2008 Malware Challenge

Scott Knight <knightsc@gmail.com>

- **Describe your malware lab.**

  My lab is a simple setup. It consists of a MacBook with 2GB of memory running MacOSX 10.5.5. I use VMWare Fusion 2.0 since this lets me take multiple snapshots and revert back to any of my previous snapshots. I have two virtual machines setup. The first one is a stock Windows XP Pro machine with minimal extra software installed. OllyDbg is the main application installed along with other utilities like Process Monitor, Regshot, Wireshark and PEiD. The other virtual machine is a stock Ubuntu 8 machine used only when I need a Linux platform for analysis.

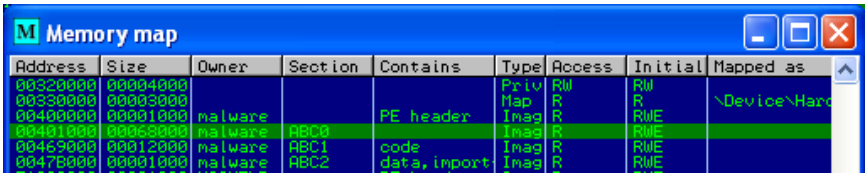
- **Is the malware packed? If so, how did you determine what it was?**

  I chose to answer this question second since if an executable is packed then there isn't a lot of static analysis you can do until it is unpacked.

  The first thing I did, was open up malware.exe in OllyDbg. OllyDbg immediately displayed the following warning:

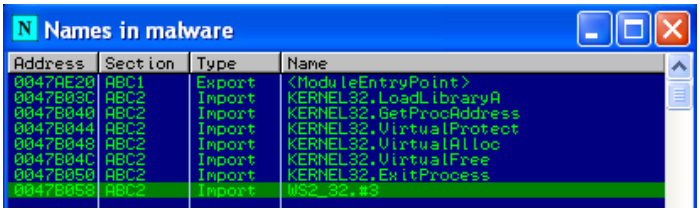
  ```
  Quick statistical test of module 'malware' reports that its code section is
  either compressed, encrypted, or contains large amount of embedded data.
  Results of code analysis can be very unreliable or simply wrong. Do you want to
  continue analysis.
  ```

  This is usually a good first indication that an executable might be packed. The next thing I did, was to look at the different sections in the executable. Usually, in typical applications, you find sections like .text, .data or .rsrc. When I opened up the memory window in OllyDbg the following is what I saw:

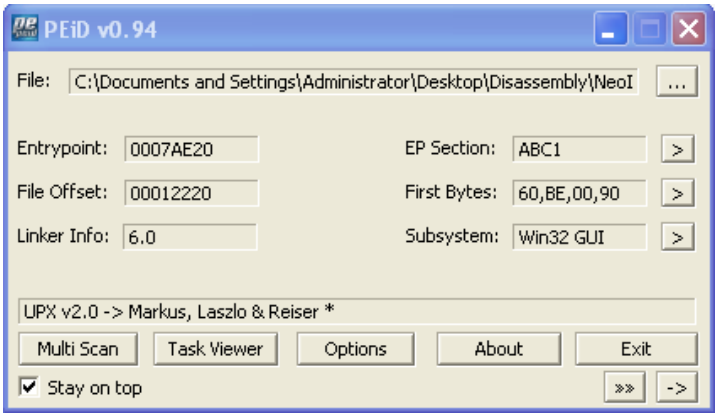
  Notice that the only sections that are present are ones called `ABC0`, `ABC1` and `ABC2` and the typical ones I listed above are not present. This was another good indication to me that the executable was packed. The way that packers typically work are to take an existing executable and transform it in some way and then include an unpacking stub that is executed first to reverse the previous transformation. A lot of the time what you will see is an extra section added and then the entry point of the executable changed to execute the unpacking stub first and then it jumps to the unpacked code. These ABC sections also stood out to me as being similar to what you see in a UPX packed executable. In a typical UPX packed executable you will see sections like `UPX0`, `UPX1` and `UPX2`. So It seems like this executable is packed and possibly with some sort of modified UPX packer.

  Next I looked at the function imports of the executable. So I opened up the Names window in OllyDbg.

  This stands out to me as a pretty short list, with almost all of the imports from KERNEL32.dll. Normally you would find a long list of imports and from many different DLLs. Additionally the functions that are imported are things for loading additionally libraries manually (LoadLibraryA, GetProcAddress) and modifying memory (VirtualProtect, VirtualAlloc, ...).

This convinced me that this is a packed executable but I wasn't a hundred percent sure what it was packed with. My guess was UPX and to double check this I opened up malware.exe in PEiD, and this was the results:



So I was correct. The executable is packed and is packed with UPX.

- **What information can you gather about the malware without executing it?**

  Before attempting static analysis I first needed to unpack the executable. I first tried downloading the UPX utility and running the `upx` command to unpack the program. Since malware.exe had non standard section names the UPX utility was not able to do the job. I did a quick google search on manual unpacking of UPX and found this article, http://vault.reversers.org/UPXUnpackingTut, that told me how to do it. When the UPX unpacking stubs starts it does a `PUSHAD` instruction to save all the registers, then unpacks, then does a `POPAD` to restore the registers and then jumps to the original entry point of the application. All that I needed to do was to locate the `POPAD` instruction and place a breakpoint on the `jmp` instruction. I found the following code:

```
0047AFBE   61               POPAD
0047AFBF   8D4424 80        LEA EAX,DWORD PTR SS:[ESP-80]
0047AFC3   6A 00            PUSH 0
0047AFC5   39C4             CMP ESP,EAX
0047AFC7  ^75 FA            JNZ SHORT malware.0047AFC3
0047AFC9   83EC 80          SUB ESP,-80
0047AFCC  -E9 FB59F9FF      JMP malware.004109CC
```

  I then set a breakpint at `0x0047AFCC`, ran OllyDbg to that line, and then used OllyDump to save the unpacked executable to my hard drive.

  Now that the executable was unpacked I started the static analysis. The first thing I did was to run the `strings` command on the unpacked file. This alone revealed a lot about the malware. The following is a list of strings that caught my attention:

  1. There were a lot of `Echo` strings that were creating a `1.reg` file. So at some point I would guess that registry modifications are made

  2. I started to see a lot of strings with `RealmBoT` in it. A quick google search turned up a handful of pages offering to let me download groups of bot source code, one of them being Crx-realmbot.VNC+RFI.

  3. I also noticed a list of web page names like PayPal, Gmail, Yahoo!, so possible the malware is looking for these pages

  4. Next I saw a long list of what seemed like random words. My guess would be that this is a password list of some sort for trying to brute force passwords for something.

  5. Then I started noticing some strings about IRC. Things like NICK, USER, PASS, for example. I would guess that this malware tries to connect to an irc server. Earlier there were some strings of #chalenge so maybe that is the channel that the bot tries to connect to

6. I noticed some strings about remote shell, so I would guess the program provides some way to spawn a remote shell
7. Some of the strings suggested that the program could start a ftp server
8. I found the string, `Normal key logger active.` This indicates to me that there is some sort of key logging functionality that can be activated.
9. There were also some strings about DDoS flooding
10. There were a handful of strings that seemed to deal with downloading files.
11. There were many strings throughout the program that suggested that there was some sort of httpd functionality.

All of these things are just what I gathered from reading the output from the `strings` command. At first guess I would think based on the name RealmBoT throughout the binary and all the different sort of commands found that this is some sort of IRC bot that installs itself on the victim machine, connects to an irc server and then waits for commands.

Next I started to analyze the main function of the program in OllyDbg to try to figure out what it does on start up. The first function, called from the main function that I noticed was at `0x00401CEE`, here's a snippet from the start of it:

```
00401CEE > 53               PUSH EBX
00401CEF   55               PUSH EBP
00401CF0   8B2D A0804100    MOV EBP,DWORD PTR DS:[<&kernel32.GetModuleHandleA>]
00401CF6   56               PUSH ESI
00401CF7   57               PUSH EDI
00401CF8   68 B4C44100      PUSH malware-.0041C4B4 ; ASCII "kernel32.dll"
00401CFD   FFD5             CALL EBP               ; GetModuleHandle
00401CFF   8B35 B0804100    MOV ESI,DWORD PTR DS:[<&kernel32.GetProcAddress>]
00401D05   8BF8             MOV EDI,EAX            ; EDI = Kernel32Handle
00401D07   33DB             XOR EBX,EBX
00401D09   3BFB             CMP EDI,EBX            ; Make sure we got a handle to kernel32
00401D0B   0F84 FD000000    JE malware-.00401E0E
00401D11   68 A4C44100      PUSH malware-.0041C4A4 ; ASCII "SetErrorMode"
00401D16   57               PUSH EDI
00401D17   FFD6             CALL ESI
00401D19   68 88C44100      PUSH malware-.0041C488 ; ASCII "CreateToolhelp32Snapshot"
00401D1E   57               PUSH EDI
00401D1F   A3 C8B64200      MOV DWORD PTR DS:[<lpSetErrorMode>],EAX
```

This function continued on like this for a long time. Basically the author of this malware is loading almost of all the libraries they needed dynamically. This can make the disassembly less obvious because OllyDbg can't always figure out what the function pointer being called really is. I went through this function and labeled the imports so that the rest of the disassembly would make more sense.

Back in the main function, after the load library function is called, the program calls `CreateMutex` and `WaitForSingleObject` so that it makes sure only one instance of itself is running at the same time. After that it initializes the winsock library for network functionality.

```
004038D0   BF 04010000      MOV EDI,104
004038D5   8D85 18FCFFFF    LEA EAX,DWORD PTR SS:[EBP-3E8]
004038DB   57               PUSH EDI
004038DC   50               PUSH EAX
004038DD   FF15 5C804100    CALL DWORD PTR DS:[<&kernel32.GetWindowsDirectoryA>]
004038E3   8D85 1CFDFFFF    LEA EAX,DWORD PTR SS:[EBP-2E4]
004038E9   57               PUSH EDI
004038EA   50               PUSH EAX
004038EB   56               PUSH ESI
004038EC   FF15 A0804100    CALL DWORD PTR DS:[<&kernel32.GetModuleHandleA>]
004038F2   50               PUSH EAX
004038F3   FF15 9C804100    CALL DWORD PTR DS:[<&kernel32.GetModuleFileNameA>]
```

This code above is the start of the program trying to figure out if this is the first time it has run or not. It does this by calling `GetWindowsDirectoryA` and `GetModuleFileNameA` and looking to see if the

fully-qualified path returned from `GetModuleFileNameA` contains the windows directory. If it does not contain the windows directory then it considers this the first run. If this is the first run then it will make a copy of itself in the windows directory as a file named `Winsec32.exe`. Then the code will call `CreateProcessA` to start the copy running and then `ExitProcess` to stop the first copy from running.

When the `Winsec32.exe` instance starts up it will go through the same check again to determine if the running process is running in the Windows directory. This time however the code will determine that this is not the first time the program has run and continue running at `0x00403AF9`.

The next function that is called is at `0x00408E8A`. This function calls 4 other functions: `RegCreateKeyExA`, `RegSetValueExA`, `RegDeleteValueA`, and `RegCloseKey`. This function takes a single argument passed in from the main function. The main function calls it passing in the `Winsec32.exe` string. The functions purpose it to call `RegSetValueExA` on a list of key and sub key values if the argument is not NULL. If the argument is NULL, then the function will call `RegDeleteValueA` and remove all of the values that might have been previously set. The following is the list of keys/sub keys at `0x0041E5F4` that get set:

Key	Sub Key
0x80000002 (HKLM)	Software\Microsoft\Windows\CurrentVersion\Run
0x80000002 (HKLM)	Software\Microsoft\Windows\CurrentVersion\RunServices
0x80000001 (HKCU)	Software\Microsoft\OLE

For each of these keys `RegSetValueExA` is called setting `lpValueName` equal to `"Microsoft Svchost local services"` and the `lpData` to `"Winsec32.exe"`.

Next comes some strings that I would assume are related to IRC based on the first one.

```
00403B5B   6A 7F            PUSH 7F
00403B5D   68 00C84100      PUSH malware-.0041C800        ; ASCII "testirc1.sh1xy2bg.NET"
00403B62   68 4C554700      PUSH malware-.0047554C
00403B67   E8 D4B90000      CALL malware-.0040F540
00403B6C   A1 B8C74100      MOV EAX,DWORD PTR DS:[41C7B8]
00403B71   6A 3F            PUSH 3F
00403B73   68 18C84100      PUSH malware-.0041C818        ; ASCII "#chalenge"
00403B78   68 CC554700      PUSH malware-.004755CC
00403B7D   A3 9C564700      MOV DWORD PTR DS:[47569C],EAX
00403B82   E8 B9B90000      CALL malware-.0040F540
00403B87   83C4 40          ADD ESP,40
00403B8A   6A 3F            PUSH 3F
00403B8C   68 24C84100      PUSH malware-.0041C824        ; ASCII "happy12"
00403B91   68 0C564700      PUSH malware-.0047560C
00403B96   E8 A5B90000      CALL malware-.0040F540
```

It looks like th function at `0x0040F540` is some sort of string copy and these strings deal with IRC. `testirc1.sh1xy2bg.NET` looks like a host name, so probably the IRC server to connect to. `#chalenge` is a channel and I would guess that `happy12` is either a user name or a password.

After this comes the final part of the main function that looks to be a run loop

```
00403B9B   83C4 0C          ADD ESP,0C
00403B9E   8935 A0564700    MOV DWORD PTR DS:[4756A0],ESI
00403BA4   33FF             XOR EDI,EDI
00403BA6   3935 00B74200    CMP DWORD PTR DS:[42B700],ESI
00403BAC   75 16            JNZ SHORT malware-.00403BC4
00403BAE   8D45 F8          LEA EAX,DWORD PTR SS:[EBP-8]
00403BB1   56               PUSH ESI
00403BB2   50               PUSH EAX
00403BB3   FF15 20B54200    CALL DWORD PTR DS:[<lpInternetGetConnectedState>]
00403BB9   85C0             TEST EAX,EAX
00403BBB   75 07            JNZ SHORT malware-.00403BC4
00403BBD   68 30750000      PUSH 7530
00403BC2   EB 26            JMP SHORT malware-.00403BEA
00403BC4   68 48554700      PUSH malware-.00475548
00403BC9   8935 A85D4700    MOV DWORD PTR DS:[475DA8],ESI
00403BCF   E8 3C000000      CALL malware-.00403C10
00403BD4   83F8 02          CMP EAX,2
00403BD7   8945 FC          MOV DWORD PTR SS:[EBP-4],EAX
00403BDA   74 20            JE SHORT malware-.00403BFC
00403BDC   3935 A85D4700    CMP DWORD PTR DS:[475DA8],ESI
```

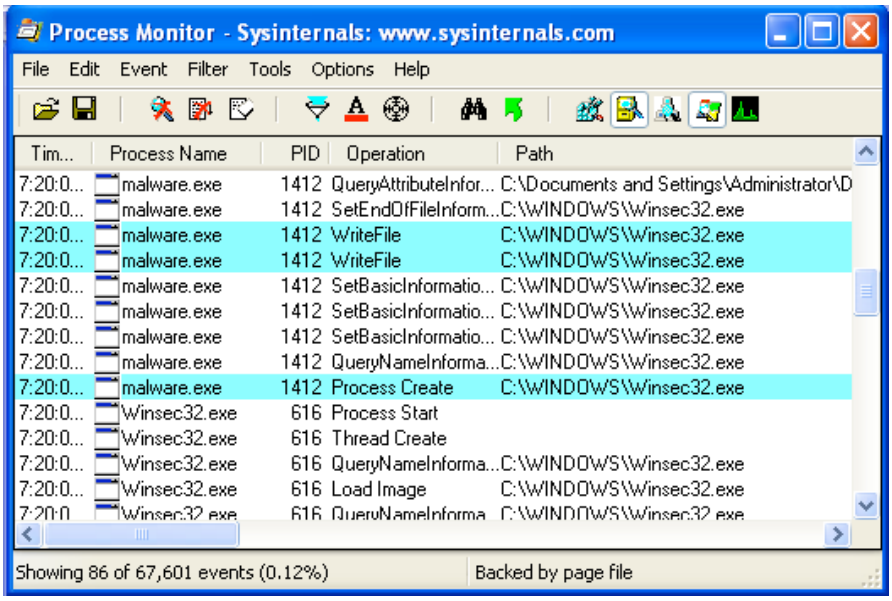
```
00403BE2   74 01            JE SHORT malware-.00403BE5
00403BE4   4F               DEC EDI
00403BE5   68 B80B0000      PUSH 0BB8
00403BEA   FF15 4C804100    CALL DWORD PTR DS:[<&kernel32.Sleep>]
00403BF0   47               INC EDI
00403BF1   83FF 06          CMP EDI,6
00403BF4  ^7C B0            JL SHORT malware-.00403BA6
00403BF6   837D FC 02       CMP DWORD PTR SS:[EBP-4],2
00403BFA  ^75 A8            JNZ SHORT malware-.00403BA4
00403BFC   E8 7EFAFFFF      CALL malware-.0040367F
00403C01   FF15 38B54200    CALL DWORD PTR DS:[<lpWSACleanup>]
00403C07   5F               POP EDI
00403C08   5E               POP ESI
00403C09   33C0             XOR EAX,EAX
00403C0B   5B               POP EBX
00403C0C   C9               LEAVE
00403C0D   C2 1000          RETN 10
```

What we see here is at `0x00403BA4` the `EDI` register is initialized to 0. Instructions are executed and then at `0x00403BF0` we increment the `EDI` register and check to see if it is 6. If it is not 6 then we jump back to the beginning of the loop. If it is 6 then we check another variable to see if it is equal to 2 and if it's not then we loop back to zeroing out the `EDI` register. Essentially the code in here will run until the variable at [EBP-4] is equal 2. When it is equal to 2 then we call the function at `0x0040367F` and then `WSACleanup`. Inside of the loop the first function call is a call to InternetGetConnectedState. If this machine is not connected to the internet, then we end up at the call to `kernel32.Sleep` and then we loop. If it is connected to the internet then the next function to be called is at `0x00403C10`. Just glancing inside the function at `0x00403C10` I see calls to things like `htons`, `socket`, `connect` and `closesocket`. My guess base on the fact that the code right before the main loop appeared to be setting a host name and channel is that the function at `0x00403C10` is what tries to do the IRC connection.

- **Describe the malware's behavior. What files does it drop? What registry keys does it create and/or modify? What network connections does it create? How does it auto-start, etc?**

  Based on the malware's disassembly the behavior seems to be as follows: Create a copy of the malware in the Windows directory named Winsec32.exe, create three registry keys to get the malware to start at system startup, and then try to connect to an IRC server. To verify these findings I set out to actually run and observer the malware. I started by taking a snapshot of my virtual machine so that I could revert back to a clean state after running the malware. I then started up some utilities to help me observe the actions of the malware. I started `regshot` and took an initial snapshot of the registry. I also started `Process Monitor` so I could watch what files were being written to. I then started up a network sniffer to see what the malware tried to connect to. Unfortunately the VMWare Fusion network driver on MacOSX doesn't have BPF support so I couldn't use Wireshark directly. Instead I started up the `vmnet-sniffer` utility that comes with VMWare Fusion which will log virtual machine network traffic in a raw format that can be opened in Wireshark at a later time.

  I then started malware.exe and watched what happened in `Process Monitor`. In the screenshot below, I've highlighted all the `WriteFile` calls and the `CreateProcess` calls.



This confirms what I found while following the disassembly of the main function. When the malware starts up it will make a copy of itself in the Windows directory called Winsec32.exe and then start Winsec32.exe running and terminate. Now that it seemed like the malware had "installed" itself, I took a second snapshot of the registry with `regshot` and then compared it to the original. The results below also confirm what I found during static analysis.

```
Regshot 1.8.2
Comments:
Datetime:2008/10/18 11:30:28  ,  2008/10/18 11:30:50
Computer:SCOTT-8879C7D10 , SCOTT-8879C7D10
Username:Administrator , Administrator

----------------------------------
Keys added:2
----------------------------------
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
HKU\S-1-5-21-746137067-1409082233-839522115-500\Software\Microsoft\OLE

----------------------------------
Values added:3
----------------------------------
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Microsoft Svchost local services: "Winsec32.exe"
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices\Microsoft Svchost local services: "Winsec32.exe"
HKU\S-1-5-21-746137067-1409082233-839522115-500\Software\Microsoft\OLE\Microsoft Svchost local services: "Winsec32.exe"
```

On startup the malware writes these three registry keys. These keys will ensure that when the computer is restarted the Winsec32.exe application will start as well.

The final thing I looked at was the logs from `vmnet-sniffer`. The logs were full of DNS lookup requests for the host name `testirc1.sh1xy2bg.NET`. My virtual machine was set up in host only mode so the lookup failed. In order to further check the network traffic. I edited the host file on my virtual Windows machine and set the `testirc1.sh1xy2bg.NET` host name to point to the IP address of my MacBook's network interface. After I set the host file my network sniffer started showing the following packets:

```
174 152.241528      172.16.99.128   172.16.99.1     TCP     1427 > 6667 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
175 152.241588      172.16.99.1     172.16.99.128   TCP     6667 > 1427 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

172.168.99.128 is the IP address of my virtual Windows XP machine and 172.16.99.1 is the IP address of the VMWare network interface on my MacOSX machine. Port 6667 is the default port for IRC, so the malware was now trying to connect it's IRC server.

So to summarize, on startup the malware drops a copy of itself called Winsec32.exe into the Windows directory. It creates three registry keys setting the values to the Winsec32.exe. These registry keys are used to auto-start the malware when the system starts up. After that it tries to connect to an IRC

server at `testirc1.sh1xy2bg.NET`.

- **What type of command and control server does the malware use? Describe the server and interface this malware uses as well as the domains and URLs accessed by the malware.**

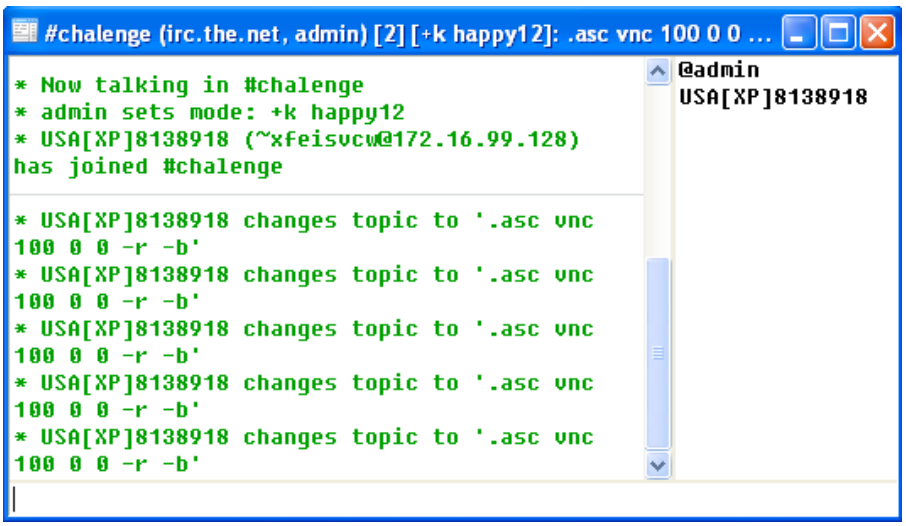
  Based on everything that was discovered during the previous two questions, I think it's safe to say that this malware uses IRC as it's command and control server. The interface it uses is to connect to a specific server, join the #chalenge channel and then wait for commands. The only domain accessed is that of the IRC server that it connects to which is `testirc1.sh1xy2bg.NET`.

- **What commands are present within the malware and what do they do? If possible, take control of the malware and run some of these commands, documenting how you did it.**

  The first thing I did in order to answer this question was to set up an IRC server on my MacBook. I download `ngircd` from http://ngircd.barton.de/. I chose `ngircd` because it was small, easy to compile and has MacOSX support. Next I reverted my VMWare machine back to a clean state. I then used mIRC to connect to my MacBook's IRC server and joined the #chalenge channel so I could watch what happened. Then I ran `malware.exe`. As `malware.exe` ran I saw the following entries show up in the `ngircd` logs:

```
[1106:6] Accepted connection 7 from 172.16.99.128:2744 on socket 4.
[1106:5] User "USA[XP]8138918!~xfeisvcw@172.16.99.128" registered (connection 7)
```

  In mIRC, I saw the following:



  I tried sending a couple private messages to the bot but there was no response. I started to look through the disassembly to try to figure out what commands the bot understood. I went back to the output from my `strings` command and started looking through it for things that looked like possible commands and where they were referenced from. I started to come across strings like `httpd.on`, `ftpd.on` and `proc.on`. All of these strings were referenced from the same procedure located at `0x00403EF2`. I then went through and made a list of all referenced strings in this function. There seemed to be quite a bit of logging messages containing either the string "REALMBOT" or "RealmBoT" so I then dropped all strings with either of those two words in it. Next I went through the list by hand removing items that didn't look like commands and came up with the following list:

```
00423050: login
00425988: rndnick
00425984: rn
00425980: die
00425978: irc.di
00421400: logout
00425974: lo
00425968: versionship
00425964: ver
0042595c: chghttp
00425934: secure
```

```
00425930: sec
00425920: lockdown.off
00425918: ld.off
00425910: visit
00425908: irc.v
00425900: web.off
004258e0: ftpd.off
004258c8: log.off
0042589c: proxy.redirect.off
00425870: ddos.off
00425850: syn.off
00425830: udp.off
0042580c: ping.off
004257e8: proc.off
004257dc: com.ps.off
004257d0: clone.off
004257ac: secure.stop
00425788: scanstop
00425768: stats
00425764: st
00425758: reconnect
00425750: irc.r
00425744: disconnect
0042573c: irc.d
00425734: quit
0042572c: irc.q
00425724: status
0042571c: irc.s
00425718: id
00425710: irc.i
00425708: reboot
0042569c: threads
00425690: threads.l
00422df4: aliases
00425688: irc.al
00425684: log
0042567c: irc.lg
00425670: clearlog
0042566c: clg
00425664: netinfo
00425660: ni
00425654: supersyn
004255b4: sysinfo
0041f718: sys
004255ac: remove
004255a8: rm
004255a0: proc.on
00425598: com.ps
00425590: uptime
00425588: com.up
0042557c: driveinfo
00425574: com.drv
00425568: testdlls
00425560: com.dll
00425558: opencmd
00425550: cmdl
00425544: closecmd
00425524: irc.who
004254d4: getclip
004254cc: com.gc
004254c0: flusharp
004254b8: farp
004254ac: flushdns
004254a0: util.fdns
00425494: currentip
00425490: cip
00425484: httpd.on
0042547c: web.on
00425474: ftpd.on
00425468: d.ftpd.on
0042545c: irc.nick
00425454: irc.n
0042544c: join
00425444: irc.j
0042543c: part
00425434: irc.pt
00425430: raw
00425428: irc.ra
0042541c: killthreads
00425414: killt
00425408: clone.quit
00425400: clone.q
004253f0: clone.rndnick
```

```
004253e4: clone.rn
004253dc: prefix
004253d4: irc.pr
00420e48: open
004253cc: com.o
004253c0: setserver
004253b8: irc.se
004253b4: dns
004253ac: irc.dn
004253a0: killprocess
0042539c: kpc
00425390: prockillid
00425388: pkid
00425380: delete
0042537c: del
00425374: list
0042536c: com.fl
00425360: mirc.cmd
00425360: mirc.cmd
0042535c: cmd
00425550: cmd1
00425350: readfile
00425348: com.rf
0042533c: keylog.on
00425330: cmd.kl.on
00425328: stop
00425328: stop
0041f818: net
004203b4: start
00425328: stop
00425270: pause
00422584: continue
00425380: delete
00425268: share
004251fc: user
0041c040: send
00425134: gethost
0042512c: irc.gh
00425120: addalias
00425118: irc.aa
00425110: privmsg
00425108: irc.pm
00422e90: action
00425100: irc.ac
004250f8: cycle
004250f0: irc.cy
004210fc: mode
004250e8: irc.m
004250dc: rawclone
004250d0: clone.ra
004250c4: clone.mode
004250bc: clone.m
004250b0: clone.nick
004250a4: clone.ni
00425098: clone.join
00425090: clone.j
00425084: clone.part
0042507c: clone.p
00425070: irc.repeat
00425068: irc.rp
00425060: delay
00425058: irc.de
00425050: update
00425048: execute
00425040: com.e
00425038: rename
00425030: com.mv
00425024: clone.make
00425018: clone.start
0042500c: synflood
00425000: ddos.ack
00424ff4: ddos.random
004221e4: download
00424ff0: dl
00424fe4: redirect
00424fd8: daemon.rd
00424fc8: clone.privmsg
00424fbc: clone.pm
00424fac: clone.action
00424fa0: clone.ac
00424f98: advscan
00424f94: asc
00424f88: udpflood
```

```
00424f7c: ddos.udpf
00424f78: u
00424f6c: pingflood
00424f60: ddos.pingf
00424f5c: p
00424f54: httpcon
00424f48: util.hcon
00424f3c: ftp.upload
```

Next I started going through the list of possible commands, sending them to the bot and looking to see if I got any response. I went through the whole list and there was no response, so I started debugging the procedure so I could watch what happened when I sent a message. Eventually I stumbled upon this line:

```
004045A6   3A05 D0C74100    CMP AL,BYTE PTR DS:[41C7D0]
004045AC   897D 28          MOV DWORD PTR SS:[EBP+28],EDI
004045AF  ^0F85 B4FDFFFF    JNZ Winsec32.00404369
```

The data at `0x0041C7D0` is the '.' character. If the first character in the command wasn't a period then the procedure didn't do anything. So then I went through trying the commands with a period in front of the command. Still no response. Next I started debugging/looking at the login command in more detail. It looked like it wanted a password sent to it so I tried the .login command with a password and got this response:

```
-USA[XP]8929677- Are you a Fucker?. (admin!admin@172.16.99.128).
-USA[XP]8929677- No pass for you.
```

After more debugging through the login handling code around `0x004045AF` I figured out that the password it wanted was `gemp123` and my hostname on IRC needed to be `legalize.it`. I eventually just ended up patching Winsec32.exe in memory so that it would accept my login. After I was logged in then the commands above started working. See the example session below.

```
<USA[XP]8929677> [REALMBOT] : Thank for trying.
<admin> .httpd.on
<USA[XP]8929677> [REALMBOT] << Server listening on IP: 172.16.99.128:80, Directory: \. >>
<admin> .ftpd.on
<USA[XP]8929677> [REALMBOT-FTP] : Server started on Port: 0, File: C:\WINDOWS\Winsec32.exe, Request: Winsec32.exe.
<admin> .proc.on
<USA[XP]8929677> RealmBoT (processes.p.l.g) .Â»Â».  Listing processes:
<USA[XP]8929677>  System (4)
<USA[XP]8929677>  smss.exe (428)
<USA[XP]8929677>  csrss.exe (660)
<USA[XP]8929677>  winlogon.exe (692)
<USA[XP]8929677>  services.exe (736)
<USA[XP]8929677>  lsass.exe (748)
<USA[XP]8929677>  vmacthlp.exe (908)
<USA[XP]8929677>  svchost.exe (924)
<USA[XP]8929677>  svchost.exe (1004)
<USA[XP]8929677>  svchost.exe (1116)
<USA[XP]8929677>  svchost.exe (1168)
<USA[XP]8929677>  svchost.exe (1328)
<USA[XP]8929677>  spoolsv.exe (1572)
<USA[XP]8929677>  explorer.exe (1708)
<USA[XP]8929677>  VMwareTray.exe (1844)
<USA[XP]8929677>  VMwareUser.exe (1860)
<USA[XP]8929677>  ctfmon.exe (1868)
<USA[XP]8929677>  VMwareService.exe (260)
<USA[XP]8929677>  alg.exe (1220)
<USA[XP]8929677>  svchost.exe (456)
<USA[XP]8929677>  mirc.exe (144)
<USA[XP]8929677>  Winsec32.exe (512)
<USA[XP]8929677>  OLLYDBG.EXE (1944)
<USA[XP]8929677>  idag.exe (1652)
<USA[XP]8929677>  notepad.exe (3908)
<USA[XP]8929677> RealmBoT (processes.p.l.g) .Â»Â».  Process list completed.
<admin> .prockillid 3908
<USA[XP]8929677> [REALMBOT] << Process killed ID: 3908 >>
```

- **How would you classify this malware? Why?**

  I would classify this malware as a Bot. I would classify it as such since the sole purpose of this program seems to be to connect to an IRC server and await commands.

- **What do you think the purpose of this malware is?**

See the question above. The purpose seems to be to connect to an IRC server and await commands from someone. The malware also seems to be able to attempt to infect new machines via a VNC exploit.

Bonus questions:

- **Is it possible to find the malware's source code? If so, how did you do it?**

  I was able to find the malware's source code. The first thing I did was search for "RealmBot" on Google which turned up a lot of hits for a World Of Warcraft bot. So next I searched on google for "RealmBot Malware" (note my use of quotes are just to detail what I searched for on google but I didn't actually quote my search). The second hit on google had a title of bot source code and mentioned something called crx-realmbot.VNC+RFI. The link however didn't have the source. I then searched for this more detailed name "crx realmbot vnc download" After looking through lots of results to different RAR files of multiple bots, I came across a link, http://darksun.ws/download/Bots/, that let me download just the individual crx realmbot files. After comparing the strings in the downloaded source code to the strings in the malware.exe and comparing the entry point in my disassembly to the WinMain function in the source, I concluded that this was in fact the source to this bot.

- **How would you write a custom detection and removal tool to determine if the malware is present on the system and remove it?**

  In order to write a custom detection and removal tool there is only a few things that would need to be done. First, you would want to look at the running processes and see if there are any processes running called `Winsec32.exe`. If you find this process running then immediately stop it. Even if you don't see it running the detection program should still go on to the next step. Next look in the Windows directory for a file named `Winsec32.exe`, if this file is found then it should be deleted. Next the detection tool should look in the registry for the following three keys:

  - HKLM\Software\Microsoft\Windows\CurrentVersion\Run\Microsoft Svchost local services
  - HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices\Microsoft Svchost local services
  - HKCU\Software\Microsoft\OLE\Microsoft Svchost local services

  If any of these keys are found then they should also be removed. Stopping the malware from running, removing the executable, and cleaning the registry keys should be enough to remove the malware from the system.