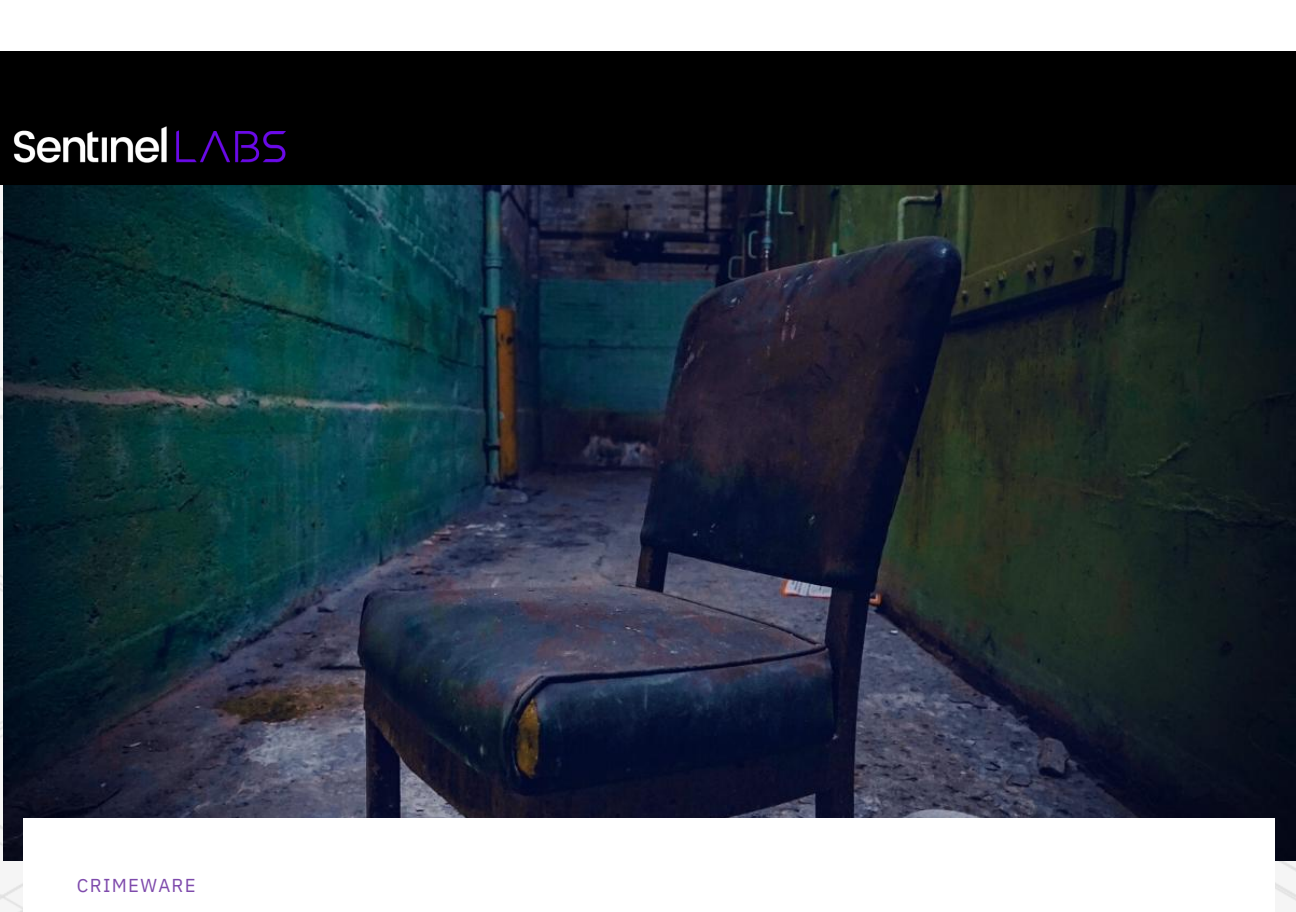


CRIMEWARE

Who Needs Macros? | Threat Actors Pivot to Abusing Explorer and Other LOLBins via Windows Shortcuts

ALEKSANDAR MILENKOSKI / AUGUST 4, 2022

By Aleksandar Milenkoski & Jim Walter

Executive Summary

- Windows Explorer (`explorer.exe`) is the top initial living-off-the-land binary (LOLbin) in the chain of LOLbins that threat actors abuse to execute malware through malicious Windows shortcuts (LNK files).
- Our mass-analysis of 27510 representative malicious `LNK` files from VirusTotal revealed Windows Explorer at the top of the list (with 87.2% prevalence), followed by `powershell.exe` (7.3%), `wscript.exe` (4.4%), and `rundll32.exe` (0.5%). `LNK` files are currently immensely popular among threat actors for malware deployment and persistence.
- We have observed intensive advertising of new versions of the mLNK and QuantumBuilder tools for building malicious `LNK` files in the cybercrime web space since May 2022, with many new features for evasion and stealth.
- The mLNK and QuantumBuilder tools enable threat actors to build malicious `LNK` files in a configurable and convenient manner. Given the popularity of `LNK` files among threat actors, there is an increasing demand for such tools on the cybercrime market.
- The actors behind the QuantumBuilder tool for building malicious `LNK` files advertise the tool and the value of `LNK` files to threat actors by claiming that Office macros "*are for the most part dead*" [as a medium for deploying malware], referring to Microsoft's recent decision to disable by default Office macro execution in the context of documents that originate from untrusted sources.

Overview

This article discusses Windows shortcuts (LNK files) as a medium to deploy malware and/or establish persistence. In the initial stages of an attack, threat actors are gravitating more towards the use of malicious shortcuts that deploy malware by executing code in the context of so-called living-off-the-land binaries (LOLbins) – legitimate executables that are readily available on Windows systems, such as `powershell.exe` or `mshta.exe` – to bypass detection. Threat actors conveniently build malicious `LNK` files with Windows system capabilities or tools specifically designed for that purpose, and then distribute the files to victims, usually through phishing emails.

Because of these advantages, threat actors are widely abusing shortcuts. Since Microsoft's announcement that Office applications will by default disable the execution of Office macros in the context of documents that originate from untrusted sources, there has been a significant uptick in malicious actors using alternative mediums for deploying malware, such as malicious Windows Apps and shortcuts (LNK files). We covered malicious Windows Apps in a previous article. In this article, we focus on malicious shortcuts and provide:

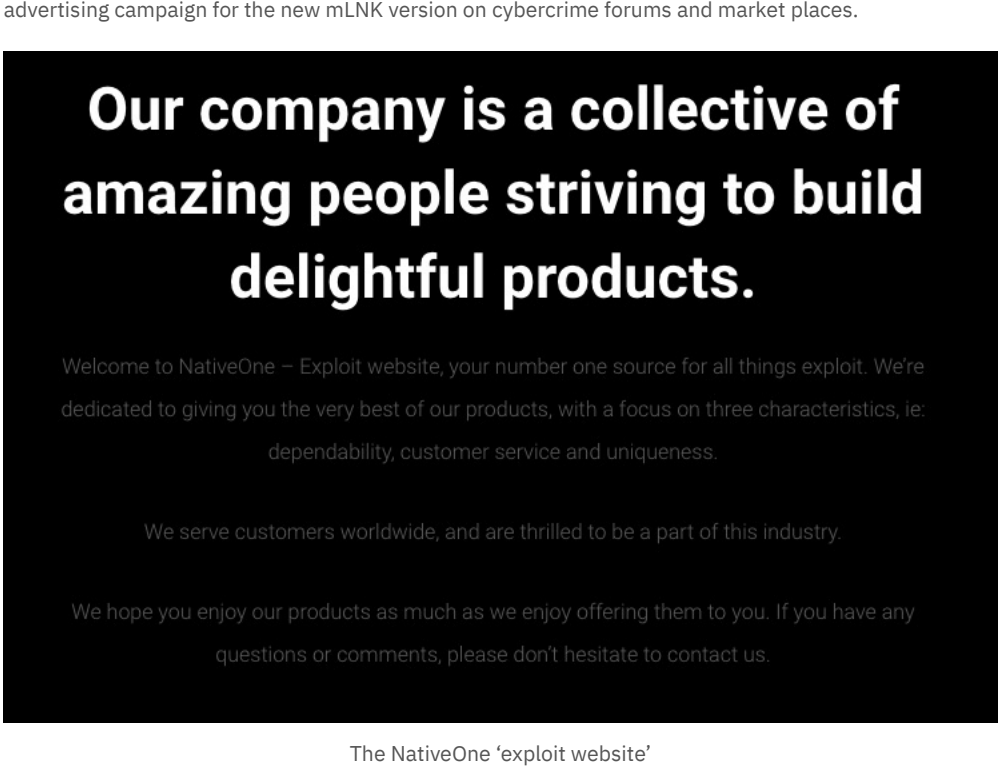
- Insights about execution chains that originate from malicious shortcuts. We base our insights on an analysis of 27510 malicious `LNK` file samples from VirusTotal that are representative of the current malicious shortcut landscape.
- An overview of active widespread attack campaigns that involve malicious shortcuts and of the dynamics of the cybercrime market for tools that build malicious `LNK` files.
- A summarizing overview of the system activities that take place when a user executes a malicious shortcut. This enables a better and generic understanding of what occurs on a system when a user falls prey to an attack that involves a malicious `LNK` file.

Current Developments in the Malicious Shortcut Threat Scene

Given the popularity of `LNK` files among threat actors, the dynamics of the cybercrime market for tools has quickly adjusted to serve the demand for tools that build malicious `LNK` files in a configurable and convenient manner. We spotlight in this section the mLNK and QuantumBuilder tools for building malicious `LNK` files. We observed that these tools have recently received updates and are currently being intensively advertised in the cybercrime web space.

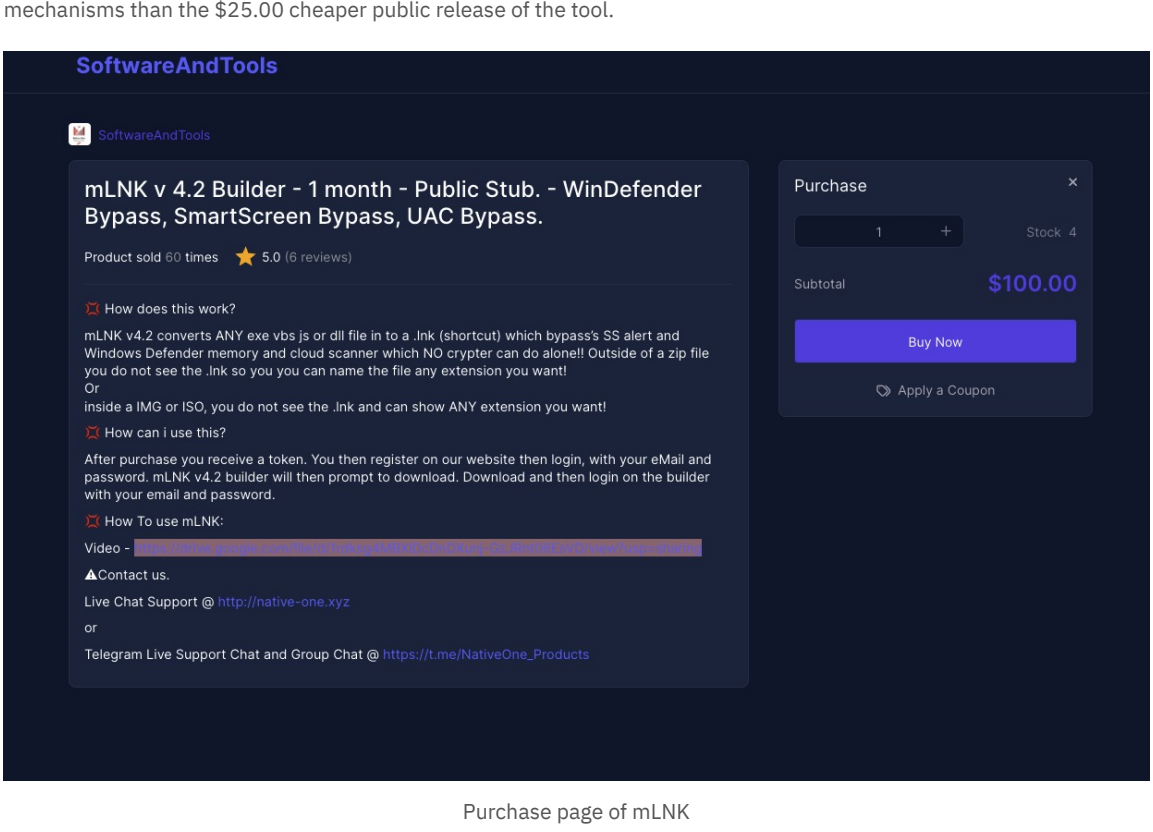
mLNK

The mLNK tool – released by NativeOne, a tool vendor on the cybercrime scene – is known for its configurability and ease of use. NativeOne released the newest version of the tool, version 4.2, in June 2022. We observed an intensive advertising campaign for the new mLNK version on cybercrime forums and market places.

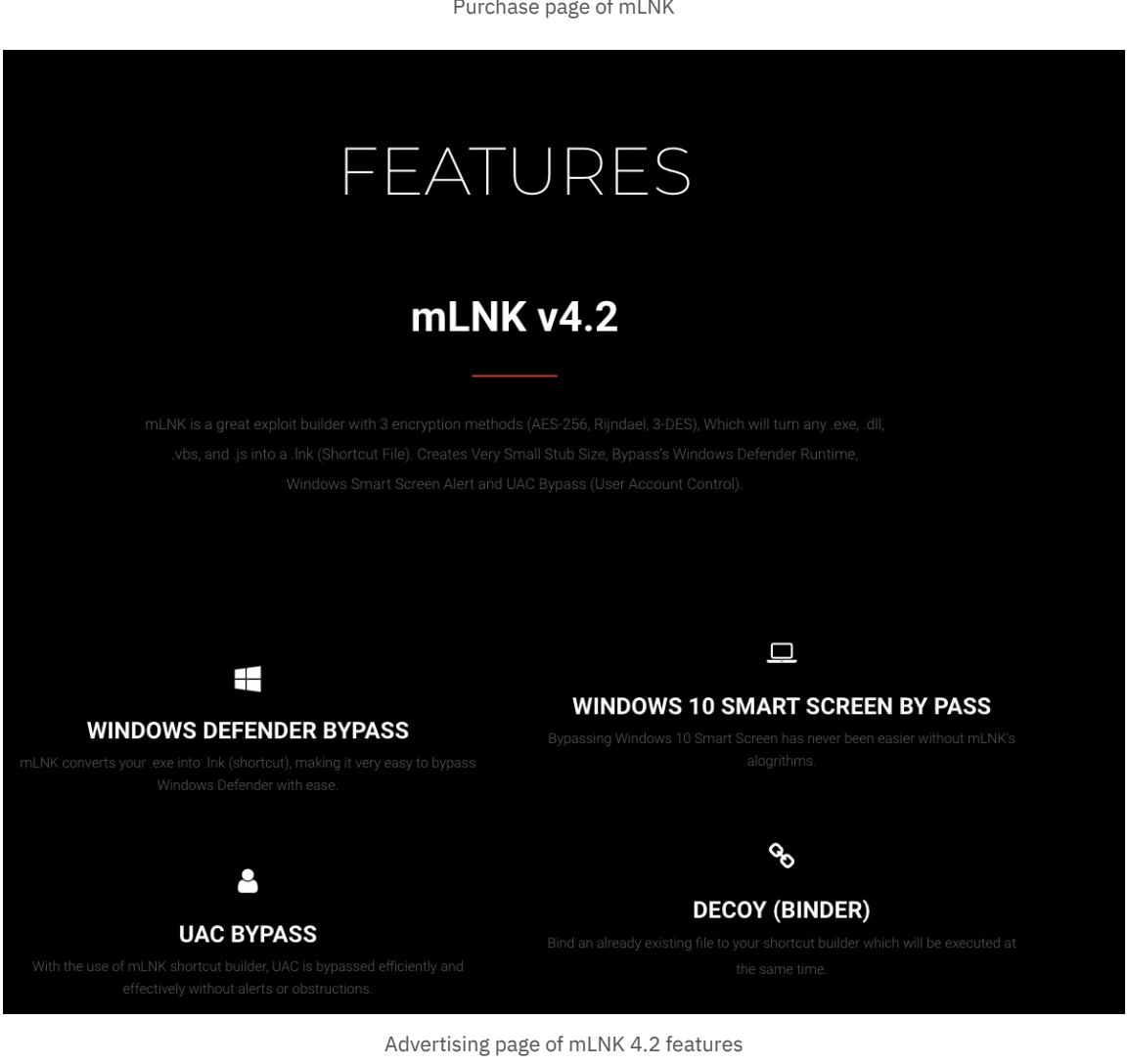


The NativeOne 'exploit website'

The new mLNK version brings new features that enable building `LNK` files that can evade Windows detection mechanisms, such as Microsoft Defender SmartScreen. The public release of mLNK currently sells for a basic price of $100 per month. NativeOne also sells a private release of mLNK 4.2 for $125.00, which bundles more evasion mechanisms than the $25.00 cheaper public release of the tool.



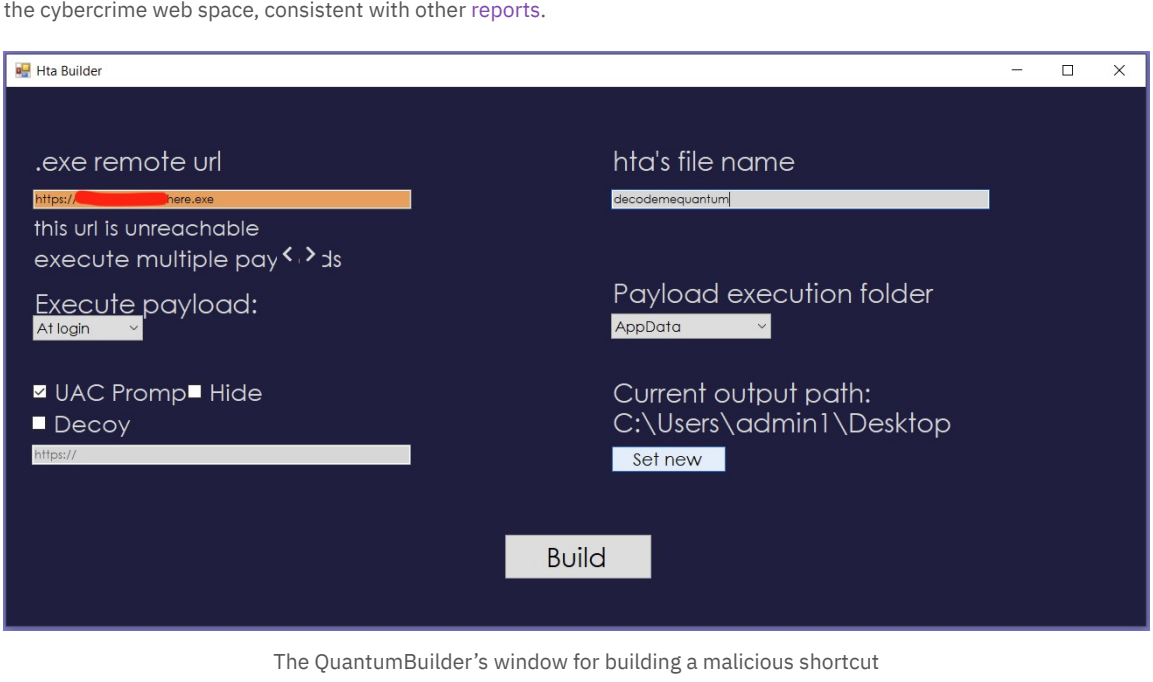
Purchase page of mLNK



Advertising page of mLNK 4.2 features

QuantumBuilder

Similar to mLNK, the QuantumBuilder tool is configurable and easy to use, enabling threat actors to conveniently create malicious `LNK` files. In May 2022, we started observing an advertising campaign for a new QuantumBuilder version in the cybercrime web space, consistent with other reports.



The QuantumBuilder's window for building a malicious shortcut

The actors behind the QuantumBuilder tool distinguish between public, VIP, and private users, and sell the tool for a basic price of €189. The following figure depicts the price list of QuantumBuilder as advertised online, including the advantages of becoming a VIP or private QuantumBuilder user.

[PUBLIC]
1 month > 189 EUR
2 months > 355 EUR
6 months > 899 EUR
Lifetime > 1500 EUR

[VIP]
1 month > 389 EUR
2 months > 555 EUR
6 months > 1099 EUR
Lifetime > 1700 EUR

[PRIVATE]
Lifetime > 2000 EUR

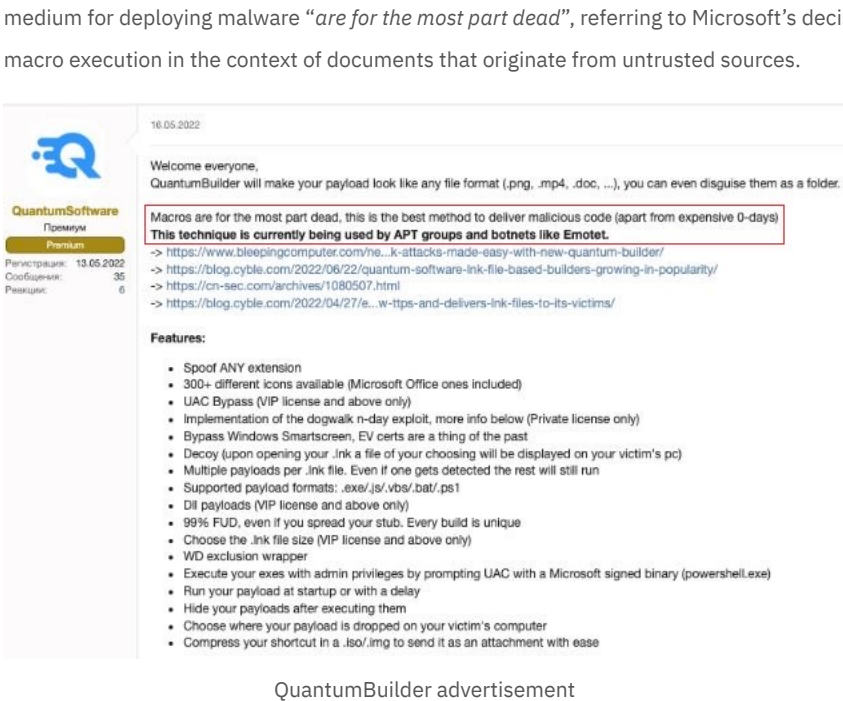
VIP license advantages:
Semi-shared UAC bypass (updated every few days)
WD exclusion wrapper
UAC disabler
Choose the .lnk file size
dll support
100% FUD payload (Semi-shared)

PRIVATE license advantages:

All VIP features
Dogwalk n-day exploit
Completely unique payload
Completely unique and FUD UAC bypass

QuantumBuilder price list

It is interesting to note that the actors behind QuantumBuilder advertise the tool by claiming that Office macros as a medium for deploying malware "*are for the most part dead*", referring to Microsoft's decision to disable by default Office macro execution in the context of documents that originate from untrusted sources.



QuantumBuilder advertisement

Active Attack Campaigns Leveraging Shortcuts

A number of widespread attack campaigns that involve malicious shortcuts are active at the time of writing this article:

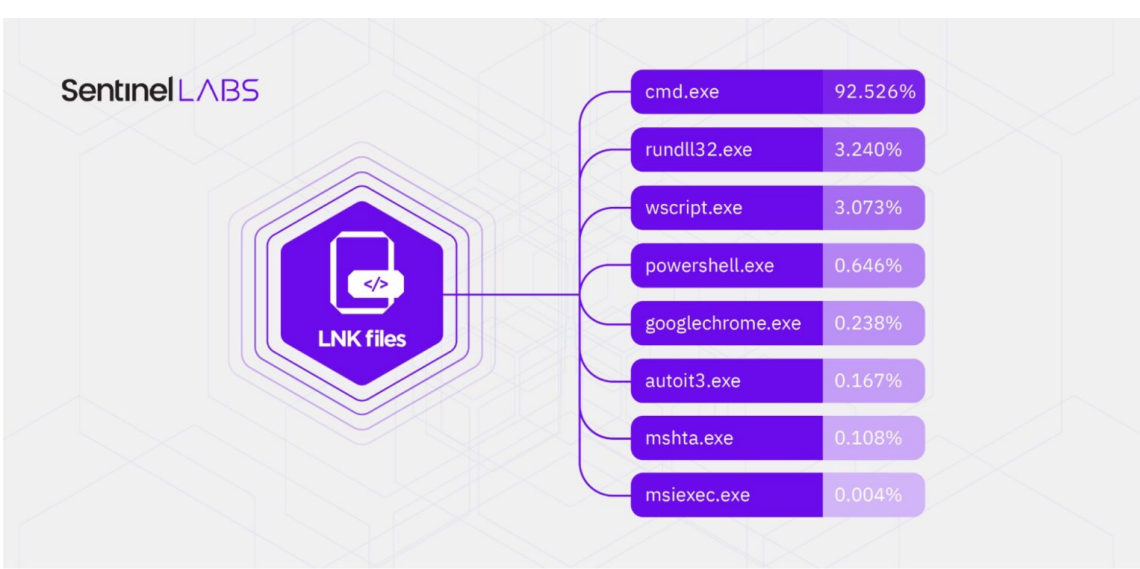
- Threat actors have started intensively distributing the major malware families QBot, Emotet, IcedID, and Bumblebee through `LNK` files since the second quarter of 2022. These malware families are capable of deploying additional malware on compromised systems, including destructive ransomware. In addition, the Threat Analysis Group (TAG) at Google has observed Exotic Lily, an initial access broker (IAB) for ransomware actors, distributing malicious `LNK` files to infect systems.
- Threat actors have been massively deploying the Raspberry Robin worm on systems through malicious `LNK` files since September 2021. These attacks specifically involve infected USB media, containing malicious `LNK` files.
- There are several Ukraine-themed attack campaigns as well as attack campaigns specifically targeting Ukrainian systems that are active since the second quarter of 2022. The Armageddon threat group, which the Security Service of Ukraine identifies as a unit of the Federal Security Service of the Russian Federation, has been distributing malicious `LNK` files through targeted phishing emails. The malicious `LNK` files deploy the GammaLoad.PS1_v2 malware on compromised systems. There are also other Ukraine-themed malicious `LNK` files currently in circulation. In addition, the GlowSand attack campaign includes malicious `LNK` files that download payloads from attacker-controlled endpoints that respond only to requests from systems with Ukrainian IP addresses.

How Threat Actors Are Abusing Shortcuts

In this section, we characterize malicious shortcuts by analyzing the filesystem path to the shortcut target and the command line arguments that the system specifies at shortcut target activation. We take a snapshot of the current malicious shortcut landscape based on VirusTotal as a mass repository of representative malicious `LNK` file samples. We analyzed 27510 `LNK` file samples submitted to VirusTotal between July 14th, 2021 and July 14th, 2022. All samples were considered malicious by at least 30 vendors. 68.89% of the `LNK` file samples were submitted in 2022, and the remaining 31.11% in 2021.

We provide current insights about execution chains that originate from malicious shortcuts to assist threat detection and hunting efforts. The section How Does Windows Execute Shortcuts? below provides background information on Windows shortcuts and the system activities that take place when a user executes a shortcut.

The following image depicts the targets of the malicious shortcuts we analyzed – the executables that the shortcuts execute at target activation – and their prevalence in the set of malicious shortcuts (expressed in percentages, rounded to three decimal places).



Targets of malicious shortcuts

The shortcut targets are LOLbins and/or enable the execution of attacker-specified code and/or executables. We observed the following targets at the top of the list:

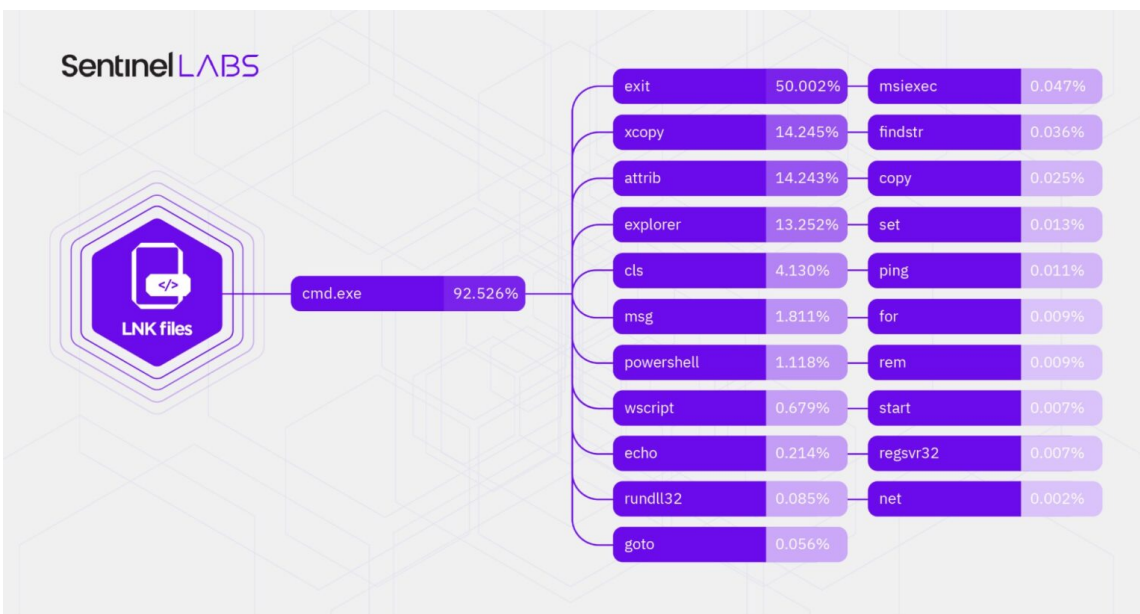
- `cmd.exe`, the Windows command interpreter, which enables the execution of Windows commands and arbitrary executables.
- `rundll32.exe`, which enables the execution of arbitrary code in a Windows DLL.
- `wscript.exe`, a Windows script execution environment, which enables the execution of arbitrary script code.
- `powershell.exe`, the command interpreter of the PowerShell scripting engine.

Malicious shortcuts activate `cmd.exe` as the shortcut target to execute one or multiple Windows commands (typically implemented as executables that reside in the `%SystemRoot%\System32` folder), and/or attacker-provided files:

- Files with the filename extension `.exe` (.exe files) and of Windows executable file format.
- Files with filename extensions different from `.exe` (non-`.exe` files) and of any file format, including the Windows executable format.

Malicious shortcuts execute multiple Windows commands and/or attacker-provided files through `cmd.exe` by specifying them as part of command statements that are chained with the `&` symbol. The chained command statements are part of the command line arguments of the shortcut target `cmd.exe`.

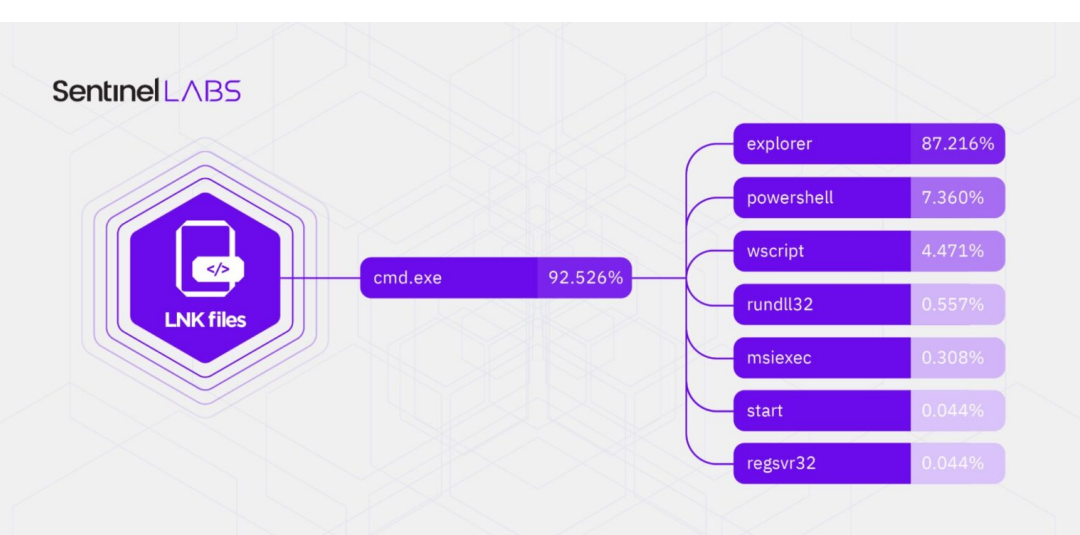
The malicious shortcuts we analyzed execute a variety of Windows commands through `cmd.exe`.



The Windows commands executed through *cmd.exe* and their prevalence

We categorize the commands as follows:

- Commands for command execution flow control, such as exit, goto, and for.
- Commands for file manipulation, such as xcopy, attrib, and copy.
- Commands that enable the execution of attacker-specified code and/or executables – LOLbins, such as explorer, powershell, wscript, rundll32, msiexec, start, and regsvr32.

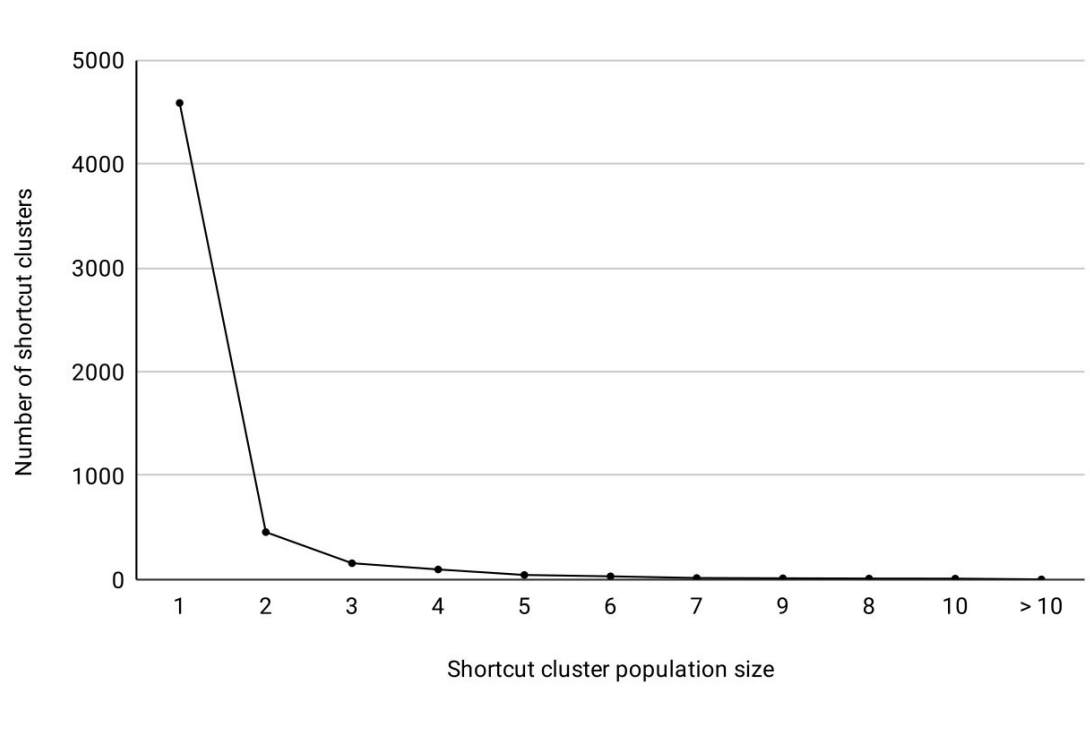


The prevalence of LOLbins in the set of the malicious shortcuts

- Commands for information gathering, reconnaissance, and system configuration, such as findstr, set, ping, and net.
- Commands for messaging and controlling the command interpreter output, such as cls, msg, echo, and rem.

The majority of the filenames of the attacker-provided `.exe` files that the malicious shortcuts we analyzed execute through `cmd.exe` are random – 99.914% of the filenames are random and only 0.086% are non-random (comprehensible), such as `streamer.exe`, `setup.exe`, or `windowsupdater.exe`.

We grouped the malicious shortcuts that execute attacker-provided `.exe` files through `cmd.exe` into clusters according to the filenames of the `.exe` files. We observed that the `.exe` files with non-random filenames are executed by a small number of shortcut clusters with large population sizes, with an average of 1177 shortcuts. On the contrary, the `.exe` files with random filenames are executed by a large number of shortcut clusters with very small population sizes, the majority of which with no more than 3 shortcuts. This shows that defenders should consider highly suspicious shortcuts that execute `.exe` files with random filenames, while staying on top of `.exe` file naming trends in the threat landscape for better detection coverage.



Number of malicious shortcut clusters vs. shortcut cluster population sizes

We observed a very diverse set of 253 different filename extensions of the attacker-provided non-`.exe` files that the malicious shortcuts we analyzed execute through `cmd.exe`.

The top 40 extensions the malicious shortcuts execute through *cmd.exe* and their prevalence

Considering filename extensions only, the malicious shortcuts executed:

- Script files, such as files with the filename extensions `.vbs`, `.vbe`, and `.js`;
- Executable files, such as files with the filename extensions `.scr` and `.dll`;
- Data files – files that store textual, audio, video, archive, and/or other arbitrary content, such as files with the filename extensions `.docx`, `.png`., `.log`., and `.dat`.

We observed that the filename extensions of the vast majority of the apparent data files, such as `.docx` or `.avi`, spoof filename extensions of executable or script files, such as `.exe` or `.vbs`, to masquerade executable or script files as files of other formats.

For approximately 0.5% of the malicious shortcuts we analyzed, the combined length of the filesystem path to the shortcut target and the command line arguments that the system specifies at target activation is greater than 260 characters. Visual inspection of the **Properties > Shortcut > Target** field of an `LNK` file in the Explorer utility, which displays the path to the shortcut target and any command line arguments, does not reveal anything beyond 260 characters. Attackers are known to abuse this for obfuscation – they craft `LNK` files such that command line arguments are padded with characters, such as newline or space, so that the combined length of the path to the shortcut target and the command line arguments significantly exceeds 260 characters.

We observed character padding mostly in shortcuts that targeted `powershell.exe`. In addition, we observed string concatenation and the use of the caret (`^`) symbol for target and/or command line argument obfuscation in approximately 2.5% of the samples.

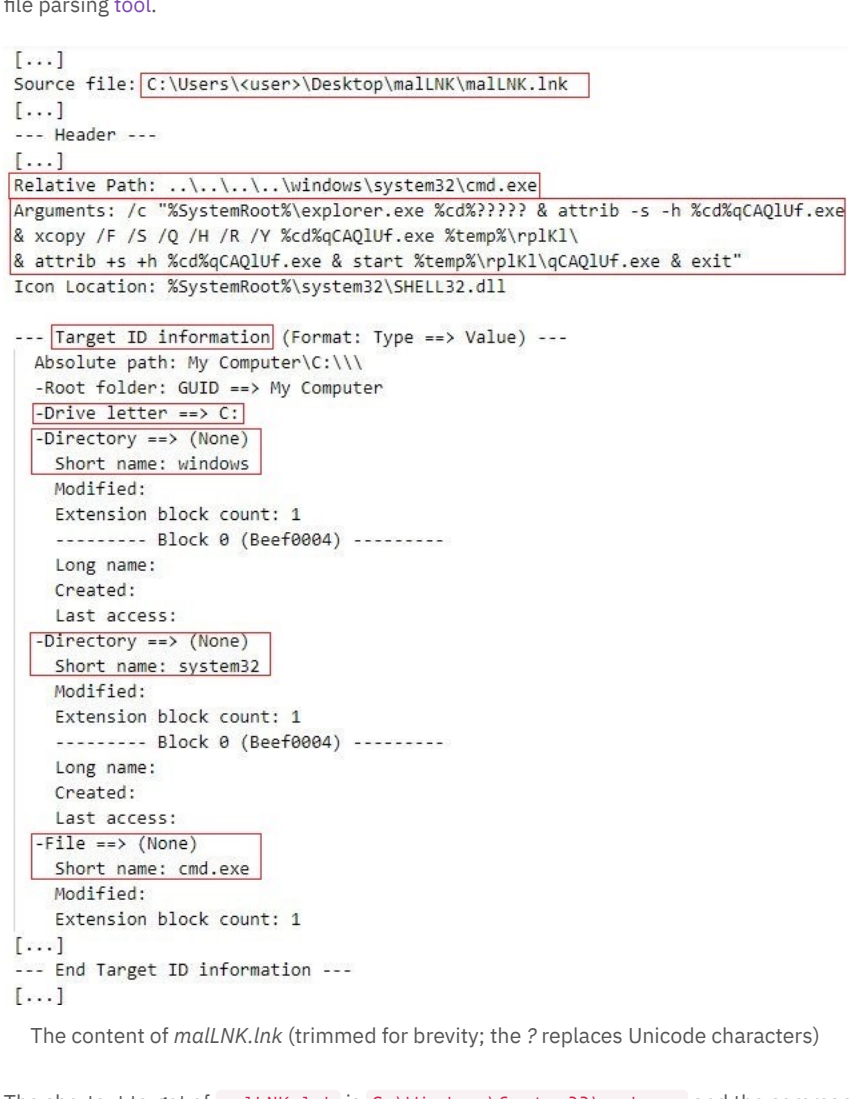
How Does Windows Execute Shortcuts?

The user interface of the Windows operating system, a component referred to as the Windows Shell, manages and conceptually represents as objects entities that users interact with. Objects include entities that reside on the filesystem, such as files and folders, as well as other entities, such as networked computers. The Windows Shell structures these objects into a namespace – the Shell namespace.

When a user creates a shortcut to another object (also referred to as the *shortcut target*) using the **Create shortcut** command, the Windows Shell creates a Shell Link object and an `LNK` file – a file with the `.lnk` filename extension. An `LNK` file is in the binary Shell Link file format and stores information that Windows needs to access (activate) the shortcut target in data structures. This information includes:

- The filesystem path to the shortcut target, for example, the path relative to the location of the `LNK` file (in the `RELATIVE_PATH` structure) and the absolute path (in the `LinkTargetIDList` structure).
- The parameters (command line arguments) that the system specifies at shortcut target activation (in the `COMMAND_LINE_ARGUMENTS` structure).
- The filesystem path to the shortcut icon that the system displays for the `LNK` file in icon view (in the `ICON_LOCATION` structure).

The figure below depicts the content of the malicious `LNK` file that we named `malLNK.lnk` (SHA-1 hash value: 5b241d50f1a662d69c96d824d7567d4503379c37). We displayed the content of `malLNK.lnk` using the `LECmd LNK` file parsing tool.



```
[...]
Source file: C:\Users\<user>\Desktop\malLNK\malLNK.lnk
[...]
--- Header ---
[...]
Relative Path: ..\..\..\..\windows\system32\cmd.exe
Arguments: /c "%SystemRoot%\explorer.exe %cd%????? & attrib -s -h %cd%qCAQlUf.exe
& xcopy /F /S /Q /H /R /Y %cd%qCAQlUf.exe %temp%\rplKl\
& attrib +s +h %cd%qCAQlUf.exe & start %temp%\rplKl\qCAQlUf.exe & exit"
Icon Location: %SystemRoot%\system32\SHELL32.dll

--- Target ID information (Format: Type ==> Value) ---
  Absolute path: My Computer\C:\\\
  -Root folder: GUID ==> My Computer
  -Drive letter ==> C:
  -Directory ==> (None)
    Short name: windows
    Modified:
    Extension block count: 1
    --------- Block 0 (Beef0004) ---------
    Long name:
    Created:
    Last access:
  -Directory ==> (None)
    Short name: system32
    Modified:
    Extension block count: 1
    --------- Block 0 (Beef0004) ---------
    Long name:
    Created:
    Last access:
  -File ==> (None)
    Short name: cmd.exe
    Modified:
    Extension block count: 1
[...]
--- End Target ID information ---
[...]
```

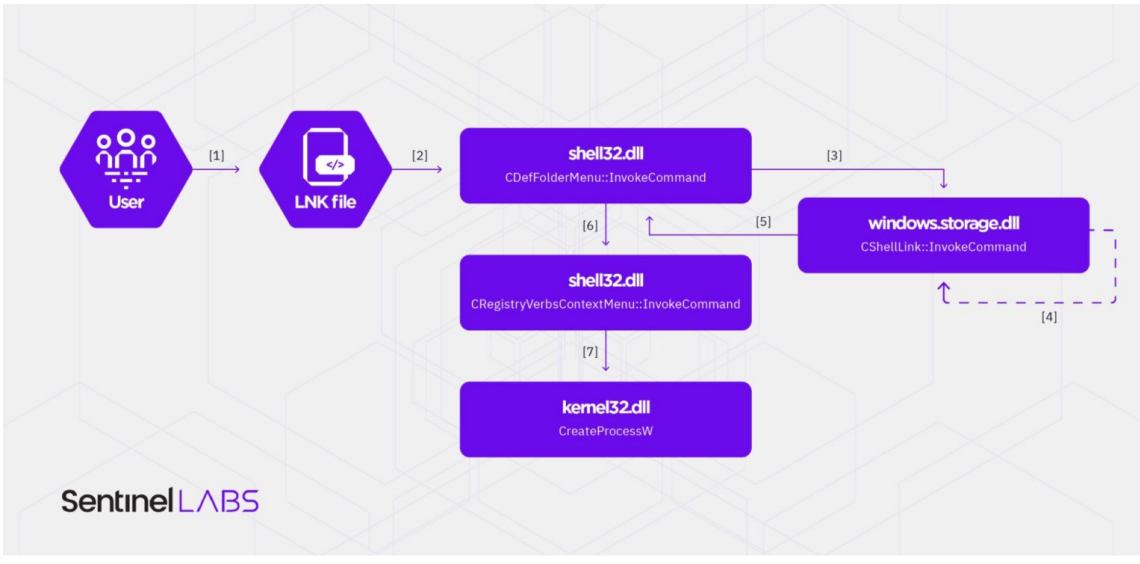
The content of *malLNK.lnk* (trimmed for brevity; the *?* replaces Unicode characters)

The shortcut target of `malLNK.lnk` is `C:\Windows\System32\cmd.exe` and the command line argument is:

```
/c "%SystemRoot%\explorer.exe %cd%新建文件夹 & attrib -s -h %cd%qCAQlUf.exe & xcopy /F /S /Q /H /R /Y %
```

In summary, the activated shortcut target uses the Explorer utility to execute an executable, manipulates the `System` and `Hidden` attributes of executables, copies an executable, and executes the copied executable.

The following figure depicts a simplified overview of the activities that the Windows operating system conducts to activate a shortcut target through an `LNK` file. We take `malLNK.lnk` as a running example.



Overview of system activities at shortcut target activation. The numbers label the transitions between the activities.

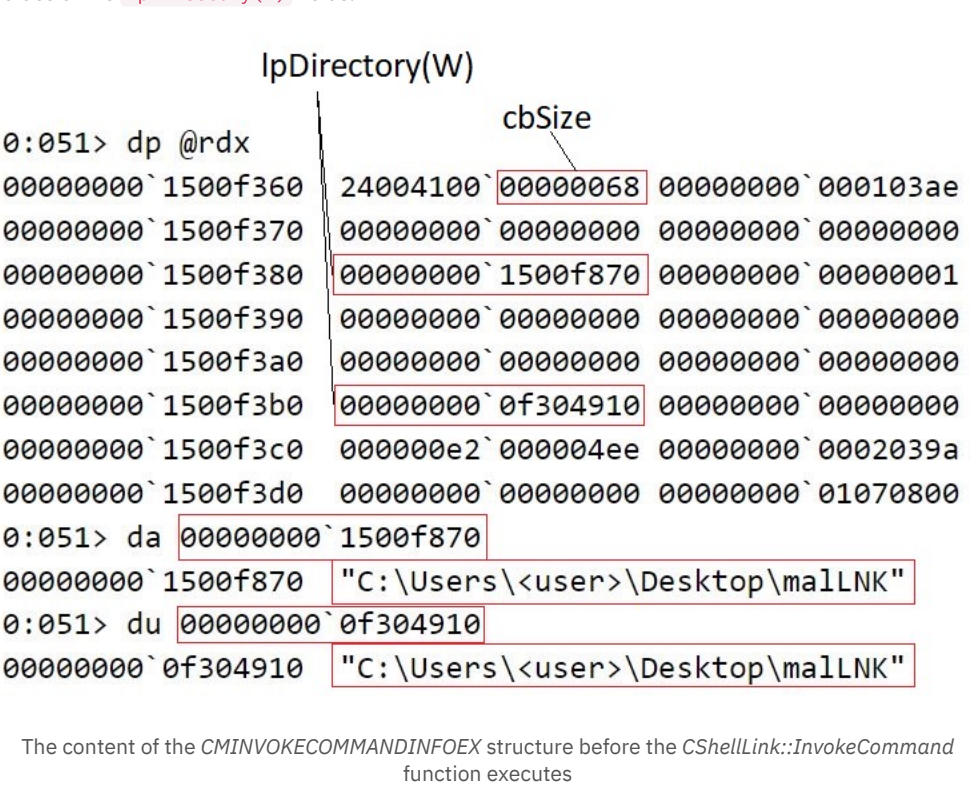
Windows handles shortcut target activation using implementations of the IContextMenu::InvokeCommand Windows Shell method. This function takes a single parameter of type `CMINVOKECOMMANDINFO` or CMINVOKECOMMANDINFOEX. The `CMINVOKECOMMANDINFO(EX)` data structure stores information about the command that the Windows Shell executes when a user triggers the execution of `IContextMenu::InvokeCommand`. In the context of shortcuts, the command is the shortcut target with any command line arguments.

The information that `CMINVOKECOMMANDINFO(EX)` stores includes the working directory at command execution (the `lpDirectory(W)` structure fields) and command parameters (the `lpParameters(W)` structure fields). In contrast to `CMINVOKECOMMANDINFO`, `CMINVOKECOMMANDINFOEX` allows for Unicode structure field values.

When a user double-clicks `malLNK.lnk` (label [1]), the system executes the `CDefFolderMenu::InvokeCommand` function (label [2]). `CDefFolderMenu::InvokeCommand` is implemented in the `%SystemRoot%\System32\shell32.dll`. This function populates a `CMINVOKECOMMANDINFOEX` structure and passes the execution flow to the `CShellLink::InvokeCommand` function with the populated `CMINVOKECOMMANDINFOEX` structure as the function's parameter.

`CShellLink::InvokeCommand` is implemented in `%SystemRoot%\System32\windows.storage.dll` (label [3]). The `CMINVOKECOMMANDINFOEX` data structure that the `CShellLink::InvokeCommand` function takes as its parameter has only a few fields populated, for example, the mandatory `cbSize` field (specifies the size of `CMINVOKECOMMANDINFOEX` in bytes) and `lpDirectory(W)`.

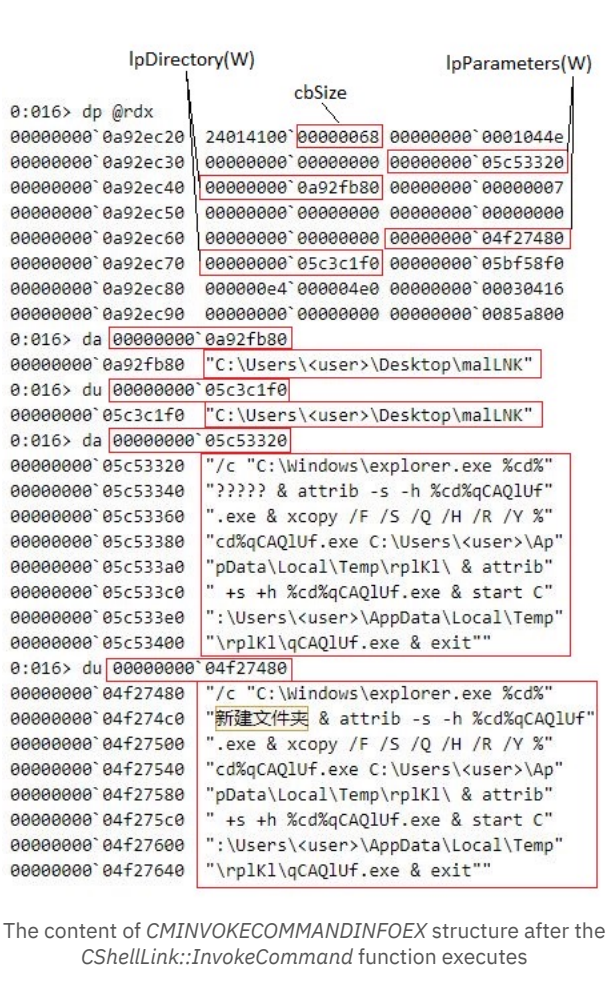
The figure below depicts the content of the `CMINVOKECOMMANDINFOEX` structure that `CShellLink::InvokeCommand` takes as its parameter. `malLNK.lnk` resides in the `C:\Users\<user>\Desktop\malLNK` folder – this determines the values of the `lpDirectory(W)` fields.




```
0:051> dp @rdx
00000000`1500f360  24004100`00000068 00000000`000103ae
00000000`1500f370  00000000`00000000 00000000`00000000
00000000`1500f380  00000000`1500f870 00000000`00000001
00000000`1500f390  00000000`00000000 00000000`00000000
00000000`1500f3a0  00000000`00000000 00000000`00000000
00000000`1500f3b0  00000000`0f304910 00000000`00000000
00000000`1500f3c0  000000e2`000004ee 00000000`0002039a
00000000`1500f3d0  00000000`00000000 00000000`01070800
0:051> da 00000000`1500f870
00000000`1500f870  "C:\Users\<user>\Desktop\malLNK"
0:051> du 00000000`0f304910
00000000`0f304910  "C:\Users\<user>\Desktop\malLNK"
```

The content of the *CMINVOKECOMMANDINFOEX* structure before the *CShellLink::InvokeCommand* function executes

The `CShellLink::InvokeCommand` function conducts the central activities related to shortcut handling. This includes locating the shortcut target on the filesystem, expanding environment variables, and fully populating a `CMINVOKECOMMANDINFOEX` structure (label [4]). `CShellLink::InvokeCommand` passes the execution flow back to the `CDefFolderMenu::InvokeCommand` function with a fully populated `CMINVOKECOMMANDINFOEX` structure (label [5]). For example, the populated `CMINVOKECOMMANDINFOEX` structure stores the command parameter in the `lpParameters(W)` structure fields – this is the data in the `COMMAND_LINE_ARGUMENTS` structure that resides in `malLNK.lnk`.



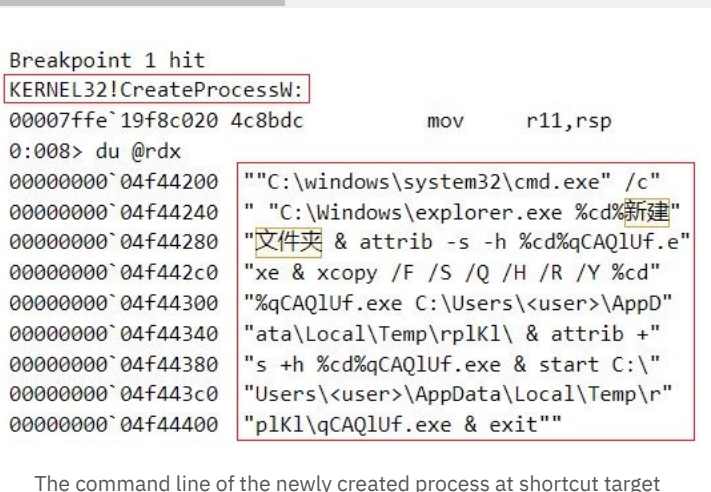

```
0:016> dp @rdx
00000000`0a92ec20  24014100`00000068 00000000`0001044e
00000000`0a92ec30  00000000`00000000 00000000`05c53320
00000000`0a92ec40  00000000`0a92fb80 00000000`00000007
00000000`0a92ec50  00000000`00000000 00000000`00000000
00000000`0a92ec60  00000000`00000000 00000000`04f27480
00000000`0a92ec70  00000000`05c3c1f0 00000000`05bf58f0
00000000`0a92ec80  000000e4`000004e0 00000000`00030416
00000000`0a92ec90  00000000`00000000 00000000`0085a800
0:016> da 00000000`0a92fb80
00000000`0a92fb80  "C:\Users\<user>\Desktop\malLNK"
0:016> du 00000000`05c3c1f0
00000000`05c3c1f0  "C:\Users\<user>\Desktop\malLNK"
0:016> da 00000000`05c53320
00000000`05c53320  "/c "C:\Windows\explorer.exe %cd%"
00000000`05c53340  "????? & attrib -s -h %cd%qCAQlUf"
00000000`05c53360  ".exe & xcopy /F /S /Q /H /R /Y %"
00000000`05c53380  "cd%qCAQlUf.exe C:\Users\<user>\Ap"
00000000`05c533a0  "pData\Local\Temp\rplKl\ & attrib"
00000000`05c533c0  " +s +h %cd%qCAQlUf.exe & start C"
00000000`05c533e0  ":\Users\<user>\AppData\Local\Temp"
00000000`05c53400  "\rplKl\qCAQlUf.exe & exit""
0:016> du 00000000`04f27480
00000000`04f27480  "/c "C:\Windows\explorer.exe %cd%"
00000000`04f274c0  "新建文件夹 & attrib -s -h %cd%qCAQlUf"
00000000`04f27500  ".exe & xcopy /F /S /Q /H /R /Y %"
00000000`04f27540  "cd%qCAQlUf.exe C:\Users\<user>\Ap"
00000000`04f27580  "pData\Local\Temp\rplKl\ & attrib"
00000000`04f275c0  " +s +h %cd%qCAQlUf.exe & start C"
00000000`04f27600  ":\Users\<user>\AppData\Local\Temp"
00000000`04f27640  "\rplKl\qCAQlUf.exe & exit""
```

The content of *CMINVOKECOMMANDINFOEX* structure after the *CShellLink::InvokeCommand* function executes

The `CDefFolderMenu::InvokeCommand` function then passes the execution flow to the `CRegistryVerbsContextMenu::InvokeCommand` function with the fully populated `CMINVOKECOMMANDINFOEX` structure as the function's parameter (label [6]). `CRegistryVerbsContextMenu::InvokeCommand` is implemented in the `shell32.dll` DLL.

The invocation of `CRegistryVerbsContextMenu::InvokeCommand` leads to the creation of a new process by invoking the CreateProcessW function that is implemented in `%SystemRoot%\System32\kernel32.dll` (label [7]). The command line of this process is the shortcut target and the command line argument, as shown below.

```
C:\windows\system32\cmd.exe /c "%SystemRoot%\explorer.exe %cd%新建文件夹 & attrib -s -h %cd%qCAQlUf.exe
```



```
Breakpoint 1 hit
KERNEL32!CreateProcessW:
00007ffe`19f8c020 4c8bdc          mov     r11,rsp
0:008> du @rdx
00000000`04f44200  ""C:\windows\system32\cmd.exe" /c"
00000000`04f44240  " "C:\Windows\explorer.exe %cd%新建"
00000000`04f44280  "文件夹 & attrib -s -h %cd%qCAQlUf.e"
00000000`04f442c0  "xe & xcopy /F /S /Q /H /R /Y %cd"
00000000`04f44300  "%qCAQlUf.exe C:\Users\<user>\AppD"
00000000`04f44340  "ata\Local\Temp\rplKl\ & attrib +"
00000000`04f44380  "s +h %cd%qCAQlUf.exe & start C:\"
00000000`04f443c0  "Users\<user>\AppData\Local\Temp\r"
00000000`04f44400  "plKl\qCAQlUf.exe & exit""
```

The command line of the newly created process at shortcut target activation

Recommendations for Investigators and Users

Investigators should consider highly suspicious any Windows shortcut (LNK file) that exhibits the following in the execution chain that originates from the shortcut:

- Execution of executables (including activation of shortcut targets) that are LOLbins and/or enable the execution of attacker-specified code and/or executables. We observed the following such executables to be among the most prevalent in the set of malicious shortcuts we analyzed: `explorer.exe`, `powershell.exe`, and `wscript.exe`.
- Execution of files with a filename extension different from `.exe` (non-`.exe` files) through `cmd.exe` as the shortcut target. We observed 253 different extensions of the non-`.exe` files that the malicious shortcuts we analyzed execute. The majority of these non-`.exe` files are files that store executable code (for example, Windows executables or script files) masquerading as files of other formats, such as audio or video files.
- Execution of files with the `.exe` extension and random filenames through `cmd.exe` as the shortcut target. For `.exe` files with non-random (comprehensible) filenames, investigators should stay on top of `.exe` file naming trends in the threat landscape for better detection coverage.

Users should stay vigilant against phishing attacks and refrain from executing attached files that originate from unknown sources. Threat actors are distributing malicious `LNK` files through phishing emails at a mass scale and there is a substantial number of active widespread attack campaigns that involve malicious shortcuts. The malicious `LNK` files often come with misleading filenames and icons masquerading as important documents or critical software to lure users into activating the shortcuts.

LNK | MLNK | WINDOWS

ALEKSANDAR MILENKOSKI

Aleksandar Milenkoski is a Senior Threat Researcher at SentinelLabs, with expertise in reverse engineering, malware research, and threat actor analysis. Aleksandar has a PhD in system security and is the author of numerous research papers, book chapters, blog posts, and conference talks. His research has won awards from SPEC, the Bavarian Foundation for Science, and the University of Würzburg.