Trust me, I'm a shortcut 

New LNK abuse methods

@Wietze

BLACK ALPS 2025

Wietze Beukema

Lead Threat Detection & Response Engineer

- → Passion for cyber security research
- → Loves open-source, community projects
- → Presented at various cyber conferences

Let’s Start.Ink

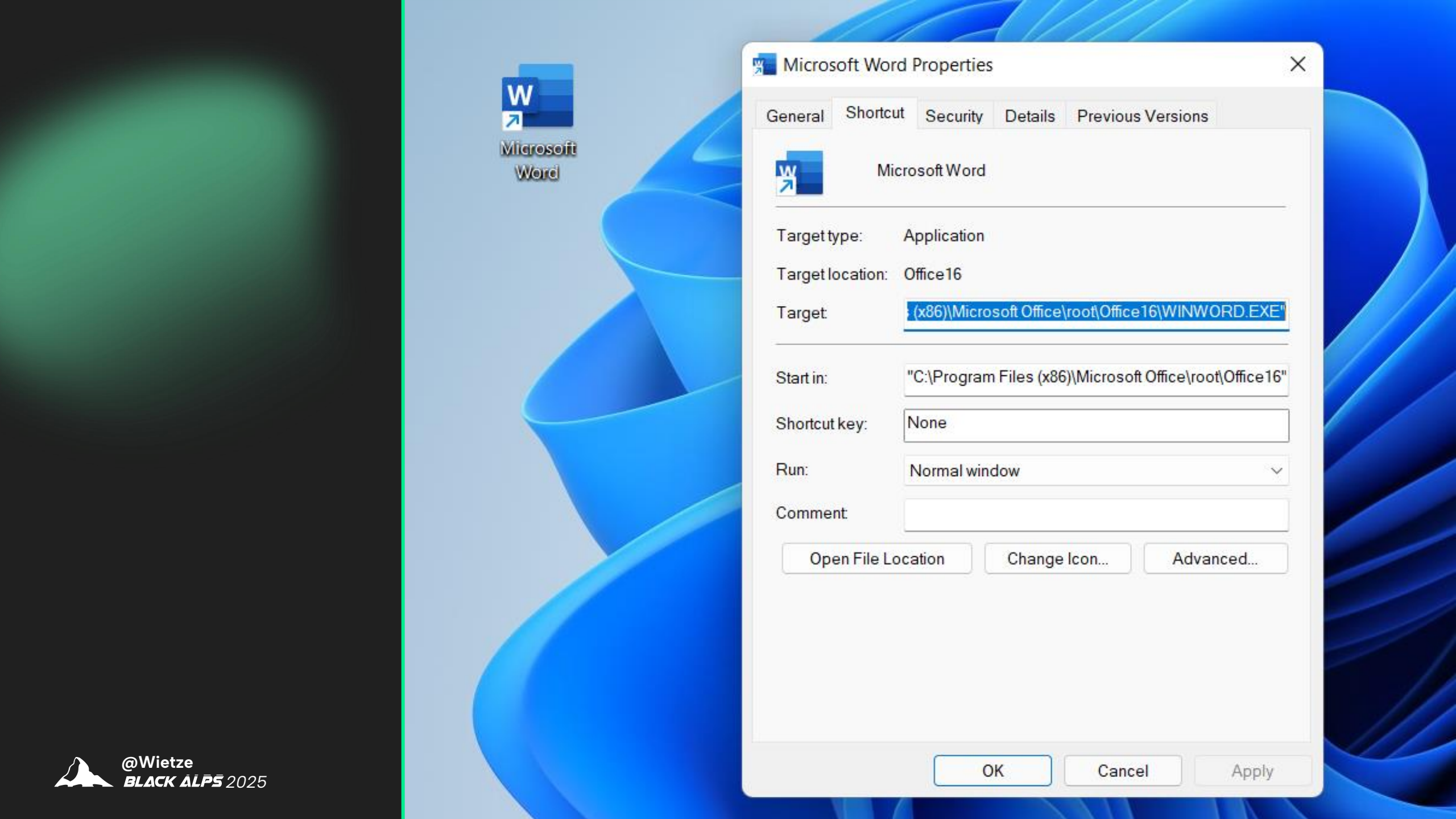
Shortcuts

Access **resources** quickly – on Windows:

- .LNK files
- .URL files

Mainly a **UI concept**

Custom binary format

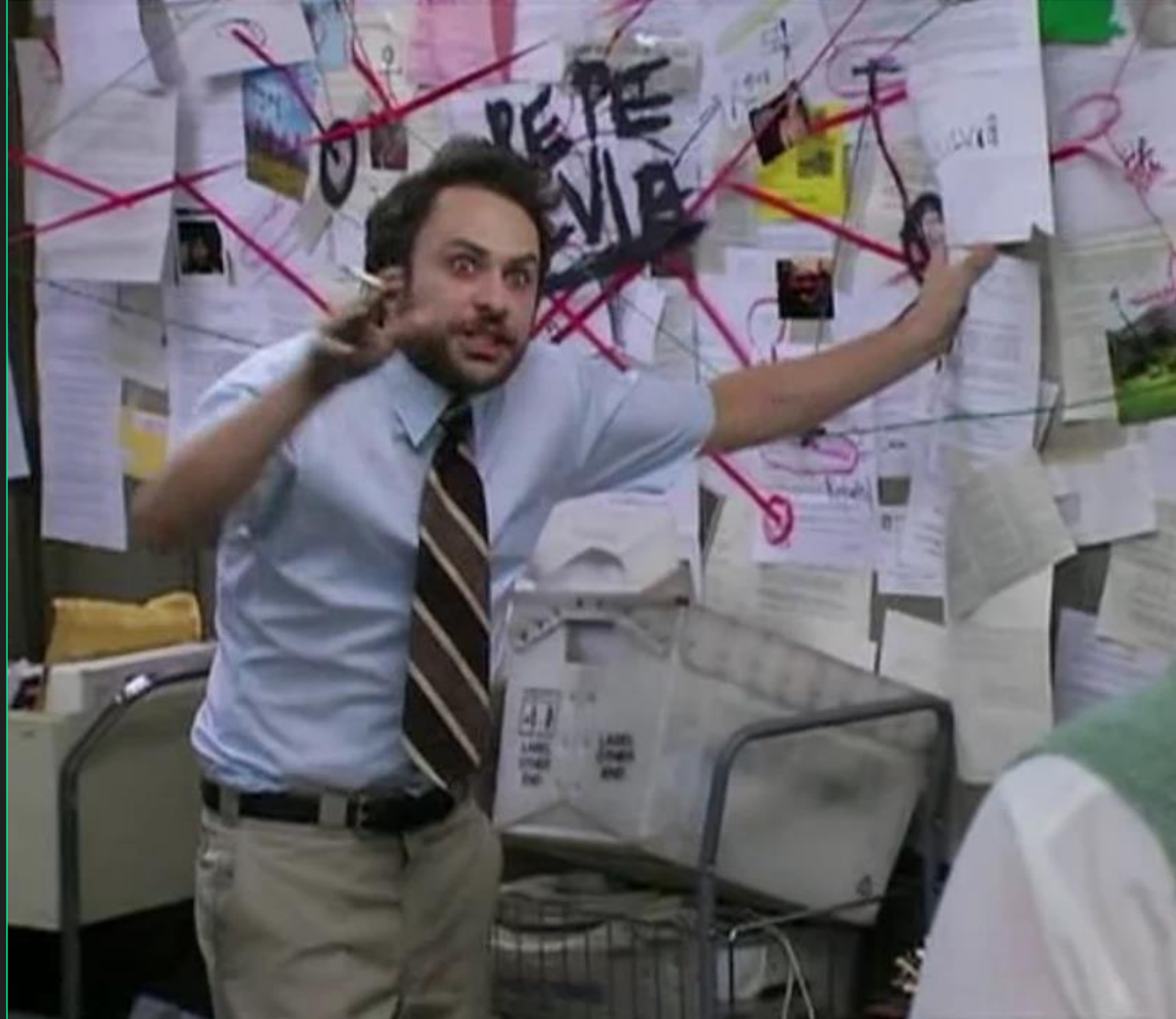


.LNK Abuse

As old as Windows 95 itself:

- **Exploits**: e.g. Stuxnet (CVE-2010-2568)
- **Initial access**: USB worms, loaders/droppers
- **Persistence**: create/update LNKs
- **Defence Evasion**: masquerading

.LNK format



Value	Description
FILE_ATTRIBUTE_DIRECTORY	
FILE_ATTRIBUTE_ARCHIVE	The file or directo flag to mark files
Reserved2	A bit that MUST b
FILE_ATTRIBUTE_NORMAL	The file or directo other bits in this s
FILE_ATTRIBUTE_TEMPORARY	The file is being u
FILE_ATTRIBUTE_SPARSE_FILE	The file is a spar
FILE_ATTRIBUTE_REPARSE_POINT	The file or directo
FILE_ATTRIBUTE_COMPRESSED	The file or directo all data in the file that compression subdirectories.

Value	Description
	BlockSignature

BlockSize (
structure

BlockSignat
SpecialFo

SpecialFold

Offset (4 by
child seg
the link t

2.5.10 Trac

The TrackerD
found in its d
[MS-DLTW] t

Value	Description
	Meaning
C	Offsets to the optional fields are not specified.
4 ≤ *value*	Offsets to the optional fields are specified.

(4 bytes): Flags that specify whether the **Volume**
PathUnicode, and **CommonNetworkRelativeLink**

bits are defined as:

Value	Description
ndLocalBasePath	If set, the **VolumeID** a and their locations are s **VolumeIDOffset** and **I** respectively. If the valu greater than or equal to **LocalBasePathUnicod** specified by the value o field. If not set, the **VolumeI LocalBasePathUnicod** values of the **VolumeI** fields are zero. If the va is greater than or equal **LocalBasePathOffset**
etworkRelativeLinkAndPathSuffix	If set, the **CommonNe** and its location is specif **NetworkRela**

Vendor name	Value
WNNC_NET_EXTENDNET	0x00290000
WNNC_NET_STAC	0x002A0000
WNNC_NET_FOXBAT	0x002B0000
WNNC_NET_YAHOO	0x002C0000
WNNC_NET_EXIFS	0x002D0000
WNNC_NET_DAV	0x002E0000
WNNC_NET_KNOWARE	0x002F0000
WNNC_NET_OBJECT_DIRE	0x00300000
WNNC_NET_MASFAX	0x00310000
WNNC_NET_HOB_NFS	0x00320000
WNNC_NET_SHIVA	0x00330000
WNNC_NET_IBMAL	0x00340000
WNNC_NET_LOCK	0x00350000
WNNC_NET_TERMSRV	0x00360000
WNNC_NET_SRT	0x00370000
WNNC_NET_QUINCY	0x00380000
WNNC_NET_OPENAFS	0x00390000
WNNC_NET_AVID1	0x003A0000
WNNC_NET_DFS	0x003B0000
WNNC_NET_KWNP	0x003C0000
WNNC_NET_ZENWORKS	0x003D0000

CommonNetworkRelativeLinkFlags (4 bytes): Flags that specify the contents of the **DeviceNameOffset** and **NetProviderType** fields.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
A	B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Where the bits are defined as:

Value	Description
A ValidDevice	If set, the **DeviceNameOffset** field contains an offset to the device name. If not set, the **DeviceNameOffset** field does not contain an offset to the device name, and its value MUST be zero.
	iderType field contains the network provider type. **roviderType** field does not contain the network provider type, and its

t, unsigned integer that specifies the location of the **NetName**
ytes, from the start of the CommonNetworkRelativeLink structure.

2-bit, unsigned integer that specifies the location of the
Device flag is set, this value is an offset, in bytes, from the start
ink structure; otherwise, this value MUST be zero.

A 32-bit, unsigned integer that specifies the type of network
ag is set, this value MUST be one of the following; otherwise, this

A0000
B0000
C0000
D0000
E0000
F0000
200000
210000
220000
230000
240000
250000
260000
270000

CommonNetworkRelativeLink structure. This field MUST be present if the value of the **DeviceNameOffset** field is greater than 0x00000014; otherwise, this field MUST NOT be present.

NetName (variable, optional): An optional, NULL–terminated string, as defined by the system default code page, which specifies a server share path; for example, "\\server\share".

DeviceName (variable, optional): An optional, NULL–terminated string, as defined by the system default code page, which specifies a device; for example, the drive letter "D:". This field MUST be present if the **ValidDevice** flag is set; otherwise, this field MUST NOT be present.

NetNameUnicode (variable, optional): An optional, NULL–terminated, **Unicode** string that is the Unicode version of the **NetName** string. This field MUST be present if the value of the **NetNameOffset** field is greater than 0x00000014; otherwise, this field MUST NOT be present.

DeviceNameUnicode (variable, optional): An optional, NULL–terminated, Unicode string that is the Unicode version of the **DeviceName** string. This field MUST be present if the **ValidDevice** flag is set and the value of **NetNameOffset** field is greater than 0x00000014; otherwise, this field MUST NOT be present.

2.4 StringData

StringData refers to a set of structures that convey user interface and path identification information. The presence of these optional structures is controlled by LinkFlags (section 2.1.1) in the ShellLinkHeader (section 2.1).

The StringData structures conform to the following **ABNF** rules [RFC5234].

```
STRING_DATA = [NAME_STRING] [RELATIVE_PATH] [WORKING_DIR]
              [COMMAND_LINE_ARGUMENTS] [ICON_LOCATION]
```

NAME_STRING: An optional structure that specifies a description of the **shortcut** that is displayed to end users to identify the purpose of the **shell link**. This structure MUST be present if the **HasName** flag is set.

RELATIVE_PATH: An optional structure that specifies the location of the **link target** relative to the file that contains the shell link. When specified, this string SHOULD be used when resolving the **link**. This structure MUST be present if the **HasRelativePath** flag is set.

WORKING_DIR: An optional structure that specifies the file system path of the working directory to be used when activating the link target. This structure MUST be present if the **HasWorkingDir** flag is set.

COMMAND_LINE_ARGUMENTS: An optional structure that stores the command-line arguments that are specified when activating the link target. This structure MUST be present if the **HasArguments** flag is set.

ICON_LOCATION: An optional structure that specifies the location of the icon to be used when displaying a shell link item in an icon view. This structure MUST be present if the **HasIconLocation** flag is set.

All StringData structures have the following structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CountCharacters																String (variable)															

HeaderSize
LinkCLSID (16 bytes)
...
...

R RunWithShimL
S ForceNoLinkTra
T EnableTargetM
U DisableLinkPat
V

mounted at the path.
umb drive, or flash card
rive.

0x00000004	
DRIVE_CDROM 0x00000005	The drive is a CD-ROM drive.
DRIVE_RAMDISK 0x00000006	The drive is a RAM drive.

DriveSerialNumber (4 bytes): A 32-bit, unsigned integer that specifies the drive serial number of the volume the link target is stored on.

VolumeLabelOffset (4 bytes): A 32-bit, unsigned integer that specifies the location of a string that contains the volume label of the drive that the link target is stored on. This value is an offset, in bytes, from the start of the VolumeID structure to a NULL-terminated string of characters, defined by the system default code page. The volume label string is located in the **Data** field of this structure.

If the value of this field is 0x00000014, it MUST be ignored, and the value of the **VolumeLabelOffsetUnicode** field MUST be used to locate the volume label string.

VolumeLabelOffsetUnicode (4 bytes): An optional, 32-bit, unsigned integer that specifies the location of a string that contains the volume label of the drive that the link target is stored on. This value is an offset, in bytes, from the start of the VolumeID structure to a NULL-terminated string of **Unicode** characters. The volume label string is located in the **Data** field of this structure.

CommonPathSuffixOffset (4 bytes): A 32-bit, unsigned integer that specifies the location of the **CommonPathSuffix** field. This value is an offset, in bytes, from the start of the LinkInfo structure.

LocalBasePathOffsetUnicode (4 bytes): location of the **LocalBasePathUnicode** this value is an offset, in bytes, from the MUST be zero. This field can be present greater than or equal to 0x00000024.

CommonPathSuffixOffsetUnicode (4 byt the location of the **CommonPathSuffix** start of the LinkInfo structure. This field **LinkInfoHeaderSize** field is greater tha

VolumeID (variable): An optional VolumeI about the volume that the **link target** w the **VolumeIDAndLocalBasePath** flag

LocalBasePath (variable): An optional, NU page, which is used to construct the full string in the **CommonPathSuffix** field. flag is set.

CommonNetworkRelativeLink (variable) (section 2.3.2) that specifies information stored.

CommonPathSuffix (variable): A NULL–te page, which is used to construct the full the string in the **LocalBasePath** field.

LocalBasePathUnicode (variable): An opt construct the full path to the link item or **CommonPathSuffixUnicode** field. This **VolumeIDAndLocalBasePath** flag is s greater than or equal to 0x00000024.

CommonPathSuffixUnicode (variable): A to construct the full path to the link item **LocalBasePathUnicode** field. This field **LinkInfoHeaderSize** field is greater tha

2.3.1 VolumeID

The VolumeID structure specifies information **link** was created. This information is useful f

...

CountCharacters (2 bytes): A 16-bit, unsigned integer that specifies either the number of characters, defined by the system default code page, or the number of **Unicode** characters f in the **String** field. A value of zero specifies an empty string.

String (variable): An optional set of characters, defined by the system default code page, or a Unicode string with a length specified by the **CountCharacters** field. This string MUST NOT b NULL-terminated.

2.5 ExtraData

ExtraData refers to a set of structures that convey additional information about a **link target**. Th optional structures can be present in an **extra data section** that is appended to the basic Shell L Binary File Format.

The ExtraData structures conform to the following **ABNF** rules [RFC5234]:

```
EXTRA_DATA       = *EXTRA_DATA_BLOCK TERMINAL_BLOCK

EXTRA_DATA_BLOCK = CONSOLE_PROPS / CONSOLE_FE_PROPS / DARWIN_PROPS /
                   ENVIRONMENT_PROPS / ICON_ENVIRONMENT_PROPS /
                   KNOWN_FOLDER_PROPS / PROPERTY_STORE_PROPS /
                   SHIM_PROPS / SPECIAL_FOLDER_PROPS /
                   TRACKER_PROPS / VISTA_AND_ABOVE_IDLIST_PROPS
```

EXTRA_DATA: A structure consisting of zero or more property data blocks followed by a termina block.

EXTRA_DATA_BLOCK: A structure consisting of any one of the following property data blocks.

- **CONSOLE_PROPS**: A ConsoleDataBlock structure (section 2.5.1).
- **CONSOLE_FE_PROPS**: A ConsoleFEDataBlock structure (section 2.5.2).
- **DARWIN_PROPS**: A DarwinDataBlock structure (section 2.5.3).
- **ENVIRONMENT_PROPS**: An EnvironmentVariableDataBlock structure (section 2.5.4).
- **ICON_ENVIRONMENT_PROPS**: An IconEnvironmentDataBlock structure (section 2.5.5).
- **KNOWN_FOLDER_PROPS**: A KnownFolderDataBlock structure (section 2.5.6).
- **PROPERTY_STORE_PROPS**: A PropertyStoreDataBlock structure (section 2.5.7).
- **SHIM_PROPS**: A ShimDataBlock structure (section 2.5.8).
- **SPECIAL_FOLDER_PROPS**: A SpecialFolderDataBlock structure (section 2.5.9).
- **TRACKER_PROPS**: A TrackerDataBlock structure (section 2.5.10).
- **VISTA_AND_ABOVE_IDLIST_PROPS**: A VistaAndAboveIDListDataBlock structure (section 2.5.11).

TERMINAL_BLOCK A structure that indicates the end of the extra data section.

3 Structure Examples

3.1 Shortcut to a File

This section presents a sample of the Shell Link Binary File Format, consisting of a **shortcut** to a file with the path "C:\test\a.txt".

The following is the hexadecimal representation of the contents of the **shell link**.

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0000	4C	00	00	00	01	14	02	00	00	00	00	00	C0	00	00	00
0010	00	00	00	46	9B	00	08	00	20	00	00	00	D0	E9	EE	F2
0020	15	15	C9	01	D0	E9	EE	F2	15	15	C9	01	D0	E9	EE	F2
0030	15	15	C9	01	00	00	00	00	00	00	00	00	01	00	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	BD	00	14	00
0050	1F	50	E0	4F	D0	20	EA	3A	69	10	A2	D8	08	00	2B	30
0060	30	9D	19	00	2F	43	3A	5C	00	00	00	00	00	00	00	00
0070	00	00	00	00	00	00	00	00	00	00	00	46	00	31	00	00
0080	00	00	00	2C	39	69	A3	10	00	74	65	73	74	00	00	32
0090	00	07	00	04	00	EF	BE	2C	39	65	A3	2C	39	69	A3	26
00A0	00	00	00	03	1E	00	00	00	00	F5	1E	00	00	00	00	00
00B0	00	00	00	00	00	74	00	65	00	73	00	74	00	00	00	14
00C0	00	48	00	32	00	00	00	00	00	2C	39	69	A3	20	00	61
00D0	2E	74	78	74	00	34	00	07	00	04	00	EF	BE	2C	39	69
00E0	A3	2C	39	69	A3	26	00	00	00	2D	6E	00	00	00	00	96
00F0	01	00	00	00	00	00	00	00	00	00	00	61	00	2E	00	74
0100	00	78	00	74	00	00	00	14	00	00	00	3C	00	00	00	1C
0110	00	00	00	01	00	00	00	1C	00	00	00	2D	00	00	00	00
0120	00	00	00	3B	00	00	00	11	00	00	00	03	00	00	00	81
0130	8A	7A	30	10	00	00	00	00	43	3A	5C	74	65	73	74	5C
0140	61	2E	74	78	74	00	00	07	00	2E	00	5C	00	61	00	2E
0150	00	74	00	78	00	74	00	07	00	43	00	3A	00	5C	00	74
0160	00	65	00	73	00	74	00	60	00	00	00	03	00	00	A0	58

BlockSize (4 bytes): A 32-bit, unsigned integer that specifies the size structure. This value MUST be 0x00000060.

BlockSignature (4 bytes): A 32-bit, unsigned integer that specifies th TrackerDataBlock **extra data section**. This value MUST be 0xA000

Length (4 bytes): A 32-bit, unsigned integer that specifies the size of TrackerDataBlock structure, including this **Length** field. This value M

This value MUST be 0x0

aracter string, as define the machine where the

epresentation ([MS-DT king service, as describ

cket representation that

cifies an alternate **IDLi** atforms that support it.

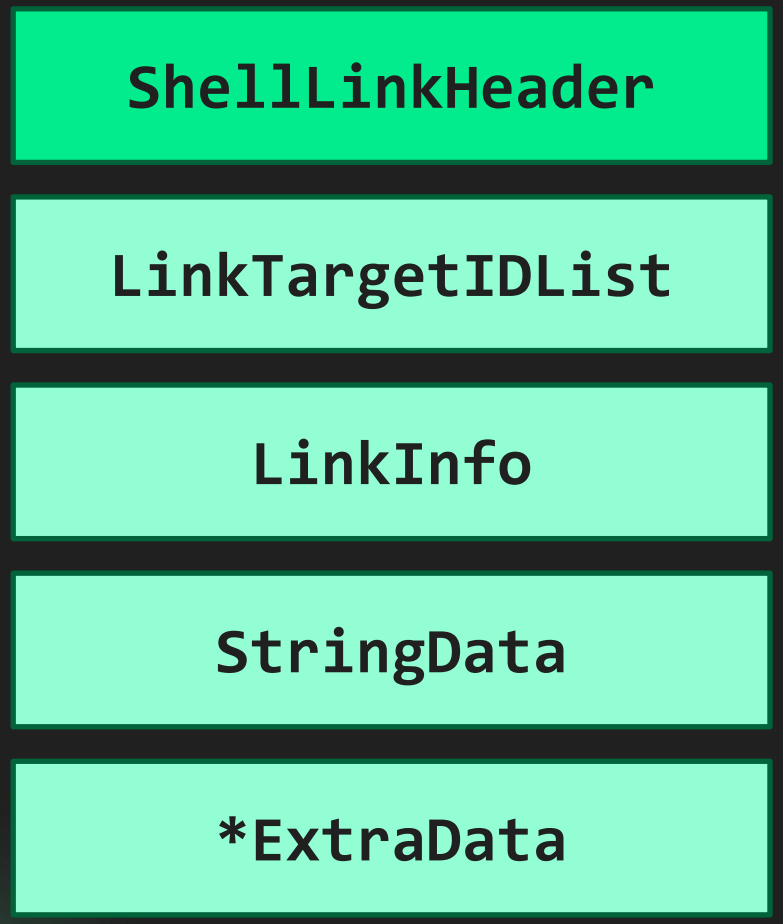
ize

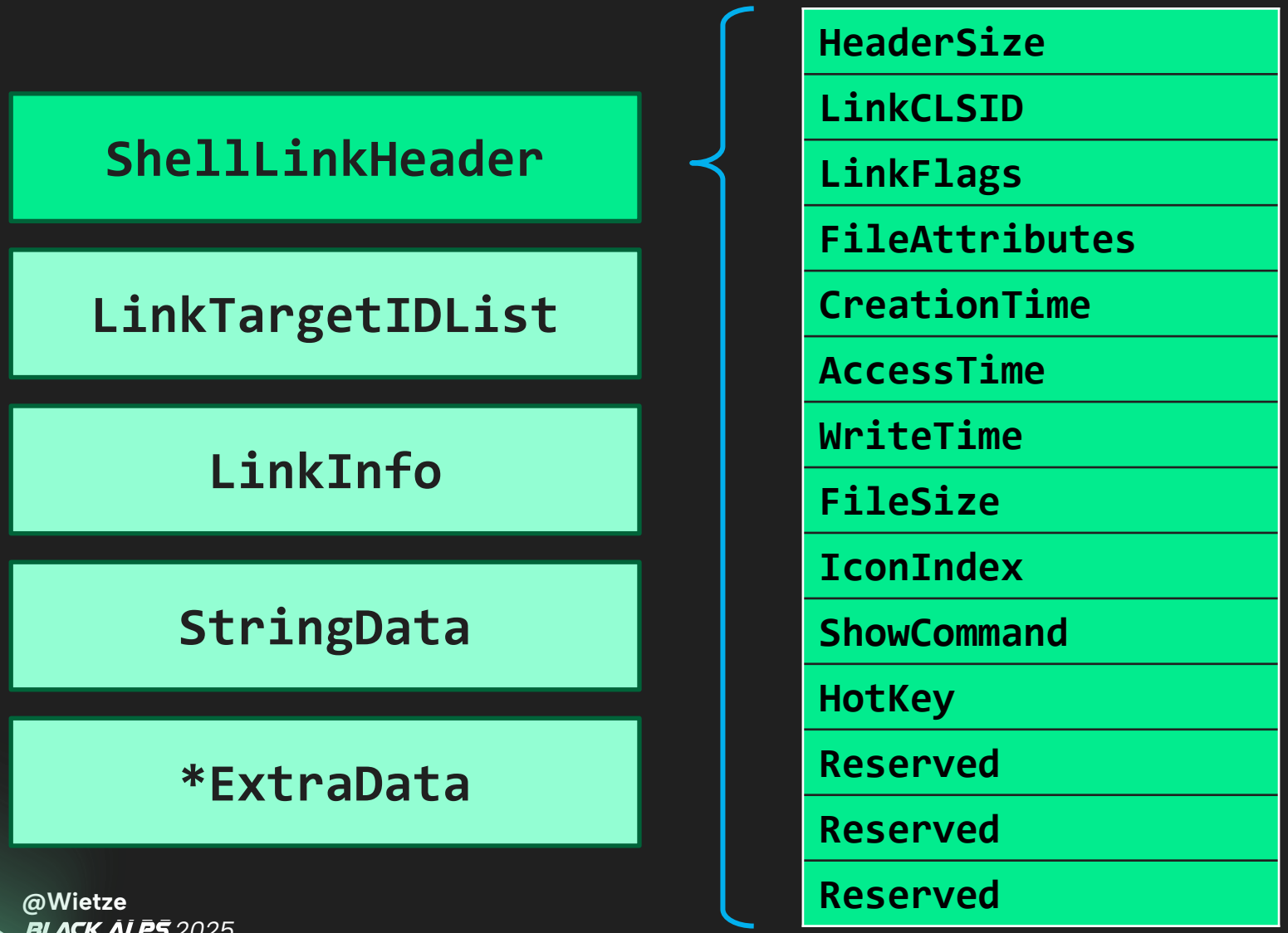
pecifies the size MUST be greate

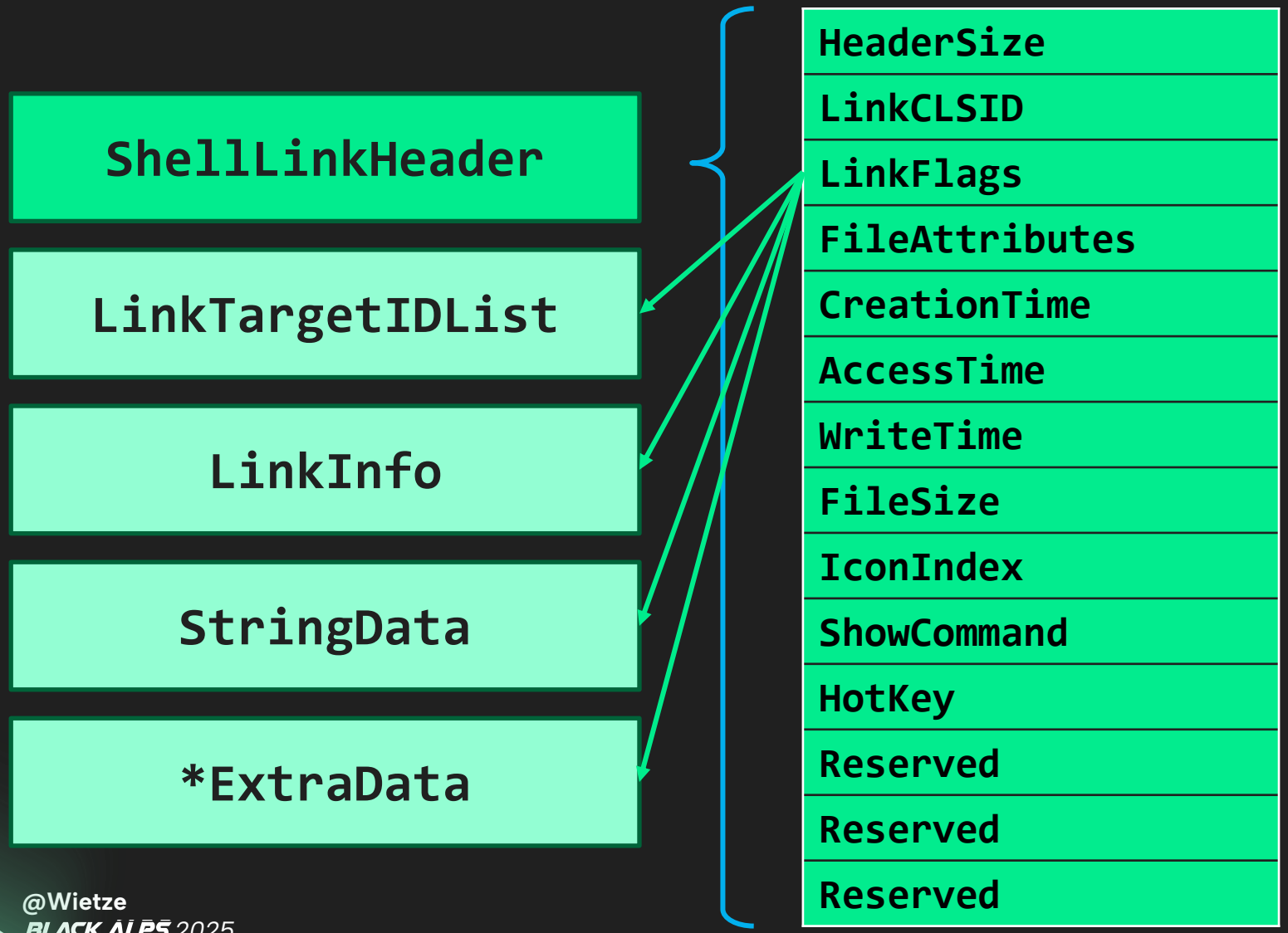
that specifies th This value MUS

4 Security

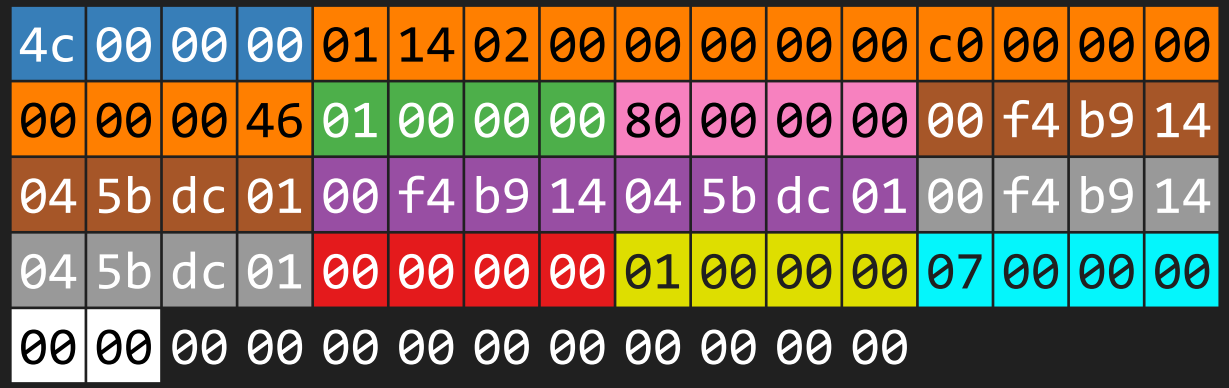
None.


ShellLinkHeader
LinkTargetIDList
LinkInfo
StringData
*ExtraData


ShellLinkHeader
LinkTargetIDList
LinkInfo
StringData
*ExtraData
HeaderSize
LinkCLSID
LinkFlags
FileAttributes
CreationTime
AccessTime
WriteTime
FileSize
IconIndex
ShowCommand
HotKey
Reserved
Reserved
Reserved


ShellLinkHeader
LinkTargetIDList
LinkInfo
StringData
*ExtraData
HeaderSize
LinkCLSID
LinkFlags
FileAttributes
CreationTime
AccessTime
WriteTime
FileSize
IconIndex
ShowCommand
HotKey
Reserved
Reserved
Reserved

Example: header



HeaderSize	0x4c (=76)
LinkCLSID	00021401-0000-0000-C000-000000000046
LinkFlags	0x01 (HasLinkTargetIDList) 25 flags ≈ 33.5M possibilities
FileAttributes	0x80 (FILE_ATTRIBUTE_NORMAL) 13 flags = 8,192 possibilities
CreationTime	2025-11-21 16:30:00 UTC
AccessTime	2025-11-21 16:30:00 UTC
WriteTime	2025-11-21 16:30:00 UTC
FileSize	0x00 (=0)
IconIndex	0x01 (=1)
ShowCommand	0x07 (SW_SHOWMINNOACTIVE)
HotKey	0x00 (=0)

Investigate.lnk

Getting into the attacker's mindset

.LNK format is archaic and **full of surprises**

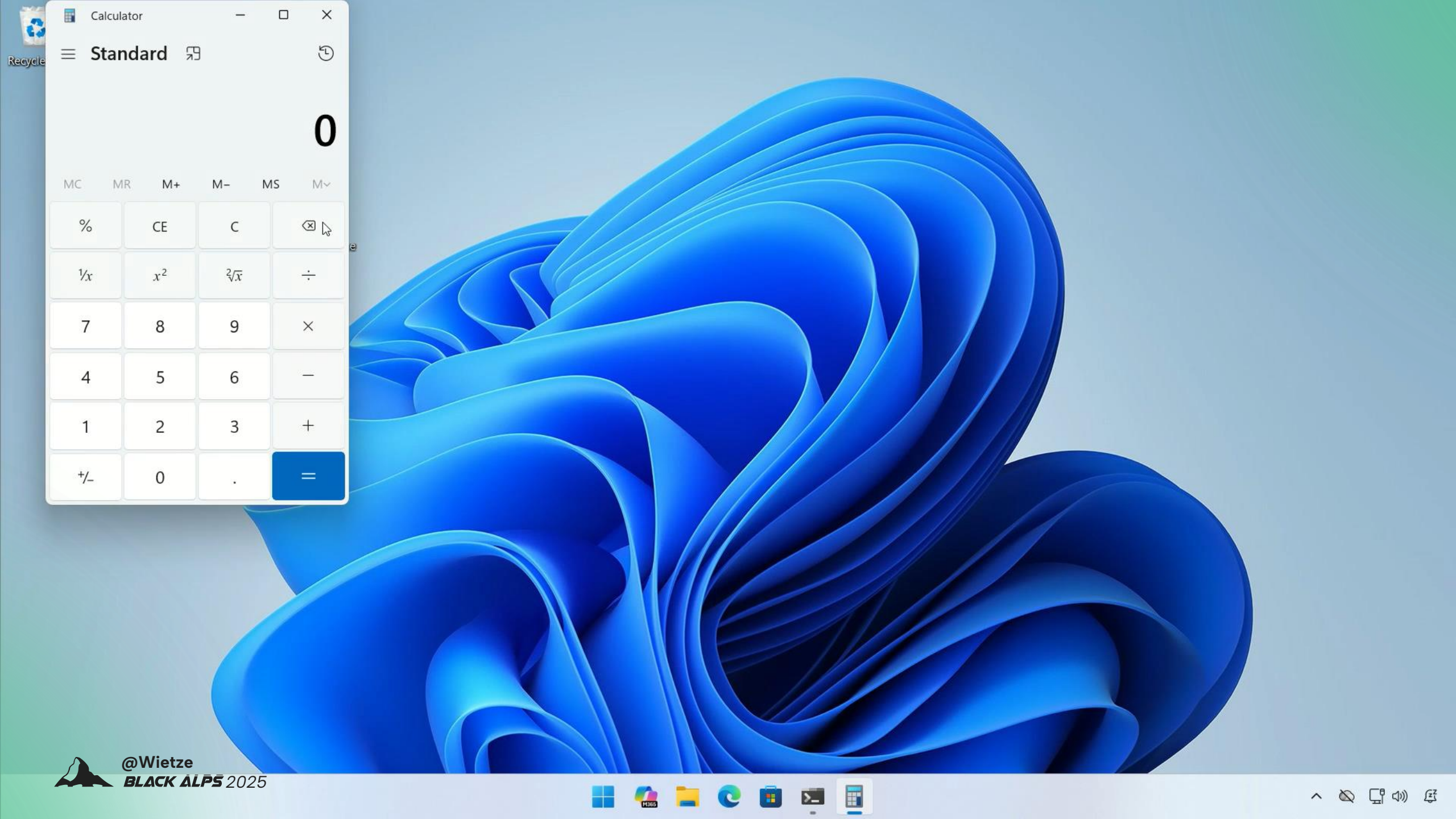
Goal

We want such surprises that help our goals:

- Creating trust
- Deception
- Stay under the radar

Means

- Unlikely combination of flags
- Unusual values
- Overflowing fields
- Mismatch

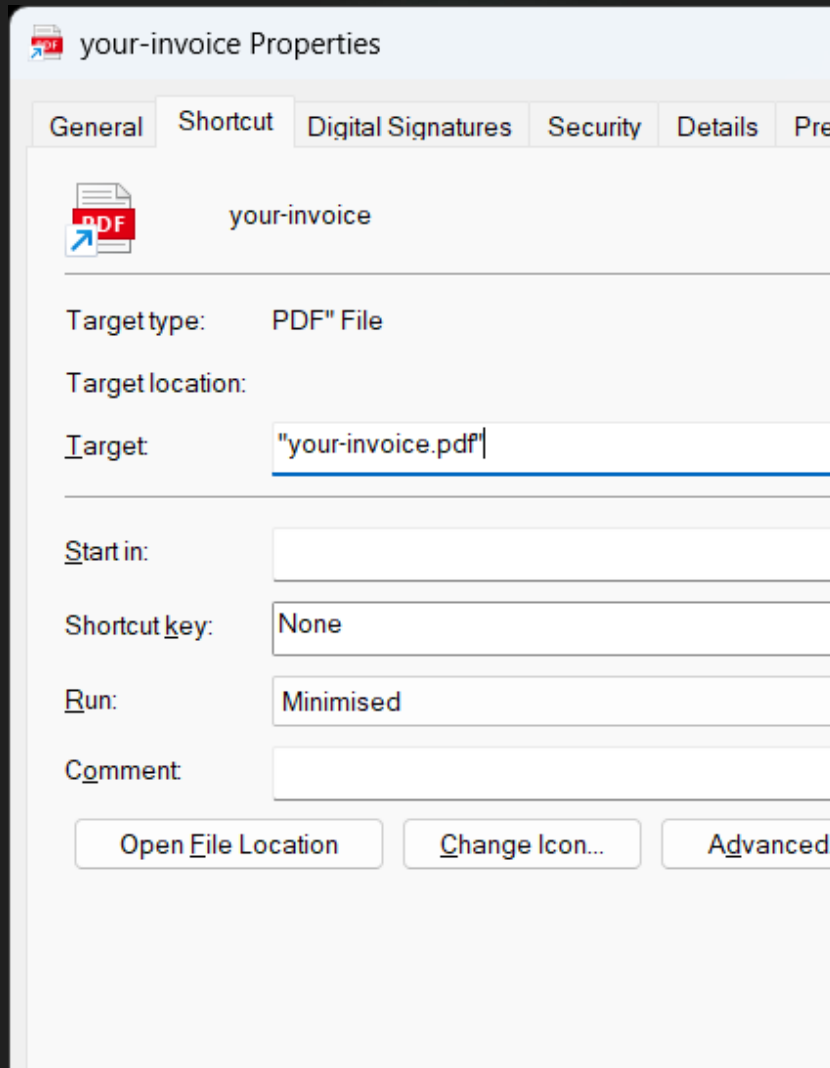

Recycle
Calculator
Standard
0
MC MR M+ M− MS M
% CE C
1/x x² ²√x ÷
7 8 9 ×
4 5 6 −
1 2 3 +
+/− 0 . =
@Wietze
BLACK ALPS 2025

Example 1: Spoofed program

Displayed:
`"your-invoice.pdf"`

Executed:
`c:\Windows\System32\calc.exe`

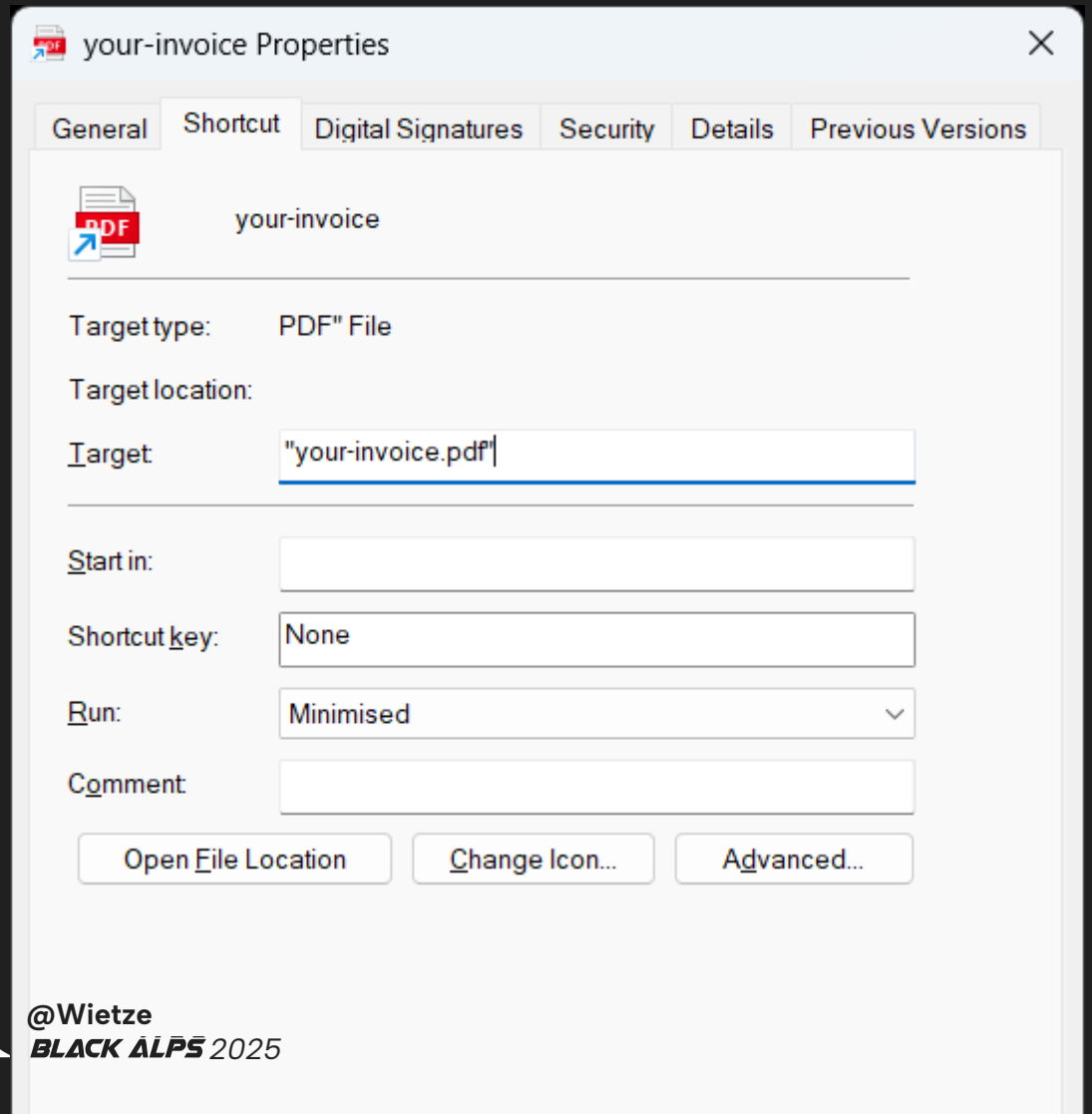
🚀 **Surprise**: main target can be spoofed
🚧 **Downside**: arguments still visible
Optionally "obfuscate" with Right-to-Left (RTL) character

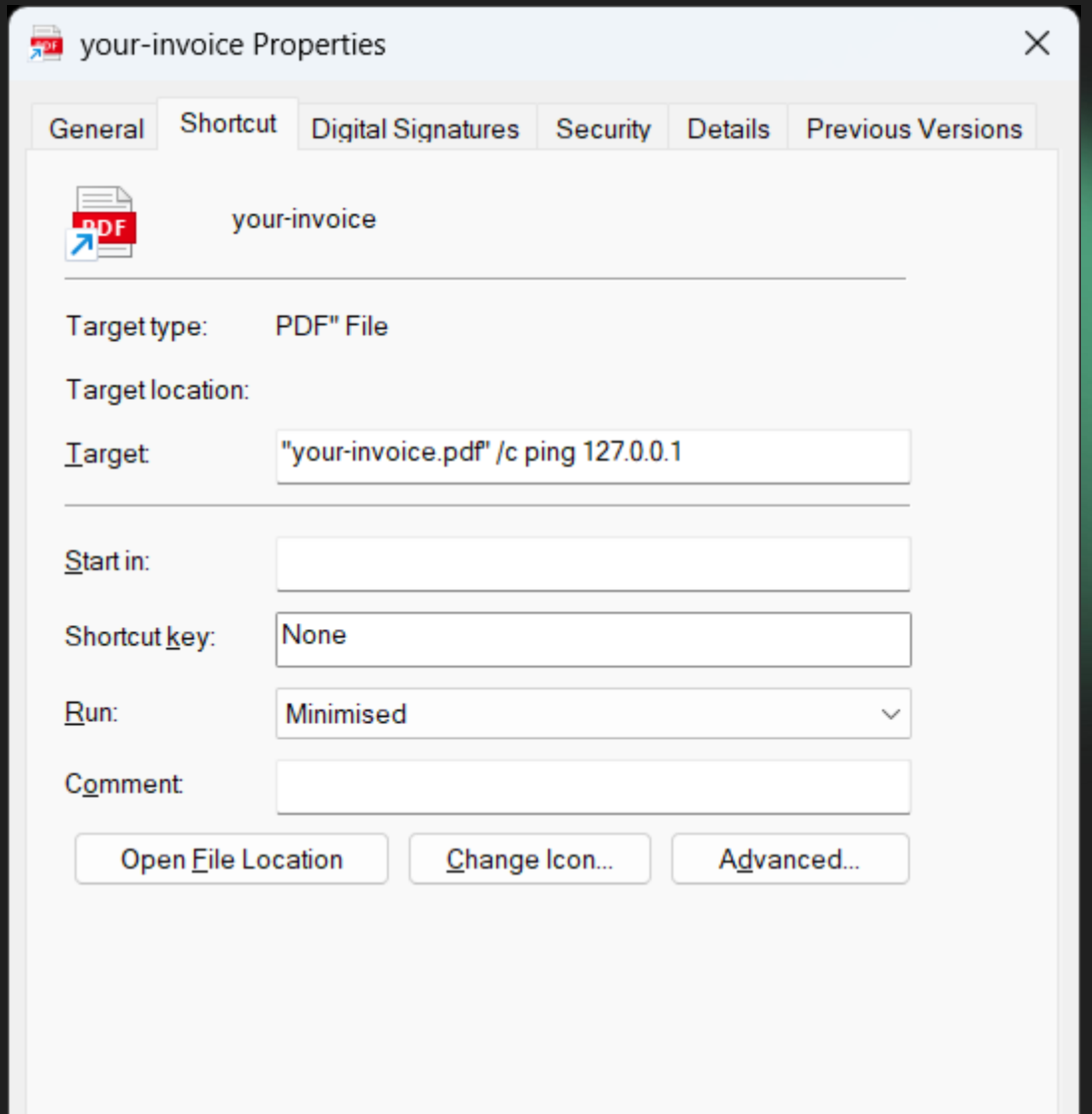


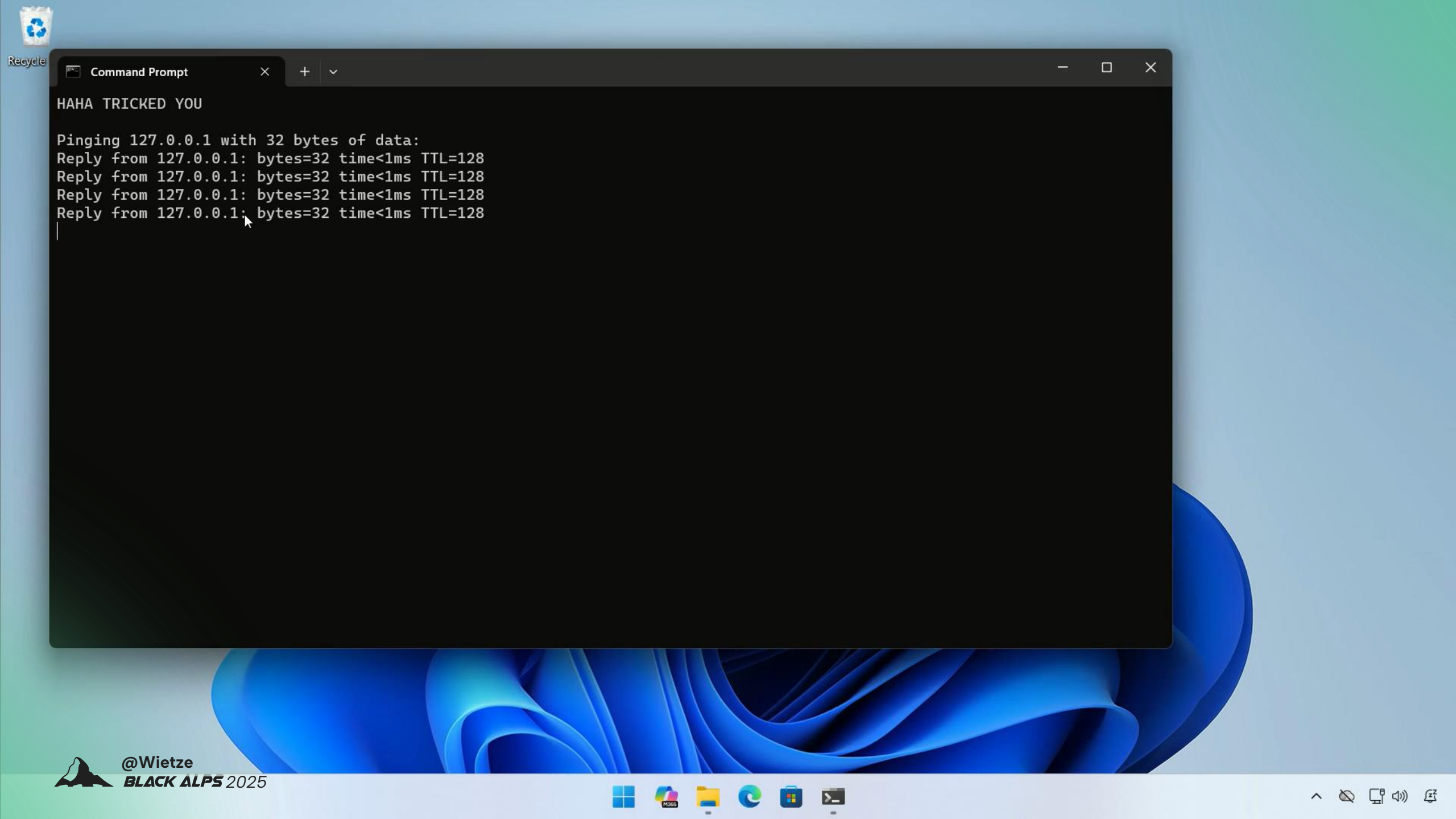
Example 1: Spoofed program

ShellLinkHeader	LinkFlags 0x00000221 HasLinkTargetIDList, HasExpString
LinkTargetIDList	C:\windows\system32\calc.exe
LinkInfo	
StringData	
*ExtraData	"your-invoice.pdf"

Example 1: Spoofed program






Recycle
Command Prompt
HAHA TRICKED YOU
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
@Wietze
BLACK ALPS 2025

Example 2: Hidden arguments

Displayed:

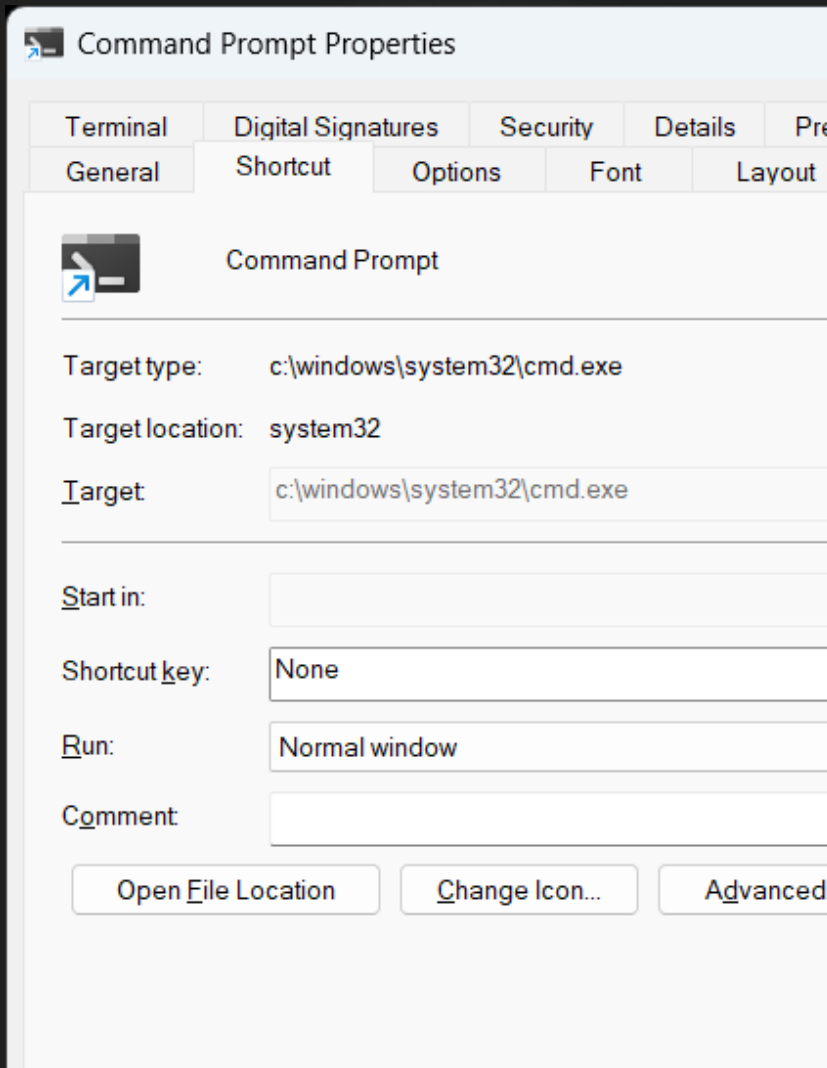
```
c:\Windows\System32\cmd.exe
```

Executed:

```
c:\Windows\System32\cmd.exe /c echo
 HAHA TRICKED YOU; ping 127.0.0.1
```

🚀 **Surprise**: arguments can be hidden, path can be disabled

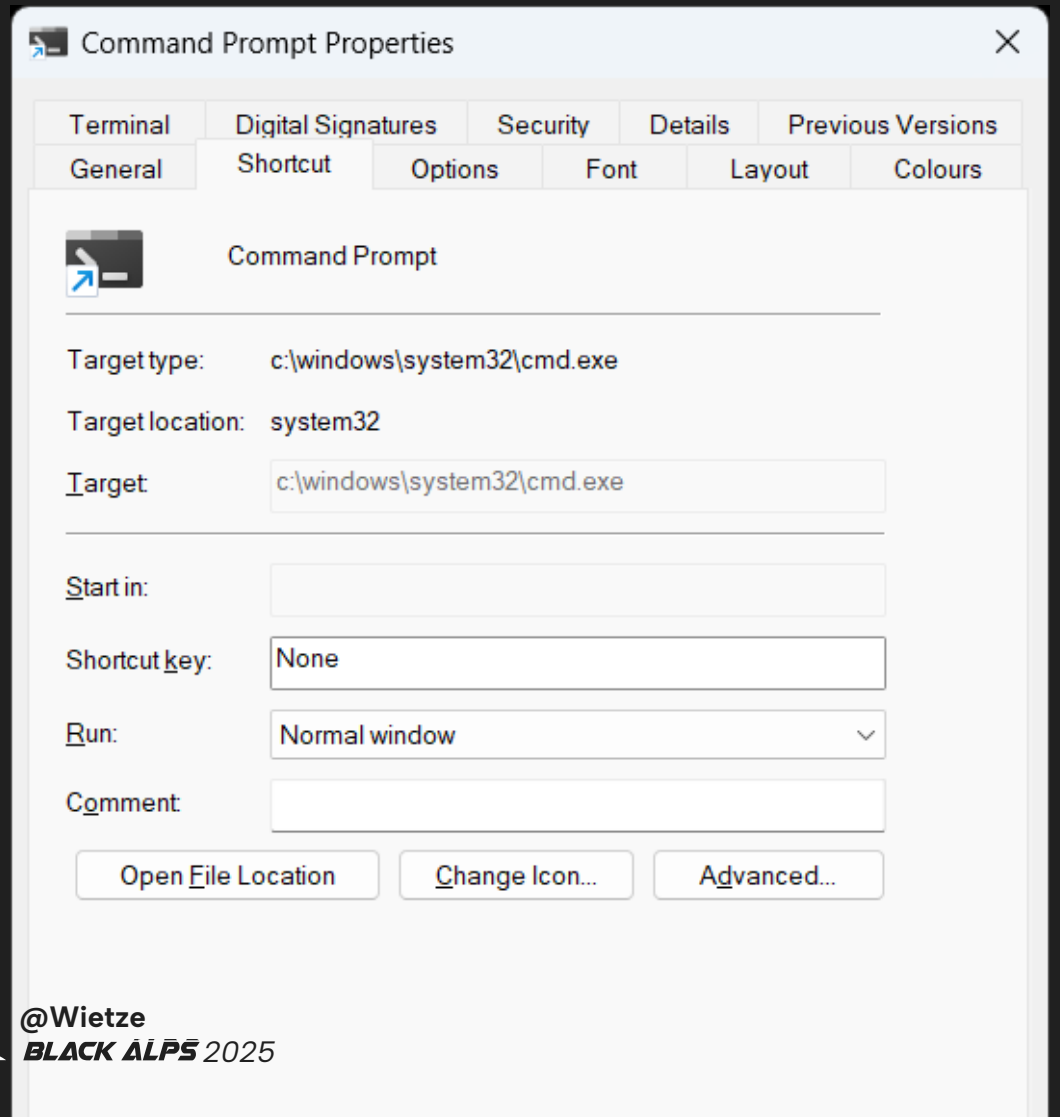
🚧 **Downside**: main program still visible

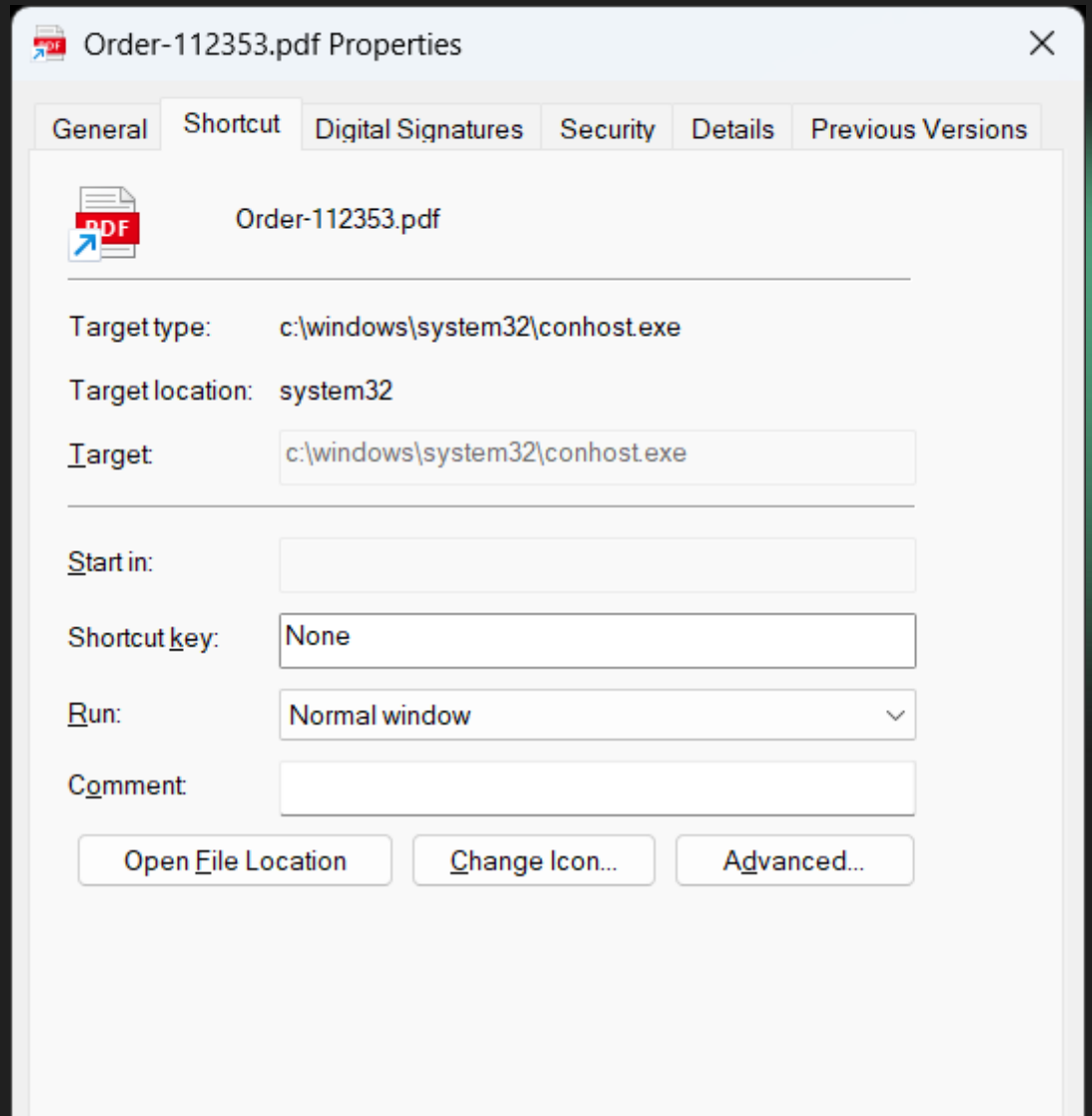


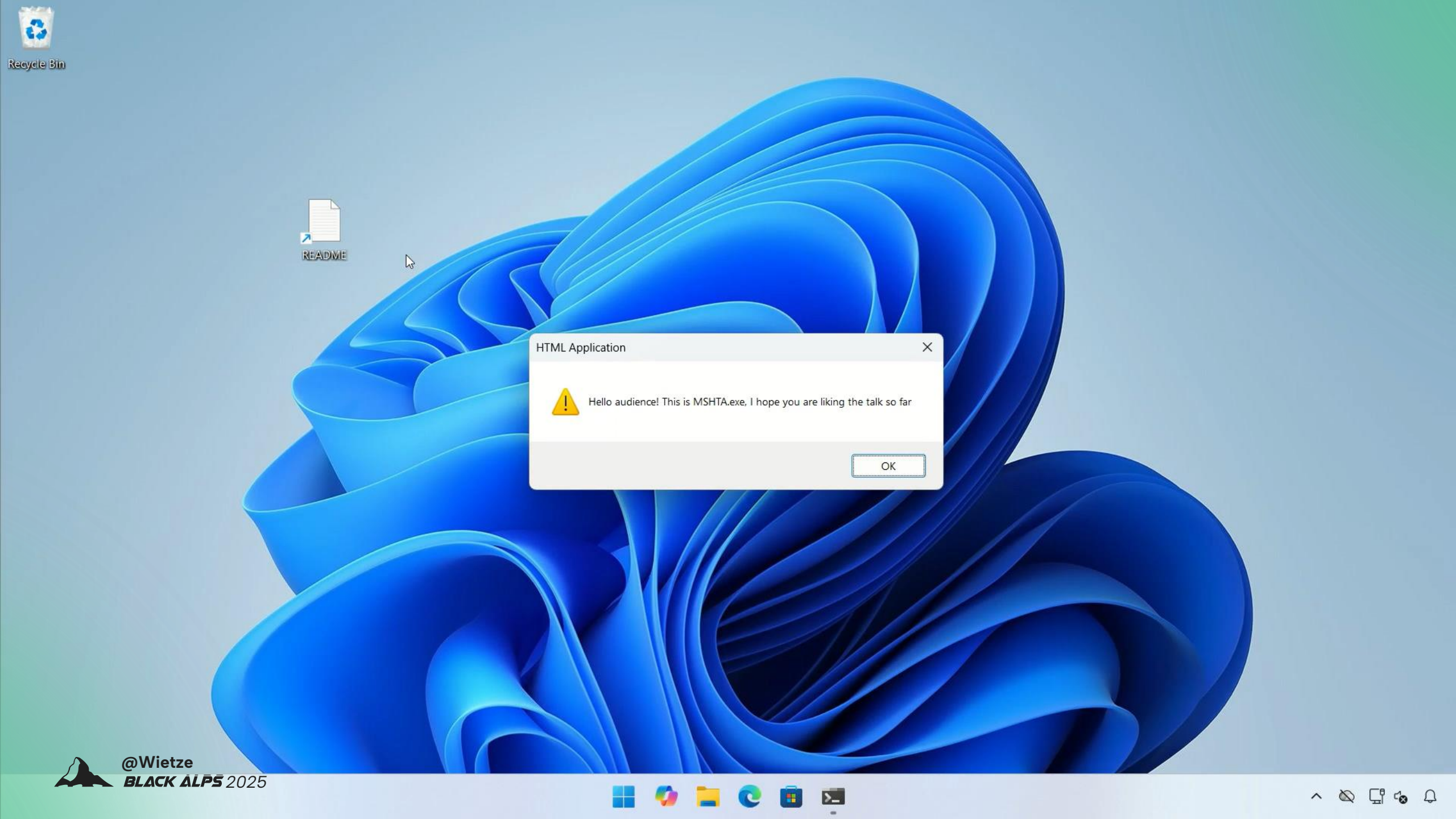
Example 2: Hidden arguments

ShellLinkHeader	LinkFlags 0x00000221: HasLinkTargetIDList, HasArguments, HasExpString
LinkTargetIDList	C:\windows\system32\cmd.exe
LinkInfo	
StringData	/c HAHA TRICKED YOU; ping 127.0.0.1
*ExtraData	(null)

Example 2: Hidden arguments






Recycle Bin
README
HTML Application
Hello audience! This is MSHTA.exe, I hope you are liking the talk so far
OK
@Wietze
BLACK ALPS 2025

Example 3: Spoofed program

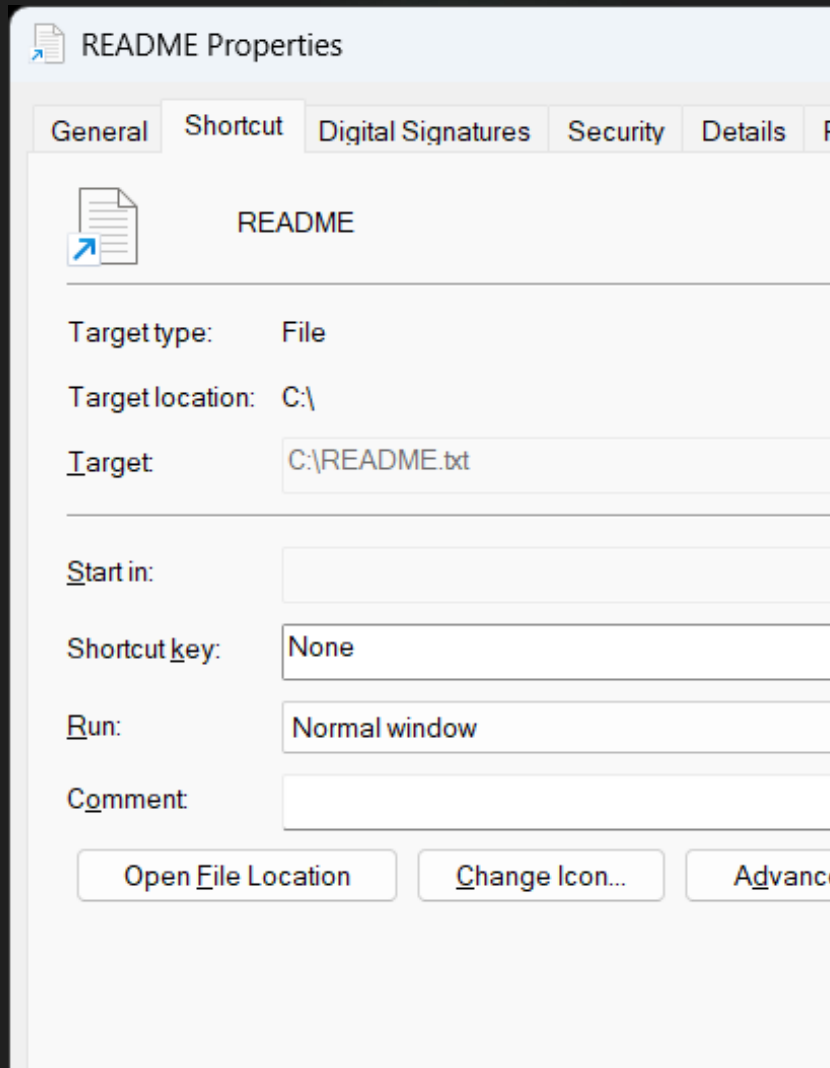
Displayed:

```
c:\README.txt
```

Executed:

```
c:\Windows\System32\mshta.exe
javascript:alert("Hello audience! …")
```

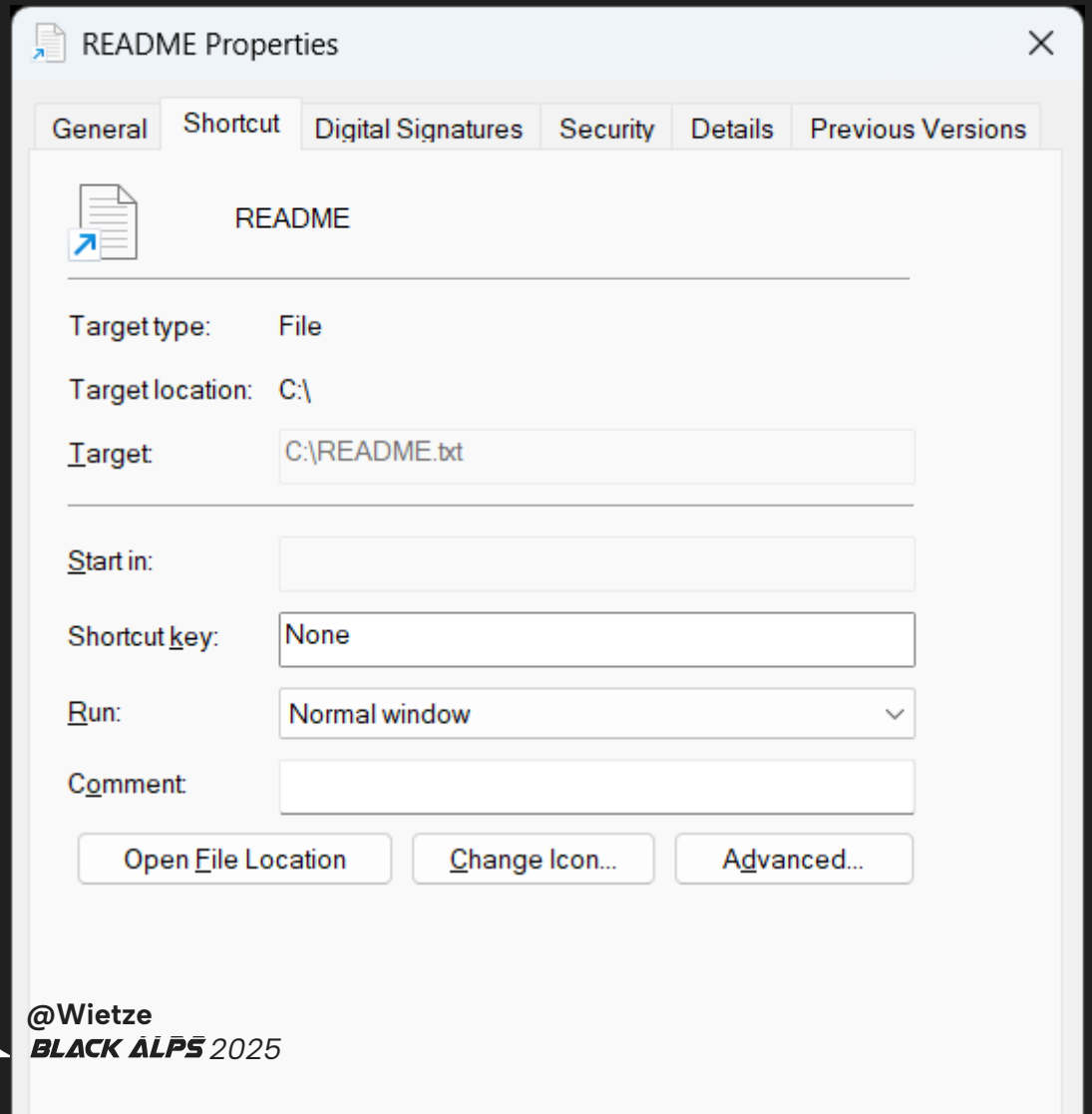
🚀 **Surprise**: entire path can be spoofed
↩️ **Side effect**: .LNK is “updated”
🚧 **Downside**: double click required

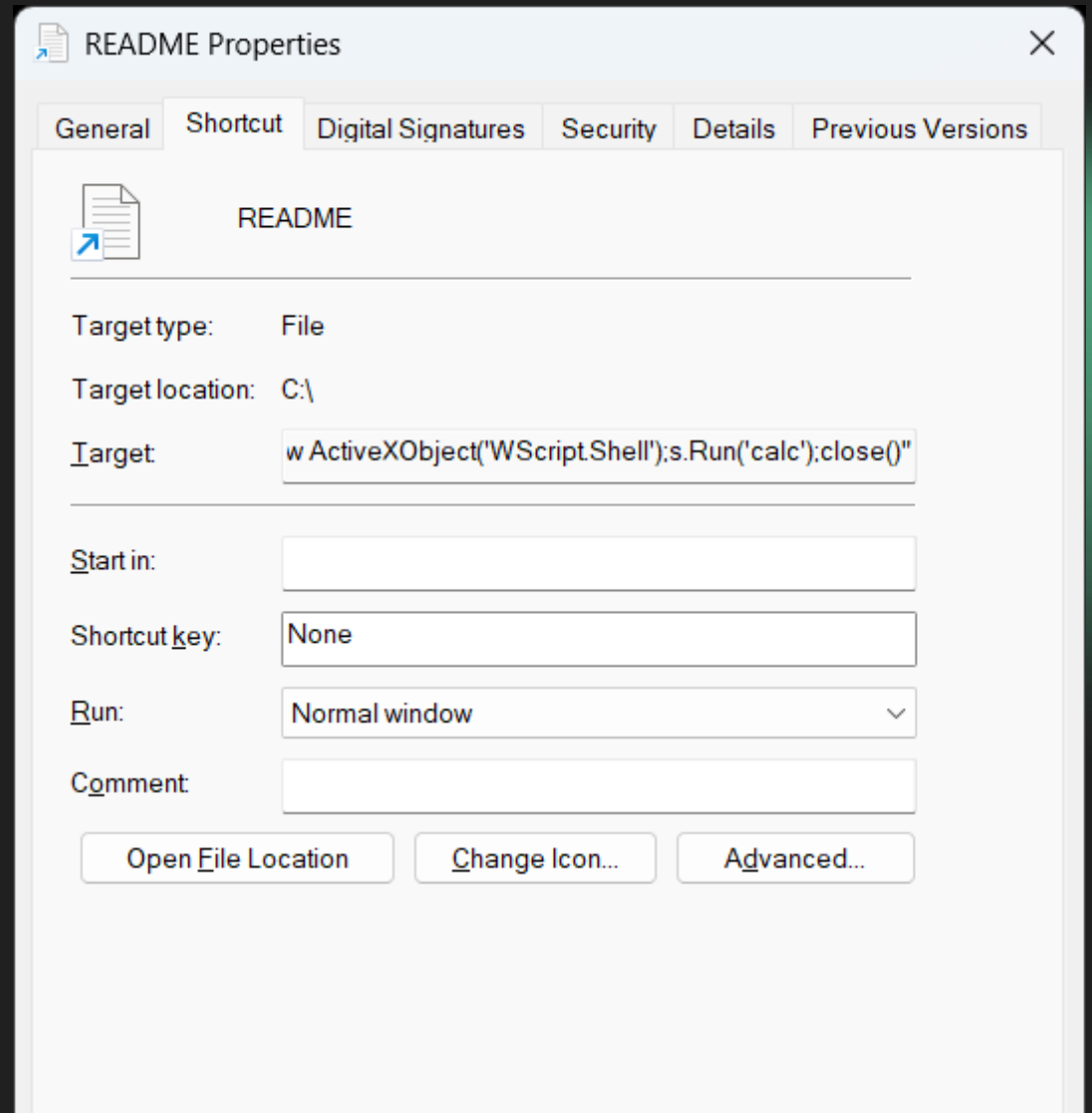


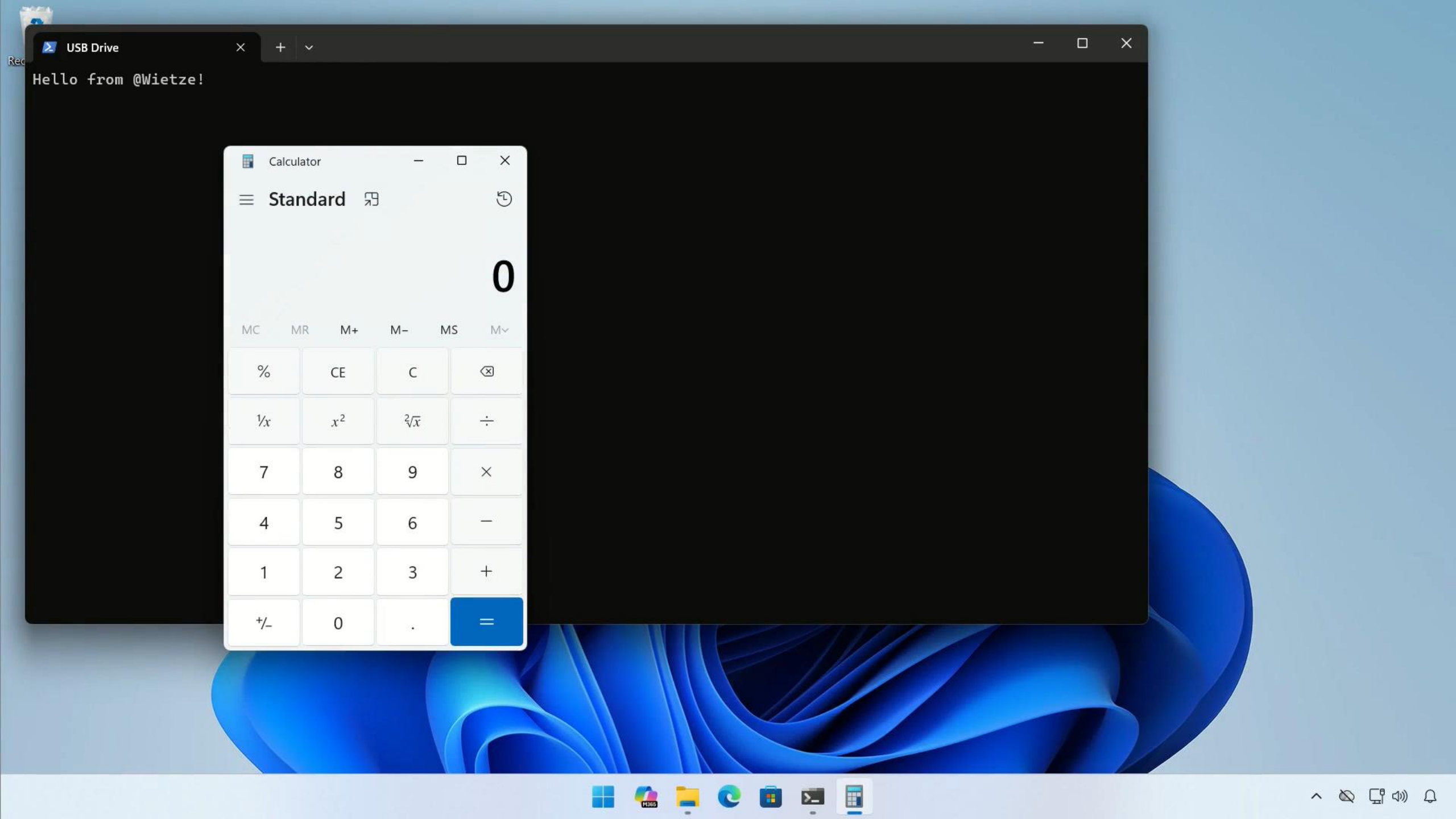
Example 3: Spoofed program

ShellLinkHeader	LinkFlags 0x00000221: HasLinkTargetIDList, HasLinkInfo, HasArguments, HasExpString
LinkTargetIDList	X:\
LinkInfo	C:\Windows\System32\mshta.exe
StringData	javascript:alert("Hello conference!"); …
*ExtraData	C:\README.txt

Example 3: Spoofed program





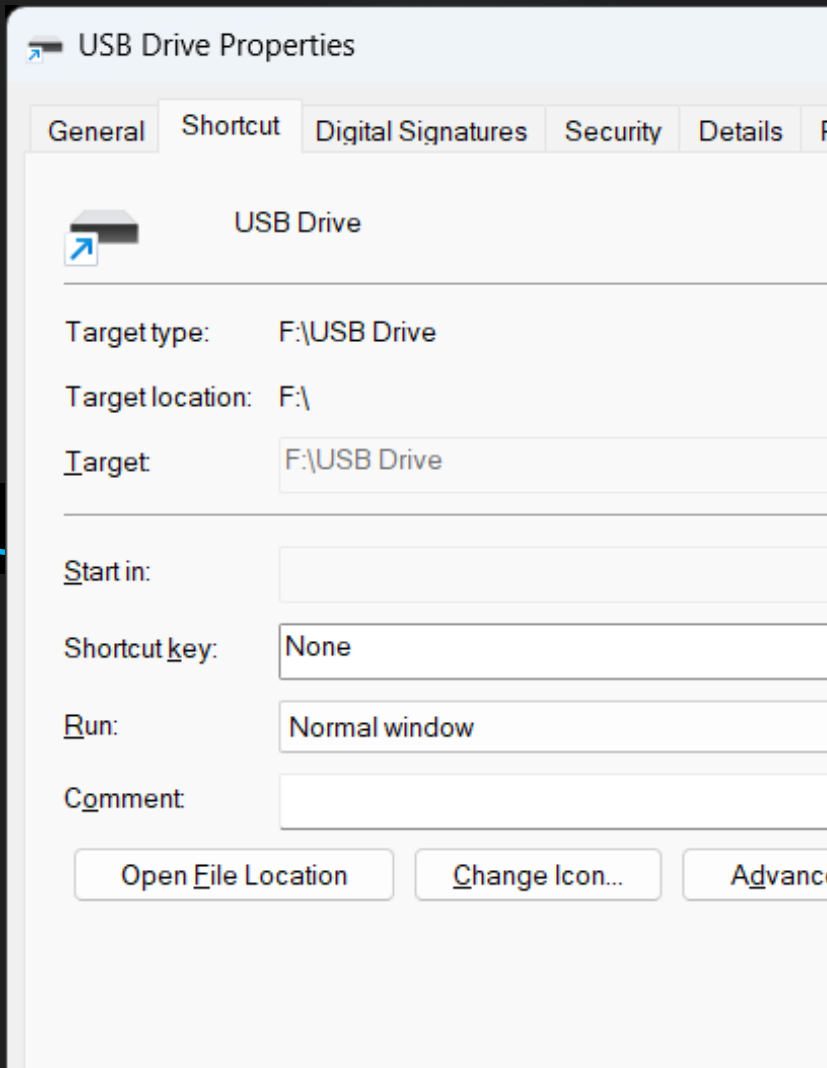

USB Drive
Hello from @Wietze!
Calculator
Standard
0
MC MR M+ M− MS M
% CE C
1/x x² ²√x ÷
7 8 9 ×
4 5 6 −
1 2 3 +
+/− 0 . =

Example 4: Spoof & Disable

Displayed:
`F:\USB Drive`

Executed:
`powershell.exe /ec SABlAGwAbABvACAAZgByAG8AbQAgAEAAVwBpAGU`

🚀 **Surprise**: everything can be spoofed
↩️ **Side effect**: .LNK is “updated”
🚧 **Downside**: none?

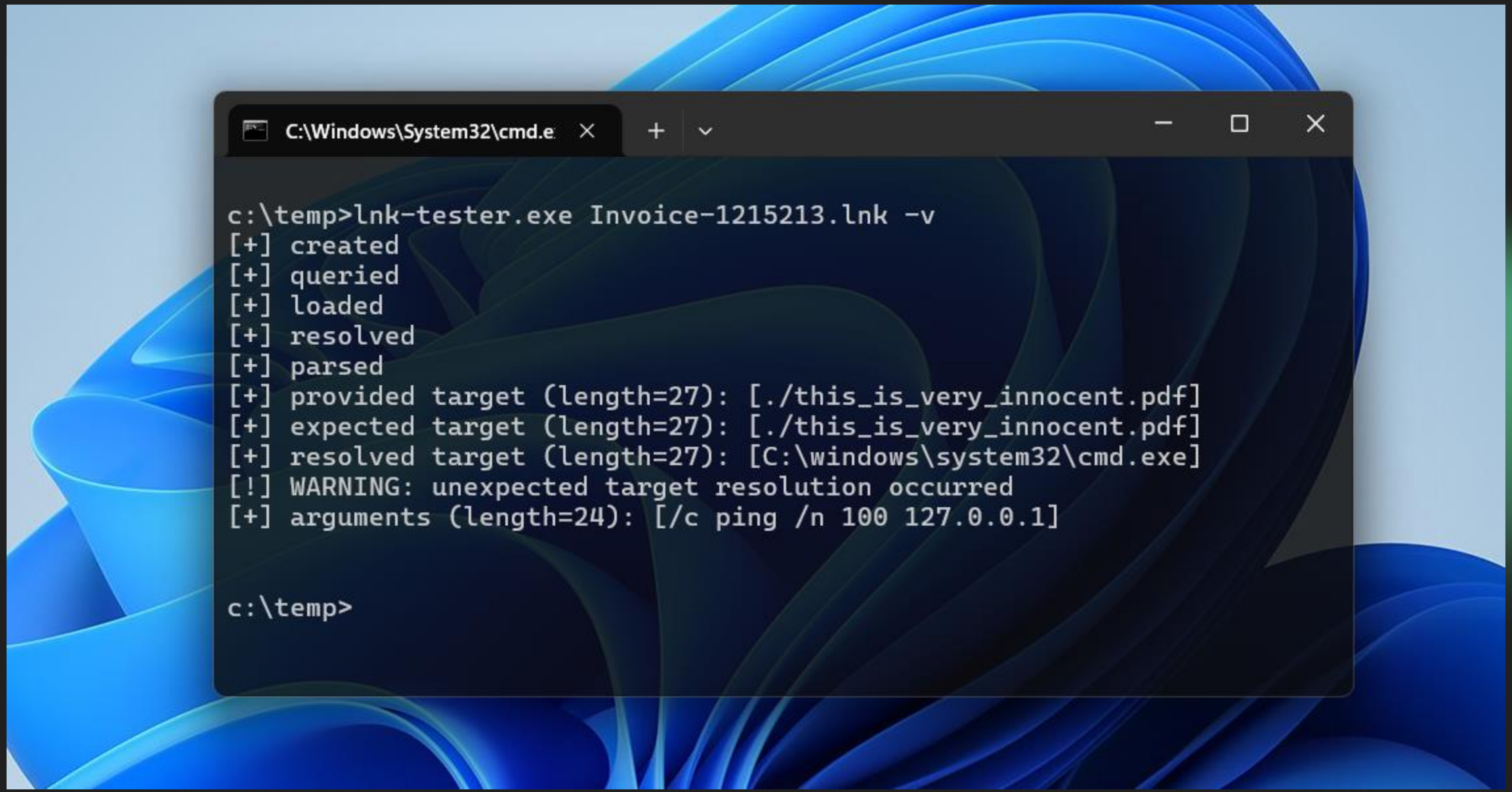


Example 4: Spoof & Disable

ShellLinkHeader	**LinkFlags** **0x00000221**: HasLinkTargetIDList, HasArguments, HasExpString
LinkTargetIDList	C:\README.txt
LinkInfo	
StringData	/c ping 127.0.0.1
*ExtraData	%WINDIR%\System32\cmd.exe | (null)

wietze/lnk-it-up

LNK generator *(Python)*
Testing framework *(C)*


C:\Windows\System32\cmd.e
c:\temp>lnk-tester.exe Invoice-1215213.lnk -v
[+] created
[+] queried
[+] loaded
[+] resolved
[+] parsed
[+] provided target (length=27): [./this_is_very_innocent.pdf]
[+] expected target (length=27): [./this_is_very_innocent.pdf]
[+] resolved target (length=27): [C:\windows\system32\cmd.exe]
[!] WARNING: unexpected target resolution occurred
[+] arguments (length=24): [/c ping /n 100 127.0.0.1]
c:\temp>

Implications: Defence

- Don’t take any .LNK at face value
- Beware of **frustrated forensics**
- **.LNK analysis tools** can be **fooled/broken**
  They are not as resilient as Windows itself

- Opportunities to detect “suspicious” .LNK files
  - `lnk-it-up/lnk-tester`
  - YARA

Conclusions

.LNK files **remain** a **popular** target in the initial access toolset

Windows' resilient handling of .LNK files gives **scope for abuse**

Malicious .LNK use **will continue in the foreseeable future**

@wietze
@wietzebeukema.nl
@wietze@infosec.exchange

/in/wjbbeukema

Thank you