ZDI-CAN-25373: Windows Shortcut Exploit Abused as Zero-Day in Widespread APT Campaigns

⁝ 3/18/2025

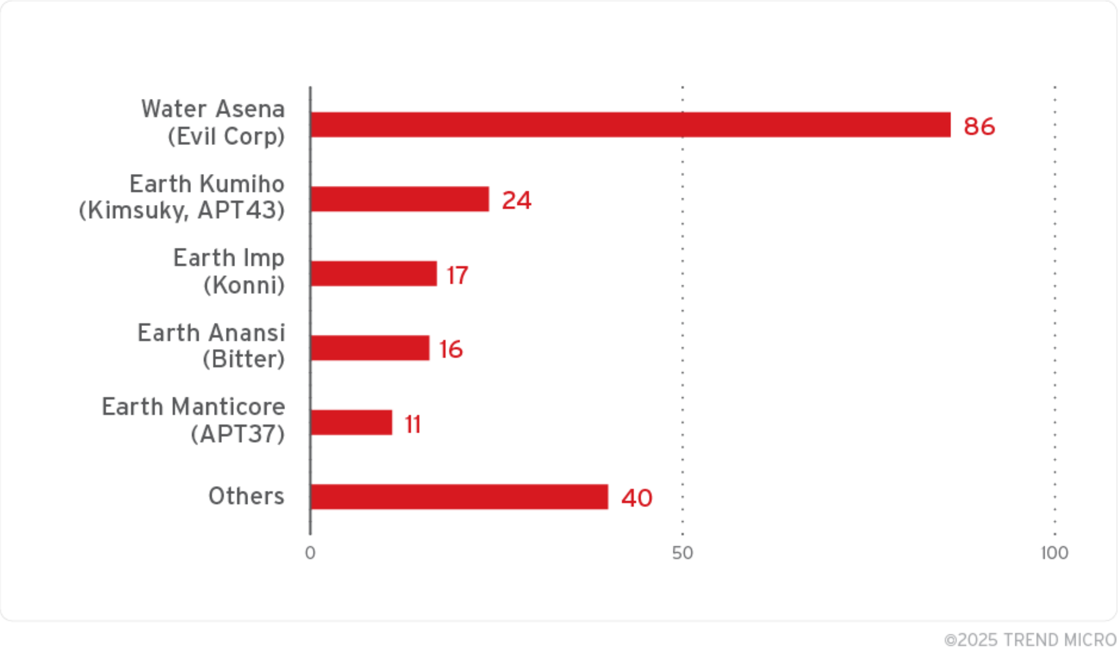


Figure 1. Number of samples from APT groups exploiting ZDI-CAN-25373
download

Summary

- Trend Zero Day Initiative™ (ZDI) identified nearly 1,000 malicious .lnk files abusing ZDI-CAN-25373, a vulnerability that allows attackers to execute hidden malicious commands on a victim's machine by leveraging crafted shortcut files.
- The attacks leverage hidden command line arguments within .lnk files to execute malicious payloads, complicating detection. The exploitation of ZDI-CAN-25373 exposes organizations to significant risks of data theft and cyber espionage.
- The vulnerability has been exploited by state-sponsored APT groups from North Korea, Iran, Russia, and China. Organizations across the government, financial, telecommunications, military, and energy sectors have been affected in North America, Europe, Asia, South America, and Australia.
- Organizations should immediately scan and ensure security mitigations for ZDI-CAN-25373, maintain vigilance against suspicious .lnk files, and ensure comprehensive endpoint and network protection measures are in place to detect and respond to this threat. Trend Micro customers are protected from possible attempts to exploit the vulnerability via rules and filters that were released in October 2024 and January 2025.

The Trend Zero Day Initiative™ (ZDI) threat hunting team identified significant instances of the exploitation of ZDI-CAN-25373 across a variety of campaigns dating back to 2017. Our analysis revealed that 11 state-sponsored groups from North Korea, Iran, Russia, and China have employed ZDI-CAN-25373 in operations primarily motivated by cyber espionage and data theft.

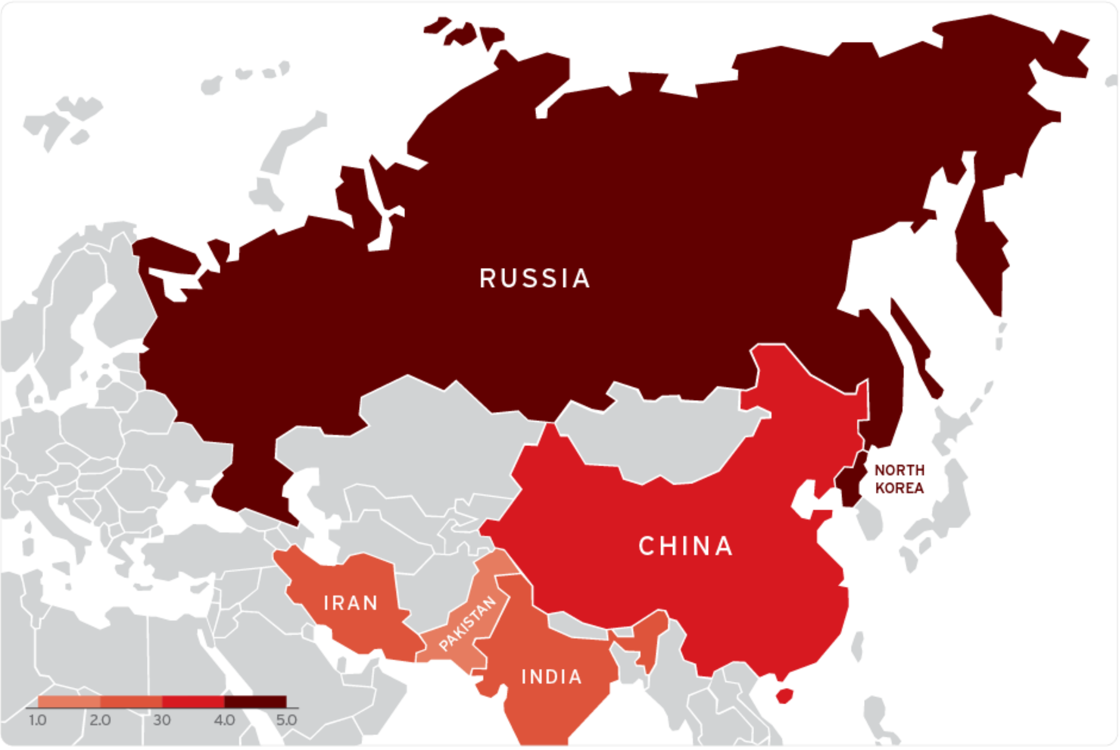
We discovered nearly a thousand Shell Link (.lnk) samples that exploit ZDI-CAN-25373; however, it is probable that the total number of exploitation attempts are much higher. Subsequently, we submitted a proof-of-concept exploit through Trend ZDI's bug bounty program to Microsoft, who declined to address this vulnerability with a security patch.

State-sponsored APT groups exploiting ZDI-CAN-25373 for espionage and data theft

While tracking the exploitation of ZDI-CAN-25373, we discovered the widespread abuse of this vulnerability by numerous threat actors and APT groups. These threats include a mix of state-sponsored as well as non-state-sponsored APT groups. Many of these groups demonstrated a high degree of sophistication in their attack chains and have a history of abusing zero-day vulnerabilities in the wild.

The raw data for Figure 1 may be found here.

The samples shown in Figure 1 have been attributed to APT groups with a high degree of confidence. The Trend ZDI threat hunting team has uncovered hundreds of additional samples that cannot be confidently attributed at this time, but nonetheless exploited ZDI-CAN-25373 in the wild.



©2025 TREND MICRO

Figure 2. APT countries of origin that have exploited ZDI-CAN-25373
download

Figure 2 highlights the number of APT groups exploiting ZDI-CAN-25373 as part of their attack chains by country of origin. These APT groups contain a mixture of state-sponsored, state-adjacent, and cybercrime groups. We've isolated the state-sponsored APT groups in Figure 3 of these APT groups with known countries of origin.

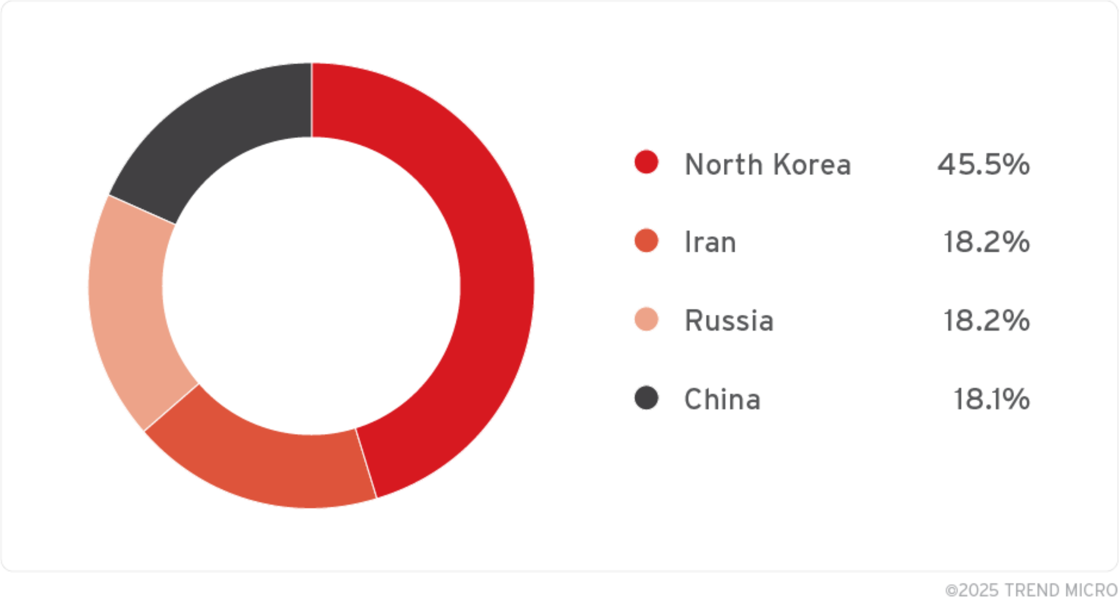


Figure 3. Countries of state-sponsored APT groups exploiting ZDI-CAN-25373
download

Nearly half of the state-sponsored threat actors that exploit ZDI-CAN-25373 are reported to originate from North Korea. It is noteworthy that a significant majority of North Korea's intrusion sets have targeted ZDI-CAN-25373 at various times. This observation underscores a trend of cross-collaboration, technique, and tool sharing among different threat groups within North Korea's cyber program.

In our analysis of the campaigns that exploit ZDI-CAN-25373 and their associated intrusion sets, we have found that nearly 70% are primarily focused on espionage and information theft (Figure 4). In comparison, over 20% are directed toward achieving financial gain. In some instances, threat actors with a primary motivation towards espionage may fund their espionage efforts with financially motivated campaigns.

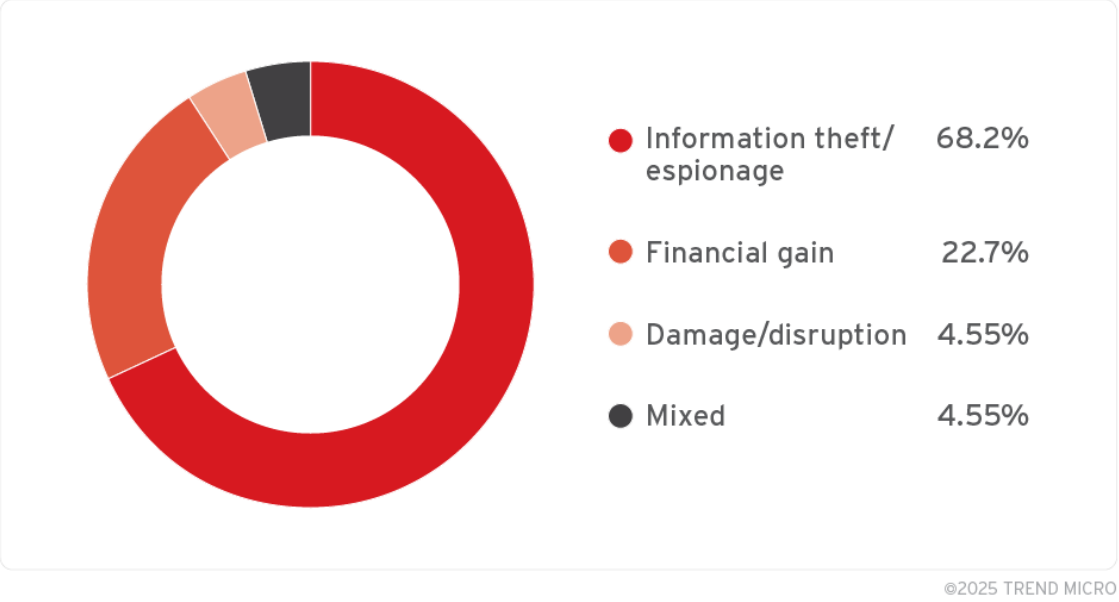


Figure 4. APT groups' motivation

Targeted sectors and risk

In the campaigns associated with ZDI-CAN-25373 and their corresponding intrusion sets that we monitor, we have identified a substantial volume of telemetry indicating that a diverse range of state-sponsored and cybercriminal threat actors have been targeting multiple sectors (Figure 5).

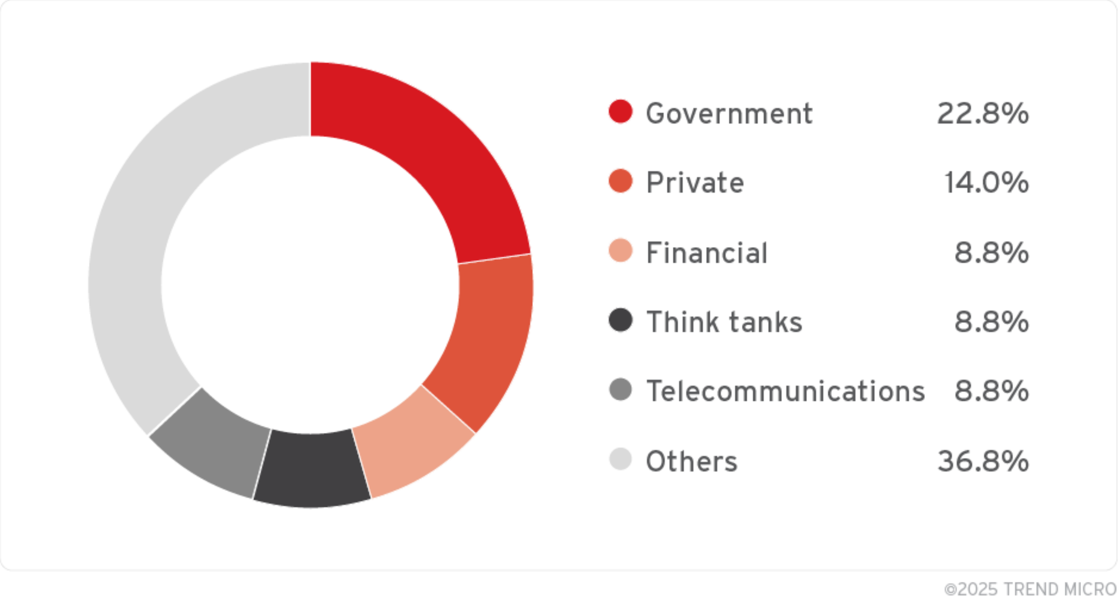


Figure 5. Targeted sectors in exploitation of ZDI-CAN-25373
download

The raw data for Figure 5 may be found here.

In our analysis of the exploitation landscape of ZDI-CAN-25373, we've identified the following sectors of particular risk for targeted attacks, including:

- Government
- Private
- Financial, including cryptocurrency-related
- Think tanks, including non-governmental organizations (NGOs)
- Telecommunications
- Military and defense
- Energy

Organizations that fall within these sectors are at higher risk for exploitation and should scan and ensure security mitigations for ZDI-CAN-25373 immediately, as well as remain vigilant of .lnk files in general. Additionally, organizations are encouraged to investigate potential compromise or attempts to compromise systems use ZDI-CAN-25373 as an intrusion vector. as an intrusion vector.

Exploring malware payloads

Of the .lnk files exploiting ZDI-CAN-25373 and campaigns we tracked, we were able to recover some of the malware payloads or correlate them through Trend Micro threat telemetry. This highlights the diverse use of various malware payloads and loaders in campaigns exploiting ZDI-CAN-25373.

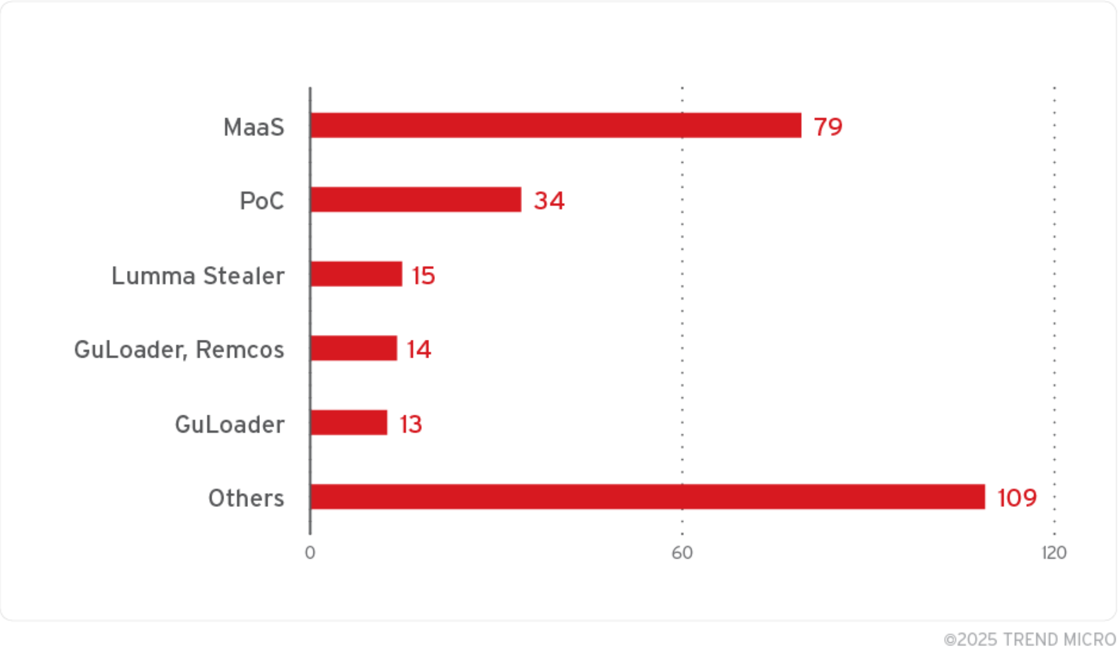


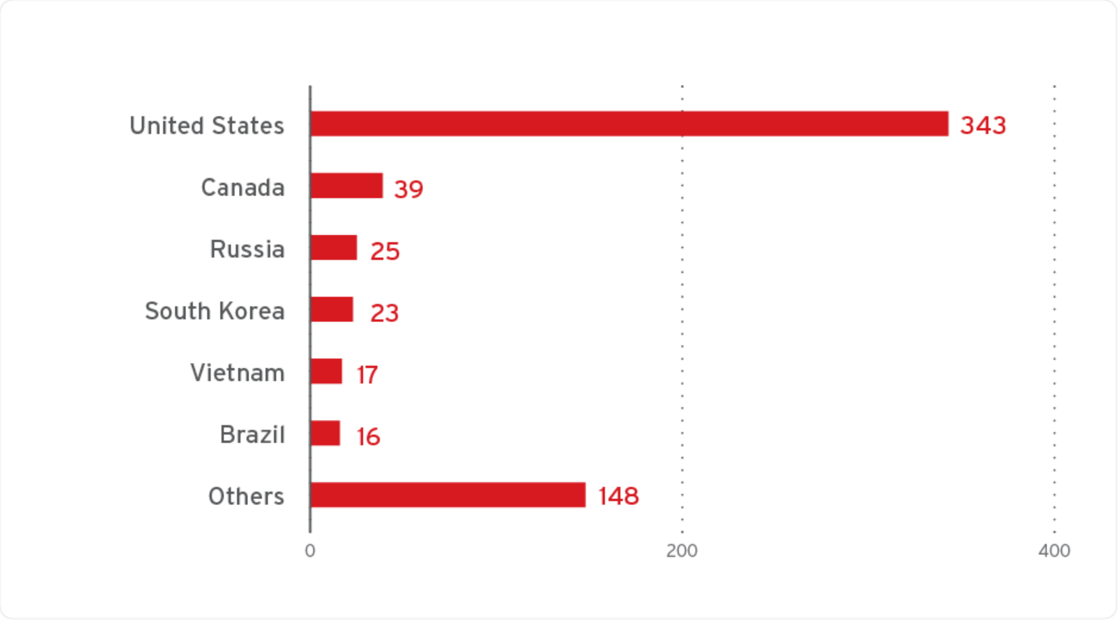
Figure 6. Malware payloads as part of attack chains exploiting ZDI-CAN-25373
download

The raw data for Figure 6 may be found here.

In Figure 6, for samples shown under "Malware-as-a-Service" (MaaS), we were not able to recover the specific malware payload used, but the observed origin domains commonly served MaaS and commodity malware payloads. The "PoC" row denotes proof-of-concept samples retrieved. These were likely used during the attack chain development lifecycle. An interesting observation we discovered while tracking these intrusion sets, and malware was that Water Asena (Evil Corp) had been exploiting ZDI-CAN-25373 in their Raspberry Robin campaigns.

Victimology

Of the samples we analyzed, a majority were submitted from North America, including the US and Canada. However, due to the number of APT groups exploiting ZDI-CAN-25373, the victimology is much broader, also affecting Europe, Asia, South America, Africa and Australia (Figure 7).



©2025 TREND MICRO

Figure 7. Files exploiting ZDI-CAN-25373 tracking countries by submission origin
download

The raw data for Figure 7 may be found here.

When placed on a map as shown in Figure 8, we can see that the exploitation of ZDI-CAN-25373 is widespread, with a focus on North America, South America, Europe, East Asia and Australia. It’s important to note that the scope of impact is likely much larger, as Figure 8 shows only samples the Trend ZDI threat hunting team was able to recover.

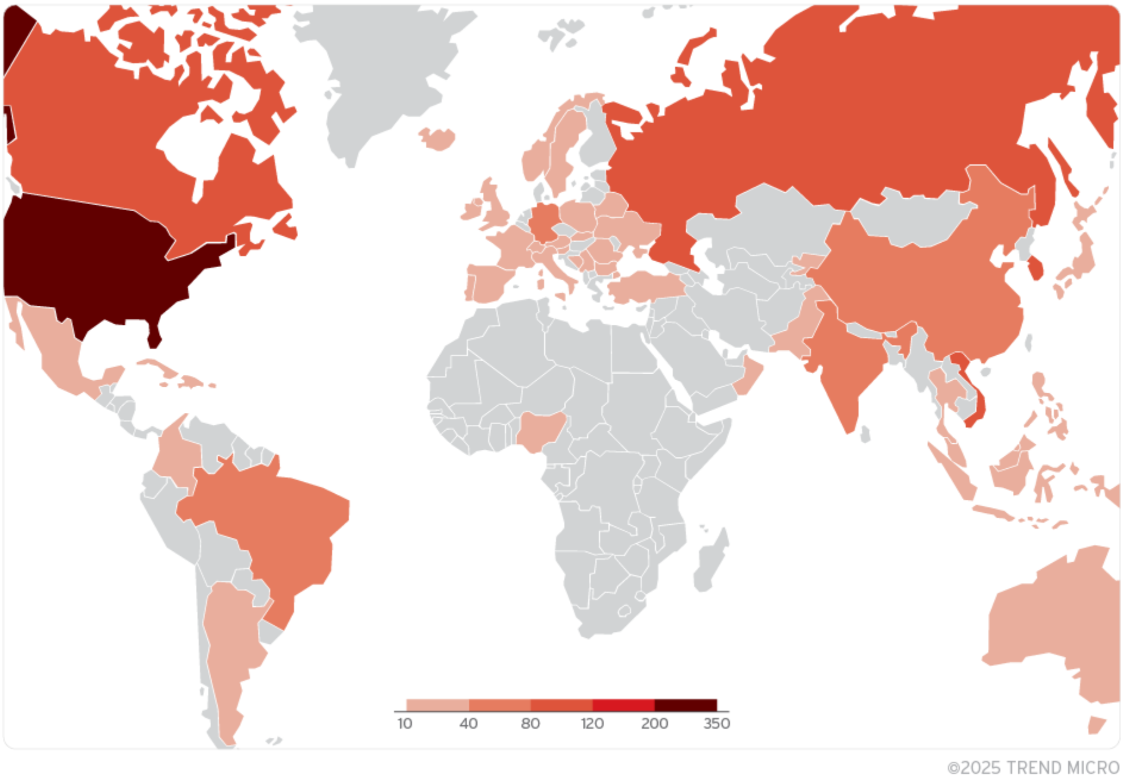


Figure 8. Files exploiting ZDI-CAN-25373 countries by file submission origin
download

The raw data for Figure 8 may be found here.

Technical details

ZDI-CAN-25373 relates to the way Windows displays the contents of shortcut (.lnk) files through the Windows UI. By exploiting this vulnerability, an attacker can prepare a malicious .lnk file for delivery to a victim. Upon examining the file using the Windows-provided user interface, the victim will not be able to tell that the file contains any malicious content. The sections below explain the vulnerability in further detail.

MS-SHLLINK (.lnk) file format

A Windows Shortcut file, also known a Shell Link file (.lnk), is a type of binary file used by the Windows operating system to act as a shortcut to a file, folder or applications. This file type is popular with threat actors due to the fact that command line arguments can be embedded inside the .lnk files ***Target*** field, which can lead to code execution on the victim machine. This section will cover the basics of the Windows Shortcut (MS-SHLLINK) binary format as they relate to understanding the exploitation of ZDI-CAN-25373.

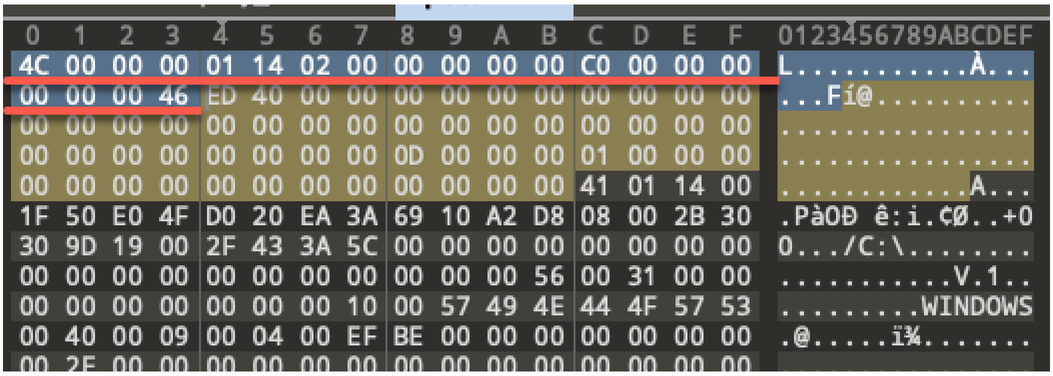
ShellLinkHeader

Every .lnk file starts with the ***ShellLinkHeader*** structure. Within ***ShellLinkHeader***, there are two mandatory fields that contain the same values across every .lnk file (Table 1). These are the fields

HeaderSize and ***LinkCLSID*** (Figure 9). These two fields can be used as magic bytes to determine whether or not a file is a .lnk file.

Field	**Value**
HeaderSize (4 bytes, offset 0x0000)	0x0000004C
LinkCLSID (16 bytes, offset 0x0004)	00021401-0000-0000- C000-000000000046

Table 1. Mandatory fields in the ShellLinkHeader structure

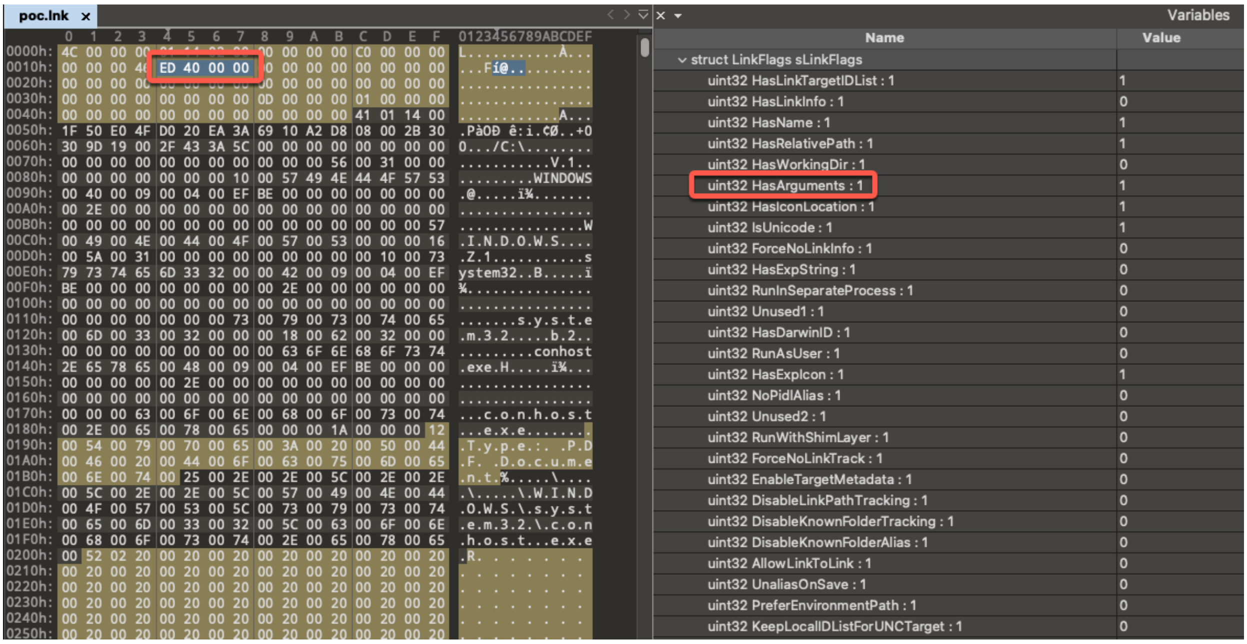


```
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00   L...........À...
00 00 00 46 ED 40 00 00 00 00 00 00 00 00 00 00   ...Fí@..........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 0D 00 00 00 01 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 41 01 14 00   ............A...
1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30   .PàOÐ ê:i.¢Ø..+0
30 9D 19 00 2F 43 3A 5C 00 00 00 00 00 00 00 00   0.../C:\........
00 00 00 00 00 00 00 00 00 00 00 56 00 31 00 00   ...........V.1..
00 00 00 00 00 00 00 10 00 57 49 4E 44 4F 57 53   .........WINDOWS
00 40 00 09 00 04 00 EF BE 00 00 00 00 00 00 00   .@.....ï¾.......
00 2E 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Figure 9. Highlighted HeaderSize and LinkCLSID within an .lnk file
download

LinkFlags

The ***LinkFlags*** structure appears after the ***HeaderSize*** and ***LinkCLSID***. It contains flags specifying which structures are additionally present in the .lnk file. Of special importance is the ***HasArguments*** flag. When this flag is set, the .lnk will contain a ***COMMAND_LINE_ARGUMENTS*** structure (Figure 10). As part of the .lnk binary format, additional COMMAND_LINE_ARGUMENTS may be passed to the .lnk file's Target. In a malicious .lnk file, these may be additional command line parameters to download and install malicious payloads through cmd.exe, powershell.exe, or other LOLbin binaries. This is often abused by threat actors who use .lnk files as part of their attack chains and is present in the exploitation of ZDI-CAN-25373.



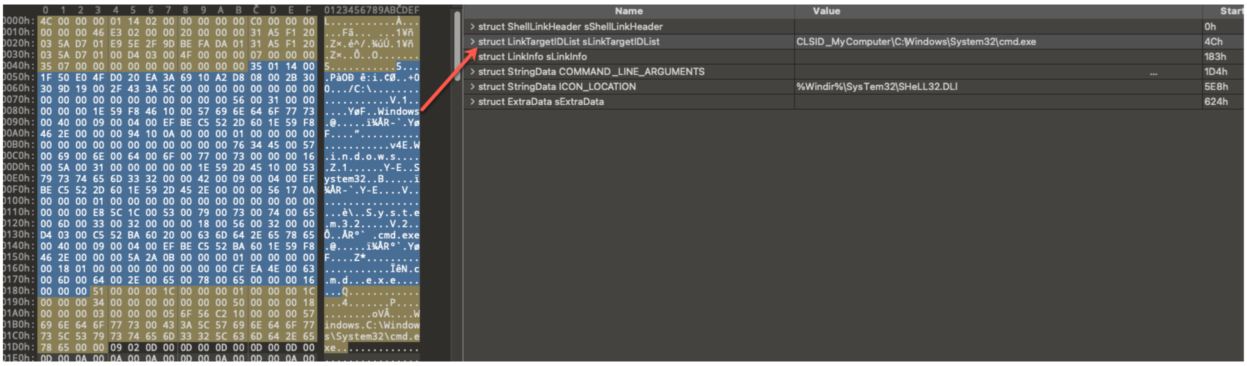
```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00   L...........À...
0010h: 00 00 00 46 ED 40 00 00 00 00 00 00 00 00 00 00   ....Fí@.........
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0030h: 00 00 00 00 00 00 00 00 0D 00 00 00 01 00 00 00   ................
0040h: 00 00 00 00 00 00 00 00 00 00 00 00 41 01 14 00   ............A...
0050h: 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30   .PàOÐ ê:i.¢Ø..+0
0060h: 30 9D 19 00 2F 43 3A 5C 00 00 00 00 00 00 00 00   0.../C:\........
0070h: 00 00 00 00 00 00 00 00 00 00 00 56 00 31 00 00   ...........V.1..
0080h: 00 00 00 00 00 00 00 10 00 57 49 4E 44 4F 57 53   .........WINDOWS
0090h: 00 40 00 09 00 04 00 EF BE 00 00 00 00 00 00 00   .@.....ï¾.......
00A0h: 00 2E 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 57   ...............W
00C0h: 00 49 00 4E 00 44 00 4F 00 57 00 53 00 00 00 16   .I.N.D.O.W.S....
00D0h: 00 5A 00 31 00 00 00 00 00 00 00 00 00 10 00 73   .Z.1...........s
00E0h: 79 73 74 65 6D 33 32 00 00 42 00 09 00 04 00 EF   ystem32..B.....ï
00F0h: BE 00 00 00 00 00 00 00 00 2E 00 00 00 00 00 00   ¾...............
0100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0110h: 00 00 00 00 00 00 00 73 00 79 00 73 00 74 00 65   .......s.y.s.t.e
0120h: 00 6D 00 33 00 32 00 00 00 18 00 62 00 32 00 00   .m.3.2.....b.2..
0130h: 00 00 00 00 00 00 00 00 00 63 6F 6E 68 6F 73 74   .........conhost
0140h: 2E 65 78 65 00 48 00 09 00 04 00 EF BE 00 00 00   .exe.H.....ï¾...
0150h: 00 00 00 00 00 2E 00 00 00 00 00 00 00 00 00 00   ................
0160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0170h: 00 00 00 63 00 6F 00 6E 00 68 00 6F 00 73 00 74   ...c.o.n.h.o.s.t
0180h: 00 2E 00 65 00 78 00 65 00 00 00 1A 00 00 00 12   ...e.x.e........
0190h: 00 54 00 79 00 70 00 65 00 3A 00 20 00 50 00 44   .T.y.p.e.:. .P.D
01A0h: 00 46 00 20 00 44 00 6F 00 63 00 75 00 6D 00 65   .F. .D.o.c.u.m.e
01B0h: 00 6E 00 74 00 25 00 2E 00 2E 00 5C 00 2E 00 2E   .n.t.%.....\....
01C0h: 00 5C 00 2E 00 2E 00 5C 00 57 00 49 00 4E 00 44   .\.....\.W.I.N.D
01D0h: 00 4F 00 57 00 53 00 5C 00 73 00 79 00 73 00 74   .O.W.S.\.s.y.s.t
01E0h: 00 65 00 6D 00 33 00 32 00 5C 00 63 00 6F 00 6E   .e.m.3.2.\.c.o.n
01F0h: 00 68 00 6F 00 73 00 74 00 2E 00 65 00 78 00 65   .h.o.s.t...e.x.e
0200h: 00 52 02 20 00 20 00 20 00 20 00 20 00 20 00 20   .R. . . . . . .
0210h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20   . . . . . . . .
0220h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20   . . . . . . . .
0230h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20   . . . . . . . .
0240h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20   . . . . . . . .
0250h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20   . . . . . . . .
```

Name	Value
struct LinkFlags sLinkFlags	
uint32 HasLinkTargetIDList : 1	1
uint32 HasLinkInfo : 1	0
uint32 HasName : 1	1
uint32 HasRelativePath : 1	1
uint32 HasWorkingDir : 1	0
uint32 HasArguments : 1	1
uint32 HasIconLocation : 1	1
uint32 IsUnicode : 1	1
uint32 ForceNoLinkInfo : 1	0
uint32 HasExpString : 1	0
uint32 RunInSeparateProcess : 1	0
uint32 Unused1 : 1	0
uint32 HasDarwinID : 1	0
uint32 RunAsUser : 1	0
uint32 HasExpIcon : 1	1
uint32 NoPidlAlias : 1	0
uint32 Unused2 : 1	0
uint32 RunWithShimLayer : 1	0
uint32 ForceNoLinkTrack : 1	0
uint32 EnableTargetMetadata : 1	0
uint32 DisableLinkPathTracking : 1	0
uint32 DisableKnownFolderTracking : 1	0
uint32 DisableKnownFolderAlias : 1	0
uint32 AllowLinkToLink : 1	0
uint32 UnaliasOnSave : 1	0
uint32 PreferEnvironmentPath : 1	0
uint32 KeepLocalIDListForUNCTarget : 1	0

Figure 10. Highlighted LinkFlags and HasArguments flag pointing to the presence of COMMAND_LINE_ARGUMENTS StringData structure
download

LinkTargetIDList

This structure contains the target of the .lnk file, as shown in Figure 11. When this structure is used, the ***HasLinkTargetIDList*** flag will be set in the ***LinkFlags*** section.



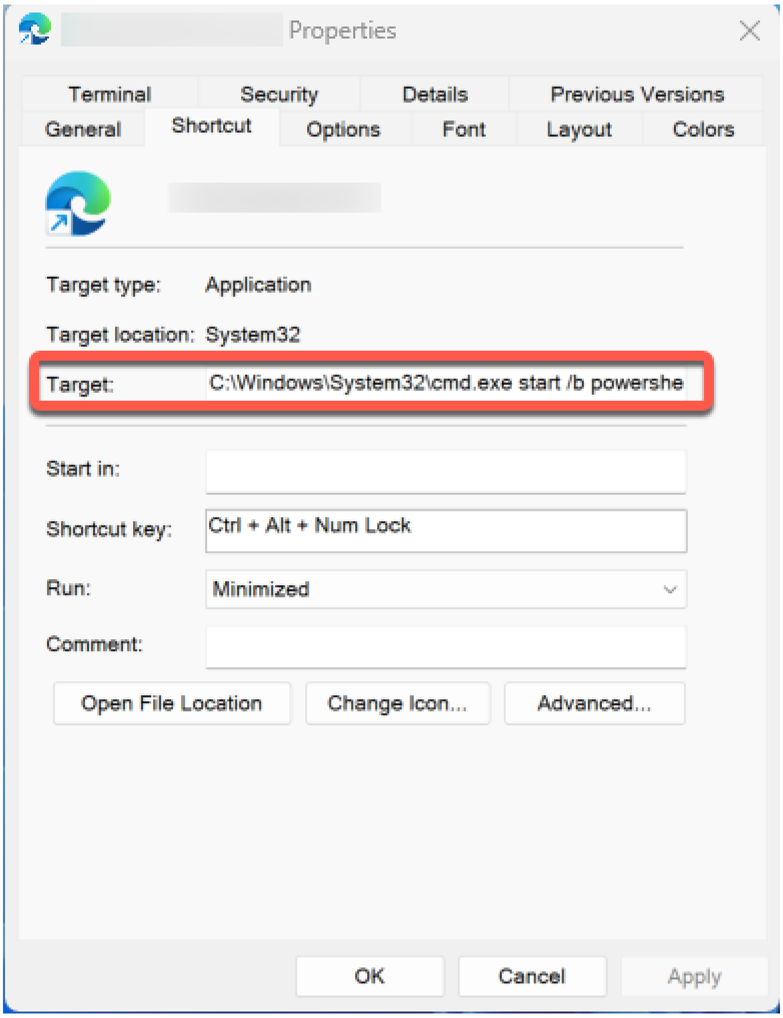
```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00  L...........À...
0010h: 00 00 00 46 E3 02 00 00 20 00 00 00 31 A5 F1 20  ...Fã... ...1¥ñ 
0020h: 03 5A D7 01 E9 5E 2F 9D BE FA DA 01 31 A5 F1 20  .Z×.é^/.¾úÚ.1¥ñ 
0030h: 03 5A D7 01 00 D4 03 00 4F 00 00 00 07 00 00 00  .Z×..Ô..O.......
0040h: 35 07 00 00 00 00 00 00 00 00 00 00 35 01 14 00  5...........5...
0050h: 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30  .PàOÐ ê:i.¢Ø..+0
0060h: 30 9D 19 00 2F 43 3A 5C 00 00 00 00 00 00 00 00  0.../C:\........
0070h: 00 00 00 00 00 00 00 00 00 00 00 56 00 31 00 00  ...........V.1..
0080h: 00 00 00 1E 59 F8 46 10 00 57 69 6E 64 6F 77 73  ....YøF..Windows
0090h: 00 40 00 09 00 04 00 EF BE C5 52 2D 60 1E 59 F8  .@.....ï¾ÅR-`.Yø
00A0h: 46 2E 00 00 00 94 10 0A 00 00 00 01 00 00 00 00  F....”..........
00B0h: 00 00 00 00 00 00 00 00 00 00 00 76 34 45 00 57  ...........v4E.W
00C0h: 00 69 00 6E 00 64 00 6F 00 77 00 73 00 00 00 16  .i.n.d.o.w.s....
00D0h: 00 5A 00 31 00 00 00 00 00 1E 59 2D 45 10 00 53  .Z.1......Y-E..S
00E0h: 79 73 74 65 6D 33 32 00 00 42 00 09 00 04 00 EF  ystem32..B.....ï
00F0h: BE C5 52 2D 60 1E 59 2D 45 2E 00 00 00 56 17 0A  ¾ÅR-`.Y-E....V..
0100h: 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00  ................
0110h: 00 00 00 E8 5C 1C 00 53 00 79 00 73 00 74 00 65  ...è\..S.y.s.t.e
0120h: 00 6D 00 33 00 32 00 00 00 18 00 56 00 32 00 00  .m.3.2.....V.2..
0130h: D4 03 00 C5 52 BA 60 20 00 63 6D 64 2E 65 78 65  Ô..ÅR°` .cmd.exe
0140h: 00 40 00 09 00 04 00 EF BE C5 52 BA 60 1E 59 F8  .@.....ï¾ÅR°`.Yø
0150h: 46 2E 00 00 00 5A 2A 0B 00 00 00 01 00 00 00 00  F....Z*.........
0160h: 00 18 01 00 00 00 00 00 00 00 00 CF EA 4E 00 63  ...........ÏêN.c
0170h: 00 6D 00 64 00 2E 00 65 00 78 00 65 00 00 00 16  .m.d...e.x.e....
0180h: 00 00 00 51 00 00 00 1C 00 00 00 01 00 00 00 1C  ...Q............
0190h: 00 00 00 34 00 00 00 00 00 00 00 50 00 00 00 18  ...4.......P....
01A0h: 00 00 00 03 00 00 00 05 6F 56 C2 10 00 00 00 57  ........oVÂ....W
01B0h: 69 6E 64 6F 77 73 00 43 3A 5C 57 69 6E 64 6F 77  indows.C:\Window
01C0h: 73 5C 53 79 73 74 65 6D 33 32 5C 63 6D 64 2E 65  s\System32\cmd.e
01D0h: 78 65 00 00 09 02 0D 00 0D 00 0D 00 0D 00 0D 00  xe..............
01E0h: 0D 00 0A 00 0A 00 0A 00 0D 00 0A 00 0D 00 0A 00  ................
```

Name	Value	Start
> struct ShellLinkHeader sShellLinkHeader		0h
> struct LinkTargetIDList sLinkTargetIDList	CLSID_MyComputer\C:\Windows\System32\cmd.exe	4Ch
struct LinkInfo sLinkInfo		183h
> struct StringData COMMAND_LINE_ARGUMENTS	...	1D4h
> struct StringData ICON_LOCATION	%Windir%\SysTem32\SHeLL32.DLl	5E8h
> struct ExtraData sExtraData		624h

Figure 11. The LinkTargetIDList contains the target of .lnk file
download

COMMAND_LINE_ARGUMENTS

When the ***HasArguments*** flag is set in the ***LinksFlags*** structure, the ***COMMAND_LINE_ARGUMENTS*** structure will be present. This structure stores command line arguments that will be passed to the specified target is when the shortcut is activated (Figure 12). This is often used in .lnk file-based attacks for code execution.



Target type: Application

Target location: System32

Target: C:\Windows\System32\cmd.exe start /b powershe

Start in:

Shortcut key: Ctrl + Alt + Num Lock

Run: Minimized

Comment:

Figure 12. The COMMAND_LINE_ARGUMENTS structure contains commands within the .lnk file's Target field
download

ICON_LOCATION

Threat actors also commonly abuse the ***ICON_LOCATION*** structure to control what icon is displayed by the .lnk file, as shown in Figure 13.

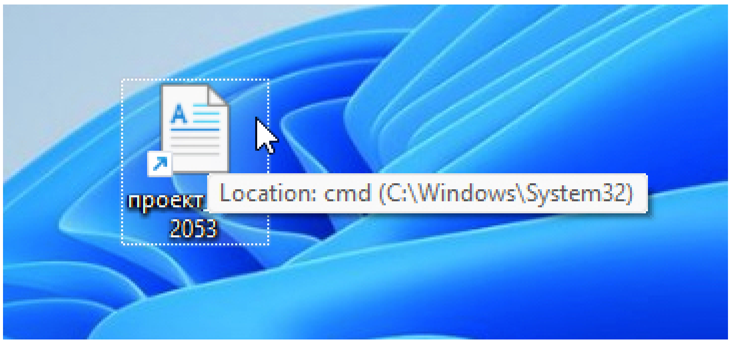


Figure 13. The ICON_LOCATION structure can be used to change the .lnk file icon
download

In attack campaigns that utilize .lnk files, threat actors will often change the icon to confuse and entice the victim into executing the shortcut. Since Windows always suppresses display of the .lnk extension, threat actors will often add a "spoof" extension such as .pdf.lnk along with a matching icon to further trick users. A .lnk file will usually have an arrow on the lower-left side of the icon.

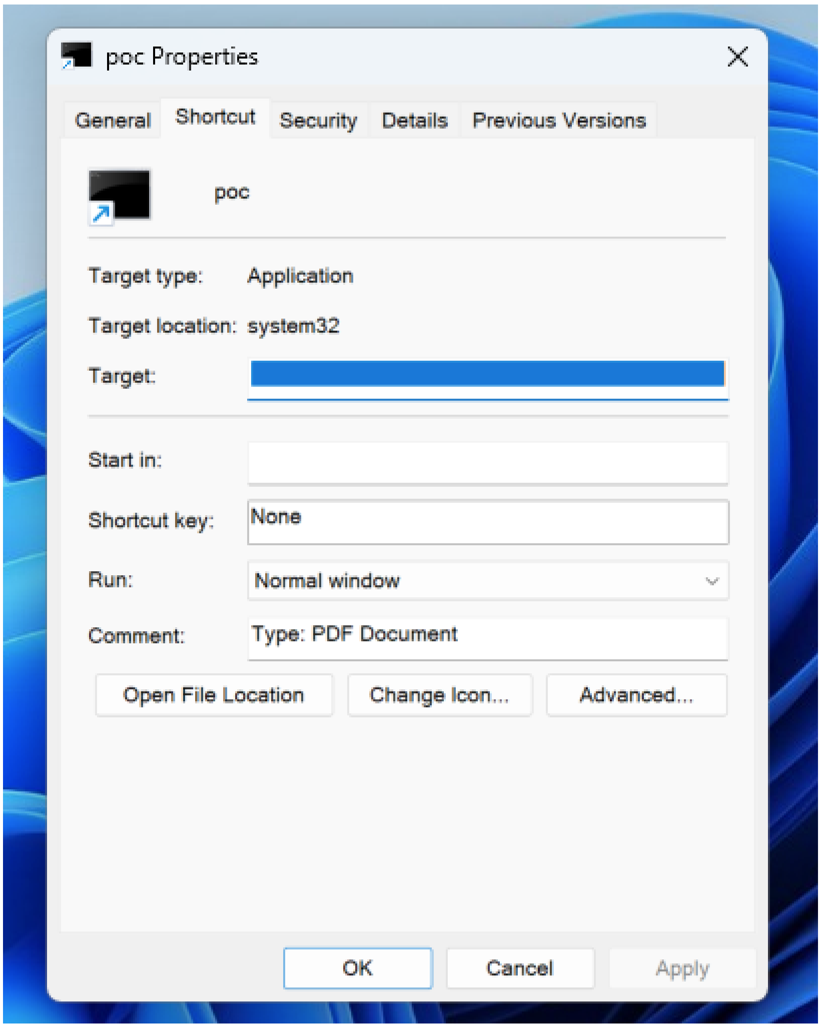
Vulnerability details

To exploit ZDI-CAN-25373, the threat actors carefully crafted .lnk files with padded whitespace characters within the ***COMMAND_LINE_ARGUMENTS*** structure (Table 2).

Hex Value	Output
\x20	Space
\x09	Horizontal Tab
\x0A	Line Feed
\x0B	Vertical Tab
\x0C	Form Feed
\x0D	Carriage Return

Table 2. Whitespace characters used to exploit ZDI-CAN-25373

If a user inspects a .lnk file containing this malicious padding, Windows will not be able to show the malicious arguments within the allotted space in the user interface (Figure 14). The impact of this exploit is that the command line arguments that will be executed by the .lnk file are completely hidden from the user's view.



poc Properties

poc

Target type: Application

Target location: system32

Target:

Start in:

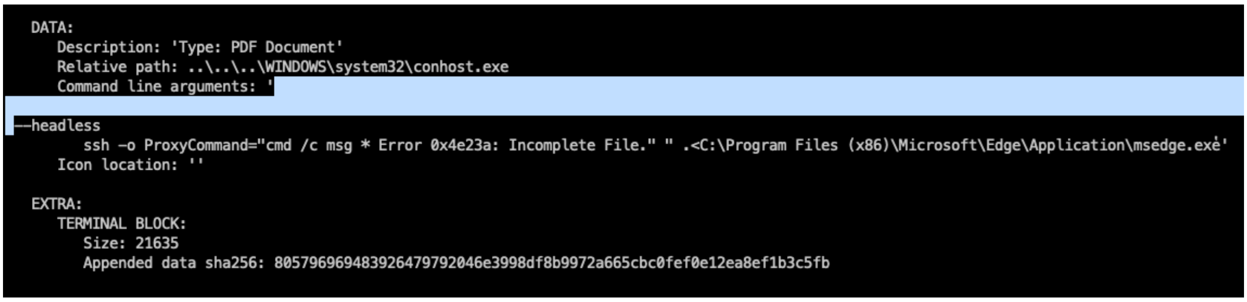
Shortcut key: None

Run: Normal window

Comment: Type: PDF Document

Figure 14. Properties of a Space (\x20) or Horizontal Tab (\x09) padded .lnk file within the Target field
download

To inspect the Target field of samples exploiting ZDI-CAN-25373 and the contents of the ***COMMAND_LINE_ARGUMENTS*** structure, third-party tools are required.



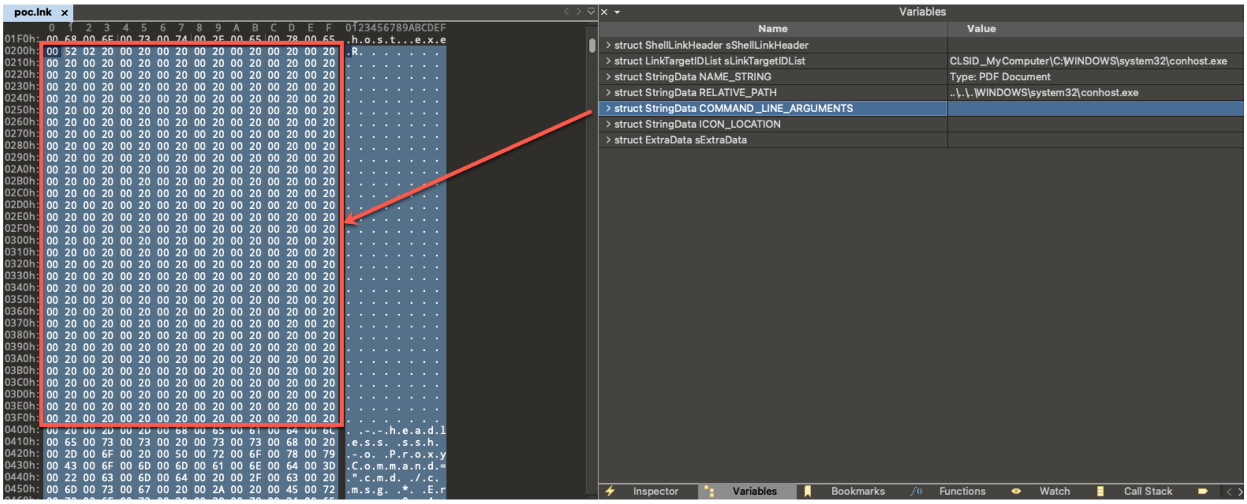
```
DATA:
   Description: 'Type: PDF Document'
   Relative path: ..\..\..\WINDOWS\system32\conhost.exe
   Command line arguments: '

--headless
      ssh -o ProxyCommand="cmd /c msg * Error 0x4e23a: Incomplete File." " .<C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe'
   Icon location: ''

EXTRA:
   TERMINAL BLOCK:
      Size: 21635
      Appended data sha256: 805796969483926479792046e3998df8b9972a665cbc0fef0e12ea8ef1b3c5fb
```

Figure 15. Whitespace placed in front the .lnk command line argument
download

When viewing a sample using a hex editor, one can observe large numbers of Space (\x20) characters within the ***COMMAND_LINE_ARGUMENTS*** structure (Figure 16).

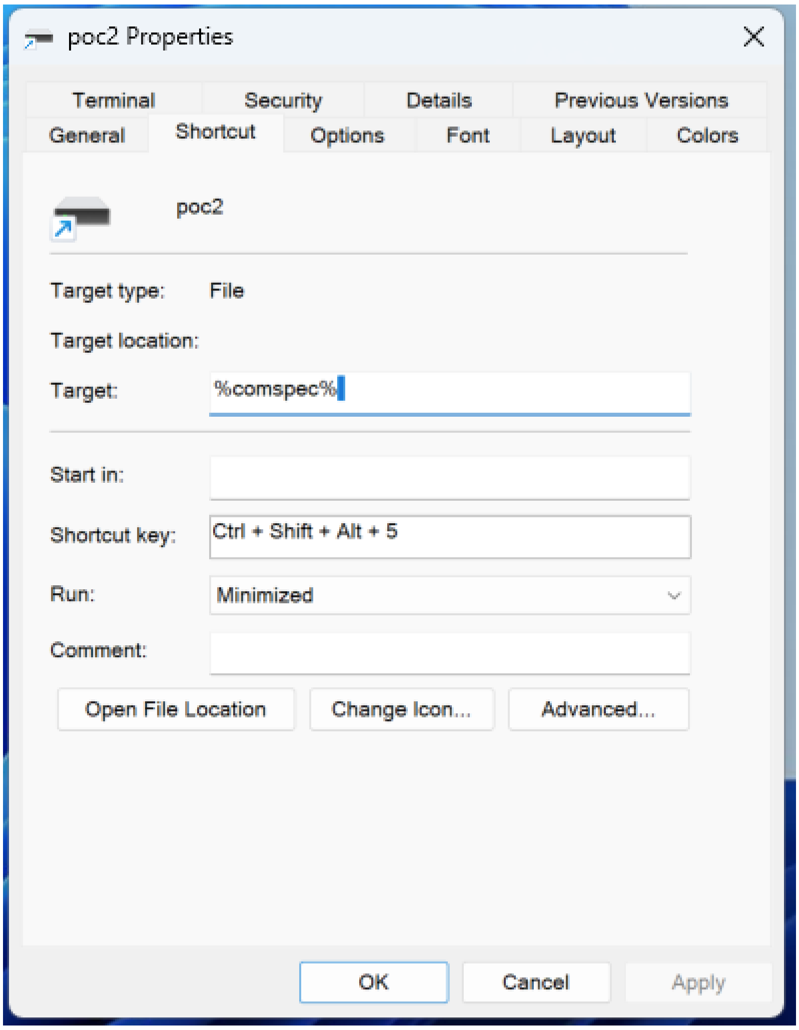


```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
01F0h: 00 68 00 6F 00 73 00 74 00 2E 00 65 00 78 00 65  .h.o.s.t...e.x.e
0200h: 00 52 02 20 00 20 00 20 00 20 00 20 00 20 00 20  .R. . . . . . .
0210h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0220h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0230h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0240h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0250h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0260h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0270h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0280h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0290h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
02A0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
02B0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
02C0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
02D0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
02E0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
02F0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0300h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0310h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0320h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0330h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0340h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0350h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0360h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0370h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0380h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0390h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
03A0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
03B0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
03C0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
03D0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
03E0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
03F0h: 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20  . . . . . . . .
0400h: 00 20 00 2D 00 2D 00 68 00 65 00 61 00 64 00 6C  . .-.-.h.e.a.d.l
0410h: 00 65 00 73 00 73 00 20 00 73 00 73 00 68 00 20  .e.s.s. .s.s.h.
0420h: 00 2D 00 6F 00 20 00 50 00 72 00 6F 00 78 00 79  .-.o. .P.r.o.x.y
0430h: 00 43 00 6F 00 6D 00 6D 00 61 00 6E 00 64 00 3D  .C.o.m.m.a.n.d.=
0440h: 00 22 00 63 00 6D 00 64 00 20 00 2F 00 63 00 20  .".c.m.d. ./.c.
0450h: 00 6D 00 73 00 67 00 20 00 2A 00 20 00 45 00 72  .m.s.g. .*. .E.r
```

Name	Value
> struct ShellLinkHeader sShellLinkHeader	
> struct LinkTargetIDList sLinkTargetIDList	CLSID_MyComputer\C:\WINDOWS\system32\conhost.exe
> struct StringData NAME_STRING	Type: PDF Document
> struct StringData RELATIVE_PATH	..\..\..\WINDOWS\system32\conhost.exe
> struct StringData COMMAND_LINE_ARGUMENTS	
> struct StringData ICON_LOCATION	
> struct ExtraData sExtraData	

Figure 16. Large amounts of Space (\x20) characters within the COMMAND_LINE_ARGUMENTS structure
download

We made an interesting observation in cases where the threat actor padded the COMMAND_LINE_ARGUMENTS with Line Feed (\x0A) and Carriage Return (\x0D) characters. In this instance, when inspecting the properties of the .lnk file, only a single highlightable character in the Target field is shown (Figure 17).


poc2 Properties

Target type: File

Target location:

Target: %comspec%

Start in:

Shortcut key: Ctrl + Shift + Alt + 5

Run: Minimized

Comment:

Figure 17. Result of padding with sequences of Line Feed (\x0A) and Carriage Return (\x0) characters
download

During our analysis, we discovered that some North Korean threat actors, such as Earth Manticore (APT37) and Earth Imp (Konni), tended to use extremely large .lnk files with large amounts of whitespace and other junk content to further evade detection. Earth Imp used files with a median size of 3.32MB, with a maximum file size of 70.1MB. Earth Manticore used files with a median size of 33.33MB, with a maximum file size of 55.16MB.

ZDI-CAN-25373 is an example of (User Interface (UI) Misrepresentation of Critical Information (CWE-451). This means that the Windows UI failed to present the user with critical information. Similar to a previous discovery we made, it is a failure to properly represent security-critical information to the user. By exploiting ZDI-CAN-25373, the threat actor can prevent the end user from viewing critical information (commands being executed) related to evaluating the risk level of the file.

Conclusion

The threat posed by APTs originating from nation-states, as well as from sophisticated cybercriminals, poses a significant risk to the confidentiality, integrity, and availability of data maintained by governments, critical infrastructure, and private organizations globally. Among the 11 state-sponsored APT groups leveraging ZDI-CAN-25373, a majority have a documented history of exploiting zero-day vulnerabilities in attacks in the wild. These vulnerabilities present substantial risks, as they target flaws that remain unknown to software vendors and lack corresponding security patches, thereby leaving governments and organizations vulnerable to exploitation. As geopolitical tensions and conflicts escalate, an increase in the sophistication of threat actors and the utilization of zero-day vulnerabilities is anticipated to rise, as both nation-states and cybercriminals endeavor to gain a competitive advantage over their adversaries. This growing prevalence of zero-day exploitation necessitates the implementation of comprehensive security solutions to safeguard critical assets and industries effectively. This vulnerability was disclosed to Microsoft via Trend ZDI's bug bounty program; Microsoft classified this as low severity and this will not be patched in the immediate future.

To make software more secure and protect customers from zero-day attacks, Trend ZDI works with security researchers and vendors to patch and responsibly disclose software vulnerabilities before APT groups can deploy them in attacks. The Trend ZDI threat hunting team also proactively hunts for zero-day attacks in the wild to safeguard the industry.

Proactive security with Trend Vision One™

Organizations can protect themselves from attacks such as those employed by APT groups with Trend Vision One™ – the only AI-powered enterprise cybersecurity platform that centralizes cyber risk exposure management, security operations, and robust layered protection. This comprehensive approach helps you predict and prevent threats, accelerating proactive security outcomes across your entire digital estate. Backed by decades of cybersecurity leadership and Trend Cybertron, the industry's first proactive cybersecurity AI, it delivers proven results: a 92% reduction in ransomware risk and a 99% reduction in detection time. Security leaders can benchmark their posture and showcase continuous improvement to stakeholders. With Trend Vision One, you're enabled to eliminate security blind spots, focus on what matters most, and elevate security into a strategic partner for innovation.

When faced with uncertain intrusions, behaviors, and routines, organizations should assume that their system is already compromised or breached and work to immediately isolate affected data or toolchains. With a broader perspective and rapid response, organizations can address breaches and protect its remaining systems, especially with robust Endpoint and Network Security solutions. The platform's security operations capabilities stop adversaries with unrivalled visibility—enriched by native sensors and third-party telemetry. Detect, investigate, and respond proactively with the power of XDR, SIEM, and SOAR. *Leaving attackers with no place left to hide.*

Trend rules and filters for ZDI-CAN-25373

The following protections have been available to Trend Micro customers:

Trend Vision One™ – Network Security

- 44844 - ZDI-CAN-25373: Zero Day Initiative Vulnerability (Microsoft Windows)

Trend Vision One™ – Endpoint Security, Trend Micro™ Deep Security™, Trend Vision One Network Sensor and Trend Micro Deep Discovery Inspector (DDI)

- 5351 - ZDI-CAN-25373 MICROSOFT WINDOWS ZERO DAY VULNERABILITY - HTTP(RESPONSE)
- 1012182 - Microsoft Windows Zero Day Vulnerability Over HTTP (ZDI-CAN-25373)

- 1012183 - Microsoft Windows Zero Day Vulnerability Over SMB (ZDI-CAN-25373)

More Trend rules and filters can be found here.

Trend Vision One Threat Intelligence

To stay ahead of evolving threats, Trend Vision One customers can access a range of Intelligence Reports and Threat Insights. Threat Insights helps customers stay ahead of cyber threats before they happen and allows them to prepare for emerging threats by offering comprehensive information on threat actors, their malicious activities, and their techniques. By leveraging this intelligence, customers can take proactive steps to protect their environments, mitigate risks, and effectively respond to threats.

Trend Vision One Threat Insights App

- **Threat Actors:** Water Glashtyn, **Earth Iktomi, Water Poukai, Water Cetus, Earth Balayang, Fire Tengu,** Earth Imp, Water Asena, **Earth Akurra**, Earth Gelert**Earth Iktomi, Water Poukai, Water Cetus, Earth Balayang, Fire Tengu,** Earth Imp, Water Asena, **Earth Akurra**, Earth Anansi, Earth Gelert, Earth Vetala, Earth Kapre, Earth Preta, Earth Tengshe, Earth Lusca
- **Emerging Threats**: ZDI-CAN-25373: Windows Shortcut Exploit Abused as Zero-Day in Widespread APT Campaigns

Hunting Queries

Trend Vision One Search App

Trend Vision One customers can use the Search App to match or hunt the malicious indicators mentioned in this blog post with data in their environment.

Detect suspicious cmd.exe or powershell.exe execution from LNK files

```
 eventSubId:2 AND (processFilePath:\"*\\cmd.exe\" OR processFilePath:\"*\\powershell.exe\") AND parentFilePath:\"*.lnk\"
```

More hunting queries are available for Trend Vision One customers with Threat Insights Entitlement enabled.

Yara Threat Hunting Rule

Use this YARA rule to find files exploiting ZDI-CAN-25373:

```
rule ZTH_LNK_EXPLOIT_A
{
  meta:
    author = "Peter Girnus"
    description = "This YARA file detects padded LNK files designed to exploit ZDI-CAN-25373."
    reference = "<LINK_TO_BLOG>"
    target_entity = "file"
  strings:
    $magic = {4C 00 00 00 01 14 02 00}
    $spoof_a = {20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00
20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20
00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00}
    $spoof_b = {09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00
09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09
00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00 09 00}
    $spoof_c = {0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A
00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A     00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00
0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00}
    $spoof_d = {0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D
00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00
0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00}
    $spoof_e = {11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11
00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11
00 11 00 11 00 11 00 11 00 11 00 11 00 11 00 11 00}
    $spoof_f = {12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12
00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00
12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00 12 00}
    $spoof_g = {13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00
13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13
00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00 13 00}
    $spoof_h = {0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A
00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A
00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00 0D 00 0A 00}
  condition:
        $magic at 0x00 and ($spoof_a or $spoof_b or $spoof_c or $spoof_d or $spoof_e or $spoof_f or
$spoof_g or $spoof_h)
}
```

Indicators of compromise

The indicators of compromise for this entry can be found here.