Source: https://github.com/ytisf/theZoo
Initial Analysis & Outside Research: Not much is known about this malware or at least not much research has been done on it. Looking it up online did little to help be analyzed further and when I started to analysis it as well I could understand why. This malware for one is a GUI so automated running of it can't allow for complete knowledge of its functionality. Also, it appears this malware needs specific criteria to even run properly or at all. These criteria I was not able to fully check but I was able to determine that it is what the malware needs.

This might have been malware which attacked the DNC but it is also part of a family of malware and this was not clear.

Static Analysis:

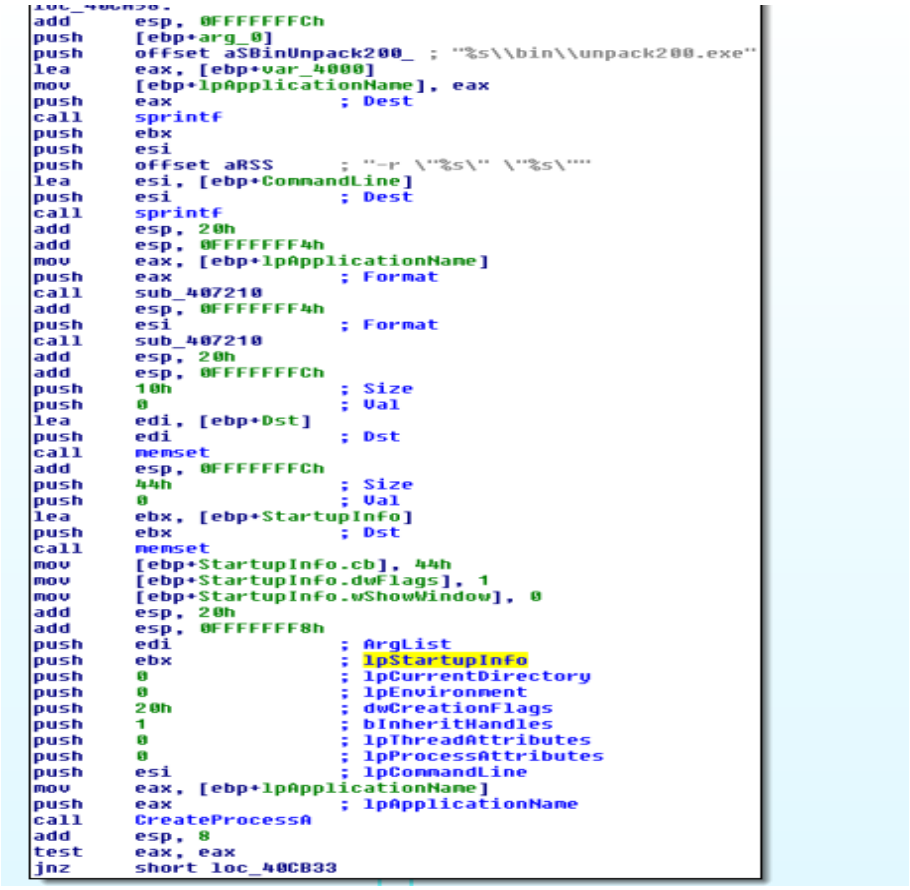
Initially, like always I check the executable for its type and its data directories and settings. This was a 32 bit PE stripped executable. It had a resource section that wasn't relevant to this malware and using PE explorer and PEID there showed to be no indication that this malware was commercially packed.

Checking the imports tables I can across something interesting. A TLS alloc function, which was weird because of the fact that there was not TLS section or Table in this PE executable. I will look at this later.

In the strings window I found the following:

```
; char aSBinUnpack200_[]
aSBinUnpack200_ db '%s\bin\unpack200.exe',0
```

As I stated before, this binary doesn't appear to be packed in anyway so this was clearly unusual. XREFing it with its sub routine of sub_40CA00.



```
add     esp, 0FFFFFFFCh
push    [ebp+arg_0]
push    offset aSBinUnpack200_ ; "%s\\bin\\unpack200.exe"
lea     eax, [ebp+var_4000]
mov     [ebp+lpApplicationName], eax
push    eax             ; Dest
call    sprintf
push    ebx
push    esi
push    offset aRSS     ; "-r \"%s\" \"%s\""
lea     esi, [ebp+CommandLine]
push    esi             ; Dest
call    sprintf
add     esp, 20h
add     esp, 0FFFFFFF4h
mov     eax, [ebp+lpApplicationName]
push    eax             ; Format
call    sub_407210
add     esp, 0FFFFFFF4h
push    esi             ; Format
call    sub_407210
add     esp, 20h
add     esp, 0FFFFFFFCh
push    10h             ; Size
push    0               ; Val
lea     edi, [ebp+Dst]
push    edi             ; Dst
call    memset
add     esp, 0FFFFFFFCh
push    44h             ; Size
push    0               ; Val
lea     ebx, [ebp+StartupInfo]
push    ebx             ; Dst
call    memset
mov     [ebp+StartupInfo.cb], 44h
mov     [ebp+StartupInfo.dwFlags], 1
mov     [ebp+StartupInfo.wShowWindow], 0
add     esp, 20h
add     esp, 0FFFFFFF8h
push    edi             ; ArgList
push    ebx             ; lpStartupInfo
push    0               ; lpCurrentDirectory
push    0               ; lpEnvironment
push    20h             ; dwCreationFlags
push    1               ; bInheritHandles
push    0               ; lpThreadAttributes
push    0               ; lpProcessAttributes
push    esi             ; lpCommandLine
mov     eax, [ebp+lpApplicationName]
push    eax             ; lpApplicationName
call    CreateProcessA
add     esp, 8
test    eax, eax
jnz     short loc_40CB33
```

This sub_routine does two things. It starts up this so called unpack200.exe executable and it DEBUGS. You see, upon further review I noticed that this executable is using the EXE4J_LOG and ISNTALL4J_LOG environment names to actually record that data about the program. This program is actually using Java libraries to run certain commands and operations. I believe this was done to in a way hide its initial functionality as these JRE functions do not show up on the Import Table.

Although because of that fact that everything is being recorded in the current directory of the name .error which isn't even hidden its hard to understand what their intensions are.

The debugging is done in sub_routine 406FF4. I will call this sub_routine DEBUG

In subroutine sub_40CC20 I see that the sub_routine (40CA00) is being called. 40CC20 gives 40CA000 an argument of jar files.

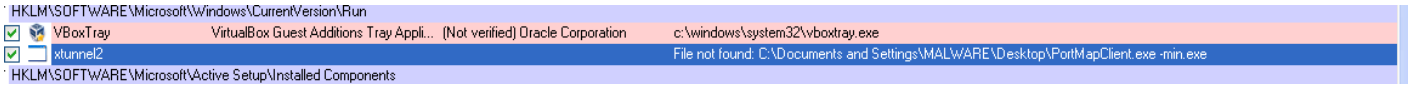
For those who do not know Jar files are an archive format. This unpack200.exe "appears" to be unpacking these jar files and using the files inside of them for some purpose.

```
add     esp, 0FFFFFFF4h
push    offset aUnpackingJre ; "Unpacking JRE"
call    DEBUG_001
add     esp, 0FFFFFFF4h
push    offset aPreparingJre__ ; "Preparing JRE ...\n"
call    printf
add     esp, 20h
add     esp, 0FFFFFFF4h
mov     eax, ds:_iob
add     eax, 20h
push    eax             ; File
call    fflush
add     esp, 0FFFFFFF8h
push    offset aLibRt_jar ; "lib\\rt.jar"
push    edi
call    unpacker
add     esp, 20h
add     esp, 0FFFFFFF8h
push    offset aLibCharsets_ja ; "lib\\charsets.jar"
push    edi
call    unpacker
add     esp, 0FFFFFFF8h
push    offset aLibPlugin_jar ; "lib\\plugin.jar"
push    edi
call    unpacker
add     esp, 20h
add     esp, 0FFFFFFF8h
push    offset aLibDeploy_jar ; "lib\\deploy.jar"
push    edi
call    unpacker
add     esp, 0FFFFFFF8h
push    offset aLibExtLocaleda ; "lib\\ext\\localedata.jar"
push    edi
call    unpacker
add     esp, 20h
add     esp, 0FFFFFFF8h
push    offset aLibJsse_jar ; "lib\\jsse.jar"
push    edi
call    unpacker
add     esp, 0FFFFFFF4h
push    offset aUnpackingJreDo ; "Unpacking JRE done"
call    DEBUG_001
```

I was able to find command line arguments. It seems as though the command line arguments -console can be used although this had no noticeable effect on the execution of the program.

From here I just decided to jump right into Dynmanic analysis. I want to see what exactly is going on.

Autoruns:



A software known as xtunnel2 was found as a new registry key for a file currently not on my system called **PortMapClient -min.exe**. This opens the door for what this malware could be.

There is actually a family of Xtunnel malware.

Through static analysis I was able to find an IP address: 45.114.10.45. I did an IP lookup and it was a chinese host name but at the same time since this malware was rather old this could be misleading seeing how someone else could have been using this IP.

I now have a method of capturing network traffic. Using the IP I created a host only network and redirected all network traffic to me and sent data back using Inetsim. This didn't lead to much except for the fact about every second that this program was trying to connect to and IP even when no buttons were being pressed. This was very suspicious. I will post the captured packet below.

This malware also can't have two instances of itself open at the same time. While that usually indicates a mutex that didn't appear to be the cases as I couldn't find one.

0040CC20