EventID 77 SOC138 Detected Suspicious XLS File

- SIEM Rule: SOC138 Detected Suspicious Xls File
- Alert creation time: 2021, March (Saturday) 13th at 20:20h
- Alert type: “Malware”
- Device action: “Allowed”
- MITRE ATT&CK Technique: (T1112) Defense Evasion - Modify Registry
- The suspicious file was detected on host “Sofia” with IP “172.16.17.56”
- Metadata of suspicious file:
    - Name: "ORDER SHEET & SPEC.xlsm”
    - MD5 Hash: “7ccf88c0bbe3b29bf19d877c4596a8d4”
    - Size: 2,66 MB

From the alert details I notice that it was created outside of working schedule, the suspicious file may modify Windows Registry, is identified with “.xlsm” extension, and is related to an alleged order.

The Device Action is “Allowed”: the file reached the host “Sofia”.

About Excel Format

The “.xlsm” format is a macro-enabled Excel (2007+) workbook, distinct from “.xls” (legacy binary format) and “.xlsx” (modern non-macro format). It’s a ZIP Archive containing XML and supports VBA Macros.

VBA (Visual Basic for Applications) Macros are small programs written in the VBA programming language that automate tasks in Microsoft Office applications. They are used to: save time by automating repetitive tasks, add custom functionality, and interact with other applications.

The problem is that macros can also be abused by attackers to deliver malware.

OLE (Object Linking and Embedding) is a Microsoft technology that allows embedding data from one application into another (e.g. an Excel chart in a Word doc). OLE Objects are notorious for hiding malware in Office files.

Some exploits don’t require VBA Macros but abuse OLE Objects. “oletools” is a Python toolkit to analyze OLE and Office files.

Static Analysis

After taking ownership of the alert, creating a case, and starting the playbook, I've downloaded the suspicious file as a zip.

On top of my Arch Linux host I've prepared a QEMU/KVM/Libvirt Windows 11 guest domain, in an isolated network, with a few selected tools that I'm familiar with. This took a lot of time, so I'm going to skip the details to focus in the analysis.

I've started with getting info about the ".xlsm" file without executing it:

```
Get-FileHash -Algorithm MD5 -Path '.\ORDER SHEET & SPEC.xlsm'
```

The MD5 Hash is:

- 7CCF88C0BBE3B29BF19D877C4596A8D4

```
trid '.\ORDER SHEET & SPEC.xlsm'
```

According to Trid's current database of references, the file is identified as:

- ".xlsm" Excel Microsoft Office Open XML Format Document (with macros) by 45,9%
- It also detects ".zip" and ".ubox" formats

```
mv '.\ORDER SHEET & SPEC.xlsm' '.\ORDER SHEET & SPEC.zip'
Expand-Archive -Path '.\ORDER SHEET & SPEC.zip' -DestinationPath '.\target\'
cat .\target\xl\vbaProject.bin | Select-String -Pattern "Auto_Open"
cat .\target\xl\vbaProject.bin | Select-String -Pattern "Shell"
```

The patterns found matches, meaning the VBA Macros are automatically opening other files and running shell commands.

```
mv '.\ORDER SHEET & SPEC.zip' '.\ORDER SHEET & SPEC.xlsm'
oleid '.\ORDER SHEET & SPEC.xlsm'
olevba '.\ORDER SHEET & SPEC.xlsm' --decode
```

“oleid” found metadata about the file (Macro-Enabled, MS Excel 2007+, OpenXML, unencrypted…). Presence of XML Macros and relationships with external OLE Objects is shown as negative.

“oleid” finds VBA Macros, and “olevba” finds a lot of stuff:

- “Auto_Open” & “ShellExecuteA”, which I found previously in “xl\vbaProject.bin”
- A lot of hex and base64 encoded strings. Some of these contain control structures (”Else”), I/O operations (”File”, “save”). These are obfuscated parts of code
- I/O keywords: “Open”, “write”, “Print”, “Kill”, “Create”
- Keywords that instruct the execution of programs: “vbNormal”, “run”, “SHELL32”, “exec” (Excel 4 Macros)
- Keywords that manage DLLs: “call”, “Lib”
- Uses “SHELL32.dll”, there’s code that includes this library

Three OLE files are found (”oleObject1.bin”, “oleObject2.bin”, “oleObject3.bin”) at “.\target\xl\embeddings\”.

These contain base64 encoding functions, and obfuscated code.

I also noticed that “oleObject3.bin” is much heavier (2GB) compared to the others (18,5KB & 31,5KB). But that’s not all, it has an exact size of 2,00GB, which indicates that it was programmatically obfuscated filling it with data until its size was exactly that value.

Threat Intelligence

Searching for the MD5 Hash in VirusTotal, I found an updated analysis (2 days ago) in which 47/65 security vendors flag it as:

- VBS/CVE-2017-11882
- Trojan:MSOffice/Downloader
- OLE/Cve-2017-11882
- Xls.Dropper.Valyria
- ...

The first submission was the 28th of January 2021.

Establishes contact with “ocsp[.]pki[.]goog” & “multiwaretecnologia[.]com[.]br” (requests “Podaliri4.exe”).

Checks if Microsoft Office is installed (TA0007 “Discovery”, T1082) querying Windows Registry.

Evades virtual environments (TA0005 “Defense Evasion”, T1497) sleeping to hinder dynamic analysis based on specific criteria. It seems to look for VMware environments.

Executes visual basic scripts through an Alternate Data Stream (ADS), a stealth technique to hide malicious code (TA0005 “Defense Evasion”, T1064):

```
# "script1.vbs" is hidden inside "asc.txt"
C:\Windows\System32\cscript.exe C:\programdata\asc.txt:script1.vbs
```

SHA-256 of “asc.txt:script1.vbs” is:

- 2eaa9d08d7e29c99d616aaccc4728f120e1e9a14816fecab17f388665a89b6e4

The equation editor starts “cmd.exe” (TA0002 “Execution”, T1203) which is related to CVE 2017-11882, performs TCP traffic with IP address “177.53.143.89:443” and performs HTTP requests to it.

This IP corresponds to “multiwaretecnologia[.]com[.]br” domain, from where it requests the payload. It’s later executed as “C:\programdata\Podaliri4.exe”.

I’ve cross referenced these results with Hybrid Analysis.

CVE-2017-11882

This vulnerability related to Microsoft Office allows the attacker to run arbitrary code as the current user by exploiting memory usage.

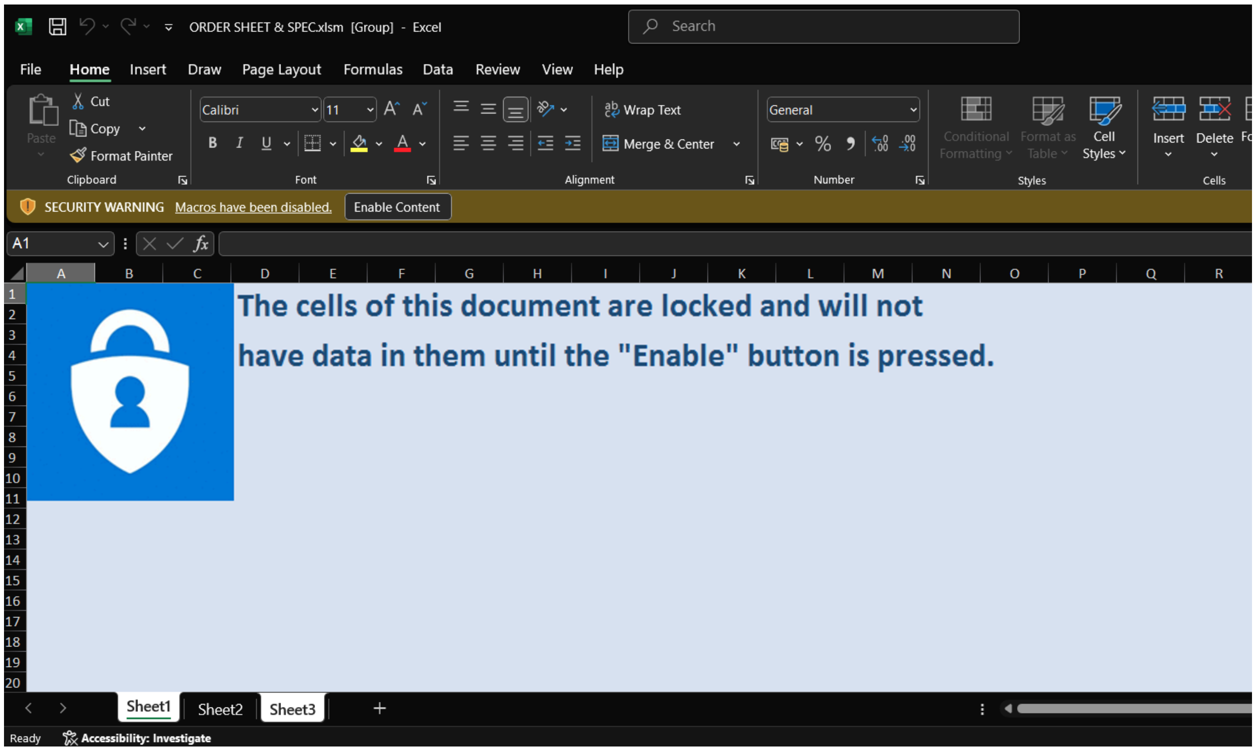
Also known as “Microsoft Office Memory Corruption Vulnerability”.

Dynamic Analysis

I’ve prepared a clean snapshot of my Windows guest domain in an isolated network, and then executed Wireshark, API Monitor, Regshot, and a few Sysinternals tools:

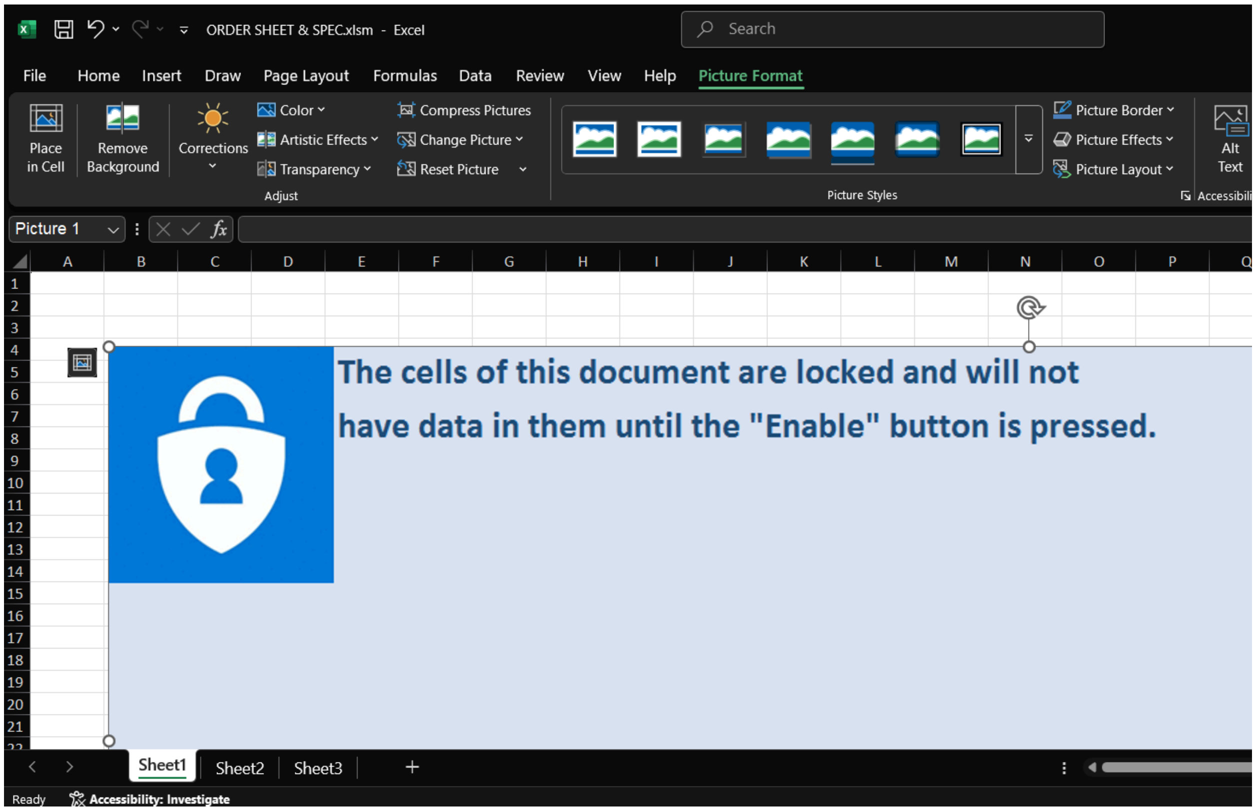
- Process Explorer
- Process Monitor
- Autoruns

Then after opening the Excel file, the “Macros have been disabled” warning is displayed, and an embedded image encourages the victim to press the “Enable Content” button.


ORDER SHEET & SPEC.xlsm [Group] - Excel
Search
File Home Insert Draw Page Layout Formulas Data Review View Help
SECURITY WARNING Macros have been disabled. Enable Content
A1
Sheet1 Sheet2 Sheet3
Ready Accessibility: Investigate

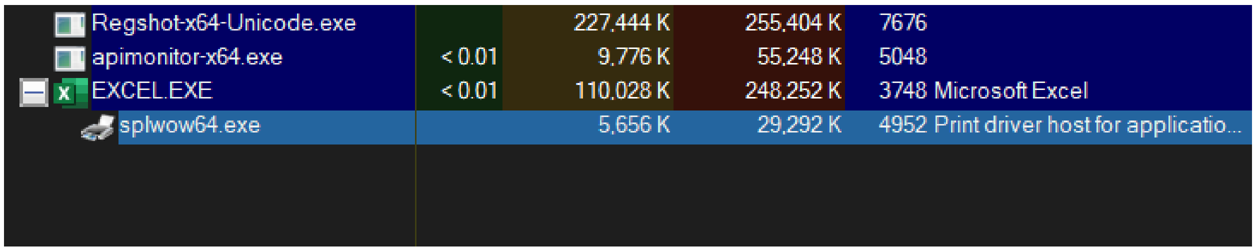
The cells of this document are locked and will not have data in them until the "Enable" button is pressed.

There are three sheets in the Excel file, and are all empty. I've proceeded pressing the button to trigger VBA Macros, and dragged the picture around the Excel file without any purpose.



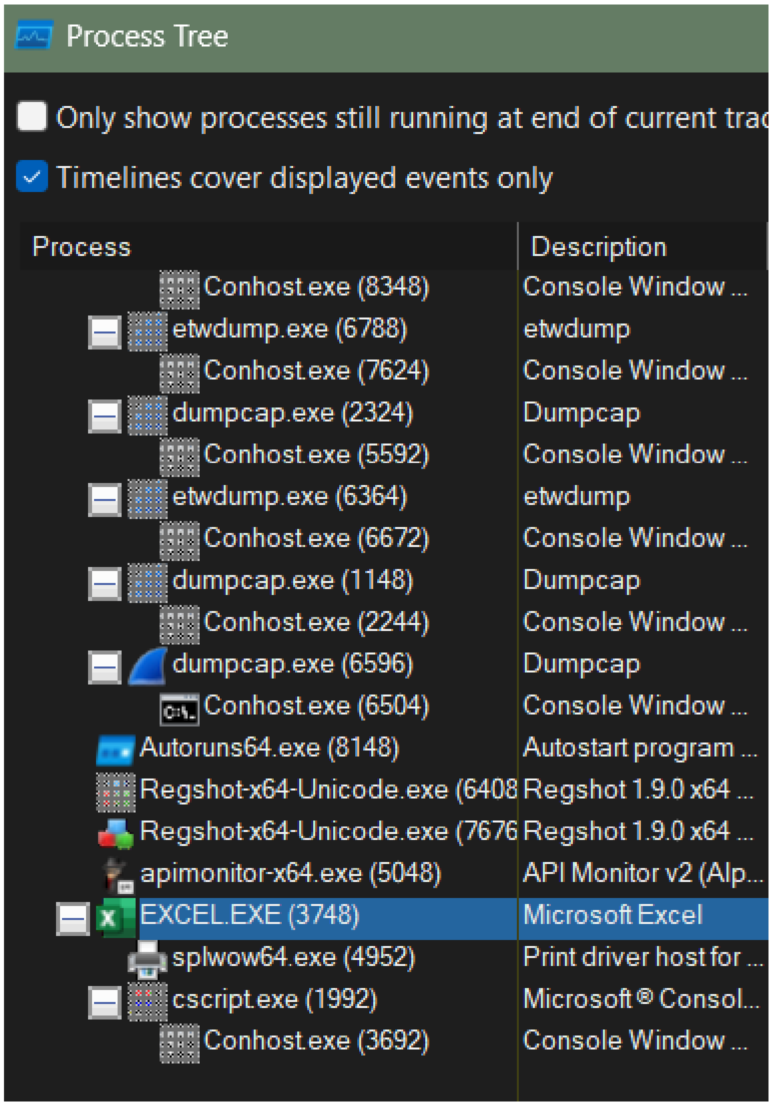
The cells of this document are locked and will not have data in them until the "Enable" button is pressed.

I looked for the "EXCEL.EXE" in the Process Explorer and noticed how it creates an instance of "splwow64.exe" in a child process.



```
Regshot-x64-Unicode.exe              227,444 K   255,404 K   7676
apimonitor-x64.exe          < 0.01     9,776 K    55,248 K   5048
EXCEL.EXE                   < 0.01   110,028 K   248,252 K   3748 Microsoft Excel
  splwow64.exe                         5,656 K    29,292 K   4952 Print driver host for applicatio...
```

The Process Tree of the Process Monitor shows two more childs: “cscript.exe” and “Conhost.exe”.



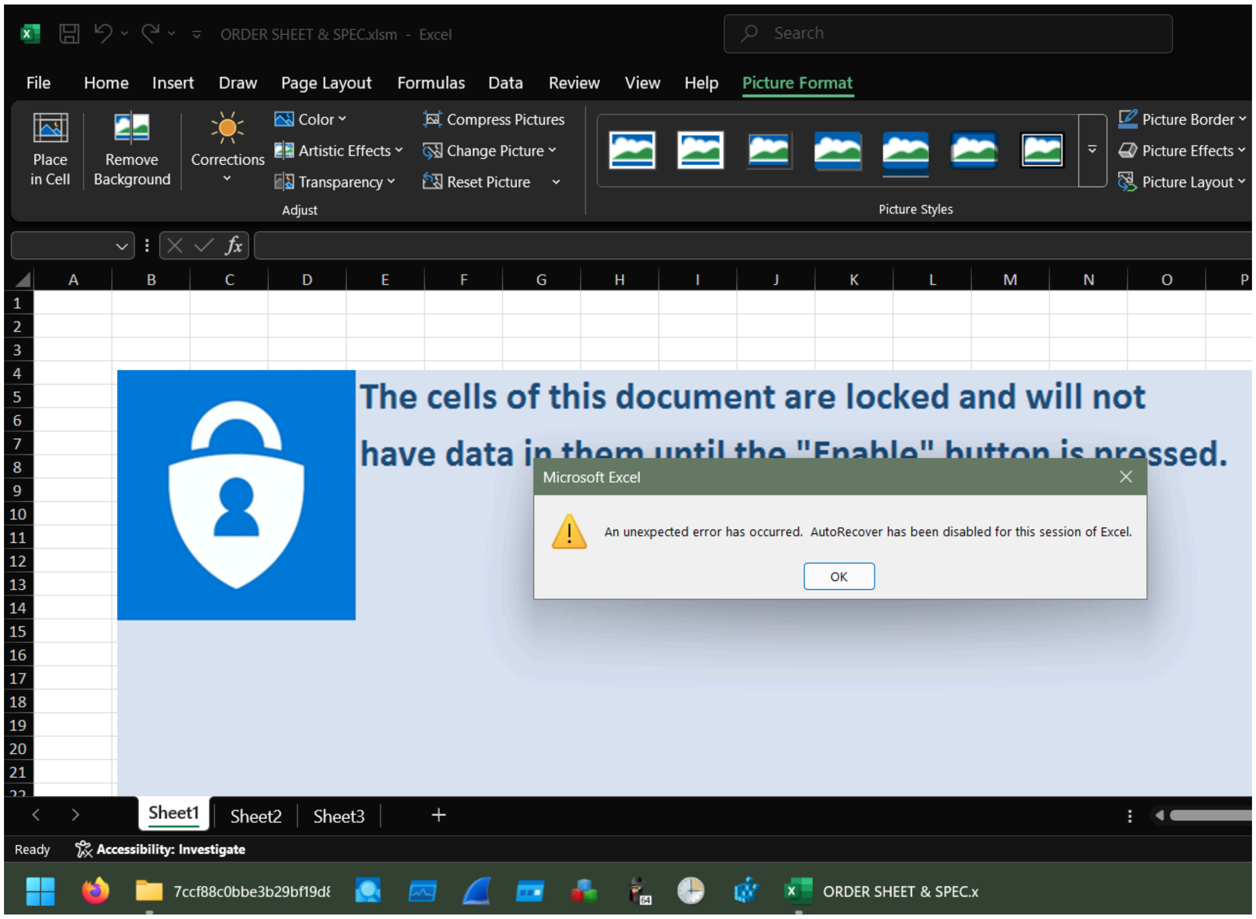
Process Tree

☐ Only show processes still running at end of current trac

☑ Timelines cover displayed events only

Process	Description
Conhost.exe (8348)	Console Window ...
etwdump.exe (6788)	etwdump
Conhost.exe (7624)	Console Window ...
dumpcap.exe (2324)	Dumpcap
Conhost.exe (5592)	Console Window ...
etwdump.exe (6364)	etwdump
Conhost.exe (6672)	Console Window ...
dumpcap.exe (1148)	Dumpcap
Conhost.exe (2244)	Console Window ...
dumpcap.exe (6596)	Dumpcap
Conhost.exe (6504)	Console Window ...
Autoruns64.exe (8148)	Autostart program ...
Regshot-x64-Unicode.exe (6408	Regshot 1.9.0 x64 ...
Regshot-x64-Unicode.exe (7676	Regshot 1.9.0 x64 ...
apimonitor-x64.exe (5048)	API Monitor v2 (Alp...
EXCEL.EXE (3748)	Microsoft Excel
splwow64.exe (4952)	Print driver host for ...
cscript.exe (1992)	Microsoft ® Consol...
Conhost.exe (3692)	Console Window ...

After a while, I noticed an error message displayed in Excel. My guess is that the malware is corrupting memory through “CVE-2017-11882” with VBA Macros to execute arbitrary code in the system, and this error message appears because the vulnerability is triggered.



The cells of this document are locked and will not have data in them until the "Enable" button is pressed.

Microsoft Excel

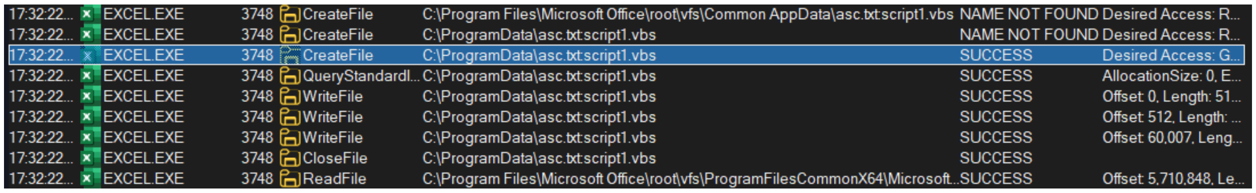
An unexpected error has occurred. AutoRecover has been disabled for this session of Excel.

OK

I left the malware running for a while, because it has sleeps() to hinder dynamic analysis. According to VirusTotal, the behavior of this malware explicitly avoids VMware systems, and since I'm running a QEMU system with VirtIO devices, I expect it to run.

Navigating through events in the Process Monitor, I've filtered to see File System Activity and spotted the creation of "C:\programdata\asc.txt:script1.vbs". I've found this reference in VirusTotal aswell, mapped to a MITRE ATT&CK TTP.

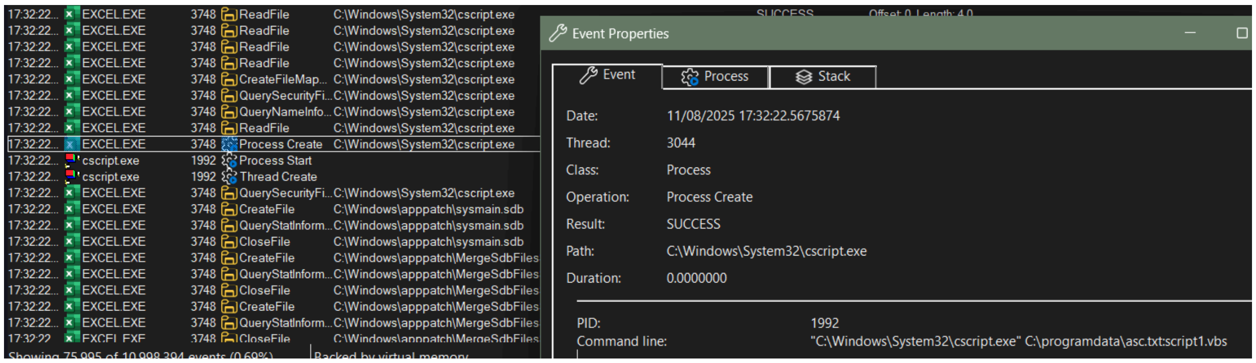
I realized how useful is to list IOCs in VirusTotal before doing dynamic analysis, because I know what to look for (IPs, Registry Keys, Files, domains...) in advance.



```
17:32:22... EXCEL.EXE  3748 CreateFile        C:\Program Files\Microsoft Office\root\vfs\Common AppData\asc.txt:script1.vbs NAME NOT FOUND Desired Access: R...
17:32:22... EXCEL.EXE  3748 CreateFile        C:\ProgramData\asc.txt:script1.vbs                                             NAME NOT FOUND Desired Access: R...
17:32:22... EXCEL.EXE  3748 CreateFile        C:\ProgramData\asc.txt:script1.vbs                                             SUCCESS        Desired Access: G...
17:32:22... EXCEL.EXE  3748 QueryStandardI... C:\ProgramData\asc.txt:script1.vbs                                             SUCCESS        AllocationSize: 0, E...
17:32:22... EXCEL.EXE  3748 WriteFile         C:\ProgramData\asc.txt:script1.vbs                                             SUCCESS        Offset: 0, Length: 51...
17:32:22... EXCEL.EXE  3748 WriteFile         C:\ProgramData\asc.txt:script1.vbs                                             SUCCESS        Offset: 512, Length: ...
17:32:22... EXCEL.EXE  3748 WriteFile         C:\ProgramData\asc.txt:script1.vbs                                             SUCCESS        Offset: 60,007, Leng...
17:32:22... EXCEL.EXE  3748 CloseFile         C:\ProgramData\asc.txt:script1.vbs                                             SUCCESS
17:32:22... EXCEL.EXE  3748 ReadFile          C:\Program Files\Microsoft Office\root\vfs\ProgramFilesCommonX64\Microsoft...SUCCESS        Offset: 5,710,848, Le...
```

"C:\Windows\System32\cscript.exe" (a child of "EXCEL.EXE") executes "script1.vbs" locked behind ADS in "C:\programdata\asc.txt" file (previously created by "EXCEL.EXE").

This is done for obfuscation purposes.



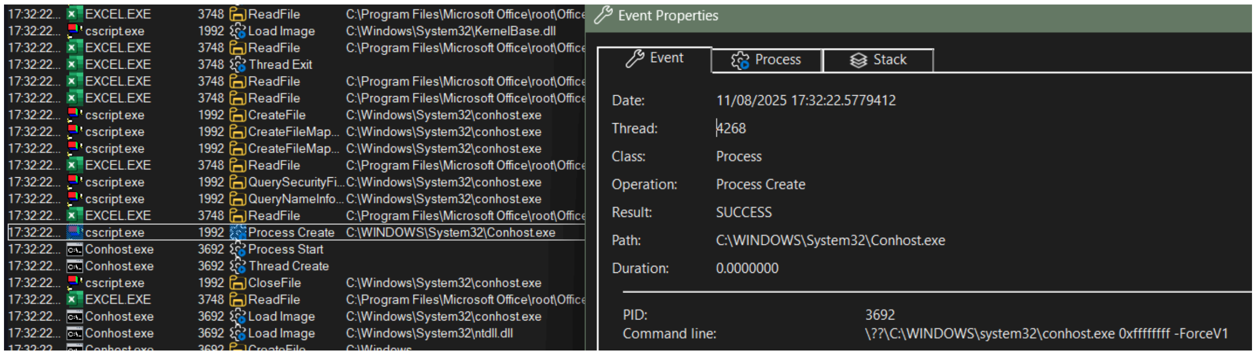
```
17:32:22... EXCEL.EXE  3748 ReadFile          C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 ReadFile          C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 ReadFile          C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 ReadFile          C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 CreateFileMap...  C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 QuerySecurityFi...C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 QueryNameInfo...  C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 ReadFile          C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 Process Create    C:\Windows\System32\cscript.exe
17:32:22... cscript.exe 1992 Process Start
17:32:22... cscript.exe 1992 Thread Create
17:32:22... EXCEL.EXE  3748 QuerySecurityFi...C:\Windows\System32\cscript.exe
17:32:22... EXCEL.EXE  3748 CreateFile        C:\Windows\apppatch\sysmain.sdb
17:32:22... EXCEL.EXE  3748 QueryStatInform...C:\Windows\apppatch\sysmain.sdb
17:32:22... EXCEL.EXE  3748 CloseFile         C:\Windows\apppatch\sysmain.sdb
17:32:22... EXCEL.EXE  3748 CreateFile        C:\Windows\apppatch\MergeSdbFiles
17:32:22... EXCEL.EXE  3748 QueryStatInform...C:\Windows\apppatch\MergeSdbFiles
17:32:22... EXCEL.EXE  3748 CloseFile         C:\Windows\apppatch\MergeSdbFiles
17:32:22... EXCEL.EXE  3748 CreateFile        C:\Windows\apppatch\MergeSdbFiles
17:32:22... EXCEL.EXE  3748 QueryStatInform...C:\Windows\apppatch\MergeSdbFiles
17:32:22    EXCEL.EXE  3748 CloseFile         C:\Windows\apppatch\MergeSdbFiles
Showing 75,995 of 10,998,394 events (0.69%)   Backed by virtual memory
```

```
Event Properties

Date:        11/08/2025 17:32:22.5675874
Thread:      3044
Class:       Process
Operation:   Process Create
Result:      SUCCESS
Path:        C:\Windows\System32\cscript.exe
Duration:    0.0000000

PID:             1992
Command line:    "C:\Windows\System32\cscript.exe" C:\programdata\asc.txt:script1.vbs
```

"C:\Windows\System32\cscript.exe" executes:

```
\??\C:\WINDOWS\System32\conhost.exe 0xffffffff -ForceV1
```

I believe the "\??\" prefix is there to attempt to obscure the process. The "-ForceV1" flag forces "conhost.exe" to use the legacy Console API V1 which is deprecated, less secure, and easier to exploit.

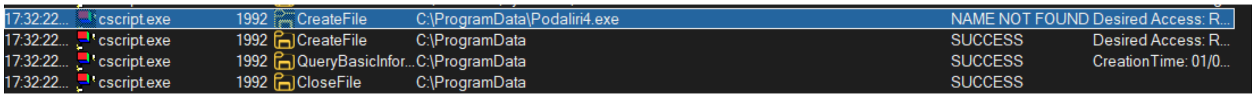


```
17:32:22... EXCEL.EXE   3748 ReadFile           C:\Program Files\Microsoft Office\root\Office
17:32:22... cscript.exe 1992 Load Image         C:\Windows\System32\KernelBase.dll
17:32:22... EXCEL.EXE   3748 ReadFile           C:\Program Files\Microsoft Office\root\Office
17:32:22... EXCEL.EXE   3748 Thread Exit
17:32:22... EXCEL.EXE   3748 ReadFile           C:\Program Files\Microsoft Office\root\Office
17:32:22... EXCEL.EXE   3748 ReadFile           C:\Program Files\Microsoft Office\root\Office
17:32:22... cscript.exe 1992 CreateFile         C:\Windows\System32\conhost.exe
17:32:22... cscript.exe 1992 CreateFileMap...   C:\Windows\System32\conhost.exe
17:32:22... cscript.exe 1992 CreateFileMap...   C:\Windows\System32\conhost.exe
17:32:22... EXCEL.EXE   3748 ReadFile           C:\Program Files\Microsoft Office\root\Office
17:32:22... cscript.exe 1992 QuerySecurityFi... C:\Windows\System32\conhost.exe
17:32:22... cscript.exe 1992 QueryNameInfo...   C:\Windows\System32\conhost.exe
17:32:22... EXCEL.EXE   3748 ReadFile           C:\Program Files\Microsoft Office\root\Office
17:32:22... cscript.exe 1992 Process Create     C:\WINDOWS\System32\Conhost.exe
17:32:22... Conhost.exe 3692 Process Start
17:32:22... Conhost.exe 3692 Thread Create
17:32:22... cscript.exe 1992 CloseFile          C:\Windows\System32\conhost.exe
17:32:22... EXCEL.EXE   3748 ReadFile           C:\Program Files\Microsoft Office\root\Office
17:32:22... Conhost.exe 3692 Load Image         C:\Windows\System32\conhost.exe
17:32:22... Conhost.exe 3692 Load Image         C:\Windows\System32\ntdll.dll
17:32:22... Conhost.exe 3692 CreateFile         C:\Windows
```

```
Date:       11/08/2025 17:32:22.5779412
Thread:     4268
Class:      Process
Operation:  Process Create
Result:     SUCCESS
Path:       C:\WINDOWS\System32\Conhost.exe
Duration:   0.0000000

PID:            3692
Command line:   \??\C:\WINDOWS\system32\conhost.exe 0xffffffff -ForceV1
```

I found an attempt to create the "Podaliri4.exe" file in the same path I found earlier in VirusTotal. All this put together could be trying to download the payload from the C2 Server and drop it in the victim's system.

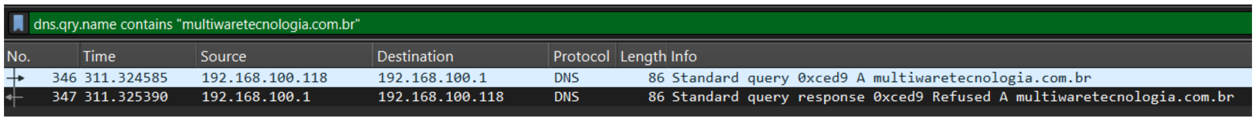
I've seen that some security vendors categorized this malware as a Dropper, and its behavior could be explained as: attempting to create the final payload from different sources. The Excel file is also categorized as "Valyria", a malware built for the purpose of stealing sensitive information from different sources: keystrokes, screenshots...



```
17:32:22... cscript.exe 1992 CreateFile        C:\ProgramData\Podaliri4.exe  NAME NOT FOUND Desired Access: R...
17:32:22... cscript.exe 1992 CreateFile        C:\ProgramData                SUCCESS        Desired Access: R...
17:32:22... cscript.exe 1992 QueryBasicInfor... C:\ProgramData               SUCCESS        CreationTime: 01/0...
17:32:22... cscript.exe 1992 CloseFile         C:\ProgramData                SUCCESS
```

The Windows guest domain is in an isolated network, but the malware attempted to resolve the domain I found in VirusTotal which is meant to transmit the "Podaliri4.exe" resource.

After filtering Wireshark results by that domain, a DNS request is found.



```
dns.qry.name contains "multiwaretecnologia.com.br"

No.    Time        Source           Destination      Protocol Length Info
   346 311.324585  192.168.100.118  192.168.100.1    DNS      86 Standard query 0xced9 A multiwaretecnologia.com.br
   347 311.325390  192.168.100.1    192.168.100.118  DNS      86 Standard query response 0xced9 Refused A multiwaretecnologia.com.br
```

Log Management

I've searched for the IP Address of the victim in Log Management section of LetsDefend and I found two firewall logs communicating with "177.53.143.89" IP, which is a C2 Server that I've found earlier in VirusTotal.

The firewall logs were created at 20:20h of March 13th, just when the alert was created.

This confirms that contact with the C2 Server was successful, so the malicious Excel file reached the victim's system and it was executed.

EDR

I've proved that the file is malicious, and since the device action was allowed, I'm activating containment in the victim's system to prevent further activity.

I've searched for the MD5 Hash of the excel file to check if other endpoints have the file, and no matches were found.

Artifacts

- “https://multiwaretecnologia.com.br/js/Podaliri4.exe” Payload location
- “177.53.143.89” IP Address of C2 Server
- “7ccf88c0bbe3b29bf19d877c4596a8d4” MD5 Hash of Excel file
- “2eaa9d08d7e29c99d616aaccc4728f120e1e9a14816fecab17f388665a89b6e4” SHA-256 of asc.txt:script1.vbs
- “C:\programdata\Podaliri4.exe” Payload execution

Conclusion

The attacker leveraged LOTL tools and social engineering to deliver a Trojan hidden as an Excel file of type “.xlsm” which supports VBA Macros.

The Trojan reached the victim and it was successfully executed. Contact with C2 Servers was established, but the target’s host has been contained to prevent further spreading.

There’s no evidence of lateral movement, because the Excel file is not found in other endpoints.

The Trojan spawns child processes by using VBA Macros and builds a script locked behind ADS that deletes after execution to frustrate forensics.

The purpose of this Trojan is to steal sensitive information from the victim’s system.