Detecting Trickbot attacks

Trickbot is a Trojan that was initially used to steal banking credentials. Now this malware is known in Microsoft Windows operating systems where it targets Microsoft Defender to prevent its detection and removal, then steals credentials using multi-component modules that collect and exfiltrate data.

These searches detect and investigate unusual activities that might relate to Trickbot, including looking for file writes associated with its payload, process injection, shellcode execution, and data collection.

Required data

- **Endpoint data**
- **Windows Security event logs**

Some commands, parameters, and field names in the searches below might need to be adjusted to match your environment. In addition, to optimize the searches shown below, you should specify an index and a time range when appropriate.

How to use Splunk software for this use case

► Account discovery commands

To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the **Endpoint data model** in the Common Information Model (CIM). For information on installing and using the CIM, see the **Common Information Model documentation**.

This search is designed to detect a potential account discovery series of commands used by malware to recon a target machine. Malware using this technique commonly runs a specific series of command processes, or drops a module that runs the series of commands. Searches that show these commands being used are a good indicator of attack if seen in machines used by a non-technical user or department.

```
```

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
Trickbot named pipe

To complete this process, your deployment needs to ingest logs with the process name and pipename from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA.

This search detects a potential Trickbot infection through a named pipe to the system. This technique is used by Trickbot to communicate with its C2 server to post or get commands during infection.

```
| search (PipeName="\\lacesomepipe" (EventCode=17 OR EventCode=18) (source=Syslog:Linux-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational OR sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational)) | stats min(_time) AS firstTime max(_time) AS lastTime count BY Computer user_id EventCode PipeName signature Image process_id | convert timeformat="%Y-%m-%d %H:%M:%S" ctime(firstTime) | convert timeformat="%Y-%m-%d %H:%M:%S" ctime(lastTime)
```

Cobalt Strike named pipes

To complete this process, your deployment needs to ingest logs with the process name, parent process, and command-line executions from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA.

This search identifies the use of default or publicly known, named pipes used with Cobalt Strike. A named pipe is a named, one-way or duplex pipe for communication between the pipe server and one or more pipe clients. Cobalt Strike uses named pipes in many ways and has default values used with the Artifact Kit and Malleable C2 Profiles.

Each EDR product presents named pipes differently. You can try taking the values and generating a search based on your product of choice.

Upon triage, review the process performing the named pipe. If it is explorer.exe, it might have been injected into by another process. Review recent parallel processes to identify suspicious patterns or behaviors. A parallel process might have a network connection. Review and follow the connection back to identify any file modifications.

Adversaries use named pipes with Cobalt Strike to blend in. Because of this, some of the named pipes identified and added might cause false positives. Filter by process name or pipe name to reduce false positives.

```
| search ((EventID=17 OR EventID=18) (source=Syslog:Linux-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational OR sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational) (PipeName=\\DserNamePipe* OR PipeName=\\UIA_PIPE* OR PipeName=\\mojo.* OR PipeName=\\msagent_* OR PipeName=\\ntsvcs* OR PipeName=\\postex_* OR PipeName=\\spoolss_* OR PipeName=\\srvsvc_* OR PipeName=\\status_* OR PipeName=\\win_svc* OR PipeName=\\winsock* OR PipeName=\\wkssvc*)) | stats count min(_time) AS firstTime max(_time) AS lastTime BY Computer, process_name, process_id | convert timeformat="%Y-%m-%d %H:%M:%S" ctime(firstTime) | convert timeformat="%Y-%m-%d %H:%M:%S" ctime(lastTime)
```

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
**Executable file written in administrative SMB shares**

To complete this process, your deployment needs to ingest [Windows Security event logs](https://docs.microsoft.com/en-us/windows/security/threat-protection/events/event-5145) with event code 5145 enabled. The Windows TA is also required. You should also ensure you enable the object "Audit Access Success/Failure" in your group policy.

This search identifies executable files (.exe or .dll) being written to Windows administrative SMB shares (Admin$, IPC$, C$). This technique is commonly used by tools like PsExec/PaExec and others to stage service binaries before creating and starting a Windows service on remote endpoints. The Trickbot malware family implements this behavior to try to infect other machines in the infected network.

False positives from this search might occur since system administrators might use tools like PsExec for troubleshooting or administration tasks. However, this activity typically comes only from certain users and certain systems that can be added to an allow list.

```splunk
| search (Access_Mask="0x2" EventCode=5145 Object_Type=File eventtype=wineventlog_security (Relative_Target_Name="*.dll" OR Relative_Target_Name="*.exe") (Share_Name="\\\"\C$" OR Share_Name="\\\"\IPC$" OR Share_Name="\\\"\admin$"))
| stats min(_time) AS firstTime max(_time) AS lastTime count BY EventCode Share_Name Relative_Target_Name Object_Type Access_Mask user src_port Source_Address
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)
```

**Scheduled task with rundll32 command trigger**

To complete this process, your deployment needs to ingest logs with the task schedule (e.g. Security log event code 4698) from your endpoints.

This search uses Windows Security event code 4698, A scheduled task was created to identify suspicious tasks registered on Windows either via schtasks.exe or via TaskService with a command to be executed with a Rundll32. This technique used by Trickbot which uses rundll32 to load the Trickbot downloader.

Schtasks.exe is natively found in C:\Windows\system32 and C:\Windows\syswow64. Taskschd.dll is loaded when schtasks.exe or TaskService is launched. If found loaded by another process, it is possible a scheduled task is being registered within that process context in memory.

If you return potential positive results from this search, identify the task scheduled source and determine whether it is schtasks.exe or whether it is via TaskService. Review the job created and the command to be run, then capture any artifacts on disk and review them. You should also identify any parallel processes within the same timeframe to identify the source.
To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Endpoint data model in the Processes node in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies rundll32.exe executing a DLL function name, Start and StartW, on the command line that is commonly observed with malware DLL payloads.

Rundll32.exe is natively found in C:\Windows\system32 and C:\Windows\syswow64. Typically, the DLL is written and loaded from a world writeable path or user location. In most instances it is unsigned and doesn't have a valid certificate.

If you return potential positive results from this search, review the parent process and other parallel application executions, capturing and triaging the affected DLL. In some attacks, rundll32.exe is the default process it opens and injects shellcode into. This default process can be changed but typically is not.

Mshta.exe spawning rundll32 or regsvr32 process

To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search detects a suspicious mshta.exe process that spawns rundll32 or regsvr32 child processes. This technique is seen malware attacks to initiate .dll stage loaders that execute and download the malware payload.
To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Endpoint data model in the Processes node in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies suspicious spawned processes of MS Office applications due to macro or malicious code. This technique is seen in IcedID malware which uses MS Office as a weapon or attack vector to infect machines.


Make the following adjustments to the search above to search for regsvr32 or rundll32 processes.

<table>
<thead>
<tr>
<th>Process to search for</th>
<th>Replacement search lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regsvr32</td>
<td>(&quot;Processes.process_name&quot;=regsvr32.exe OR &quot;Processes.original_file_name&quot;=REGSVR32.EXE))</td>
</tr>
<tr>
<td>Rundll32</td>
<td>(&quot;Processes.process_name&quot;=rundll32.exe OR &quot;Processes.original_file_name&quot;=RUNDLL32.EXE))</td>
</tr>
</tbody>
</table>

To complete this process, your deployment needs to ingest logs with the process name and ImageLoaded from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA.

This search identifies suspicious MS Office documents that use macro code. Macro code is a prevalent malware attack vector that runs malicious payloads, or downloads malware payloads or other malware components.

It is a good practice to disable macros by default to avoid automatically executing macro code while opening or...
closing a MS Office document.

False positives from this search might occur since macros can be used in normal MS Office documents for automation purposes.

```
| search (EventCode=7 (ImageLoaded="^\VBE7.DLL" OR ImageLoaded="^\VBE7INTL.DLL" OR ImageLoaded="^\VBEUI.DLL") (process_name="EXCEL.EXE" OR process_name="POWERPNT.EXE" OR process_name="WINWORD.EXE") (source=Syslog:Linux-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational)) | stats min(_time) AS firstTime max(_time) AS lastTime values(ImageLoaded) AS AllImageLoaded count BY Computer EventCode Image process_name ProcessId ProcessGuid
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)
```

**► MS Office product spawning command child process**

To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Endpoint data model in the Processes node in the Common Information Model (CIM).

For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies a suspicious MS Office product process (such as winword.exe, excel.exe, or powerpnt.exe as shown in the sample search) that spawns a command child process. This is commonly seen where a MS Office product runs a macro that runs a shell command to download or execute malicious LOLBins. This is seen in Trickbot spear phishing documents where it runs shell commands to run a MSHTA payload.

```
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)
```

**► Powershell remote thread to known Windows process**

To complete this process, your deployment needs to ingest logs with the process name and CreateRemoteThread from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA.

This search looks for a suspicious PowerShell process that tries to inject code to a known or critical Windows process, executing it using CreateRemoteThread. This technique is seen in Trickbot malware where it loads a shellcode to svchost.exe to run reverse shell to C2 and download another payload.
To complete this process, your deployment needs to ingest logs with the process name, DNS query name, process path, and query status from your endpoints. If you are using Sysmon, you must have at least version 12 of the Sysmon TA.

This search detects a suspicious wermgr.exe process that tries to connect to known IP web services. This technique used by Trickbot to recon the infected machine and look for its IP address without leaving a heavy fingerprint on the command line process.

Since wermgr.exe is designed for Windows error handling processes, this process trying to connect to these IP web services is suspicious and indicates probable malicious code injection.

To complete this process, your deployment needs to ingest logs with the process name, parent process, and command-line executions from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA.

This search detects a potential malicious wermgr.exe process that creates executable files. Since wermgr.exe is an application trigger when an error is encountered in a process, it is very unusual for this process to drop executable files. This technique is used by Trickbot malware where it injects code into this process to run malicious behavior.
like downloading a payload.

```
| search (EventCode=11 TargetFilename="*.exe" process_name="wermgr.exe" (source=Syslog:Linux-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational OR sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational))
| stats min(_time) AS firstTime max(_time) AS lastTime count BY Image TargetFilename process_name dest EventCode ProcessId
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)
```

► Wermgr.exe process spawning command or PowerShell process

To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the [Endpoint data model](#) in the Processes node in the Common Information Model (CIM). For information on installing and using the CIM, see the [Common Information Model documentation](#).

This search detects a suspicious command or PowerShell process spawned by wermgr.exe. This behavior is seen as code injection techniques used by Trickbot.

```
| rename "Processes.*" AS "*"
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)
```

Next steps

The content in this article comes from [Splunk Enterprise Security (ES)](https://www.splunk.com/solutions/security). As a Splunk premium security solution, ES solves a wide range of security analytics and operations use cases including continuous security monitoring, advanced threat detection, compliance, incident investigation, forensics and incident response. Splunk ES delivers an end-to-end view of an organization's security posture with flexible investigations, unmatched performance, and the most flexible deployment options offered in the cloud, on-premises, or hybrid deployment models. If you have questions about this use case, see the Security Research team's [support options on GitHub](https://github.com/splunk).

In addition, these Splunk resources might help you understand and implement this use case:

- **E-book:** [Ransomware, malware and cyberthreats](#)
• Blog: Hunting with Splunk: The basics