv[1]
    output_directory = sys.argv[2]

    dump_certificate_table(input_file, output_directory)
```

ConnectWise configuration abuse

We built a configuration dumper that extracts settings and embedded files from the certificate. While we do not share the script for legal reasons, most of the meaningful data that is useful for threat detection is saved in XML format and directly visible in dumped attributes (using the Python script above) or a strings listing of the sample.

Here is an example output of the configuration dumper for a malicious sample[4]:

```
Processing: cb8a1a1e90c29461b0503e2c5deac7b673617477128ee3baea4d8134676c8af4


========= Parsed Client Data =========

Is Silent: False

Components to Exclude: 0x0


--- Instance Identifier Blob ---

0602000000a400005253413100080000010001000051e42d081b84623245c42be884c3217fb43dd511451b1fbdb5019df867e7672a3e
6c5dce4fd41025f2d4ed931d3978f52105bbe293b3be6f4ba476bcdaf1806d456ffa6cc64e005958582793330c3f4dc22385d8afff1
e15a1437daa3c3b11c3ad5078b288072a7c42fa0383b5b8f17b4ddab832bc501698714cbfffa569636c46395d0d4c37171a33c710bd
b353335d93222a1dad1a26ae65c55a6711bb95a3559826db8c1dacf36df5705002433b7e2967a2b5d1975eb5ada96e627b73b9b62fb
915835897b03d61683c68c92e3c77b264990941a9504461549c4a3d07c9f70207525262f8d20c907802a3cfb5dc7fb6886dac164c10
80a3425b87f1d10d2


--- Launch Parameters ---

Launch Parameters: ?
e=Access&y=Guest&h=bookinginvoiceview.top&p=8041&k=BgIAAACkAABSU0ExAAgAAAEAAQAFHkLQgbhGIyRcQr6ITDIX%2b0PdUR
RRsfvbUBnfhn52cqPmxdzk%2fUECXy1O2THTl49SEFu%2bKTs75vS6R2vNrxgG1Fb%2fpsxk4AWVhYJ5MzDD9NwiOF2K%2f%2fHhWhQ32qP
DsRw61QeLKIByp8QvoDg7W48XtN2rgyvFAWmHFMv%2f%2blaWNsRjldDUw3FxozxxC9s1MzXZMiKh2tGiauZcVaZxG7laNVmCbbjB2s8231
cFACQzt%2bKWeitdGXXrWtqW5ie3O5ti%2b5FYNYl7A9YWg8aMkuPHeyZJkJQalQRGFUnEo9B8n3AgdSUmL40gyQeAKjz7Xcf7aIbawWTBC
Ao0Jbh%2fHRDS&c=Property%20S%20New&c=Working%20new&c=Property%20Working%20new&c=Working%20Property&c=&c=&c=
&c=

Launch Constraints: ?
h=bookinginvoiceview.top&p=8041&k=BgIAAACkAABSU0ExAAgAAAEAAQAFHkLQgbhGIyRcQr6ITDIX%2b0PdURRRsfvbUBnfhn52cqP
mxdzk%2fUECXy1O2THTl49SEFu%2bKTs75vS6R2vNrxgG1Fb%2fpsxk4AWVhYJ5MzDD9NwiOF2K%2f%2fHhWhQ32qPDsRw61QeLKIByp8Qv
oDg7W48XtN2rgyvFAWmHFMv%2f%2blaWNsRjldDUw3FxozxxC9s1MzXZMiKh2tGiauZcVaZxG7laNVmCbbjB2s8231cFACQzt%2bKWeitdG
XXrWtqW5ie3O5ti%2b5FYNYl7A9YWg8aMkuPHeyZJkJQalQRGFUnEo9B8n3AgdSUmL40gyQeAKjz7Xcf7aIbawWTBCAo0Jbh%2fHRDS


--- Additional Files ---

  app.config                    0x970 bytes
  Client.en-US.resources        0xc3d0 bytes
  Client.Override.en-US.resources 0x1f6 bytes
  Client.Override.resources     0x1f12 bytes
  Client.resources              0x73a8 bytes
  system.config                 0x3b6 bytes

--- PNG Icons by Size ---
16px : 0xed5 bytes
32px : 0xed5 bytes
48px : 0x1d8 bytes
```

The first interesting indicator is the connection URL and the port which are part of the launch parameters as well as the silent installation flag which is set to false here. But there is more: embedded additional resources and configuration files.

One of the extracted additional files for this sample is a .NET resource named Client.Override.en-US.resources. In this sample[4] it modifies the ApplicationTitle so that ConnectWise fakes a Windows update and instructs the user not to turn off the system, probably to ensure that the remote connection remains active for some time.

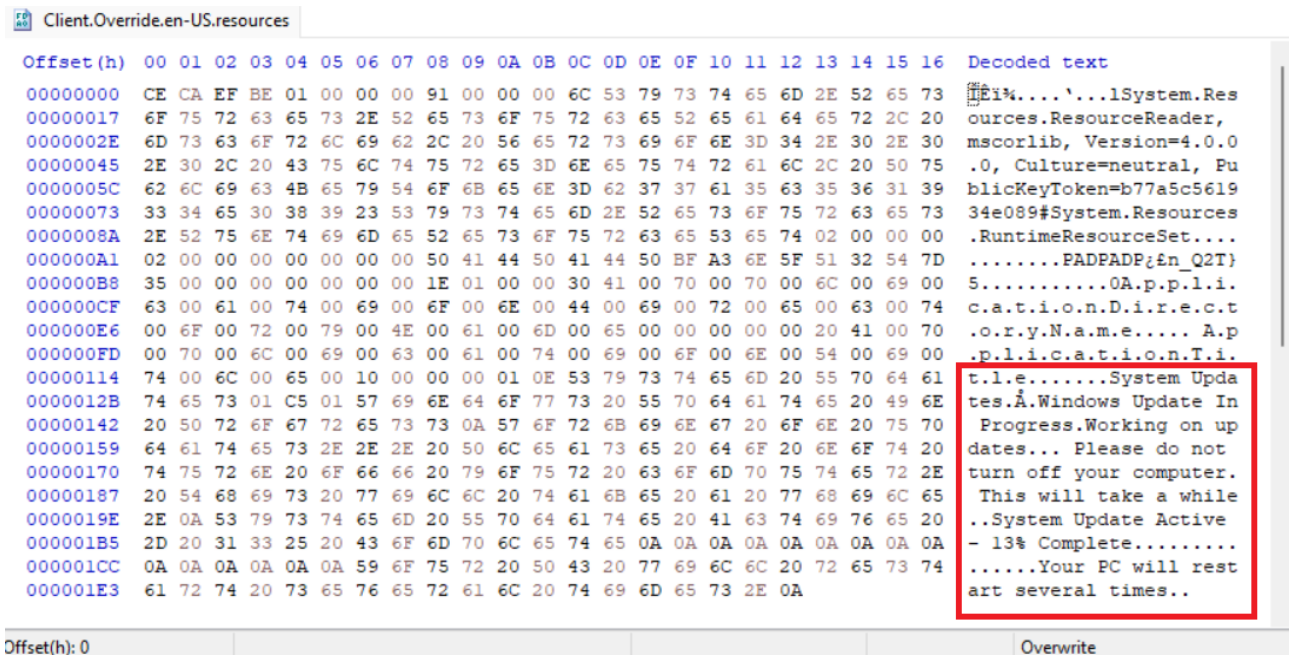

Figure 5: fake Windows update messages in a config file

Another resource named Client.Override.resources contains Google Chrome icon PNG files which override the ApplicationIcon property. Similar samples like [5] use the same file to override the property BlankMonitorBackgroundImage with a fake Windows update screen JPEG. Both images are shown below.

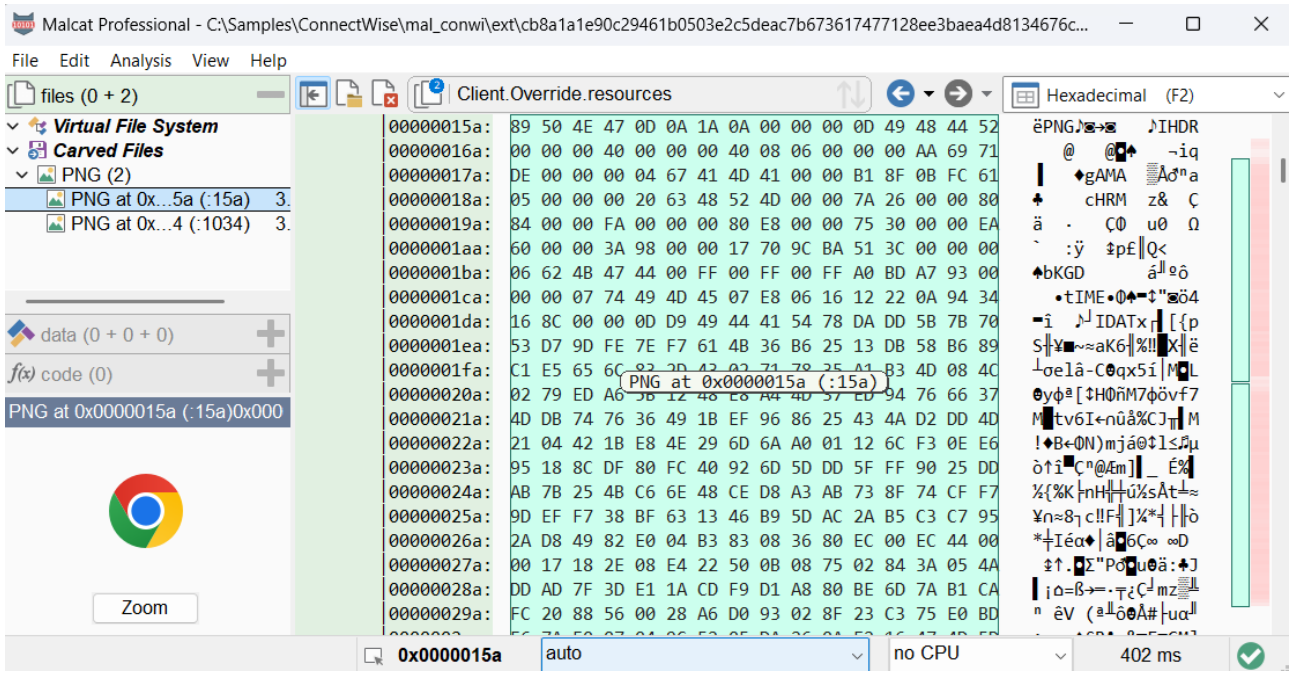

Figure 6: Configuration data that sets a Google Chrome icon, sample [4]

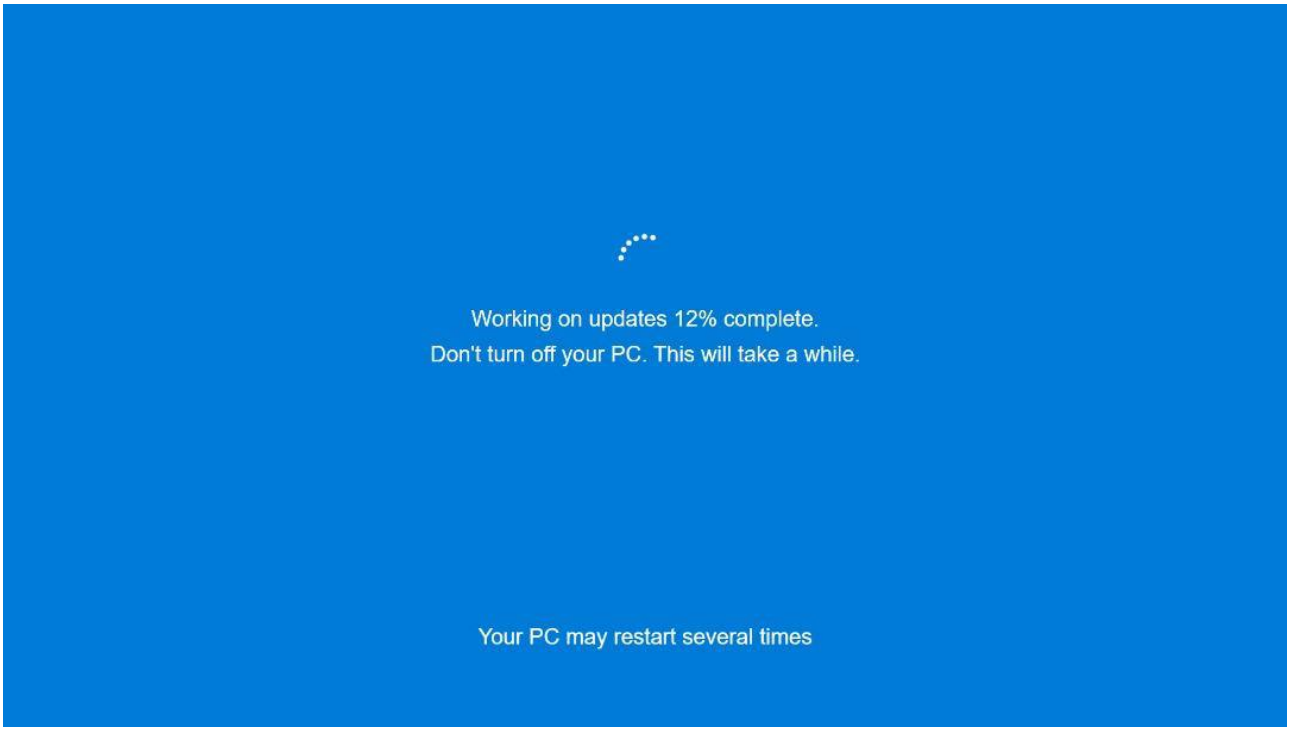

Figure7: Fake Windows update screen JPEG in ConnectWise config of sample [5], the compression artefacts are as seen in the sample

There are also two configuration files named system.config and app.config. These are XML files with more settings. The system.config usually includes another ClientLaunchParametersConstraint value—on top of the one already extracted using the config extractor—which holds the connection URL, port, and other parameters.

The app.config XML is also interesting for threat evaluation. The following image shows the contents of an app.config file that is typical for malicious ConnectWise samples:

app.config

```
    <section name="ScreenConnect.ApplicationSettings" type=
    "System.Configuration.ClientSettingsSection" />
  </configSections>
  <ScreenConnect.ApplicationSettings>
    <setting name="ShowFeedbackSurveyForm" serializeAs="String">
      <value>false</value>
    </setting>
    <setting name="SupportShowUnderControlBanner" serializeAs="String">
      <value>false</value>
    </setting>
    <setting name="AccessShowUnderControlBanner" serializeAs="String">
      <value>false</value>
    </setting>
    <setting name="SupportHideWallpaperOnConnect" serializeAs="String">
      <value>false</value>
    </setting>
    <setting name="AccessHideWallpaperOnConnect" serializeAs="String">
      <value>false</value>
    </setting>
    <setting name="HideWallpaperOnConnect" serializeAs="String">
      <value>false</value>
    </setting>
    <setting name="AccessShowBalloonOnConnect" serializeAs="String">
      <value>false</value>
    </setting>
    <setting name="AccessShowBalloonOnHide" serializeAs="String">
      <value>false</value>
    </setting>
    <setting name="AccessShowSystemTrayIcon" serializeAs="String">
      <value>false</value>
    </setting>
  </ScreenConnect.ApplicationSettings>
</configuration>
```

Figure 6: app.config with remote connection indicators set to false

This app.config disables several indicators which would alert a user that ConnectWise is present like a tray icon or a black wallpaper during an active connection.

To summarize, the settings in the certificate table of ConnectWise substantially influence the behavior of ConnectWise installers and clients. Among others the certificate saves:

- Silent installation option
- Launch parameters which include connection URL and Port
- Application icons
- Messages and window titles shown to the user
- Images used by the software, such as background images
- Indicators that show the presence of an active connection

By modifying these settings, threat actors create their own remote access malware that pretends to be a different software like an AI-to-image converter by Google Chrome. They commonly add fake Windows update images and messages too, so that the user does not turn off the system while threat actors remotely connect to them.

Threat prevention recommendations

We recommend fellow defenders disallowing any ConnectWise samples that have several of the following app.config settings set to false (using regex syntax):

- (Support|Access)?HideWallpaperOnConnect
- (Support|Access)?ShowBalloonOnHide
- (Support|Access)?ShowBalloonOnConnect
- (Support|Access)?ShowSystemTrayIcon
- (Support|Access)?ShowCloseDialogOnExit

A Yara rule may look as follows:

```
rule SilentConwi : Riskware PUP
{
        meta:
                author = "Karsten Hahn @ G DATA CyberDefense"
                description = "Settings from app.config that hide the connection of the client. These
settings are potentially unwanted"
                sha256 = "1fc7f1ef95f064b6c6f79fd1a3445902b7d592d4ff9989175b7caae66dd4aa50"

        strings:
                $rex_01 = /<setting name="(Access|Support)?ShowBalloonOnConnect"[^>]*>\s*
<value>false<\/value>/
                $rex_02 = /<setting name="(Access|Support)?HideWallpaperOnConnect"[^>]*>\s*
<value>false<\/value>/
                $rex_03 = /<setting name="(Access|Support)?ShowBalloonOnHide"[^>]*>\s*
<value>false<\/value>/
                $rex_04 = /<setting name="(Access|Support)?ShowSystemTrayIcon"[^>]*>\s*
<value>false<\/value>/

        condition:
                all of them
}
```

We also recommend detecting fake application titles, fake icons and fake background images that are embedded in .NET resources within the certificate.

GDATA products detect maliciously abused ConnectWise samples as Win32.Backdoor.EvilConwi.* and samples with questionable settings as Win32.Riskware.SilentConwi.*

Vendor practices and end-user risk

Although authenticode stuffing is common practice, ConnectWise's decision to influence critical behavior and its user interface with unauthenticated attributes is clearly dangerous. It entices threat actors to build their own remote access malware with custom icons, background images and text, that is signed by a trusted company.

Given how widely (ab-)used ConnectWise's ScreenConnect is, it is a good idea to keep an eye out for these samples. Until ConnectWise changes their authenticode stuffing practices, the possibility of signed malware being created and distributed remains a threat.

On June 12, we contacted ConnectWise prior to the release of this article to make them aware of the issues described above and give them the opportunity to issue a statement. We noticed on Tuesday, June 17, 2025 that the signature used to sign the samples was revoked. We have not received a statement by the time this article was released.

Update 30. June 2025

After this article got published, Ken Pyle reached out to us saying that EvilConwi might be related to the following CVEs:

1. https://www.cve.org/CVERecord?id=CVE-2023-25718
2. https://www.cve.org/CVERecord?id=CVE-2023-25719

New ConnectWise installers do not save the configuration in the certificate anymore but instead in separate files which are shipped alongside the installer.

These new installers are still abused by threat actors.

Samples referenced in this article

[1] ConnectWise from BleepingComputer
7287a53167db901c5b1221137b5a1727390579dffd7098b59e6636596b37bc27

[2] ConnectWise from Reddit
7180238578817d3d62fd01fe4e52d532c8b3d2c25509b5d23cdabeb3a37318fc

[3] Setup file with tracking data in certificate
a6fb2a4be91f6178d8ba0ca345727d1cb7995c3e4a659a68bef306c9eff4b18e

[4] ConnectWise sample with fake Windows update messages and Chrome icons in config
cb8a1a1e90c29461b0503e2c5deac7b673617477128ee3baea4d8134676c8af4

[5] ConnectWise sample with fake Windows update screen in config
28f46446d711208aa7686cdaea60d3a31e2b37b08db7cfb0ce350fcd357a0236

[6] ConnectWise sample used for comparison
6d9442ae6ba5a9f34a47e234b6047f61d8ac129e269199793ebb0bed1ad7e3ba

[7] ConnectWise sample used for comparison
277ef6c0dcaf0e76291fbde0199dda1ca521c03e77dc56c54f5b9af8508e6029

Sample hashes sorted by infection vectors

Based on collected samples and their naming schemes, we observed certain patterns as possible infection vectors.

Fake installers

540c9ae519ed2e7738f6d5b88b29fb7a86ebfce67914691ce17be62a9b228e0a, ZoomInstallerFull.exe

55a228f22f68b8a22967cc5b8b2fcbea66fcaf77bebedfb1f89cd134a0268653, zoom_meetingconnect.exe

C0c48de11bc4b70fb546b9a76b6126a355c0a0f4b45ed6b6564d8f3146c9f0af, ZoomInstaller-x64.exe

67b909bbcce486baba59d66e3b4ec4c74dd64782051a41198085a5b3450d00c9, OneDriveSetup.exe

b1c36552556a69ec4264d54be929e458c985b83bbc42fe09714c6dce825ac9a7, MicrosoftExcel.ClientSetup.exe

D37e804938cf0a11c111832b509fbecf8a0f3e9373133be108d471d45db75de8, Adobe-Update-ClientSetup.wSZQ5iHP.exe.part

b61aed288b4527b15907955c7521ff63cc0171087ac0f7fea6c7019a09c96c04, Adobe.ClientSetup_v7.-2.7.exe

6bce39b7d7552dbacbb4bdf06b76b4fed3fbb9fe4042b81be12fbdff92b8d95c, SSA Viewer.exe

Fake video or movie clients

6aa1b9f976624f7965219f1a243de2bebb5a540c7abd4d7a6d9278461d9edc11, Creation_Made_By_CanvaAI.mp4 Canva.com

8fc8727b6ddb28f76e46a0113400c541fb15581d2210814018b061bb250cc0e6, FULL_MOVIE_DOWNLOAD.exe

5da9a0d0830c641ffda6be3be7733de469418abedc6fac0cfcd76ba49f8ade2e, P0RN-vidz.Client.exe

72fe38ad67a26cfd89d1bfc744d33f80277e8eb564b5b92fdac46a9a24d845f3, P0RN-vid.Client.exe

5ccc9ef3e8f7113469f4a46c3aca3939fd53b3561a9fd8ffacd531aa520c5921, FULL_MOVIE_WATCH_NOW.exe

23ff4f91db852b07c7366a3c3b8be0bade2befccbfea7e183daadb5e31d325c0, Schau mir jetzt nackt vor der Webcam zu.exe

Fake documents

41037935246da6f43615d93912bc62811c795ea4082a2bfdbf3eda53a012666e, Social_Security_Statement_873164.exe

98e3f74b733d4d44bec7b1bf29f7b0e83299350143ff1e05f0459571cb49c238, Statement.pdf.Client.exe

d6844a6050d5f6c20a3fe12df28e53a2e46559e6c5017576022372e35ab44ff5, SSA-statement-osu5ma6.PDF`.exe

573f1eefac3079790a9ab40bdd3530ce34b1d2d1c6fa6703a5a8d81cb190a458, BarryStatementPDF.exe

F55c6160ed57a97c4f0e1c6aa6e3f8f01a966e96a99a29e609ec60e63be11889, FATURA-255441144227D55224QO02GX6QL.com

4e5cfd915f44dc263f29e1eaef82b3e2e903ba92b10f88c0eaf89fe5eab82ff5, ANFRAGE FÜR VORSCHLAG.exe

E7f9b9c9205162ddee72a7b7ff86b6524e19c7e8b51f64fdbffc8015c7e8934c, Important Document.exe