Looking for the ‘Sliver’ lining: Hunting for emerging command-and-control frameworks

microsoft.com/security/blog/2022/08/24/looking-for-the-sliver-lining-hunting-for-emerging-command-and-control-frameworks

August 24, 2022

Microsoft has observed the Sliver command-and-control (C2) framework now being adopted and integrated in intrusion campaigns by nation-state threat actors, cybercrime groups directly supporting ransomware and extortion, and other threat actors to evade detection. We’ve seen these actors use Sliver with—or as a replacement for—Cobalt Strike. Given Cobalt Strike’s popularity as an attack tool, defenses against it have also improved over time. Sliver thus presents an attractive alternative for actors looking for a lesser-known toolset with a low barrier for entry.

First made public in late 2019 and advertised to security professionals, Sliver is an open-source framework that’s available on GitHub and includes many common C2 framework features such as support for multiple simultaneous operators, multiple listener types, user-developed extensions, and payload generation. Since December 2020, we’ve observed threat actors adopting Sliver into their arsenal.

Among its adopters is the prolific ransomware-as-service (RaaS) affiliate DEV-0237. More recently, we've seen cybercrime actors historically tied to human-operated ransomware now deliver Sliver and various post-compromise tools using Bumblebee malware (also known as COLDTRAIN) as an initial access loader. Customers can learn more about Bumblebee in our Threat Analytics report available in the Microsoft 365 Defender portal.

In this blog, we share how the researchers behind Microsoft Defender Experts for Hunting analyzed Sliver and used both lab-simulated attacks and real-world threat activity to create hunting queries to surface Sliver and other C2 frameworks.

Threat hunting: Part art(ifact), all science

For security researchers, there's a distinction between hunting and detection. For novel threats, researchers try to strike a balance between high-fidelity detection rules identifying a specific, known malware family, threat actor, or class of behavior and low-fidelity hunting rules, which generate more false positives but also more generically capture a technique and its derivatives.

The following sections illustrate the art and science of how these lower-fidelity rules help threat hunters measure and contextualize suspicious observations to find novel or stealthy threats.

Sleuthing Sliver

Threat actors use C2 frameworks to manage their access to compromised hosts and networks during an intrusion. A C2 framework usually includes a server that accepts connections from implants on a compromised system, and a client application that allows the C2 operators to interact with the implants and launch malicious commands.

Many threat actors integrate public, open-source C2 framework options into their arsenal because these have a low barrier to entry and offer several advantages for attackers like low cost, ease of modification, and difficult attribution. As previously mentioned, Sliver is one such open-source framework. Although Sliver is somewhat new, the TTPs it implements are common across many frameworks.

Below are examples of how Defender Experts hunt for these TTPs to identify Sliver and other emerging C2 frameworks in customer environments.

Infrastructure

Sliver, like many C2 frameworks, supports various network protocols such as DNS, HTTP/TLS, MTLS, and TCP. It can also accept implant or operator connections and host files to impersonate a benign web server.

The first step in testing any C2 framework is starting listeners and scanning them to identify anomalies. Some common artifacts are unique HTTP header combinations and JARM hashes, the latter of which are active fingerprinting techniques for TLS servers. RiskIQ has shared such a methodology for Sliver and Bumblebee detection.

Payloads

Since Sliver is written in the Go programming language (GoLang), its implants are cross-platform compatible. By default, operators can generate implants in several formats, including:

- Shellcode
- Executable
- Shared library/DLL
- Service

Sliver also supports stagers—smaller payloads with few built-in features that are primarily intended to retrieve and launch a full implant. Stagers are used by many C2 frameworks to minimize the malicious code that's included in an initial payload (for example, in a phishing email). This can make file-based detection more challenging.

However, operators don't need to use Sliver's default DLL or executable payloads. Motivated threat actors can generate a Sliver shellcode and embed it in custom loaders like Bumblebee that then runs the Sliver implant on a compromised system. Detection engineers can create loader-specific detections or, if the shellcode isn't obfuscated, rules for the shellcode payload that is embedded in the loader.

Config extraction

When responding to a suspected intrusion, security analysts may find themselves with a malware payload with little context. Quickly extracting key configuration details from the malware like C2 address, network configurations, and other implant details is a crucial step in hunting for affected devices in the network.

Many implants, including Sliver, heavily obfuscate or encrypt useful information to prolong analysis and detection attempts. Sliver, like other implants based on GoLang, uses the public gobfuscate library for this purpose. Several researchers have created tools to assist with "de-gobfuscating" strings in payloads, but it is still a fairly manual process[1].

In such cases, we can extract configurations we are interested in more easily when they're loaded into memory. Sliver must de-obfuscate and decrypt its configurations to use them, so we can scan memory for these values and extract them programmatically to get results like configuration data, as illustrated in Figure 1 below:

```
{
  "MTLS_ECPrivateKey": "-----BEGIN EC PRIVATE KEY-----\nMIGkAgEBBDAztDYHsF7UBqI6JuiAOkujPZcOrJ4T
DYTa9md+huaqrppyRjFmiw1w\namTjJGcoVeWgBwYFK4EEACKhZANiAATYbkiYJdFyKSFTlwgHYayAZMfXDp2jIt3V\nn0OA
7c8O5+g3N3uC6xMYFTdiw7LMBJyMzdgGFMbD3MZ57vLsE1WXBMmXFj6gxb2J\ne/lUPN7dSq6RSPwXgdQPAzpNLBLqcaE=\n
-----END EC PRIVATE KEY-----",
  "ImplantName": "ACUTE_PITCH",
  "URL": "mtls://10.0.0.1:8888",
  "MTLS_Certificate": "-----BEGIN CERTIFICATE-----\nMIIBrTCCATOgAwIBAgIRAMlsQAcLoXh416+vNRYjtsMw
CgYIKoZIzj0EAwMwADAe\nFw0yMTA4MTgxMzQ4MDdaFw0yNDA4MTcxMzQ4MDdaMAAwdjAQBgcqhkjOPQIBBgUr\ngQQAIgNi
AASCQIa26wVXgDXcRDPJ7nR4XjcTJ2qLl7WejwCz1oxpZpxd32eODvmO\no6ai40k3ZWDYbzrkRFpVaeP8Tlo73mQzksKGtw
vUW4ZhxwfY4Tcw8l8C/bHYWKPL\nIbzMB5hblkWjcTBvMA4GA1UdDwEB/wQEAwICpDAdBgNVHSUEFjAUBggrBgEFBQcD\nAQ
YIKwYBBQUHAwIwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUtsCneQ4SMtz1\nZgMsFQefNbBRuF8wDgYDVR0RAQH/BAQw
AoIAMAoGCCqGSM49BAMDA2gAMGUCMQCF\nAbvhBl0/kfEeXekvNGvmXHIysblfM",
  "ImplantUUID": "778b2cca-90ac-4ab9-8235-bc8f43b527fe"
}
```


Figure 1. Sample configuration extraction from a Sliver test implant

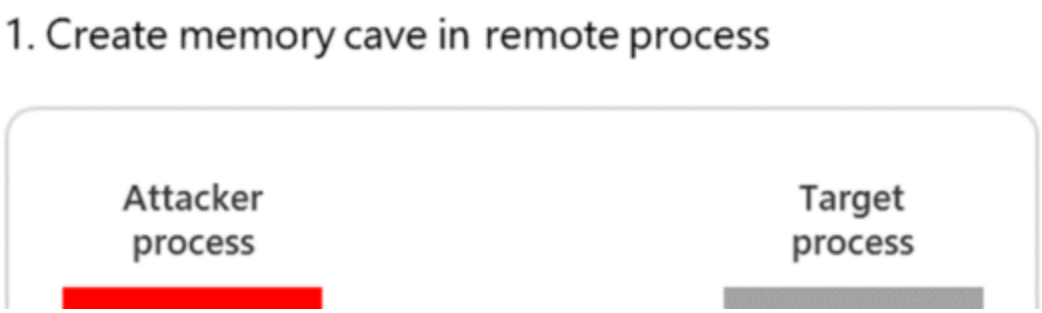
There are similar public de-obfuscation tools for Cobalt Strike, such as Apr4h/CobaltStrikeScan and CCob/BeaconEye.

Some malware will attempt to obfuscate or encrypt configurations in memory as well. Cobalt Strike's "sleep_mask" is a good example of this. However, it's important to note that even in these cases, the malware must decrypt the configurations when it wants to check in with the C2 server for new instructions. Thus, extracting configurations from memory requires intentional timing.

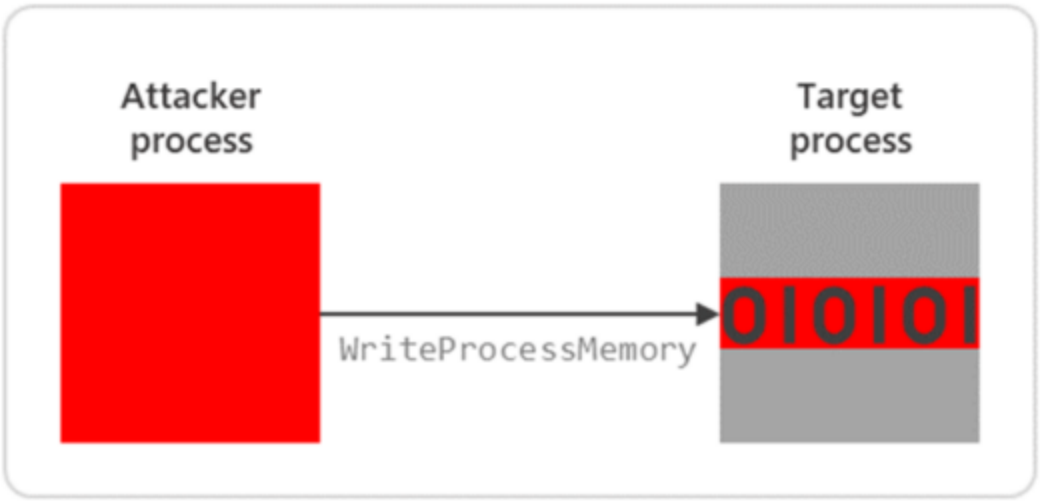
Code execution

Sliver includes a variety of built-in techniques and post-exploitation functionality. One of the most common underlying techniques used by C2 operators and frameworks is process injection, which is a method of running arbitrary code within the address space of a separate live process.

Attackers use process injection for defense evasion, access, or privilege elevation, distancing risky code execution, and many other reasons. As Microsoft researchers explained: "[P]rocess injection gives attackers the ability to run malicious code that masquerades as legitimate programs. With code injection, attackers don't have to use custom processes that can quickly be detected. Instead, they insert malicious code into common trusted processes (e.g., explorer.exe, regsvr32.exe, svchost.exe, etc.), giving their operations an increased level of stealth and persistence." Figure 2 below illustrates how process injection typically works:



2. Write malicious code to the cave



3. Make the cave "executable memory" then launch the code to spawn an implant in a new thread under the remote process

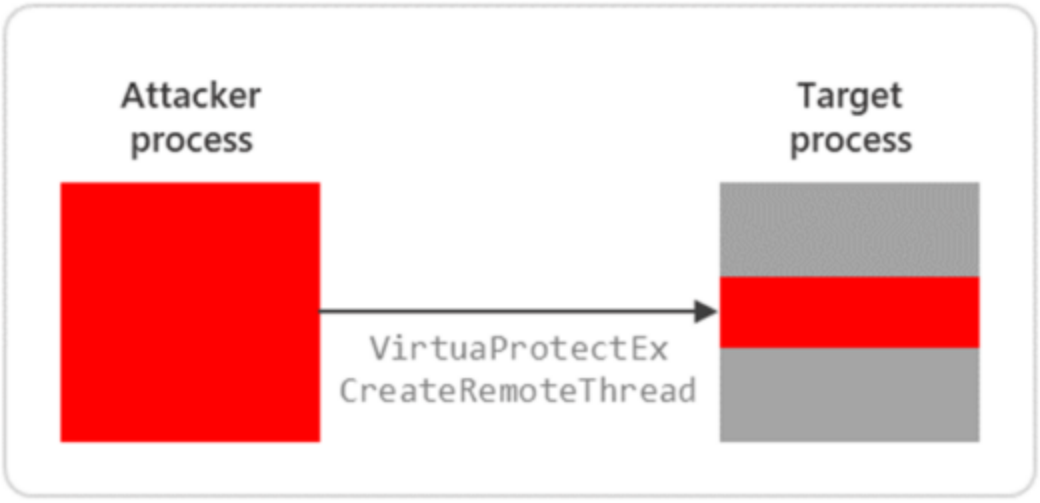


Figure 2.

How process injection works

Just like any other C2 framework, Sliver utilizes process injection as a core part of many default commands or capabilities, such as:

- migrate (command) – migrate into a remote process
- spawndll (command) – load and run a reflective DLL in a remote process
- sideload (command) – load and run a shared object (shared library/DLL) in a remote process
- msf-inject (command) – inject a Metasploit Framework payload into a process
- execute-assembly (command) – load and run a .NET assembly in a child process
- getsystem (command) – spawn a new Sliver session as the *NT AUTHORITY\SYSTEM* user
- extensions/aliases – Beacon Object Files (BOFs), .NET apps, and other third-party tooling

Sliver also uses common process injection implementations. For example, as of this writing, the built-in Sliver *migrate* command migrates to a remote process using a classic combination of *VirtualAllocEx*, *WriteProcessMemory*, *VirtualProtectEx*, and finally *CreateRemoteThread* Windows API calls. Other commands such as *Sideload*, *SpawnDll*, and *Execute-Assembly* also rely on this combination. This sequence of injection-related API calls is well documented, and Microsoft Defender for Endpoint generates alerts like *A process was injected with potentially malicious code* based on the combination of such API calls:

Figure 3. Example of Microsoft Defender for Endpoint alerts for injection-related API calls

Aside from process injection, Sliver provides additional familiar techniques such as lateral movement via a *PsExec* command. Defender for Endpoint generates multiple alerts on such default techniques:

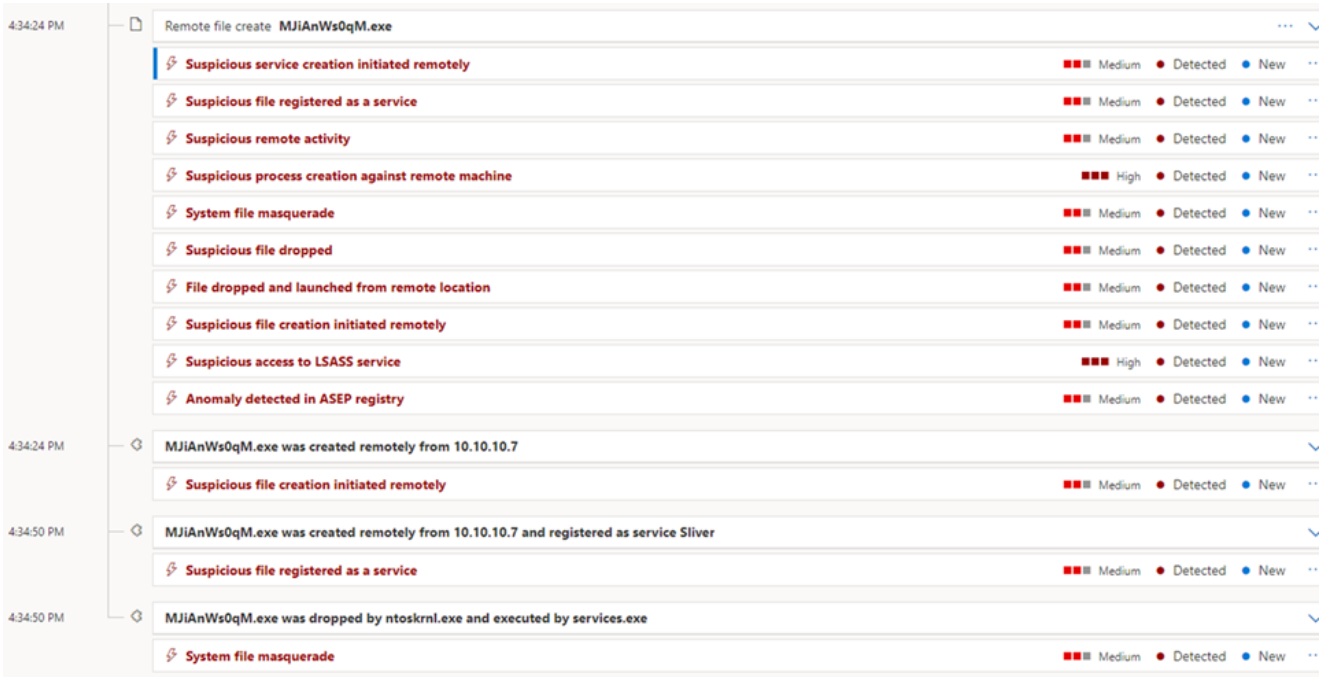


Figure 4. Example of Microsoft Defender for Endpoint alerts for default service installation created by *PsExec* command

Surfacing Sliver threat activity

Based on our analysis of the Sliver framework, Defender Experts designed advanced hunting queries to surface Sliver-related threat activity. These hunting queries leverage Kusto Query Language (KQL), a query language specifically designed to work with large datasets in Azure. Unless otherwise noted, the detection and hunting guidance in this blog are designed for official, non-customized Sliver codebase available as of this writing.

Customers can run the following queries in the Microsoft 365 Defender portal. These queries are examples of how hunters can key in on unique default configurations implemented by Sliver.

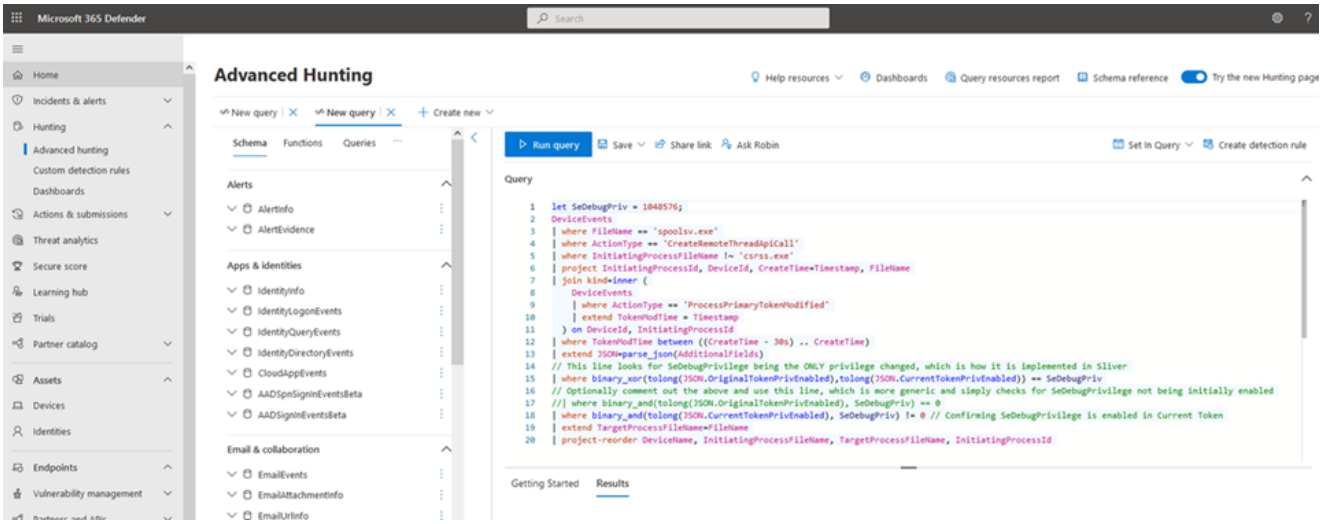


Figure 5. Running advanced hunting queries in Microsoft 365 Defender

GetSystem

The following query finds potential launch of the built-in *GetSystem* command, where the default target process of *spoolsv.exe* is used as the target process for injection. The query looks for *SeDebug* privileges being added to a process, followed by that same process creating a remote thread in *spoolsv.exe* within 30 seconds.

```
// SeDebugPrivilege constant used to identify if this privilege is enabled in Token
let SeDebugPriv = 1048576;
DeviceEvents
| where FileName == 'spoolsv.exe'
| where ActionType == 'CreateRemoteThreadApiCall'
| where InitiatingProcessFileName !~ 'csrss.exe'
| project InitiatingProcessId, DeviceId, CreateTime=Timestamp, FileName
| join kind=inner (
    DeviceEvents
    | where ActionType == 'ProcessPrimaryTokenModified'
    | extend TokenModTime = Timestamp
  ) on DeviceId, InitiatingProcessId
| where TokenModTime between ((CreateTime - 30s) .. CreateTime)
| extend JSON=parse_json(AdditionalFields)
// This line looks for SeDebugPrivilege being the ONLY privilege changed, which is
how it is implemented in Sliver
| where
binary_xor(tolong(JSON.OriginalTokenPrivEnabled),tolong(JSON.CurrentTokenPrivEnabled))
 == SeDebugPriv
// Optionally comment out the above and use this line, which is more generic and
simply checks for SeDebugPrivilege not being initially enabled
//| where binary_and(tolong(JSON.OriginalTokenPrivEnabled), SeDebugPriv) == 0
| where binary_and(tolong(JSON.CurrentTokenPrivEnabled), SeDebugPriv) != 0 //
Confirming SeDebugPrivilege is enabled in Current Token
| extend TargetProcessFileName=FileName
| project-reorder DeviceName, InitiatingProcessFileName, TargetProcessFileName,
InitiatingProcessId
```

Shell

The following query finds the default, unique PowerShell command used when Sliver creates an interactive shell with the *'Shell'* command.

```
DeviceProcessEvents
| where ProcessCommandLine == 'powershell.exe -NoExit -Command
[Console]::OutputEncoding=[Text.UTF8Encoding]::UTF8'
```

Sideload/SpawnDll/Execute-Assembly

The *Sideload*, *SpawnDll*, and *Execute-Assembly* commands spawn and inject into *notepad.exe* by default. The following query finds process creation events where the same process creates and injects into *notepad.exe* within 10 seconds.

```
DeviceProcessEvents
| where ActionType == 'ProcessCreated'
| where ProcessCommandLine =~ 'notepad.exe'
| distinct InitiatingProcessId, DeviceId
| join kind=inner (
    DeviceEvents
    | where ActionType == 'CreateRemoteThreadApiCall'
    | where ProcessCommandLine == 'notepad.exe'
    | where Timestamp between (ProcessCreationTime .. (ProcessCreationTime+10s))
) on DeviceId, InitiatingProcessId
```

PsExec

The following query finds default values for the *ImagePath*, *DisplayName*, and *Description* of the service installed on the remote system when using Sliver's *PsExec* command.

```
DeviceRegistryEvents
| where ActionType == 'RegistryValueSet'
| where (RegistryValueName == 'ImagePath' and RegistryValueData matches regex @'^[a-zA-Z]:\\windows\\temp\\[a-zA-Z0-9]{10}\.exe') or
        (RegistryValueName == 'DisplayName' and RegistryValueData == 'Sliver') or
        (RegistryValueName == 'Description' and RegistryValueData == 'Sliver implant')
```

The following query is an alternative method of searching on the same service properties but within service installation events instead of registry keys.

```
DeviceEvents
| where ActionType == 'ServiceInstalled'
| extend JSON = parse_json(AdditionalFields)
| where (FolderPath endswith_cs @':\windows\temp' and FileName matches regex @'^[a-zA-Z0-9]{10}\.exe') or (JSON.ServiceName == 'Sliver')
```

Building resilience against future attacks with threat hunting

Our analysis of Sliver's source code and functionality reveals hunting opportunities that can also be adapted for use against other malware frameworks. In addition, Sliver and many other C2 frameworks are yet another example of how threat actors are continually attempting to evade automated security detections. Threat hunting provides an added layer to other security mitigations and can help address areas of defense evasion. By focusing research efforts on the underlying attacker techniques used within Sliver, detections and threat hunting strategies are more resilient to future changes in attacker toolsets implementing those techniques.

Defender Experts is part of Microsoft's global network of more than 8,500 security experts that further enriches our vast cross-domain signals and lets us deliver coordinated threat defense in our security products and solutions. As seen in our research on Sliver, our monitoring of the threat landscape informs advanced, high-fidelity KQL queries that are then

thoroughly tested to form the basis of our Defender Experts Notifications. These notifications are designed to identify the most important risks, and provide technical information, as well as hunting and mitigation guidance.

Our insights from threat hunting and monitoring also feed into products like Microsoft Defender for Endpoint that then alert customers to malicious activity seen with C2 frameworks like Sliver. The following titles in the security center can indicate threat activity on their networks:

- Ransomware-linked emerging threat activity group detected
- Suspicious behavior by cmd.exe was observed
- Suspicious sequence of exploration activities
- Suspicious data transfer
- Suspicious System Network Configuration Discovery
- Process hollowing detected
- A process was injected with potentially malicious code
- Suspicious Peripheral Device Discovery
- Abnormal Remote Service Execution
- Suspicious file dropped
- Suspicious command launched from a remote location
- Suspicious files or content obfuscation/de-obfuscation activity

Microsoft customers can also apply the following security mitigations to reduce the impact of Sliver and other similar threats:

- Turn on network protection. Network protection helps prevent users from accessing dangerous domains and IP addresses. Check your perimeter firewall and proxy to restrict servers from making arbitrary connections to the internet to browse or download files. Such restrictions help inhibit malware downloads and C2 activity including mobile devices.
- Use Microsoft Defender Firewall, which, along with your network firewall, prevents remote procedure call (RPC) and service message block (SMB) communication along endpoints whenever possible. This limits lateral movement and other attack activities.
- Turn on cloud-delivered protection and automatic sample submission on Microsoft Defender Antivirus. These capabilities use artificial intelligence and machine learning to quickly identify and stop new and unknown threats.
- Check your Office 365 email filtering settings to ensure you block spoofed emails, spam, and emails with malware. Use Microsoft Defender for Office 365 for enhanced phishing protection and coverage against new threats and polymorphic variants. Configure Office 365 to recheck links on click and delete sent mail in response to newly acquired threat intelligence.

Organizations can also follow these general best practices to make their networks resilient against attacks:

- **Harden the cloud.** As attackers move towards cloud resources, it's important to secure cloud resources and identities as well as on-premises accounts. Security teams should focus on hardening security identity infrastructure, enforcing multifactor authentication (MFA) on all accounts, and treating cloud admins/tenant admins with the same level of security and credential hygiene as Domain Admins.
- **Close security blind spots.** Organizations should verify that their security tools are running in optimum configuration and perform regular network scans to ensure a security product protects all systems.
- **Reduce the attack surface.** Establish attack surface reduction rules to prevent common attack techniques used by C2 frameworks.
- **Evaluate the perimeter.** Organizations must identify and secure perimeter systems that attackers might use to access the network. Public scanning interfaces can be used to augment data.
- **Harden internet-facing assets.** Attackers use unpatched vulnerabilities, whether already disclosed or zero-day, especially in the initial access stage to get the C2 framework onto the target systems. They also rapidly adopt new vulnerabilities. To further reduce exposure, organizations can use endpoint detection and response (EDR) products with threat and vulnerability management capabilities, such as Microsoft Defender for Endpoint, to discover, prioritize, and remediate vulnerabilities and misconfigurations.

To find out how you can extend your ability to defend and manage your security with managed services from Microsoft, learn more about Microsoft Security Experts.

Appendix

Microsoft 365 Defender detections

Microsoft Defender Antivirus

Microsoft Defender Antivirus detects Sliver threat components as the following malware:

Microsoft Defender Antivirus also detects Bumblebee loader as the following malware:

- Trojan:Win64/Bumblebee
- Trojan:Win64/BumbleBeeLoader
- Trojan:PowerShell/Bumblebee

---

[1] Automated string de-gobfuscation, Kryptos Logic