Introduction to Immunity CANVAS

Table of Contents

Introduction

This document is intended to give you a crash course in the CANVAS attack framework. Over the next few chapters, you will learn what an exploit is, what "buffer overflow" means, what shellcode is used for, and why payloads are so important. Don't worry if you are unfamiliar with these terms, this document will walk you though everything you need to know about running CANVAS exploits from installing the application through what kind of fun you can have with a newly compromised system.

There are always helpful movies and other information here:
http://www.immunityinc.com/products-documentation.shtml

If you are a Linux user please see Install.txt for installation instructions, although this document will still come in handy to familiarize you with the product.

Obtaining and Installing CANVAS

Obtaining CANVAS

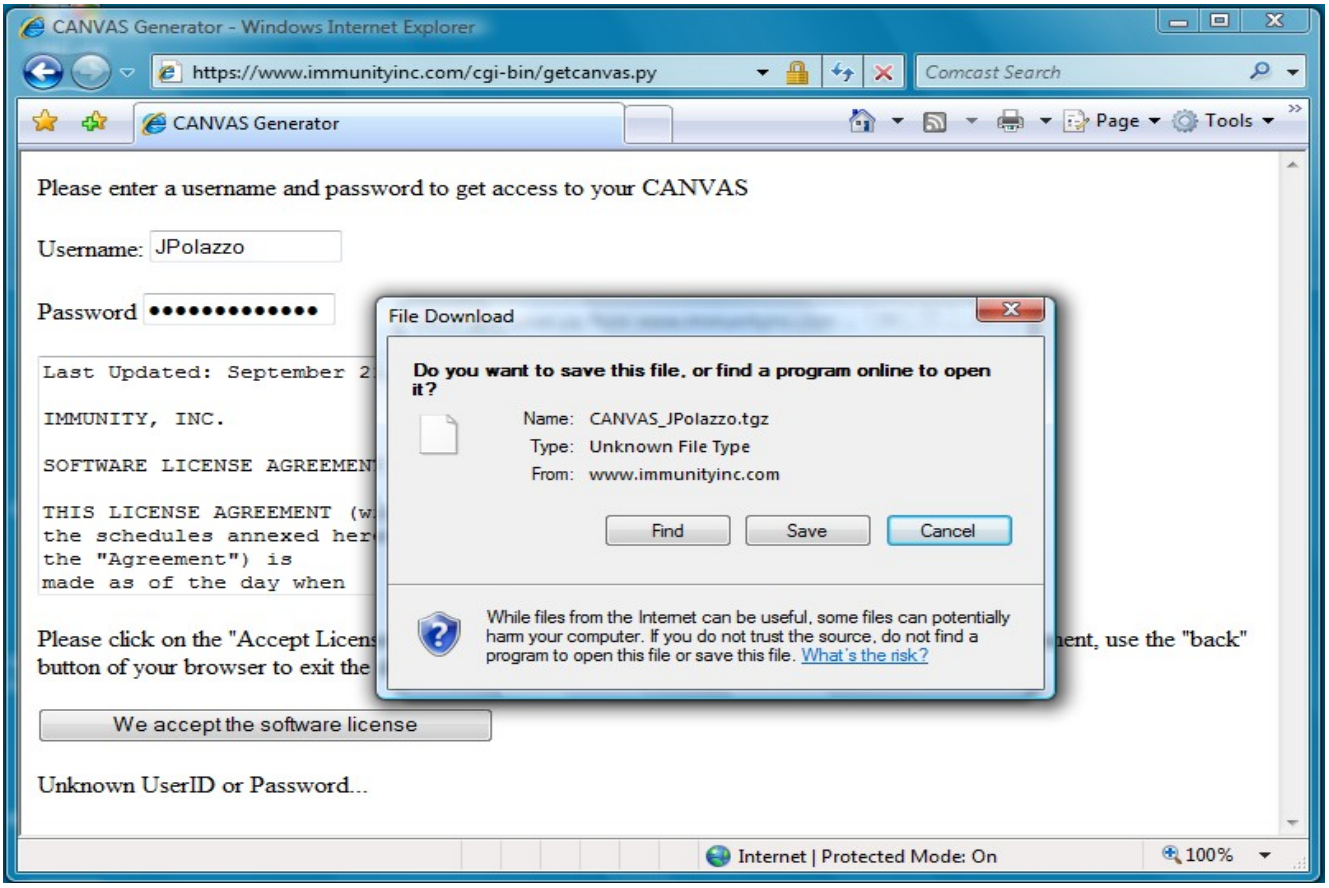
To obtain CANVAS, simply send an email to sales@immunityinc.com with your contact information and we will get in touch to arrange the creation of your immunityinc.com login UserID and Password, which will get you the latest version of CANVAS, its prerequisites, and future updates to the software.

Next, lets take a look at the prerequisites for installing CANVAS on Microsoft Windows Vista:

Prerequisites: Aside from the CANVAS application itself, all of the following prerequisites are free of charge and, with the exception of the Archive Manager, available for download from ImmunityInc.com

Prerequisite #1 The CANVAS Tarball (A "tarball" is a file with the .tar.gz or .tgz extensions – like a .zip file, essentially).

After getting your UserID and Password from Immunity, point your browser to https://www.immunityinc.com/cgi-bin/getcanvas.py. Enter in your information and agree to the software license to continue with the download process.



Watermarking uniquely identifies each copy of CANVAS to the company that purchased it.

Prerequisite #:2 an Archive Manager

The first step in installing CANVAS is to obtain a program that can unpack the tarball. This brings us to our second prerequisite:

- **WinZIP or WinRAR**

After installing an Archive Manager, extract the CANVAS_YOURNAME.tgz file to a directory of your choosing and then open it. Contained within are the next prerequisites for running CANVAS:

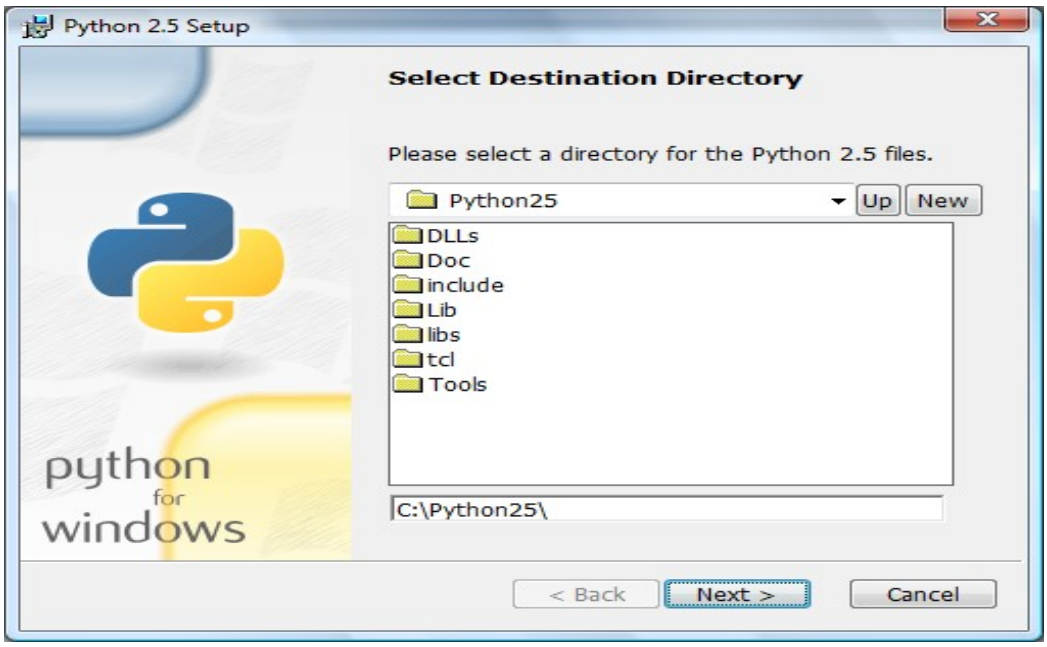
Prerequisite #:3 The PyGTK Tarball

Point your browser to: https://www.immunityinc.com/downloads/PyGTK.tar.gz. This tarball contains all of the other dependencies for CANVAS to run on Windows Vista/XP.

- **The Python Programming Language** Version 2.5
- **GTK** aka Graphical Tool Kit
- **PyGTK** the Python bindings for the Graphical Tool Kit
- **PyCairo** the Python bindings for the Cairo Image Library
- **PygObject** The Python bindings for the Gobject Library

First Lets install Python by double clicking on the Python-2.5.msi package in your PyGTK folder:

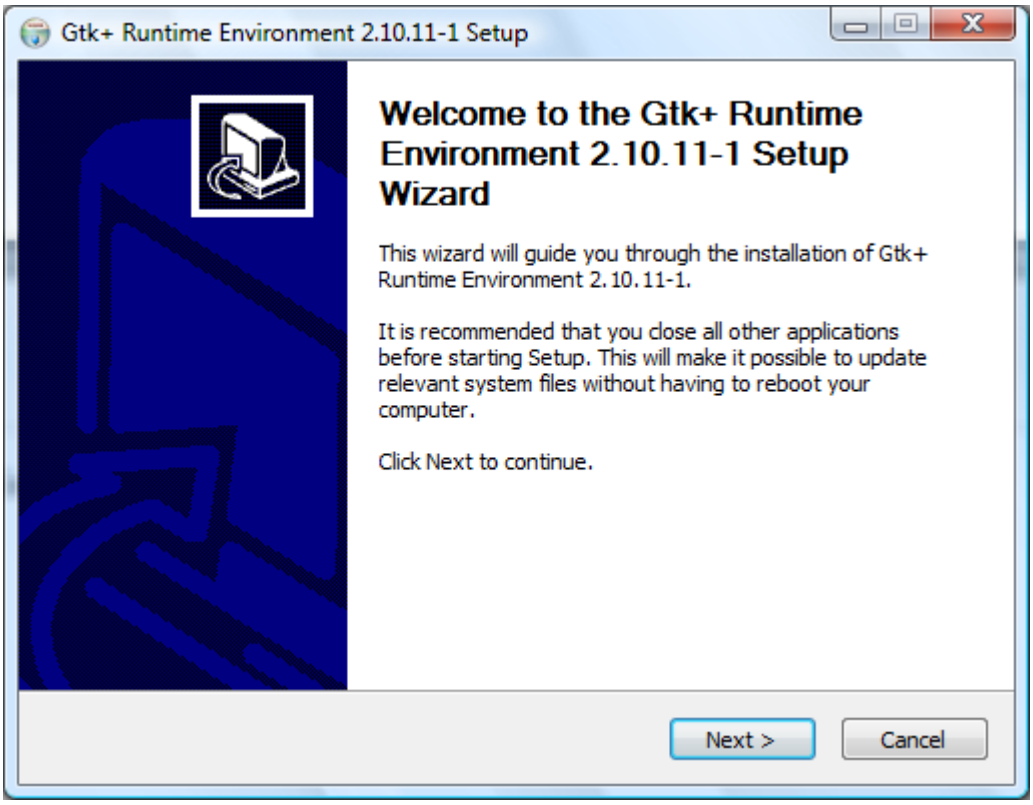
Select Next



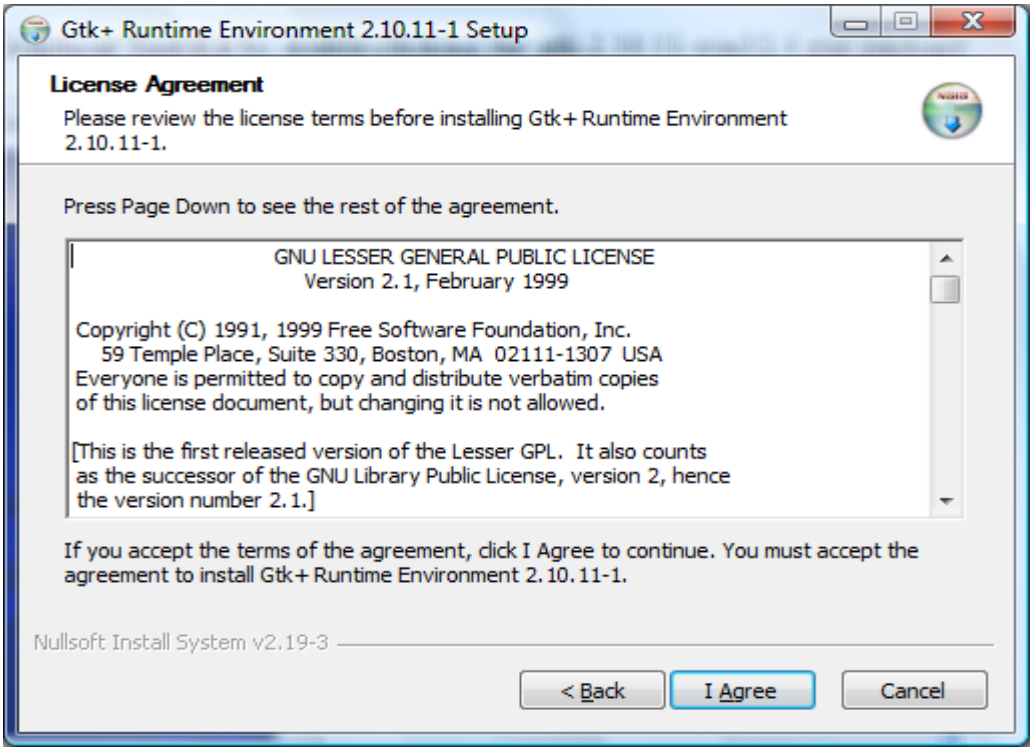
Select Next

Select Next, then Finish

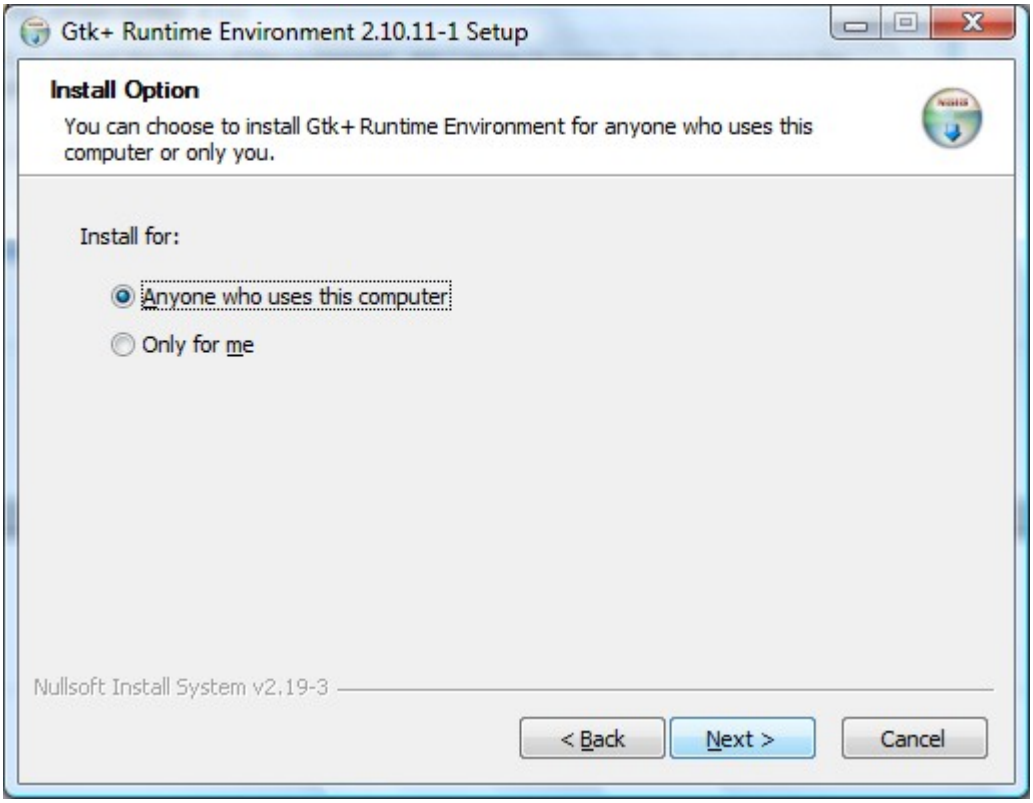
Next, install the Graphical Tool Kit by double-clicking the gtk-2.10.11-win32-1.exe package



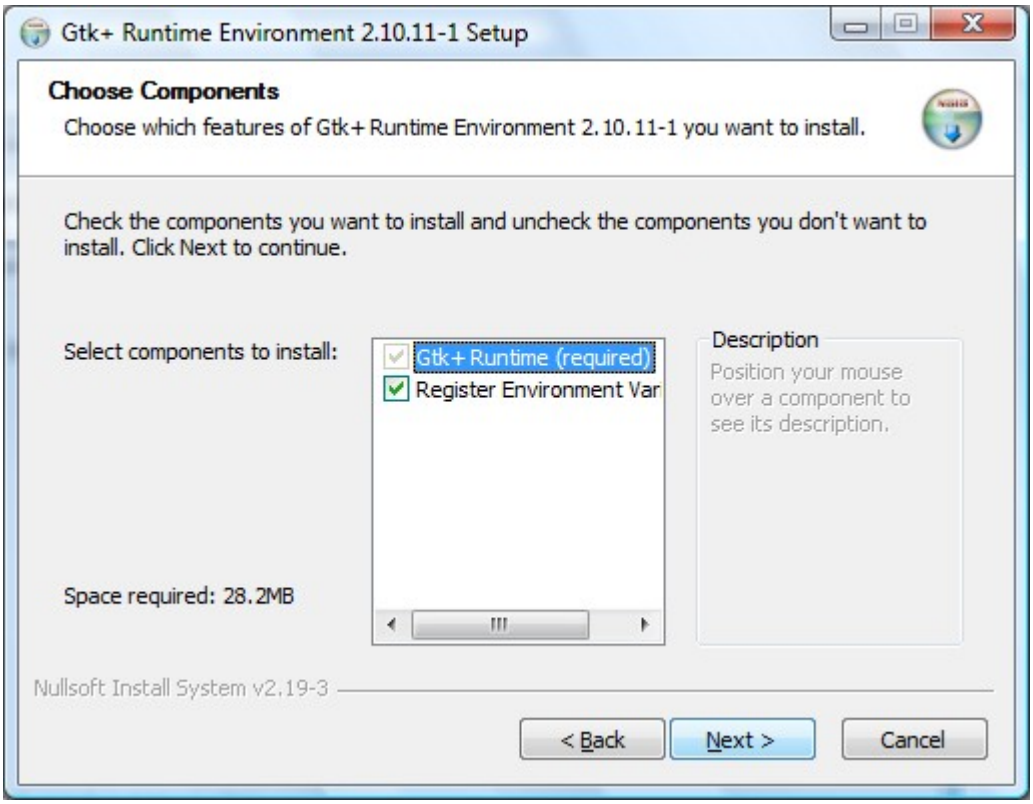
Select Next



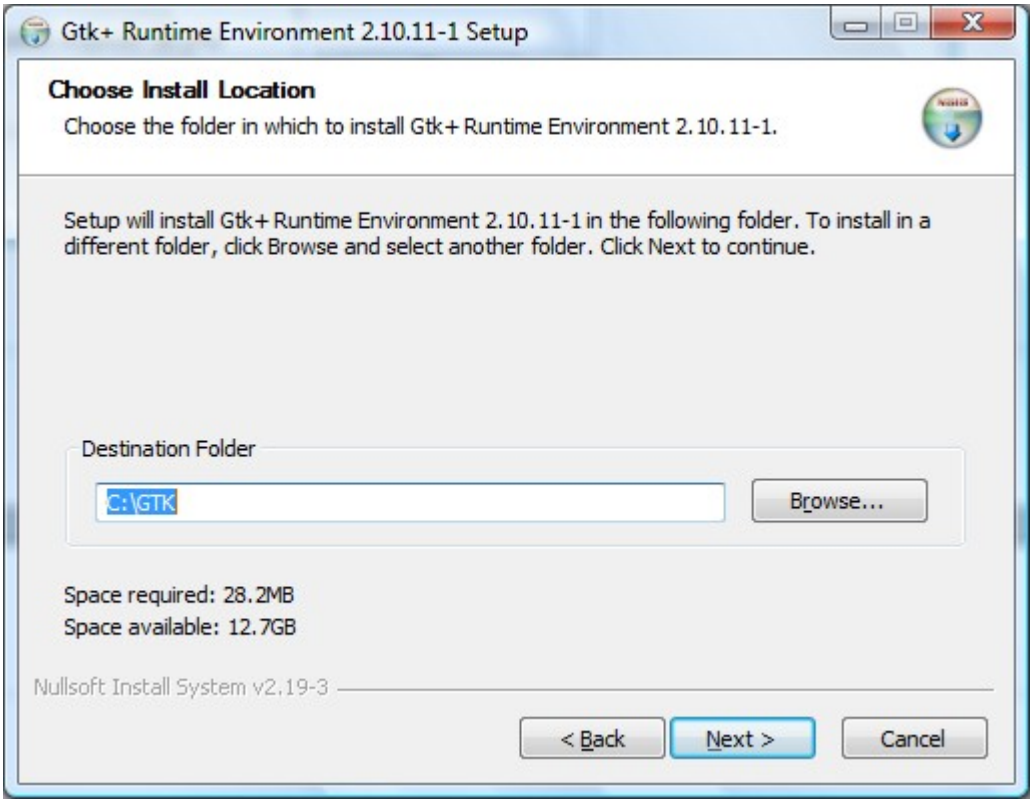
Select I agree



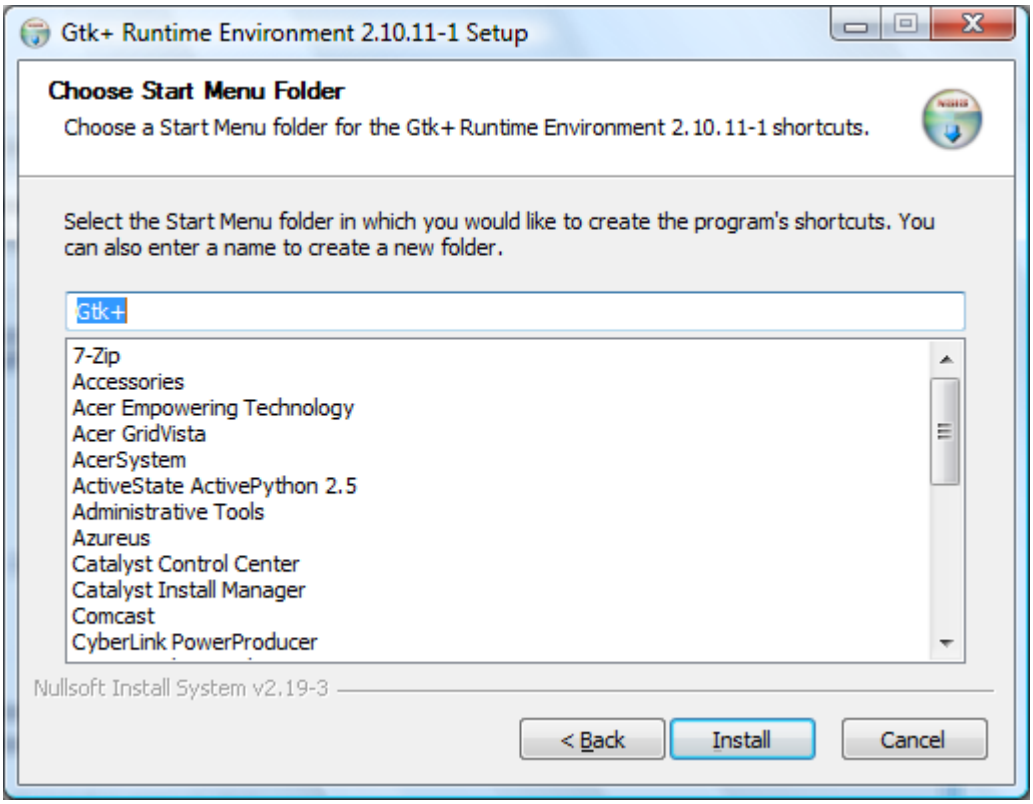
Select Next



Select Next

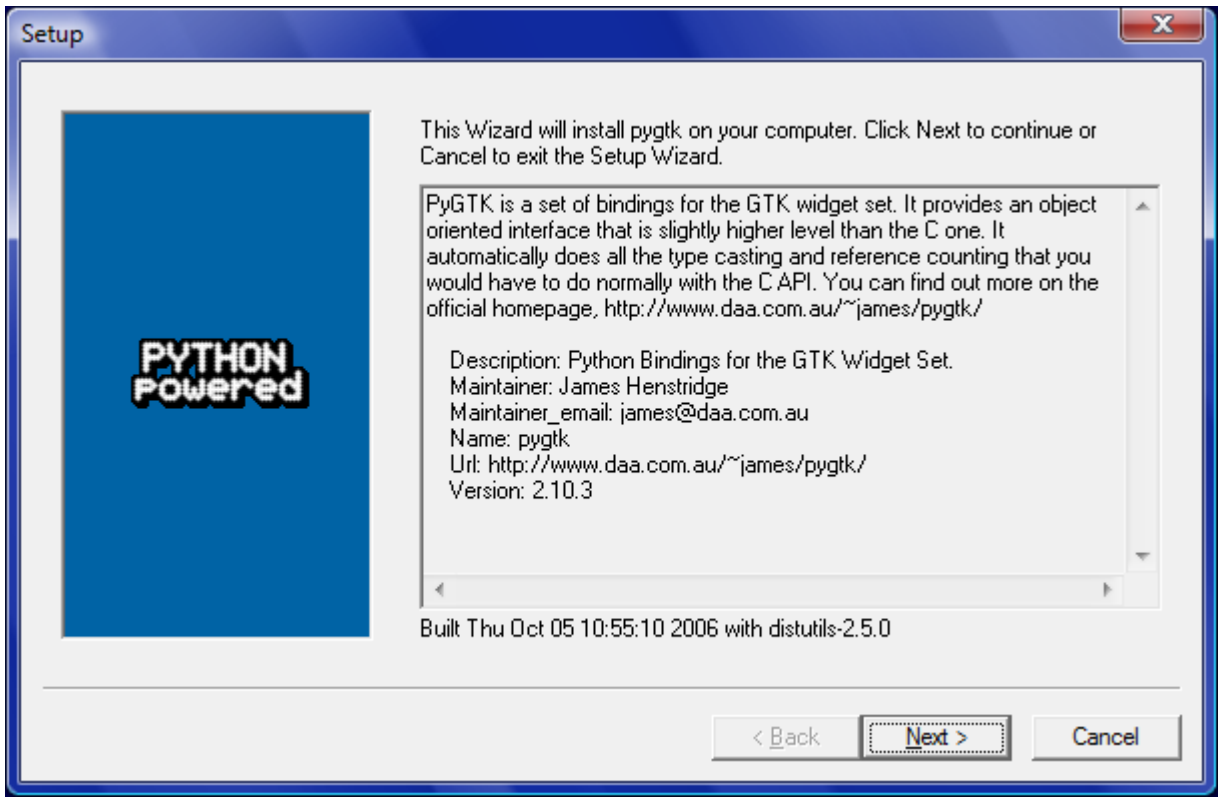


Select Next

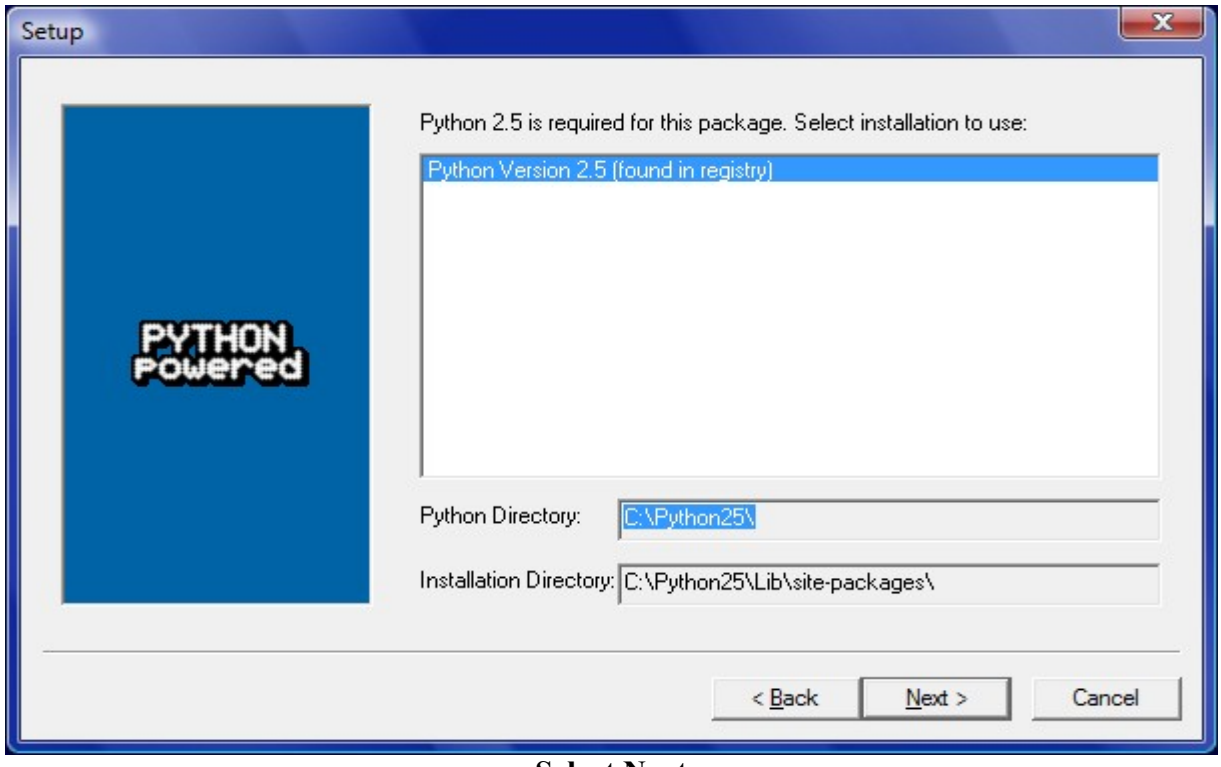


Select Install

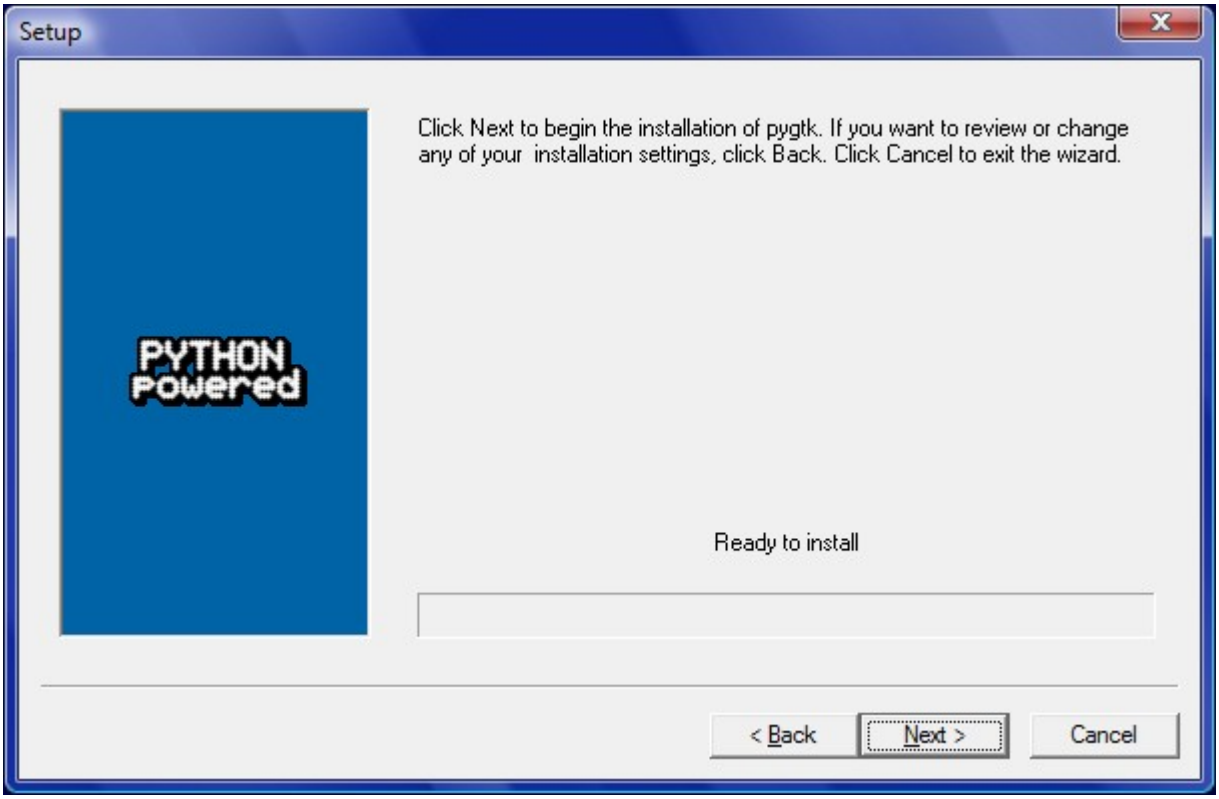
Next, Install the Python GTK bindings by clicking on the pygtk-2.10.3-1.win32-py2.5.exe package



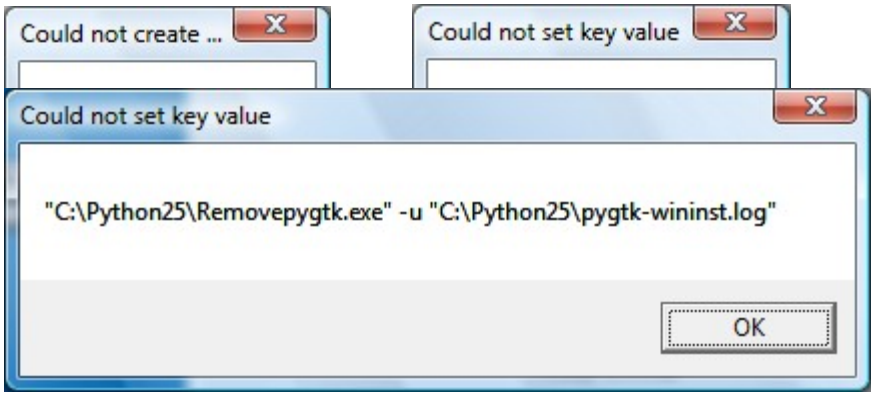
Select Next



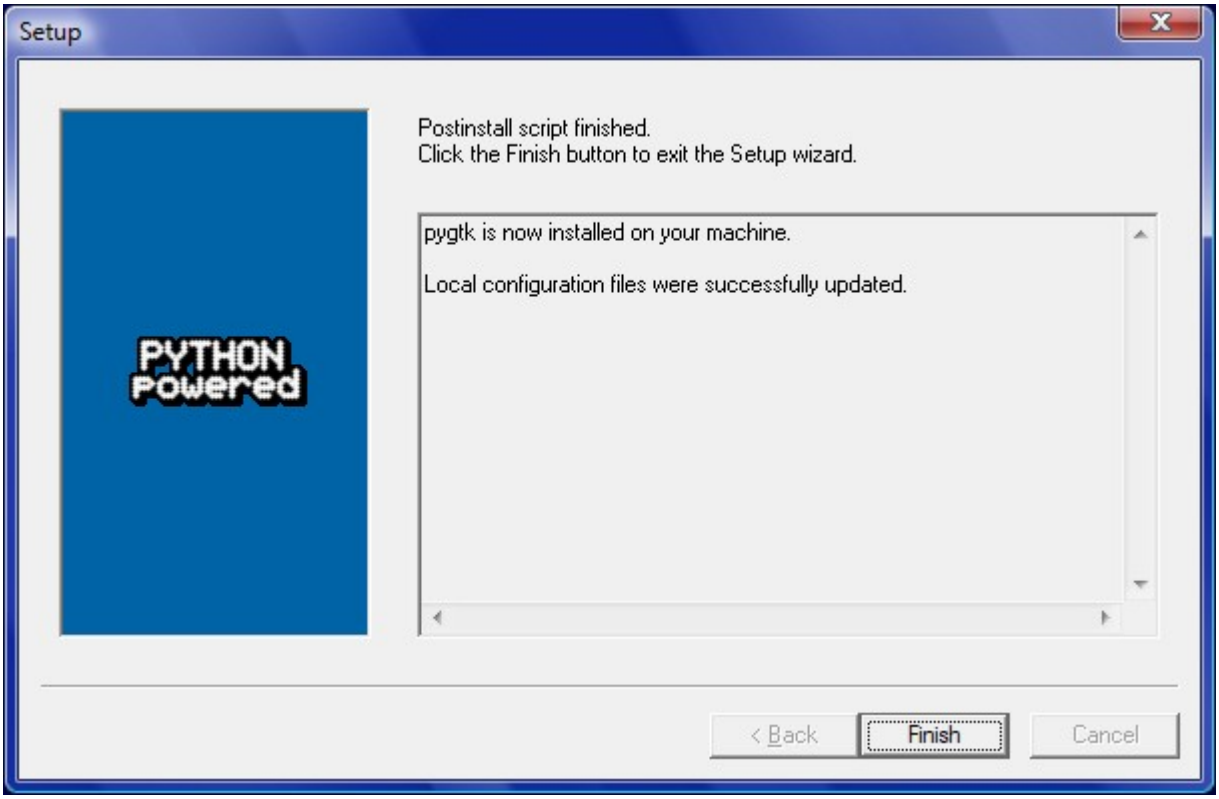
Select Next



Select Next

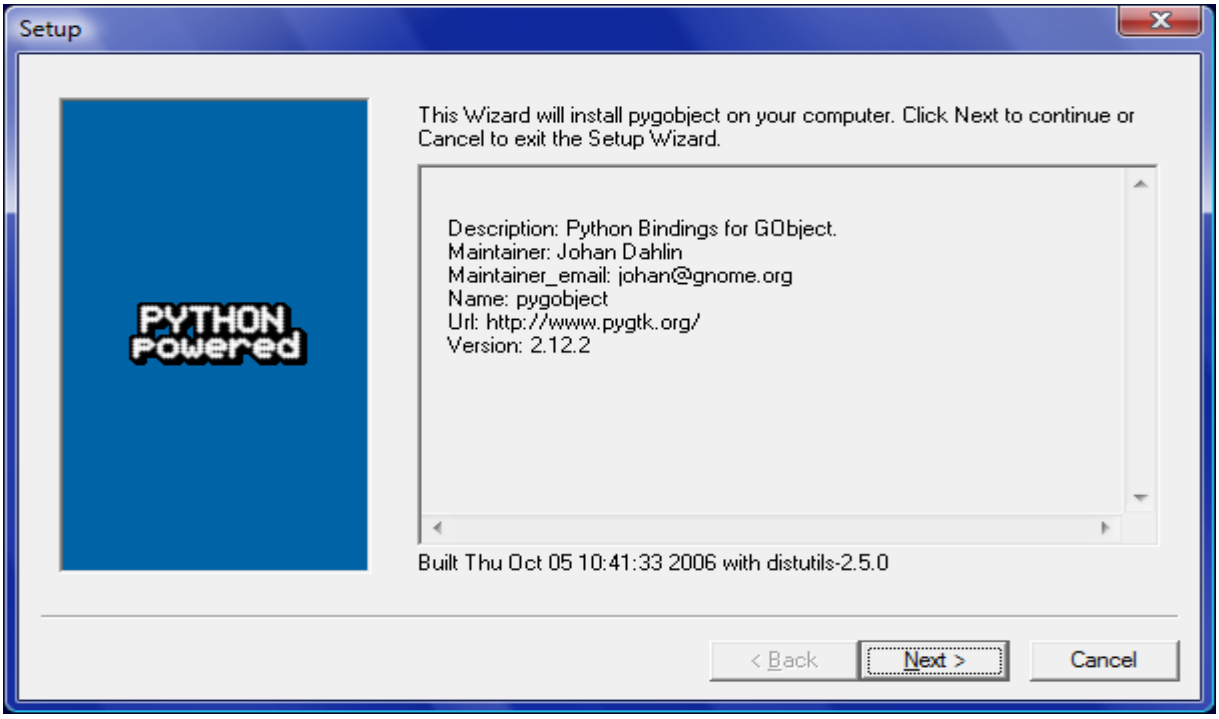


You can safely ignore these three errors

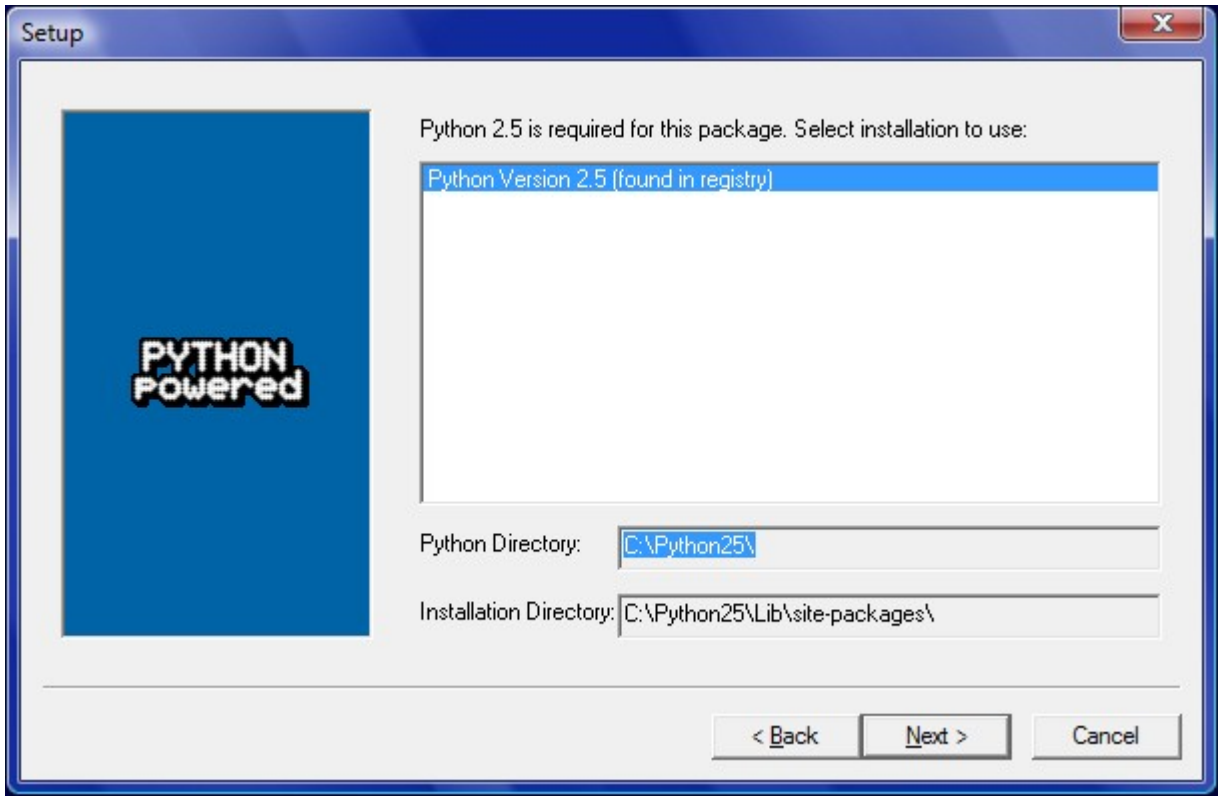


Select Finish

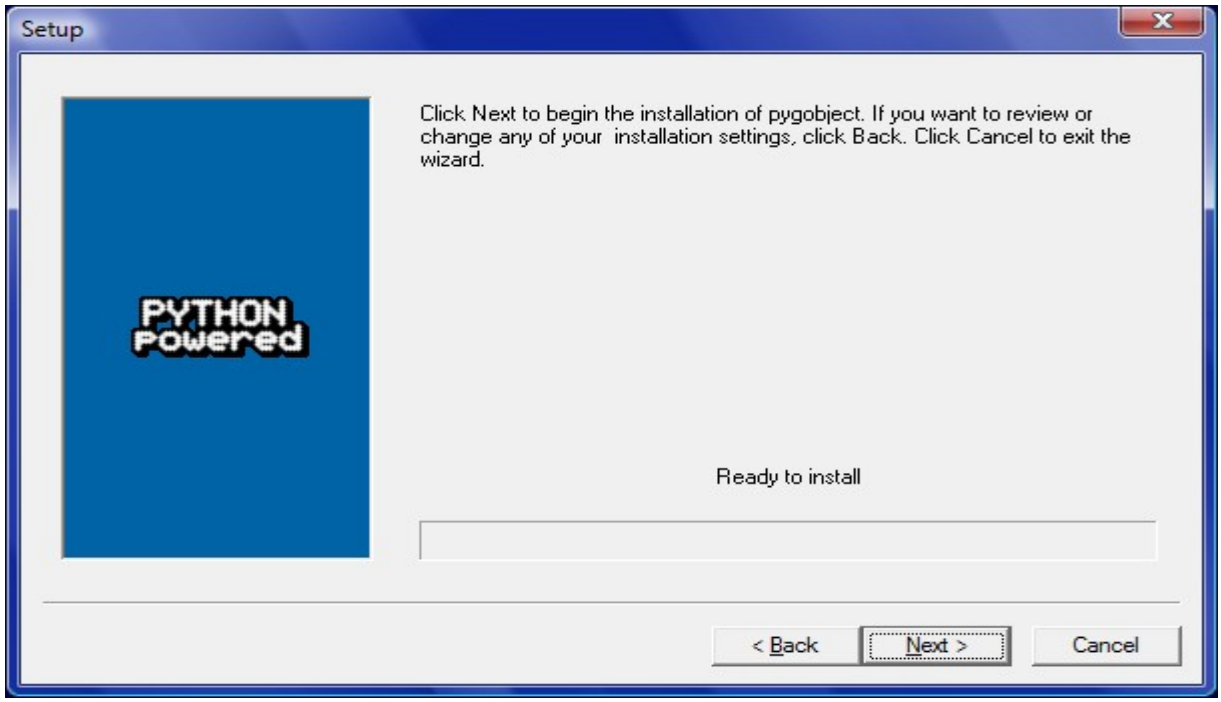
Next, install the Python Gobject bindings by clicking on the pygobject-2.12.2-1.win32-py2.5.exe Package:



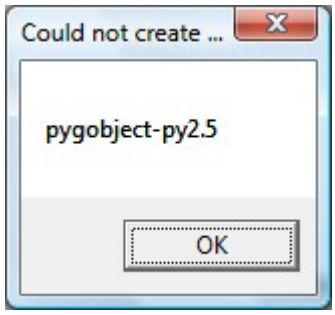
Select Next

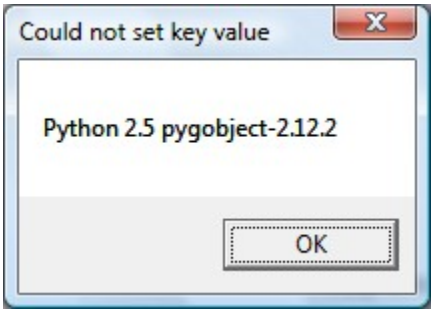


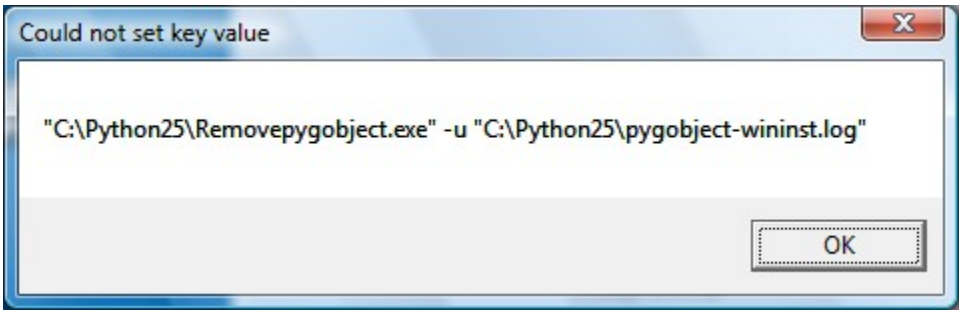
Select Next



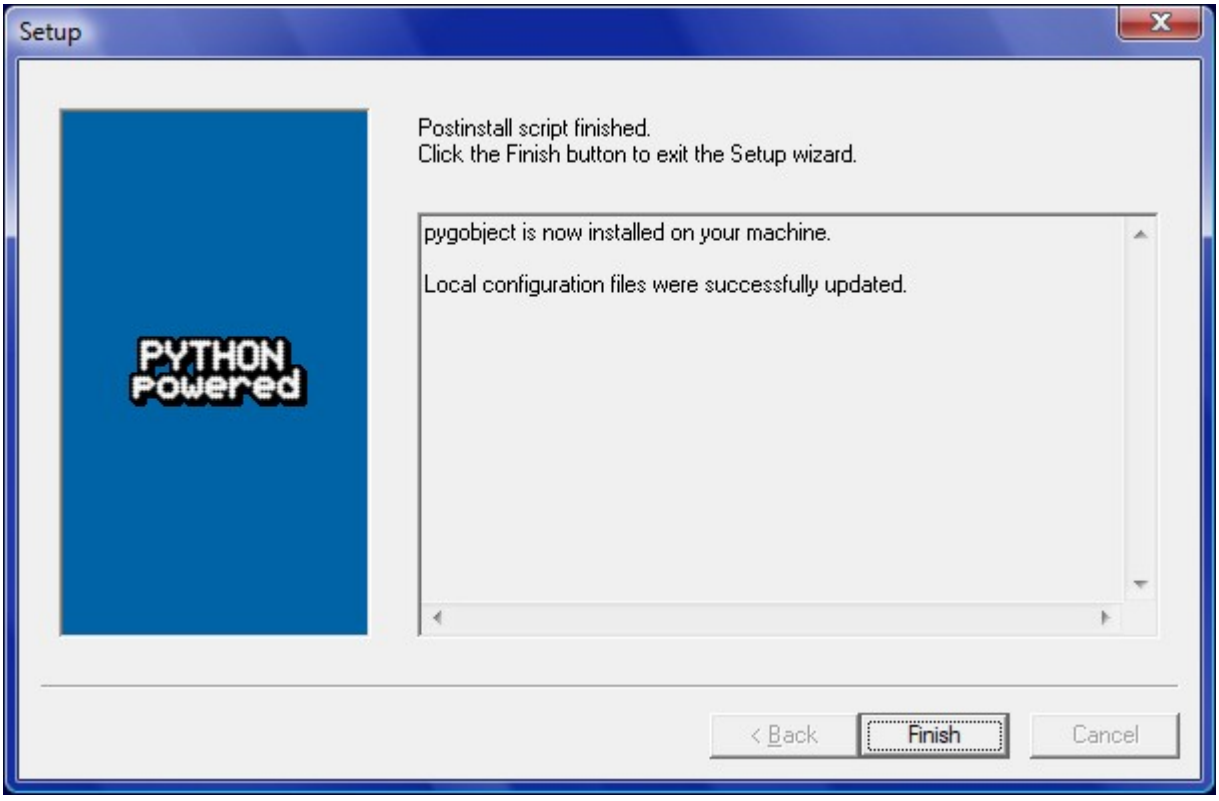
Select Next







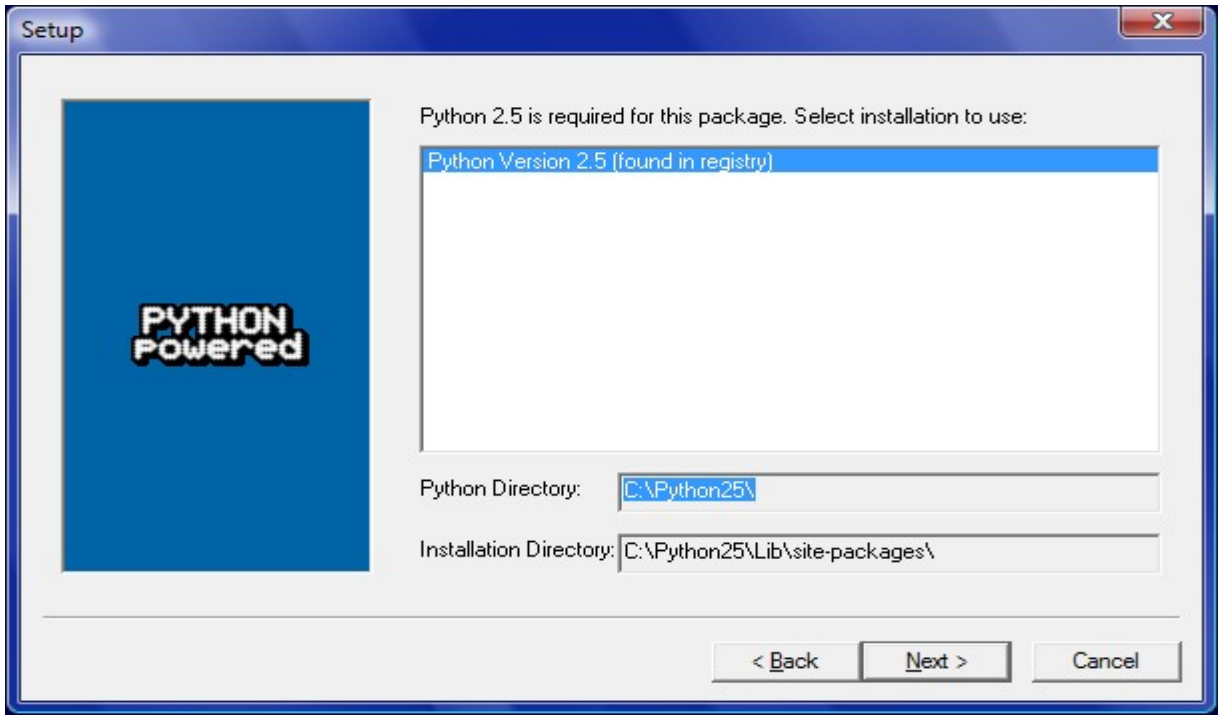
You can safely ignore these three errors



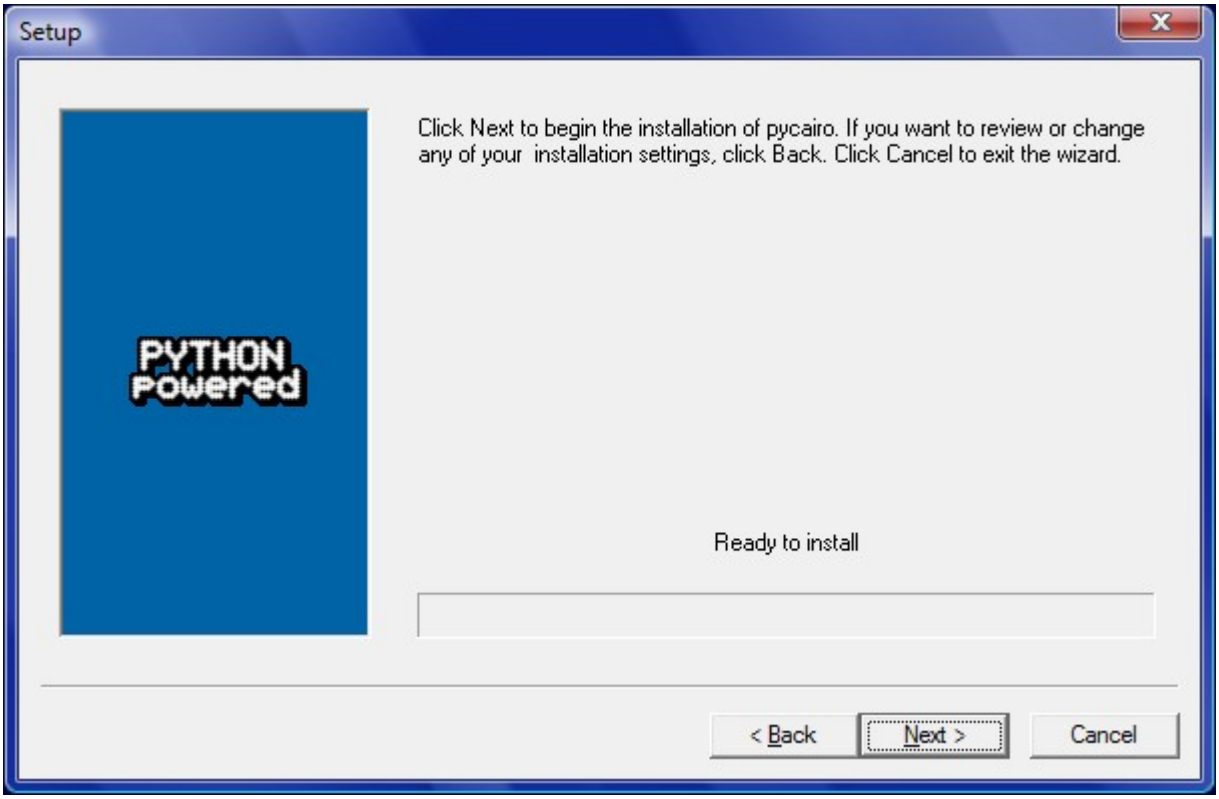
Select Finish

Finally, install the Python Cairo Image Bindings by clicking on the pycairo-1.2.6-1.win32-py2.5.exe package

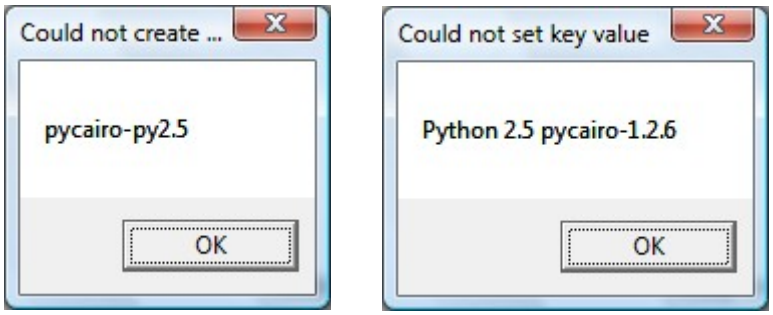
Select Next

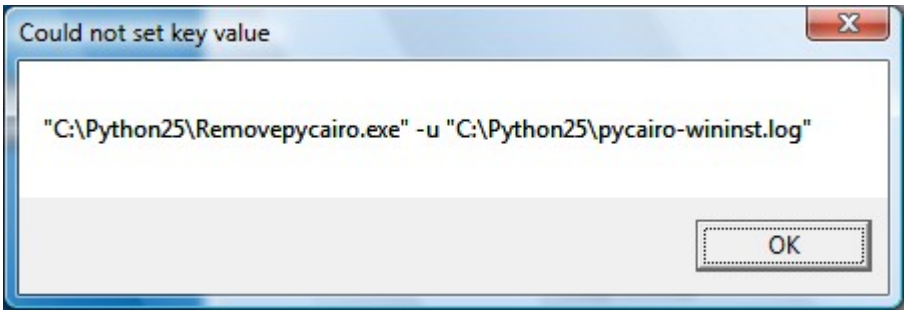


Select Next

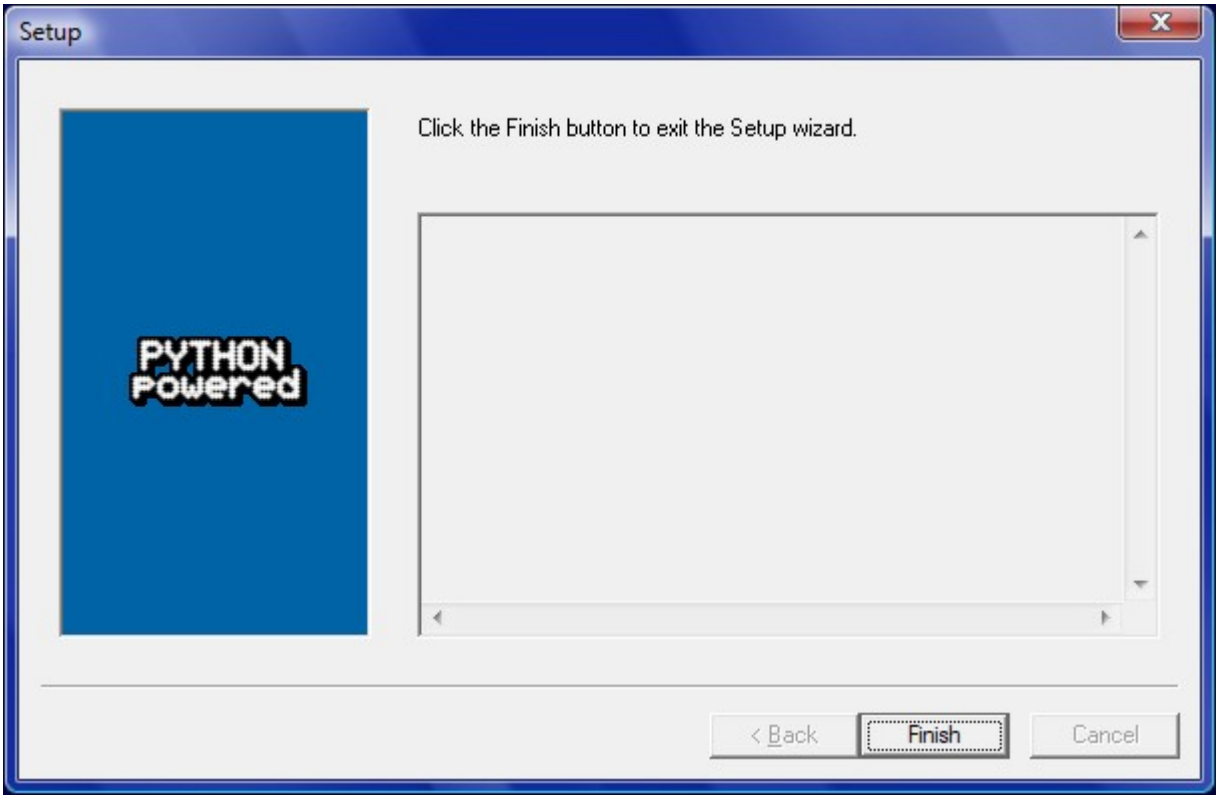


Select Next



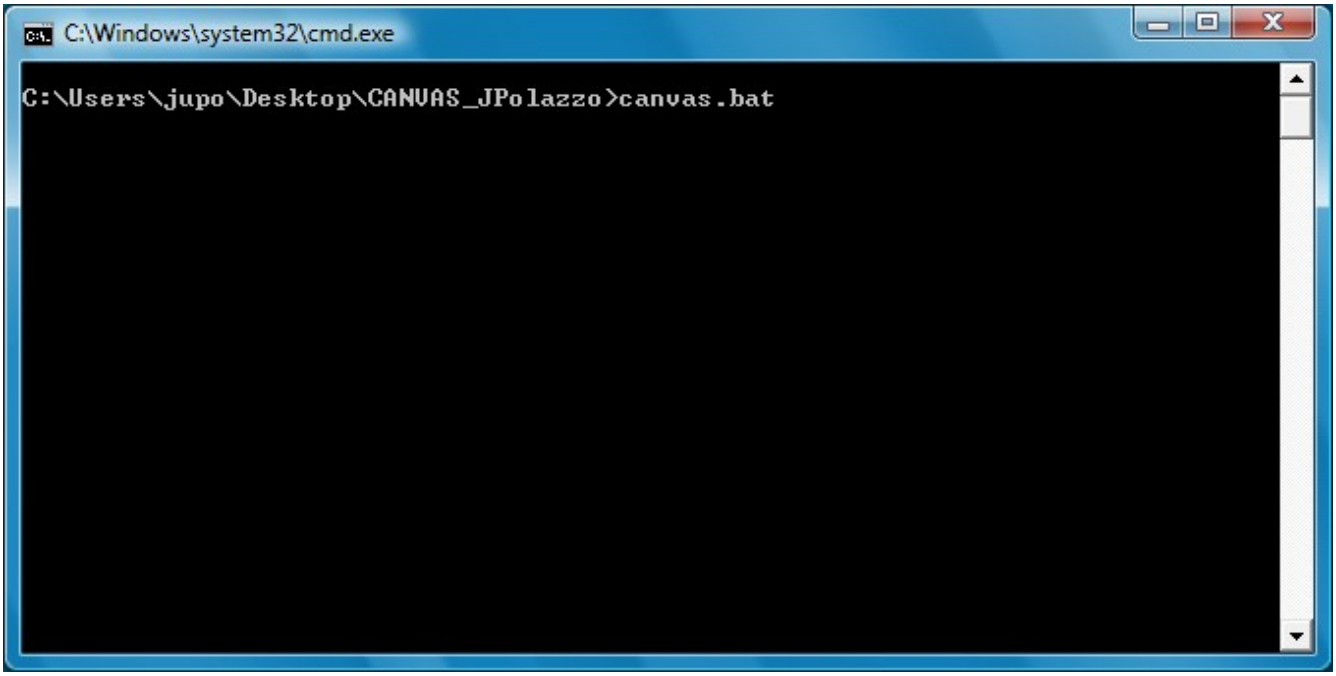


You can safely ignore these three errors

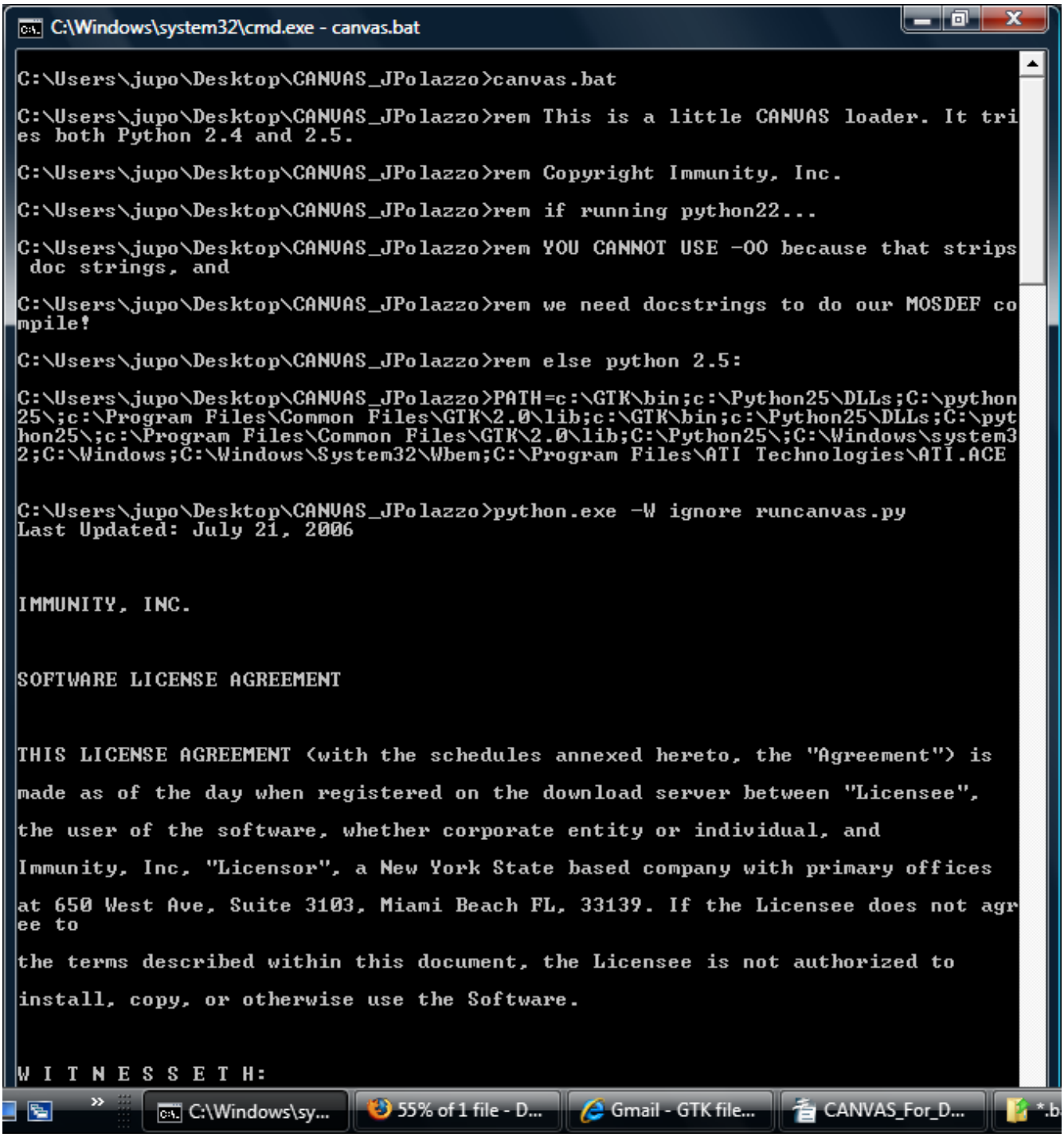


Select Finish

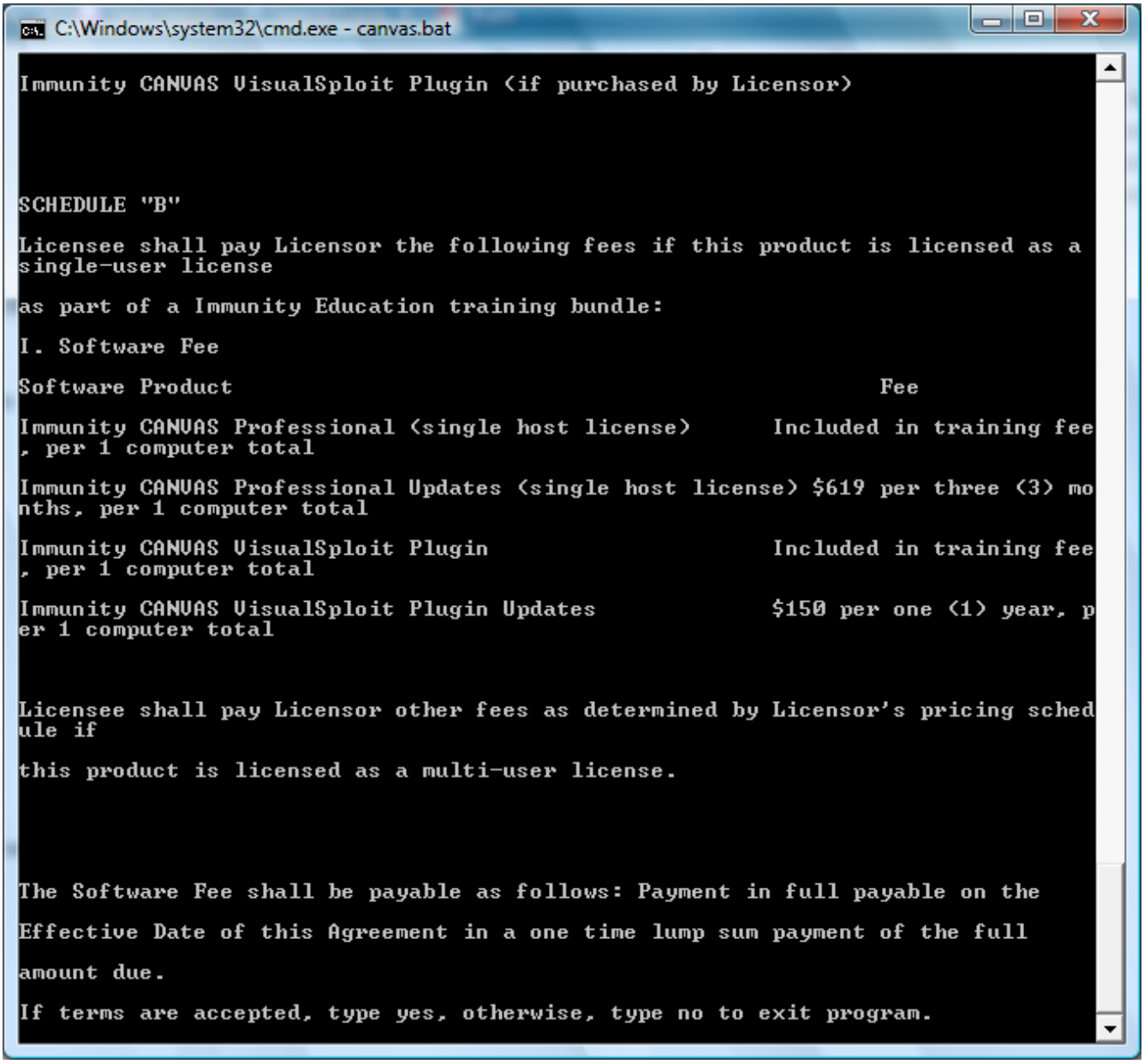
Now that we have taken care of the prerequisites, start cmd.exe and cd into where you extracted your CANVAS_<companyname>.tgz file and type in canvas.bat



Review the license agreement by pressing enter until you reach the end

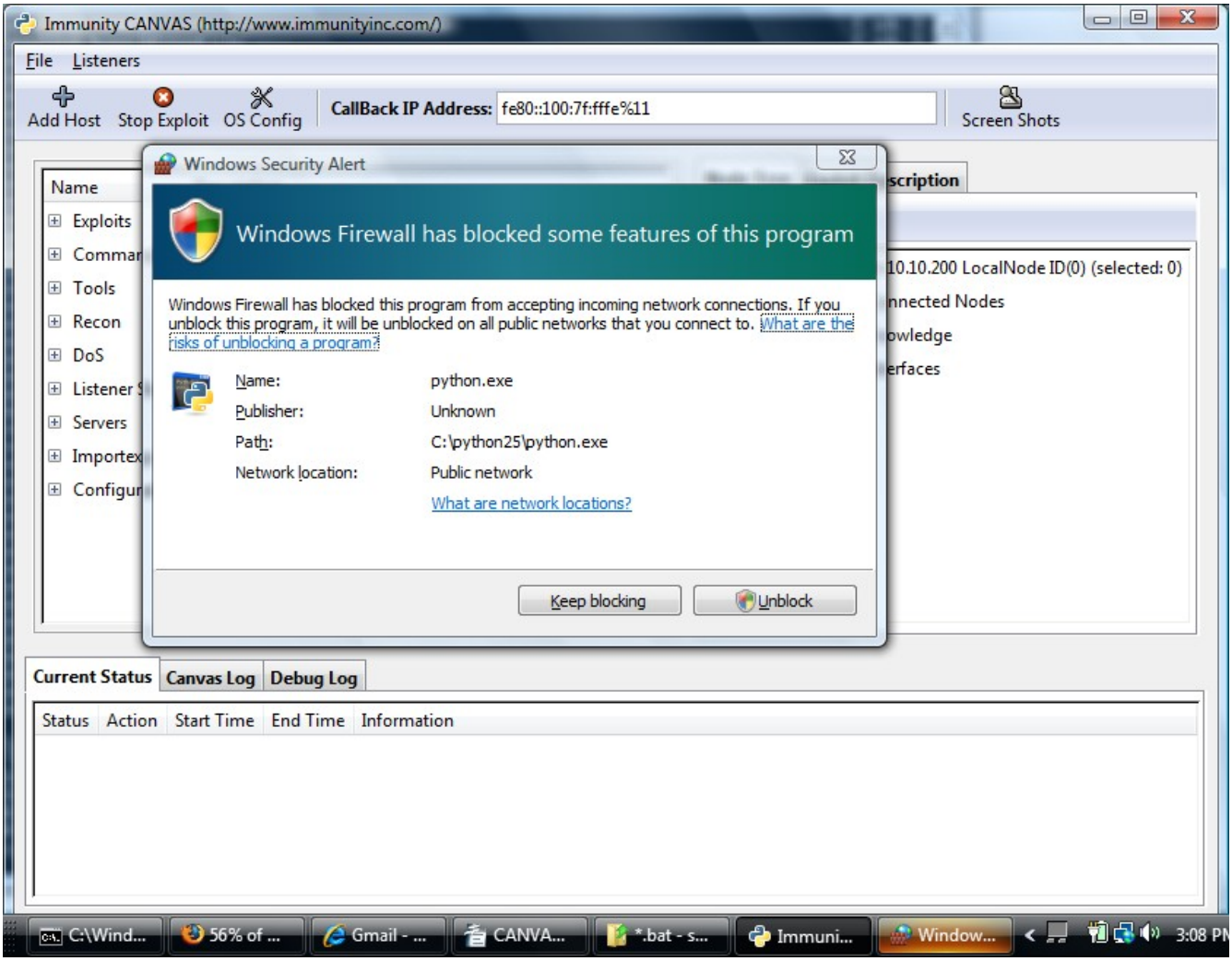


Hit enter until you reach the prompt shown below:

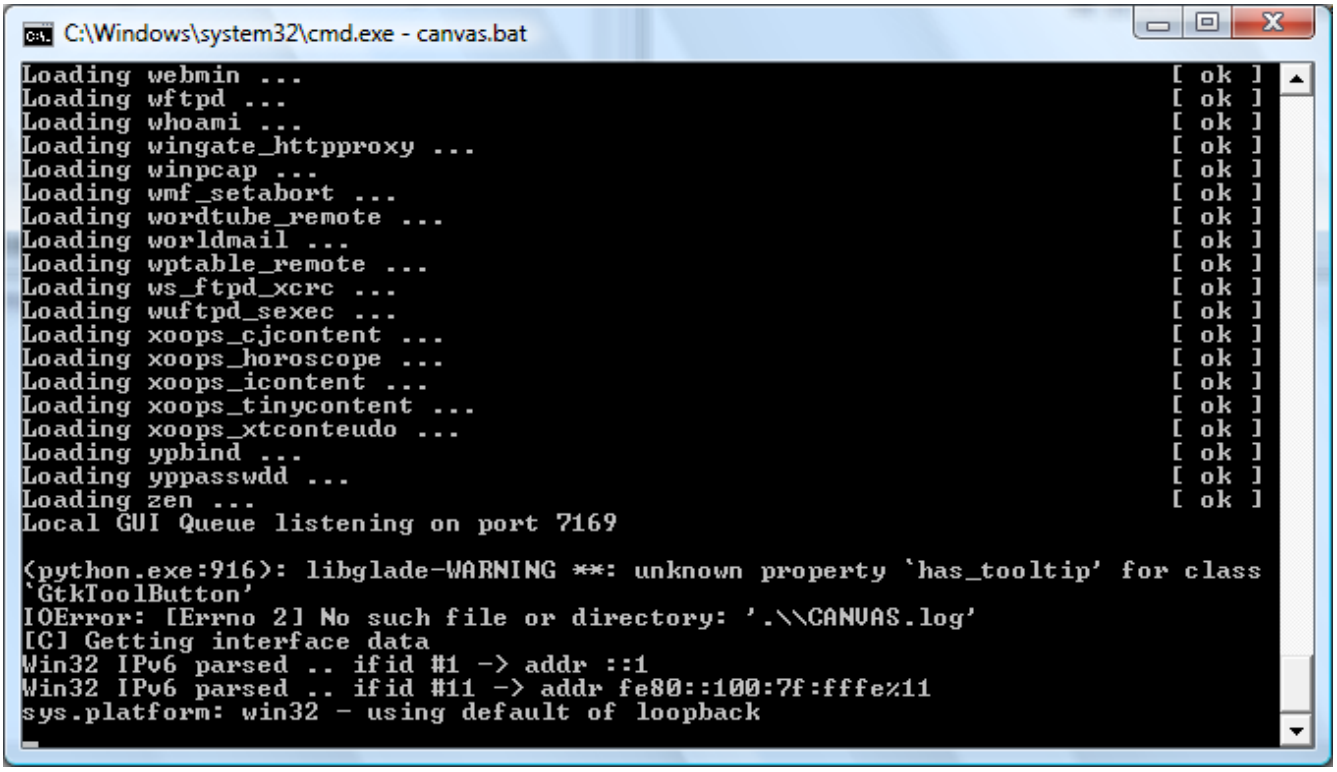


Type "yes" if you agree, otherwise , type "no" to exit the installer (and this tutorial!)

If you said yes, the batch file will load CANVAS. If you have not yet disabled your windows firewall, you will get a message like the one on the page below:



Windows Firewall needs to be disabled for CANVAS to function correctly. CANVAS sets up listeners on TCP ports of your Callback IP Address that compromised hosts will attempt to contact. It is through this method of communication that we are able to control their actions. Once again, if Windows Firewall is turned on CANVAS will not function correctly.

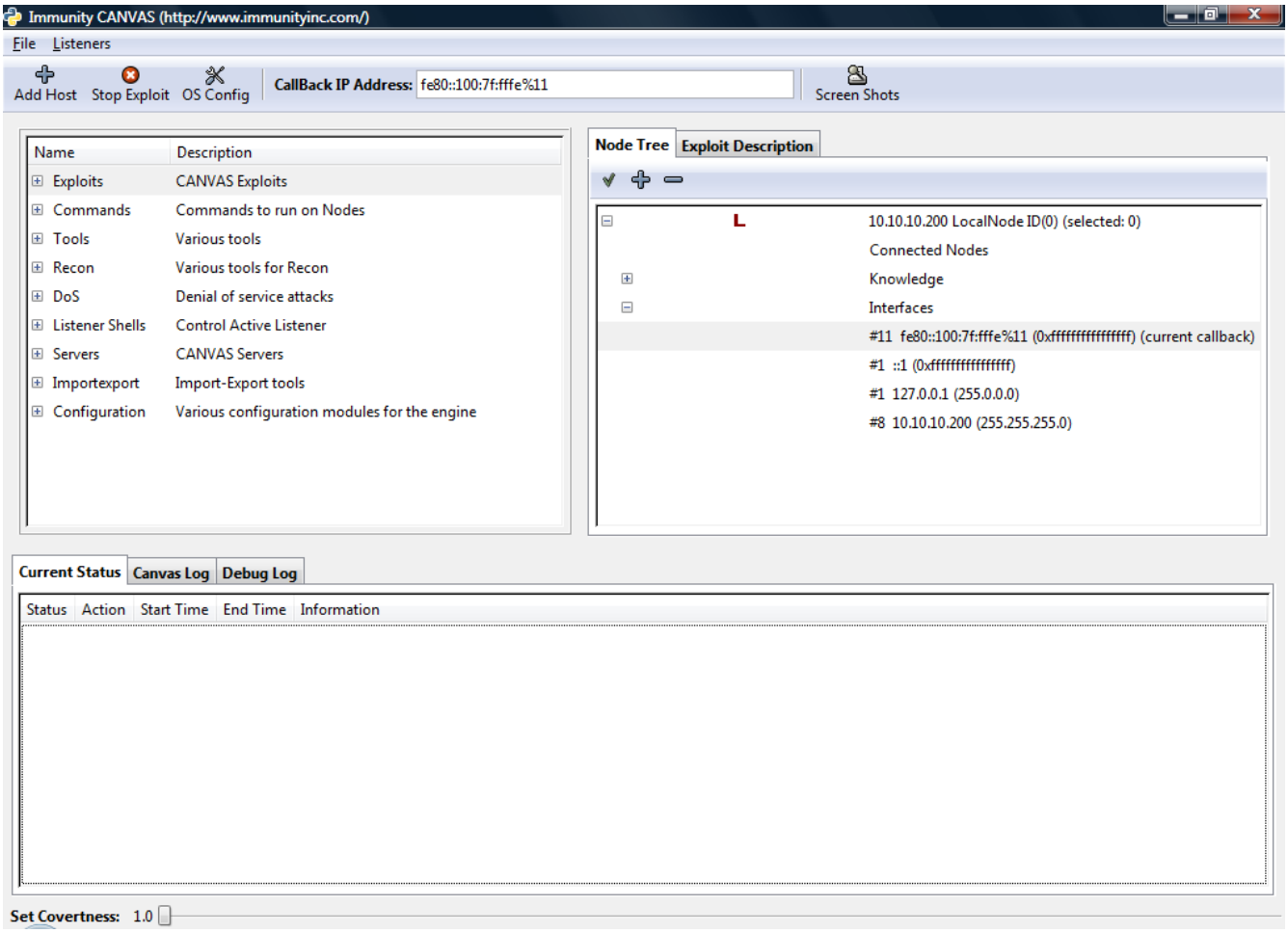


Don't close canvas.bat! This batch file has loaded all the Python modules and scripts required for the CANVAS GUI to function. Closing this window will exit the program. This window also shows each action taken by CANVAS as it happens. For convenience , everything displayed in this window is also written to your canvas.log file located in the root of your CANVAS program directory (despite the warning above ;-). This log clears itself out after reaching a size of 1 MB in order to prevent CANVAS scans from filling up your hard drive. Be aware of this feature and save off the canvas.log file regularly you wish to preserve this data.

Congratulations! CANVAS is now installed and running, and we have reached the end of the installation and the end of the first chapter. This chapter covered installing CANVAS on Windows Vista, which is still a fairly new operating system. If you ran into any unexpected errors, please email support@immunityinc.com with a description of your error and a screenshot attached if possible. We will respond as soon as humanly possible.

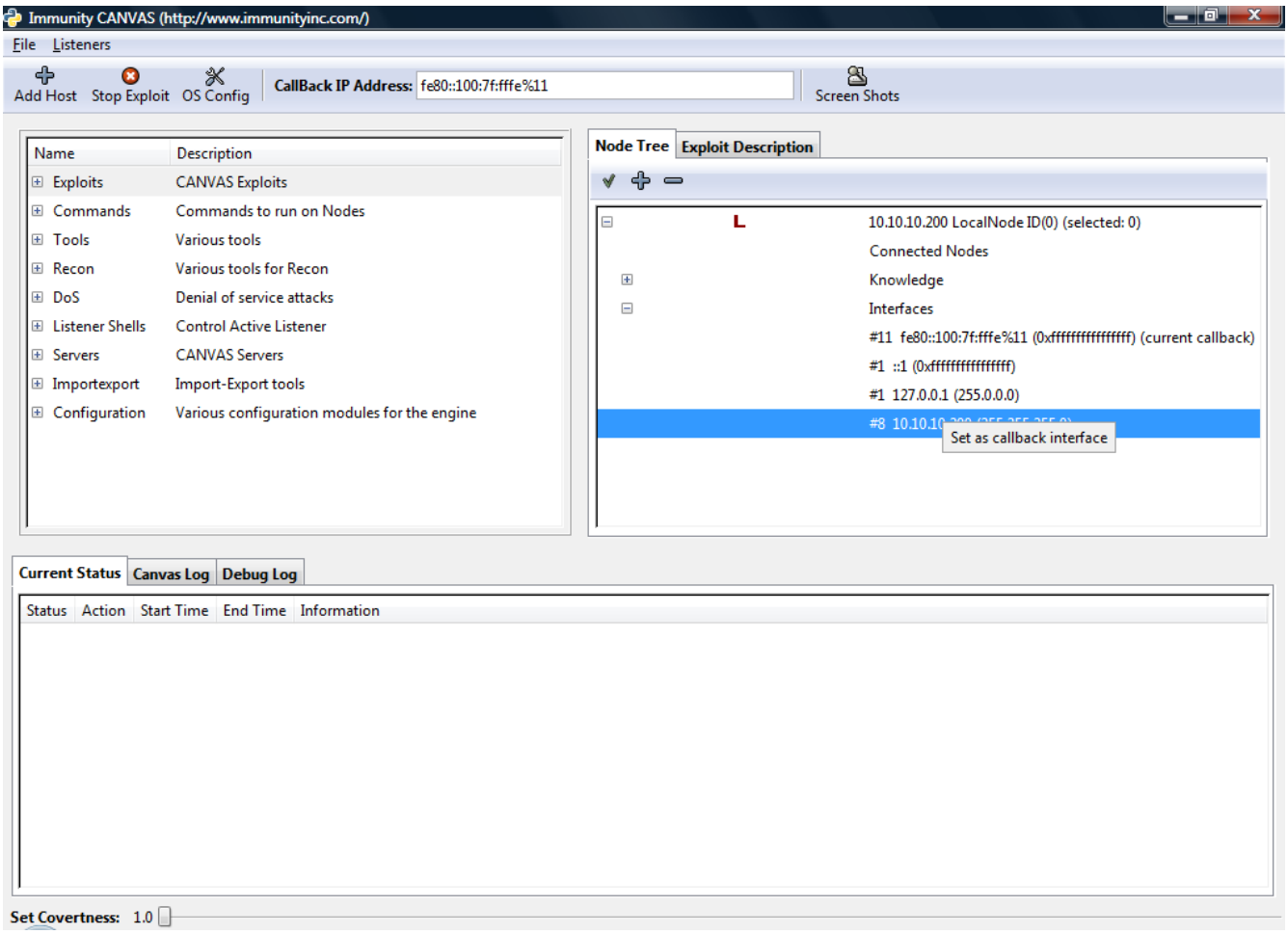
Learning the Immunity CANVAS Layout

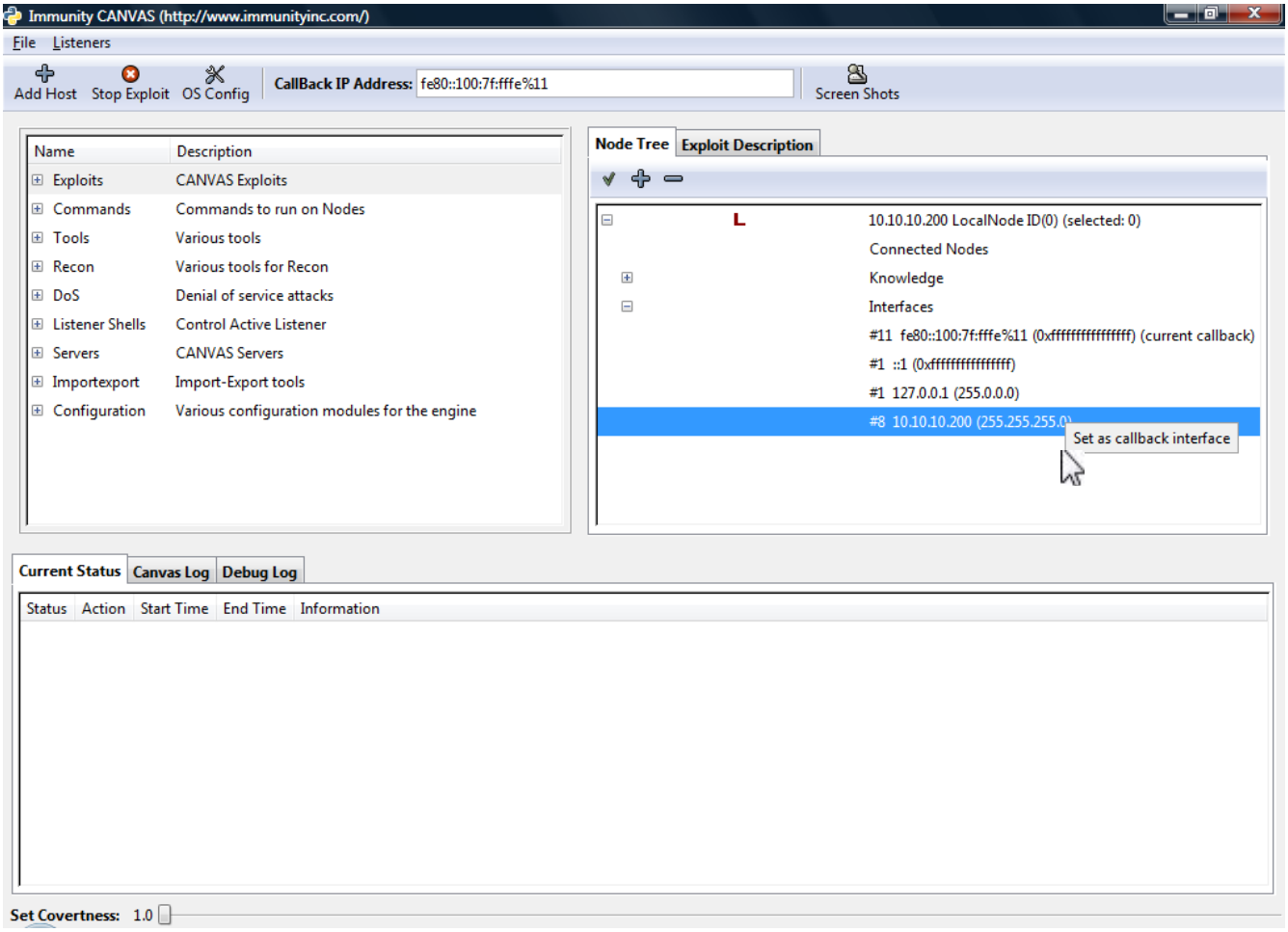
You should now see something similar to the window below:



If you closed this window for some reason, you can always get it back by CD'ing into your CANVAS directory, and then running canvas.bat. It might make your life easier if you create a shortcut to this batch file and put it on your desktop.

***note**
You may notice that on Windows Vista, the GUI sets the default callback address to Ipv6. You can fix this by highlighting the IP address from the Interfaces section of the Knowledge Window you want compromised hosts to call back to with a left click, and after the correct one is highlighted, right-click and then select “Set as callback interface” as shown below. Make sure your Callback interface is set to an IPv4 address if you are not in an IPv6 environment

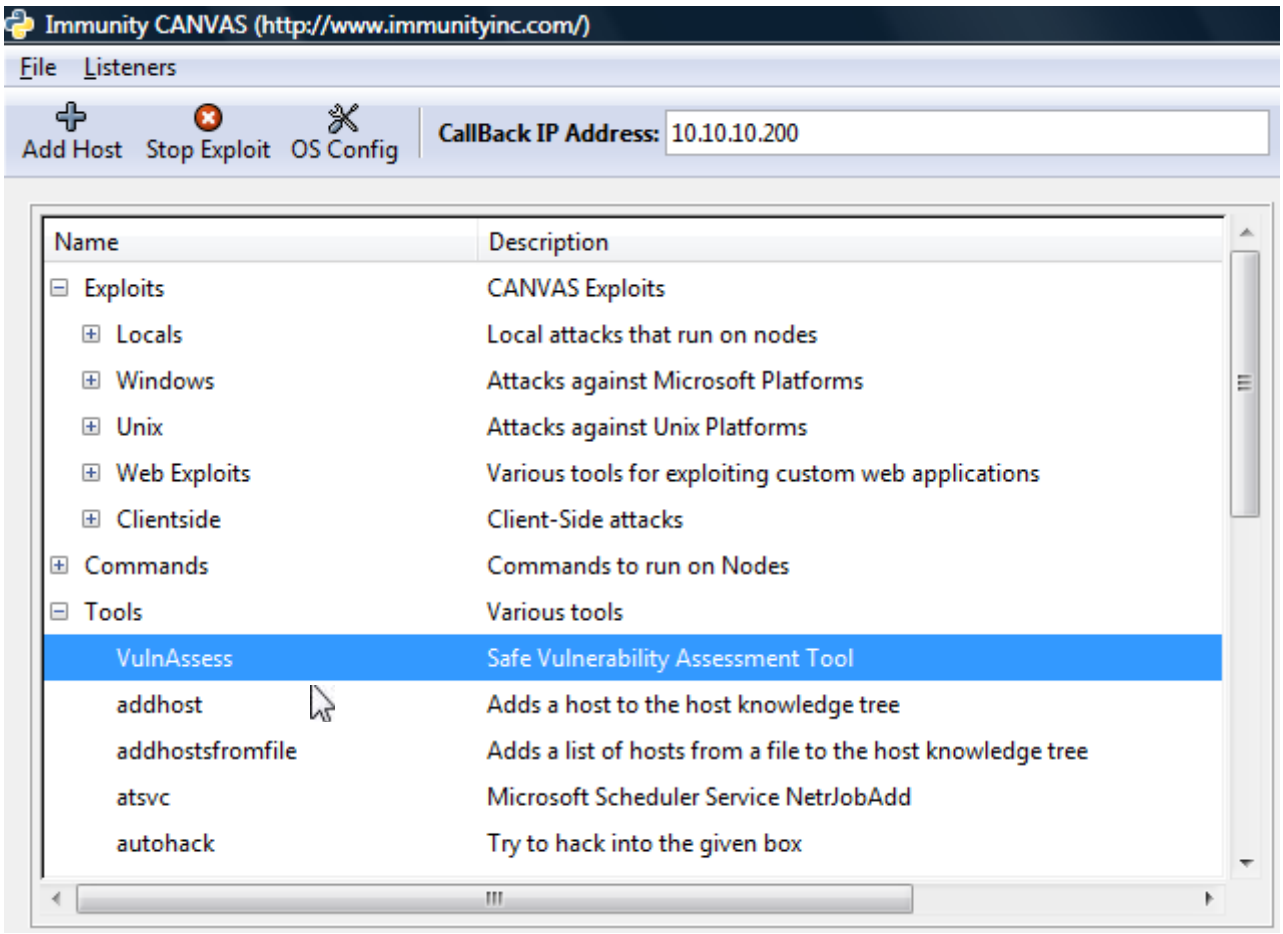

Immunity CANVAS (http://www.immunityinc.com/)
File Listeners
Add Host
Stop Exploit
OS Config
CallBack IP Address: fe80::100:7f:fffe%11
Screen Shots
Name
Description
Exploits
CANVAS Exploits
Commands
Commands to run on Nodes
Tools
Various tools
Recon
Various tools for Recon
DoS
Denial of service attacks
Listener Shells
Control Active Listener
Servers
CANVAS Servers
Importexport
Import-Export tools
Configuration
Various configuration modules for the engine
Node Tree
Exploit Description
10.10.10.200 LocalNode ID(0) (selected: 0)
Connected Nodes
Knowledge
Interfaces
#11 fe80::100:7f:fffe%11 (0xffffffffffffffff) (current callback)
#1 ::1 (0xffffffffffffffff)
#1 127.0.0.1 (255.0.0.0)
Set as callback interface
Current Status
Canvas Log
Debug Log
Status
Action
Start Time
End Time
Information
Set Covertness: 1.0



Now that you have set the correct callback interface, lets review the CANVAS GUI's three main frames: the Control Window, the Knowledge Window, and the Information Window. With these you can add hosts, set your callback IP, add a virtual interface (handy for NAT'ed external IP's), pick a host to attack, discover what exploits a host is vulnerable to, among many other things.

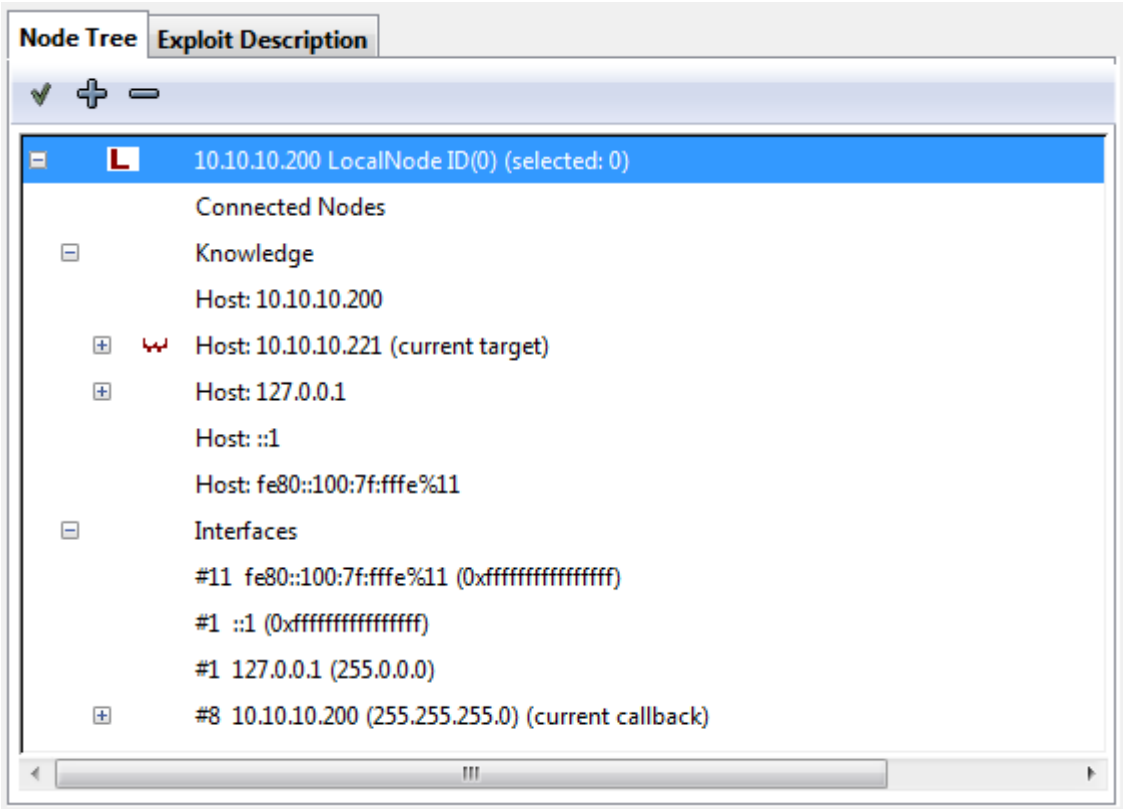
The Control Window

As you can see from the screenshot above, there are three main areas in the GUI. The top of the CANVAS GUI, on the left hand side (pictured Below) is what we will refer to as the Control Window. From this window, you can scan for vulnerabilities, launch exploits, and even "AutoHack". This is where all the network traffic will be generated from.



The Knowledge Window

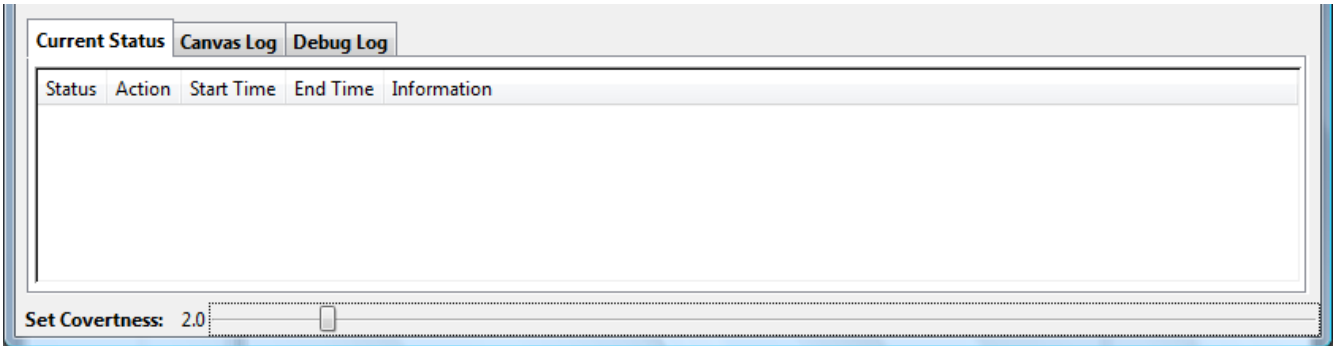
To the right of the Control Window is the area of CANVAS GUI that stores all the knowledge and information about hosts, networks, and potential targets gleaned from actions initiated by you from the Command Window. If you run a scan from the Control Window (lets say, the udpscan) and it detects 5 hosts, you will have five more entries appear in the window that I will refer to as the Knowledge Window (pictured below). This window also contains the information about your local network interfaces. From this window, you can also add virtual interfaces, such as a nat'ed external IP addess.



The Information Window

At the bottom of the CANVAS GUI is what I will refer to as the Information Window (pictured below). This window has three handy tabs: Current Status will let you know what CANVAS is doing at any given time; CANVAS Log is where you can see what is being written to the canvas.log file;. This file is important for many reasons, one of the most important being that there is a timestamp associated with each action taken. This allows you to validate or refute whether what the Server Admin sees in their log corresponds with the actions a CANVAS Admin had performed (system crash, etc). The logging

window is also handy to get a quick glance at how far along in a particular scan CANVAS is. The Debug Log will contain information about exceptions and errors that may arise while performing a scan.

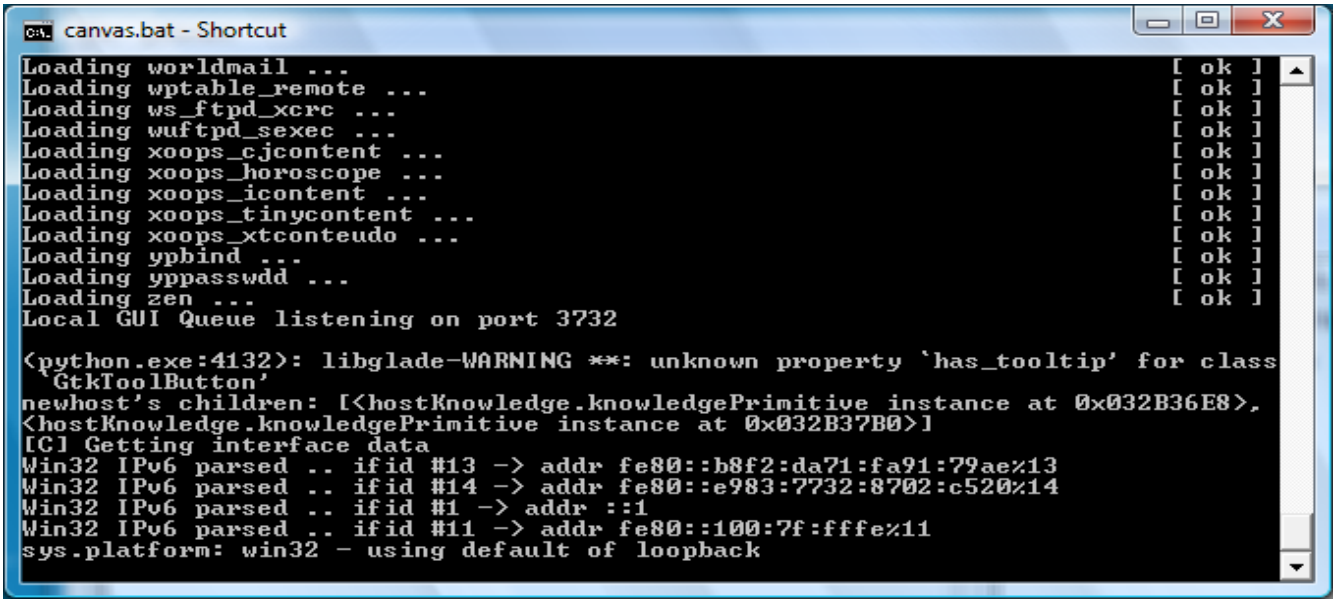


Your First Attack

Lets go over a standard attack, step-by-step before diving into the inner workings of CANVAS

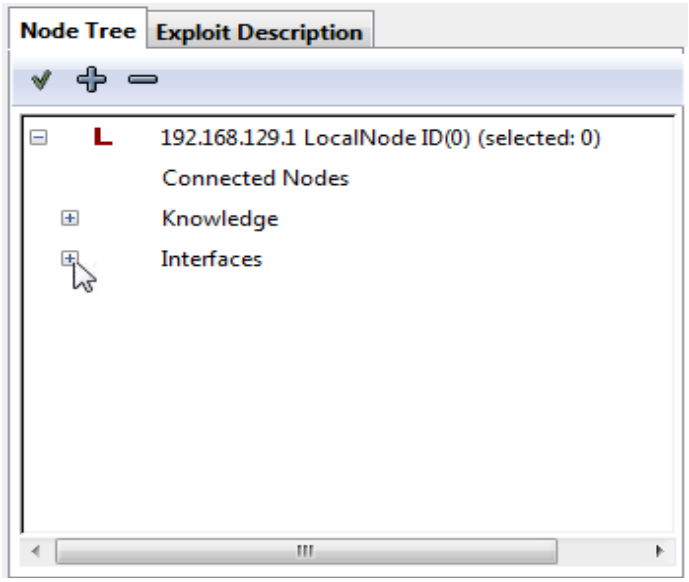
Step 1: Start CANVAS

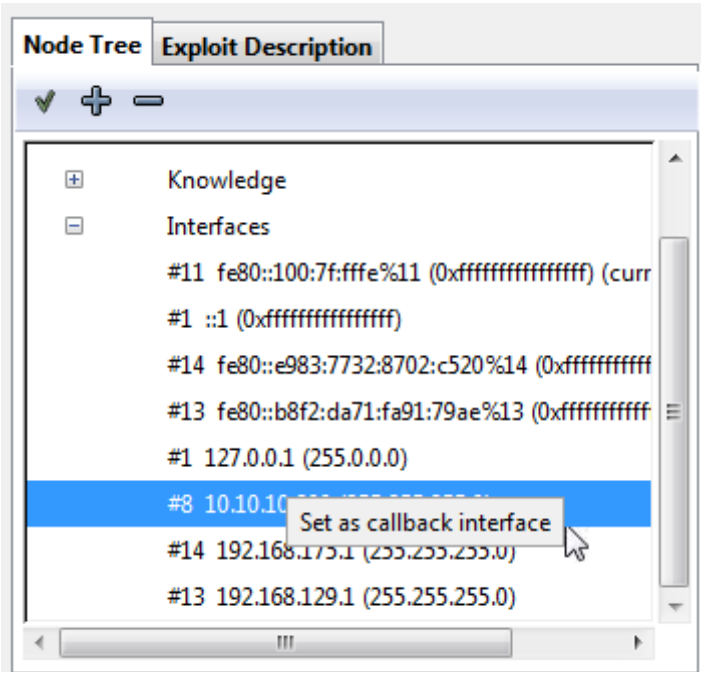
From the windows command prompt, CD into the CANVAS directory and run canvas.bat, or if you created the shortcut on your desktop as I advised earlier in this tutorial, simply click on that.



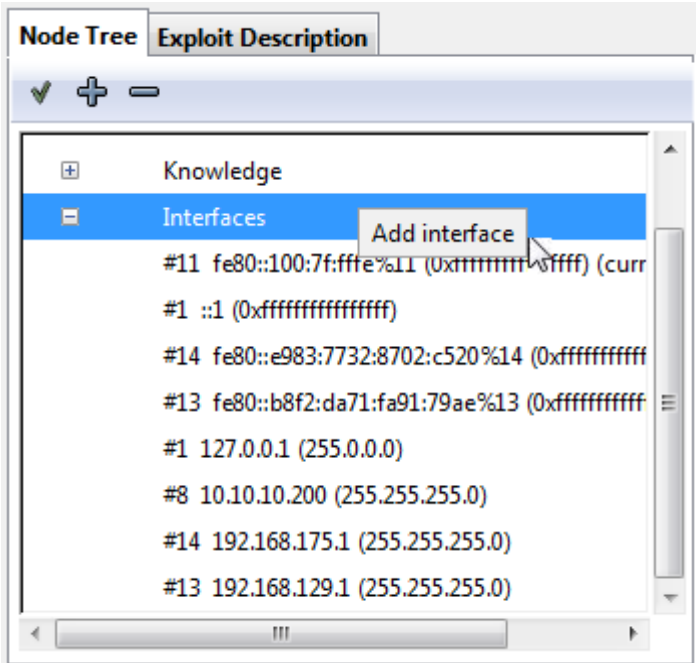
Step 2: Set your Callback IP Address

From the Node Tree tab in the Knowledge Window, expand the Interfaces and select a callback interface that will allow inbound connections from the hosts or subnets you will be attacking.

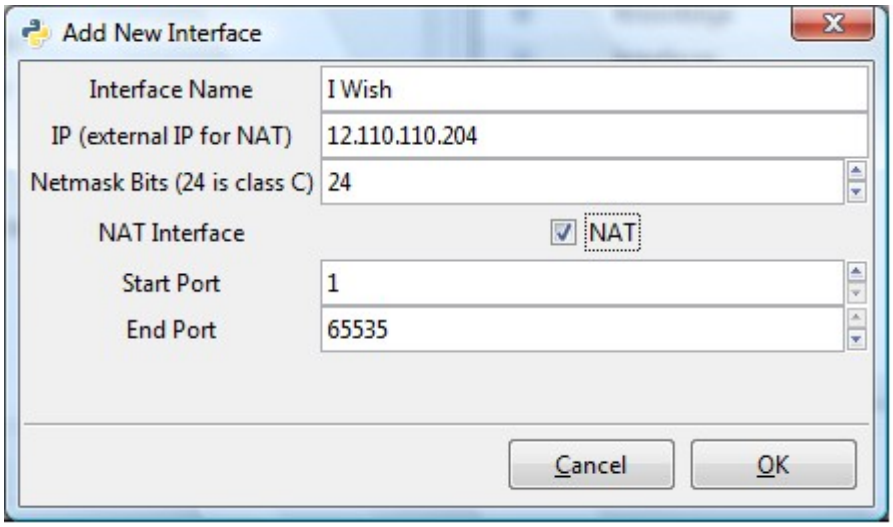




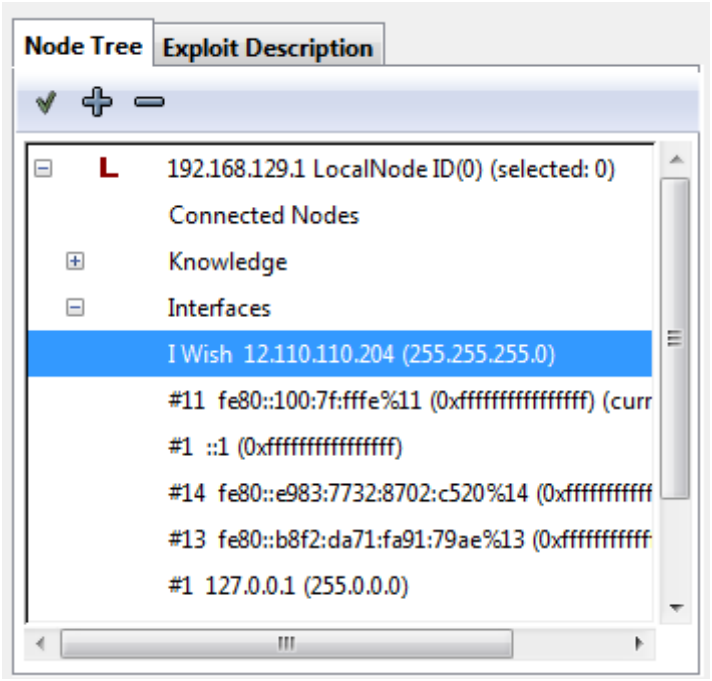
If you need to add an IP address that is not on your local machine, such as a NAT address that is routed to your workstation, left click Interfaces to highlight it, and then open the context menu with a right-click and select “Add interface”.



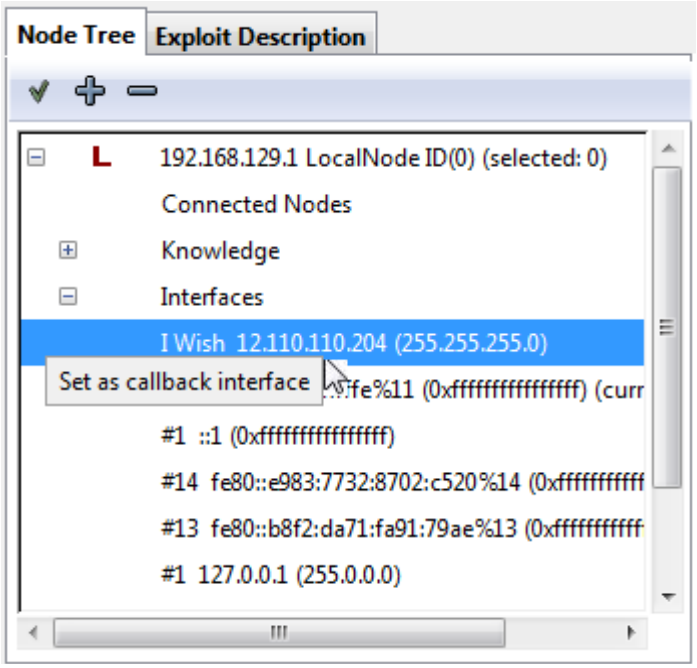
Then, type in the desired IP address and check the NAT Interface and click OK. You'll want to narrow the port range down to the specific ports you have forwarded to your local IP.



You should now see your new interface in the Knowledge Window, underneath the Interfaces tree.

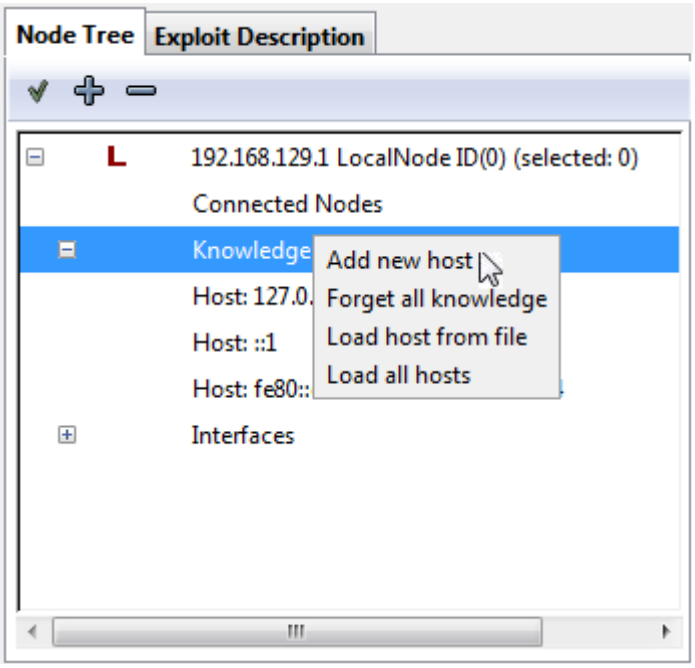


Don't forget to set the IP address you just added as the callback interface!

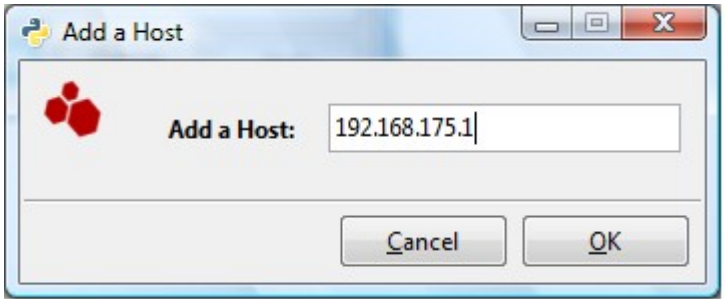


Step 3: Add a Target

Adding a target is similar to adding an Interface. Left-click Knowledge and then select “Add new host” from the context menu.

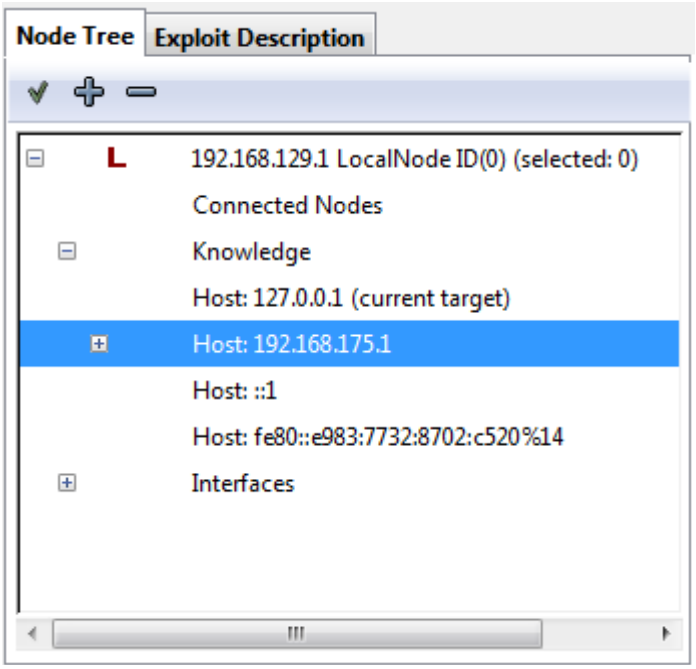


The CANVAS GUI does not differentiate between a Single IP address or 256 IP's from the Add new host window. If you want to scan a Class C range of IP address, just add the first IP in the range. For example:

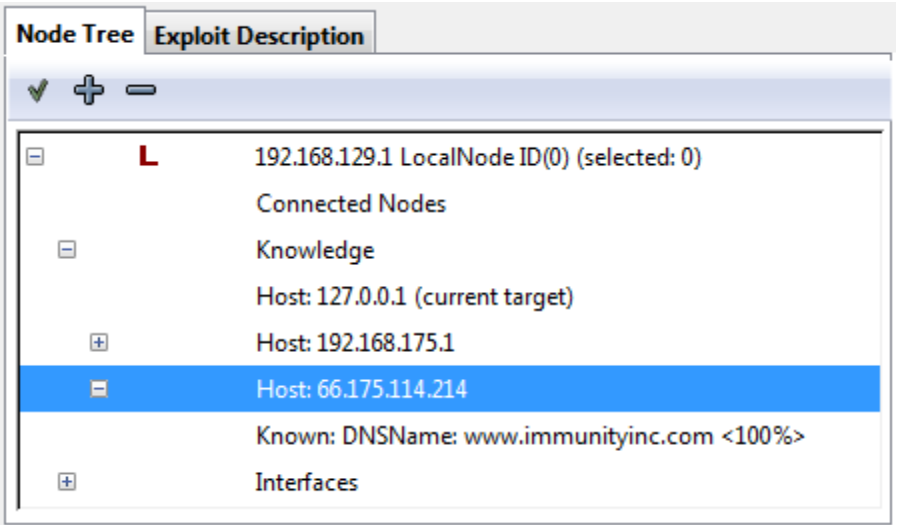


You may also input a domain name in the Add a Host field and CANVAS will perform a DNS A record lookup for you (aka, a host name lookup so for example www.immunityinc.com would result in 66.175.114.214).

You should now see your new entry in the Knowledge tree



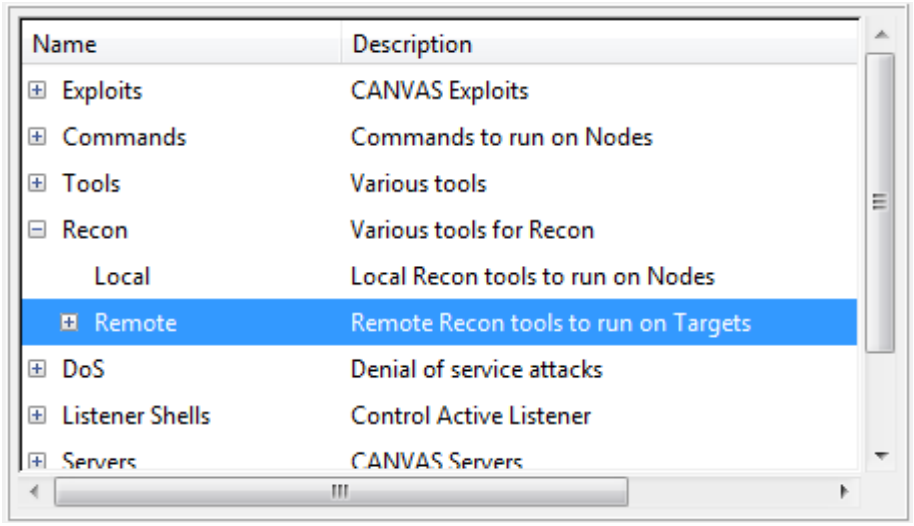
Your entry should look like this if you added a URL with a valid A record:



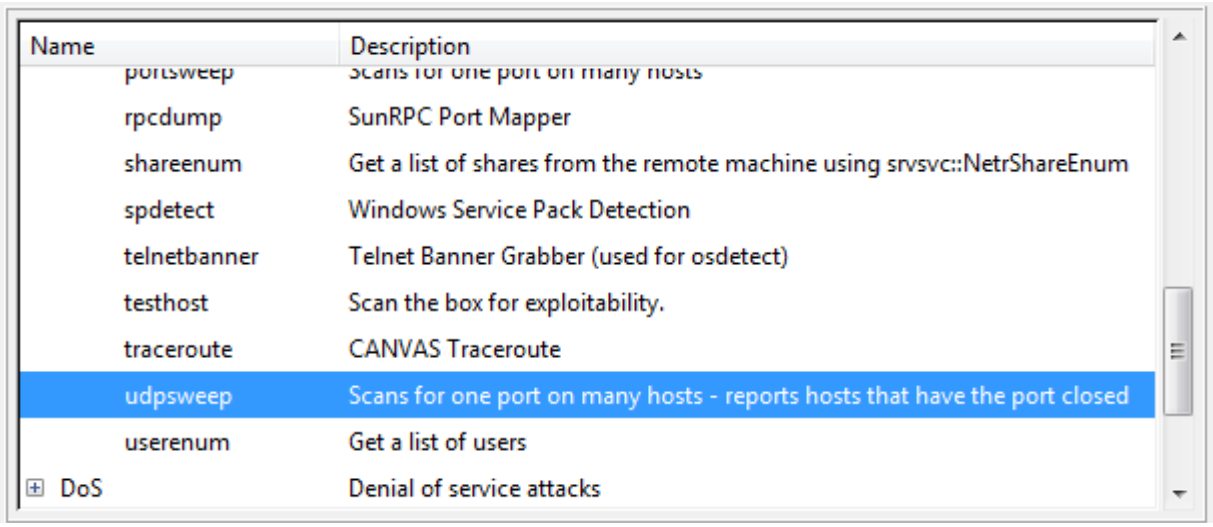
Step 4: Know Your Victim

After adding a host, you will then want to assess the potential vulnerabilities that machine has. If you have added the first IP of a subnet, you will want to discover what hosts are alive on that network. Let's take a look at how I would approach each scenario, starting with the scanning of a subnet.

From the command window, expand the Recon tree, and then the Remote tree.

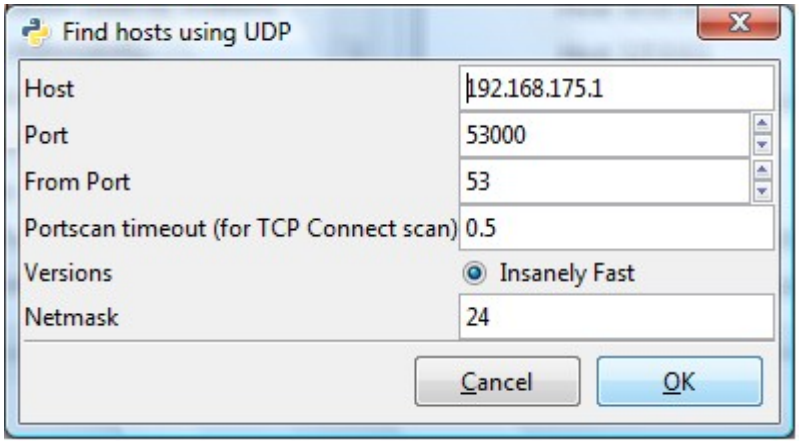


From the Remote tree, open the udpsweep module by double-clicking on it



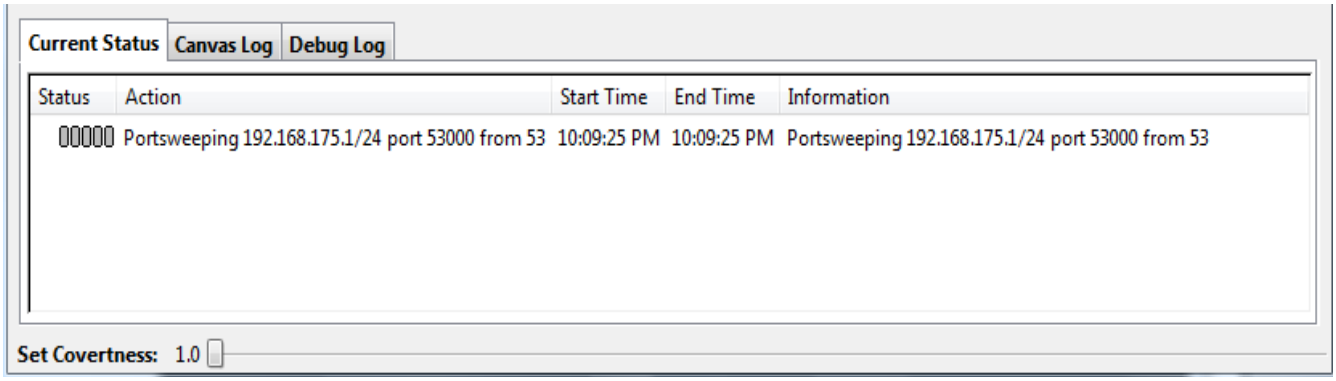
Notice the Netmask value at the bottom. Most modules will accept any accurate CIDR notation.

A value of 24 will scan the 254 IP addresses above 192.168.175.1.
A value of 32 will scan a single IP address.

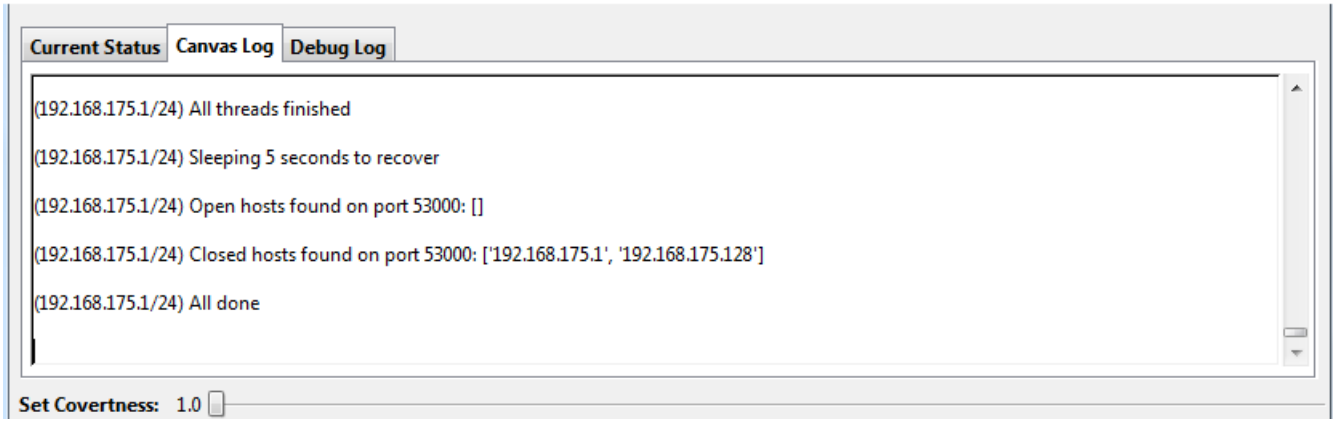


This particular module will send a UDP packet to a port of your choosing (default of 53,000). Since this port is almost always closed, a computer will respond with a "port closed" message, wile no response will be generated if there is not a computer present at that IP address. This lets us know that there is a computer to attack, even if that particular port is closed. Start the scan by clicking"Ok".

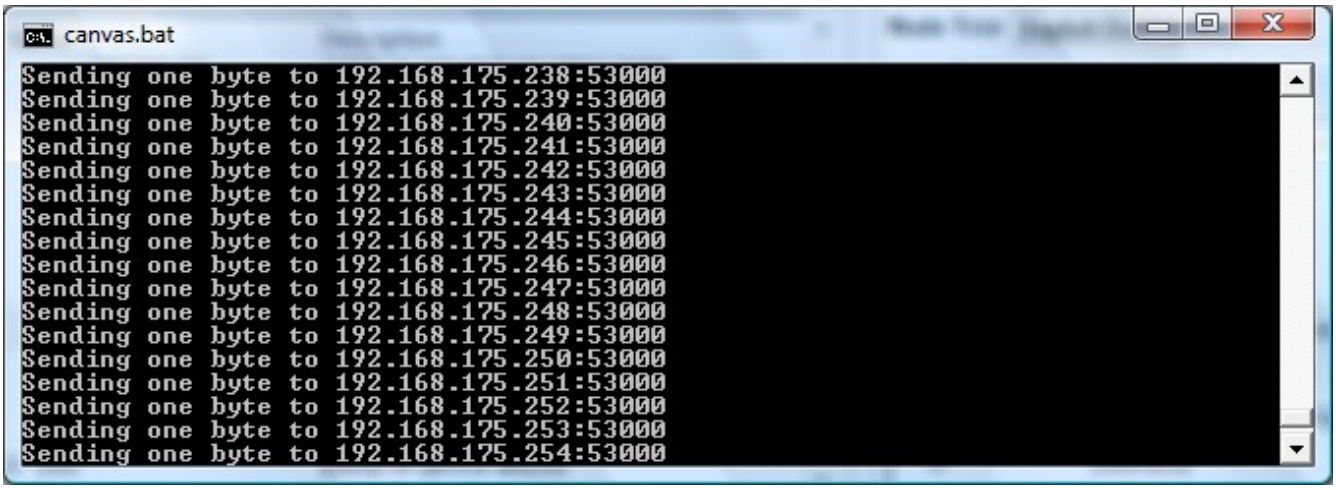
The Information window will give you the progress of the scan



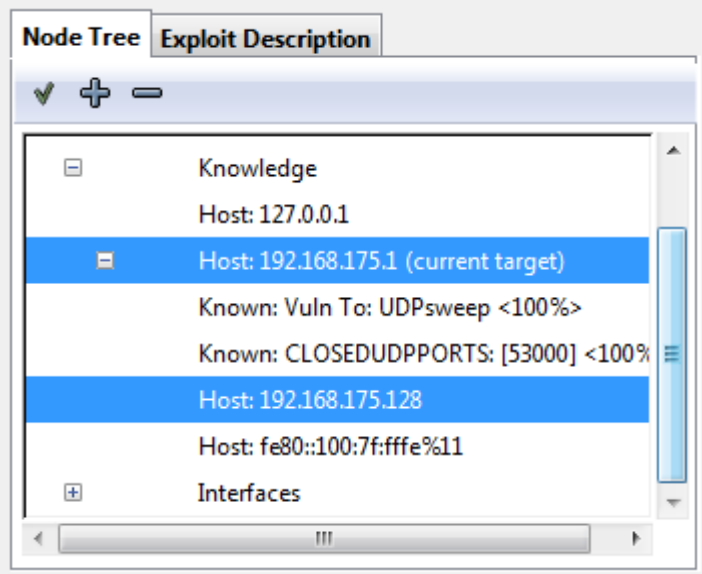
From the Information window, you can select the Canvas Log tab for a more detailed look at whats going on



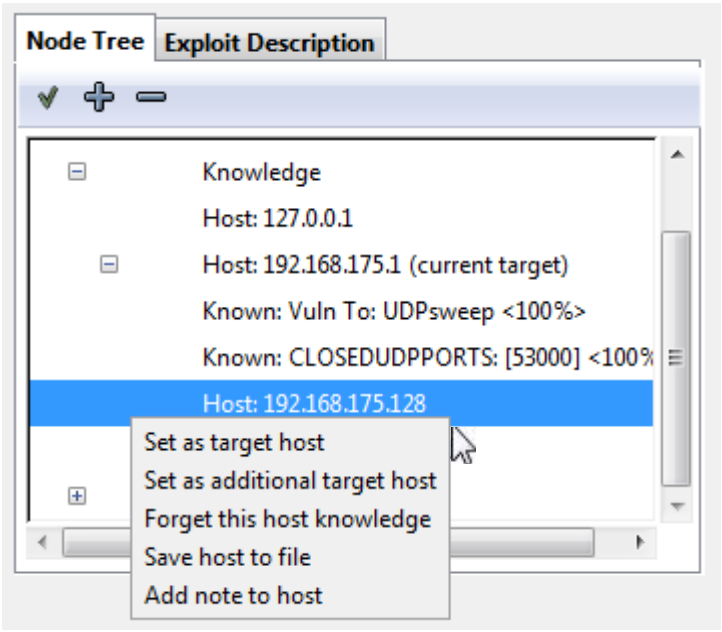
You can also take a look under the hood from the canvas.bat window



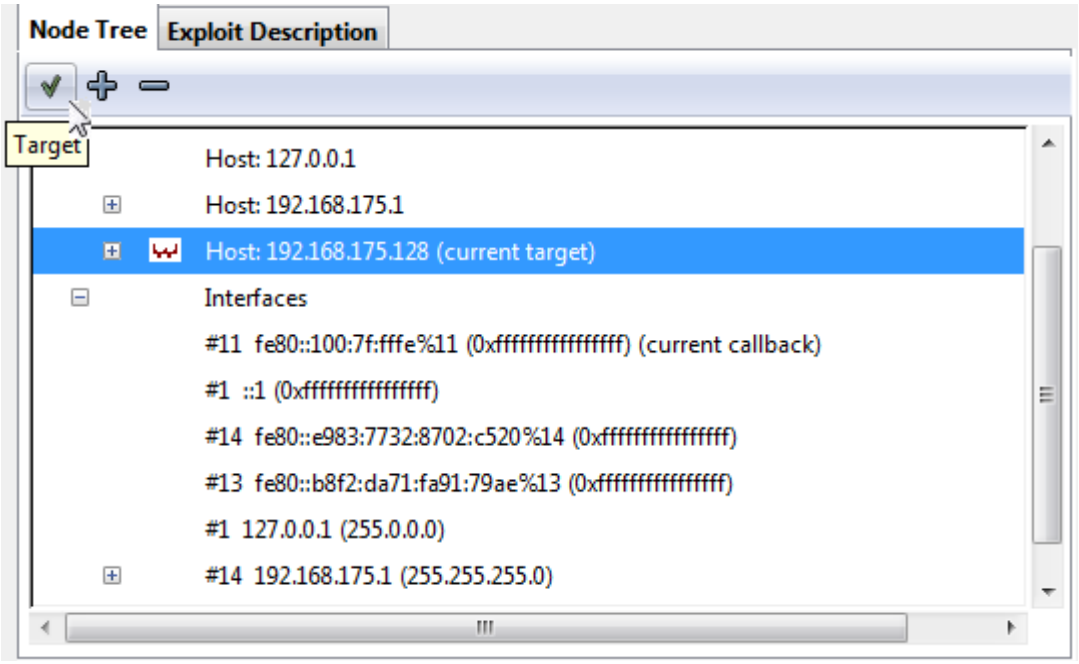
As the Canvas Log tab in the Information window reported, Two hosts were found with a closed UDP port 53,000. These closed ports will generate a new entries in the Information window under the information tree as “Host:” entries.



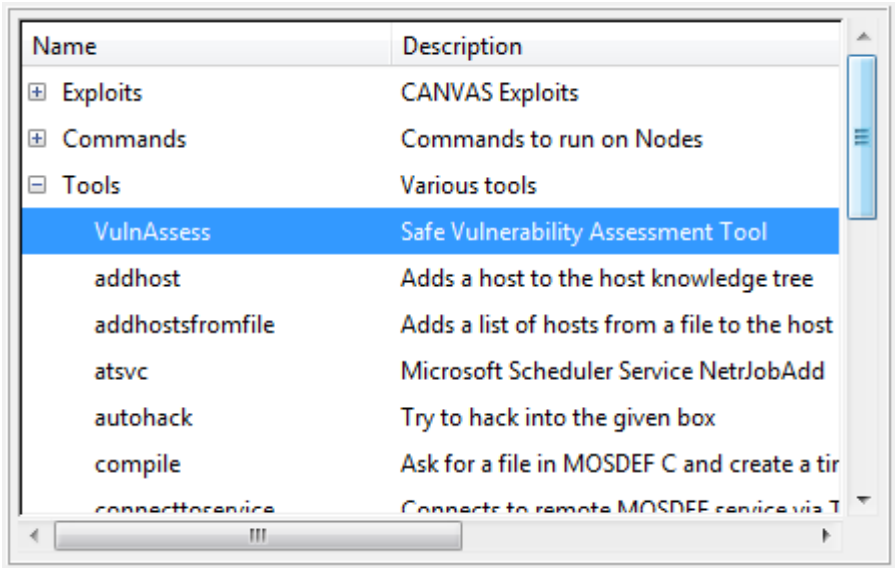
Let's continue our attack by testing out the newly discovered host at 192.168.175.128. Left click on that host and then set it to the current target by right clicking the host and selecting “Set as target host”



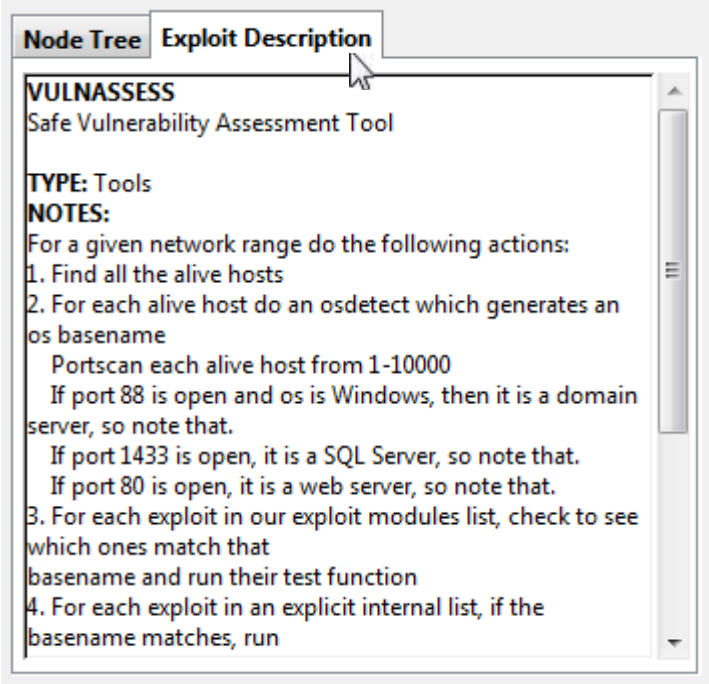
You can also set the current target by highlighting the desired Host: and then clicking on the checkmark icon in the top left corner of the Knowledge Window:



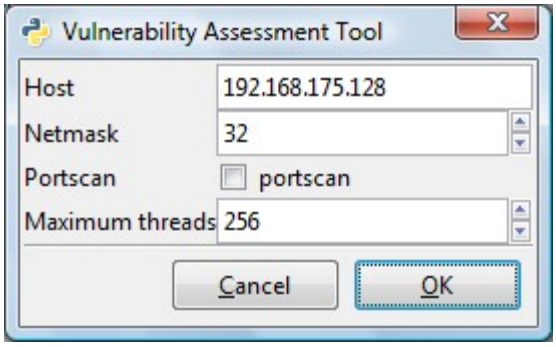
From the tools tree in the Command Window, lets highlight the VulnAssess module by left-clicking it. This module will scan for common services, drill into shares, try to determine which language of OS is running and generally let us know which type of operating system we are dealing with. VulnAssess will also try to break into services with Low Hanging Fruit (LHF) such as FTP servers with easy to guess passwords and SQL servers with blank SA passwords. Be Advised: This module will put entries into the Windows Event logs (mainly the Security Log as failed password attempts)



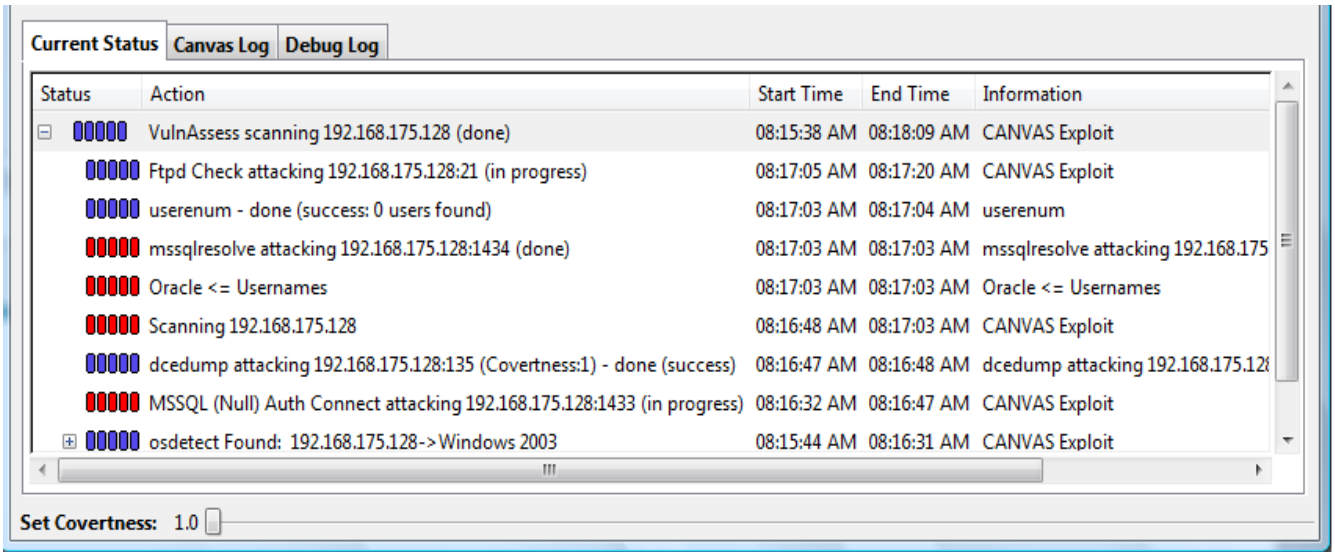
If you want a more detailed description of the individual modules select the “Exploit Description” tab from the Knowledge Window. This will give you a brief synopsis of each module available.



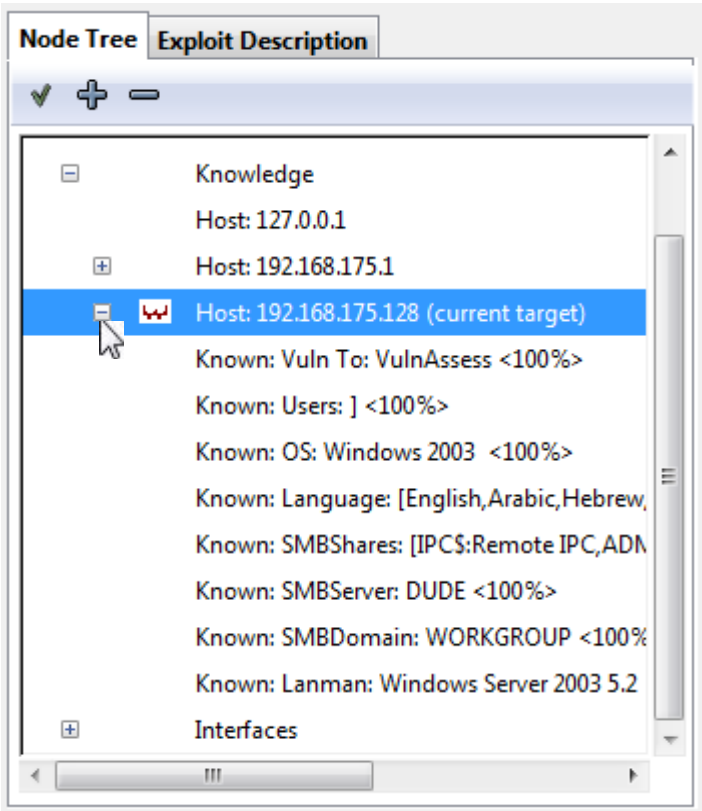
Now, open the VulnAssess module by double-clicking it. Notice that the VulnAssess module's default netmask value is 32, which will only scan a single IP address.



Once again, the Information Window will let us know when the scan is finished, as well as let us know which modules were invoked by this many-in-one vulnerability assessment tool.



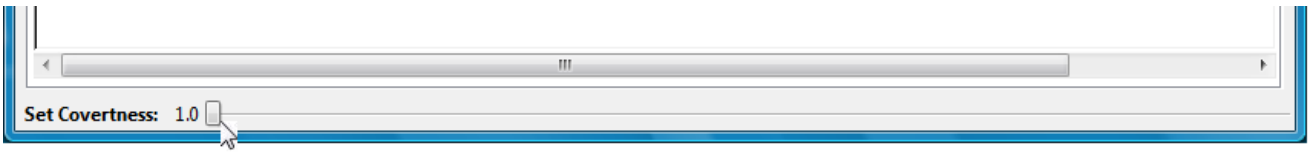
And once again, the information gleaned from the VunAssess module is added to the Host: Entry in the Information Window, underneath the information tree:



As you can see from the updated Host: entry, we are dealing with a Windows 2003 server. Most likely, we are dealing with an English language server (Unless you are in Israel or and Arabic speaking country) but it is almost impossible to tell whether a Windows server is English, Arabic, or Hebrew

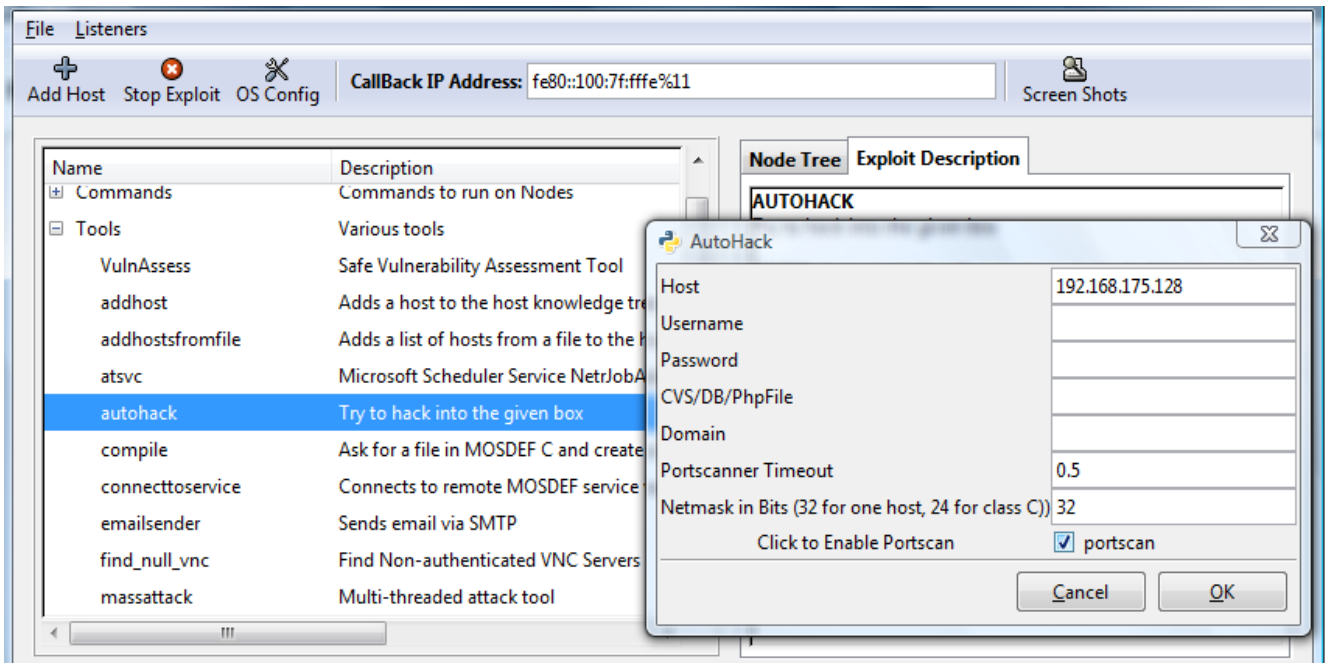
unless they have IIS running. If you are doing your scanning from a Hebrew or Arabic country, you should do some extra configuration of CANVAS to make sure you aren't trying English language attacks against a non-English server (which will probably crash the machine).

Before we try one of the more interesting modules in CANVAS, lets talk very briefly about the covertness bar you may have noticed at the bottom of the Information Window (pictured below)

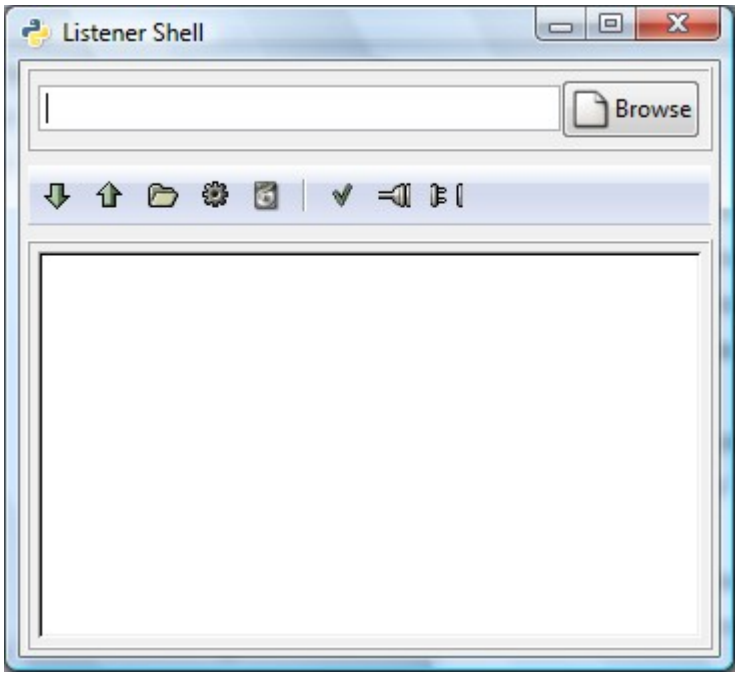


If you use a high covertness level (the bar goes to 11), then CANVAS will try to sneak past monitoring systems by fragmenting the packets while in transit. The packets will be reassembled when the reach their final destination, bypassing signature based filtering methods. This, as you might expect, slows down a module considerably. For this next example, lets try sliding the covertness bar to the right until it reads “5.0” - a good midway point for most attacks.

Now that we have raised the covertness bar to 5.0, lets try one of the more interesting modules in CANVAS by opening the AutoHack module from the Command Window. AutoHack, while noisy, does produce results by trying every applicable exploit module in CANVAS against a host. I have selected the portscan option for this demonstration



If AutoHack Succeeds , you will see a MOSDEF listener window pop up on your CANVAS workstation which will look like the one below:



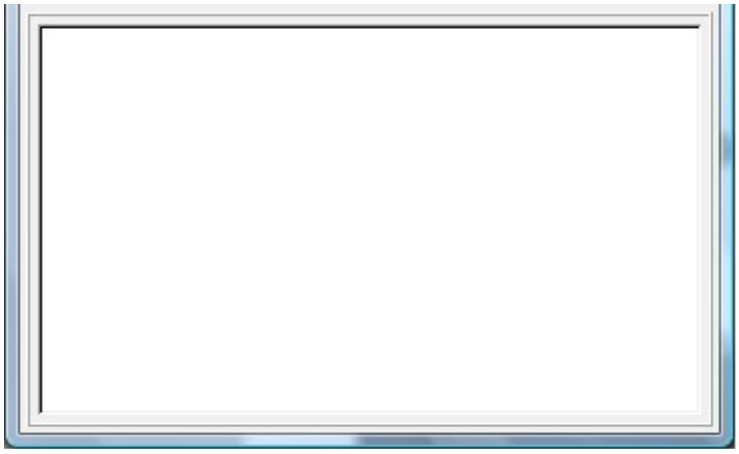
Lets familiarize ourselves with this new window by go over the different icons on this new MOSDEF shell GUI interface. For those who are unfamiliar with the MOSDEF here is a link to a very detailed description of the application: http://www.blackhat.com/presentations/win-usa-04/bh-win-04-aitel.pdf. It will suffice for this tutorial to say that MOSDEF pipes our commands to the compromised operating system and then returns the results from executing said commands.

Lets get back to our Listener Shell window. Like the CANVAS window, there are three main parts in which you will need to familiarize yourself with in order to get the full functionality from the MOSDEF shell listener window. I will refer to them as Frames in order to differentiate them from the three CANVAS Windows:

The Command Frame:

The Actions Frame

The Logging Frame



Lets start with the Action Frame (which works in conjunction with the Command Frame) in an Icon-By-Icon description:

The Down Arrow Icon will download a file from the compromised host.

The Up Arrow Icon will upload a file to the compromised host.

The Folder Icon will change the current working directory (cd) into a directory you specify

The Gear Icon will run an exe or binary program on the compromised host. This should be used when spawning a process that will interact with the GUI on the compromised system. For example, if I wanted to execute calc.exe (which would pop-up on the Windows GUI) as proof that the system was compromised, this would be the way to do it.

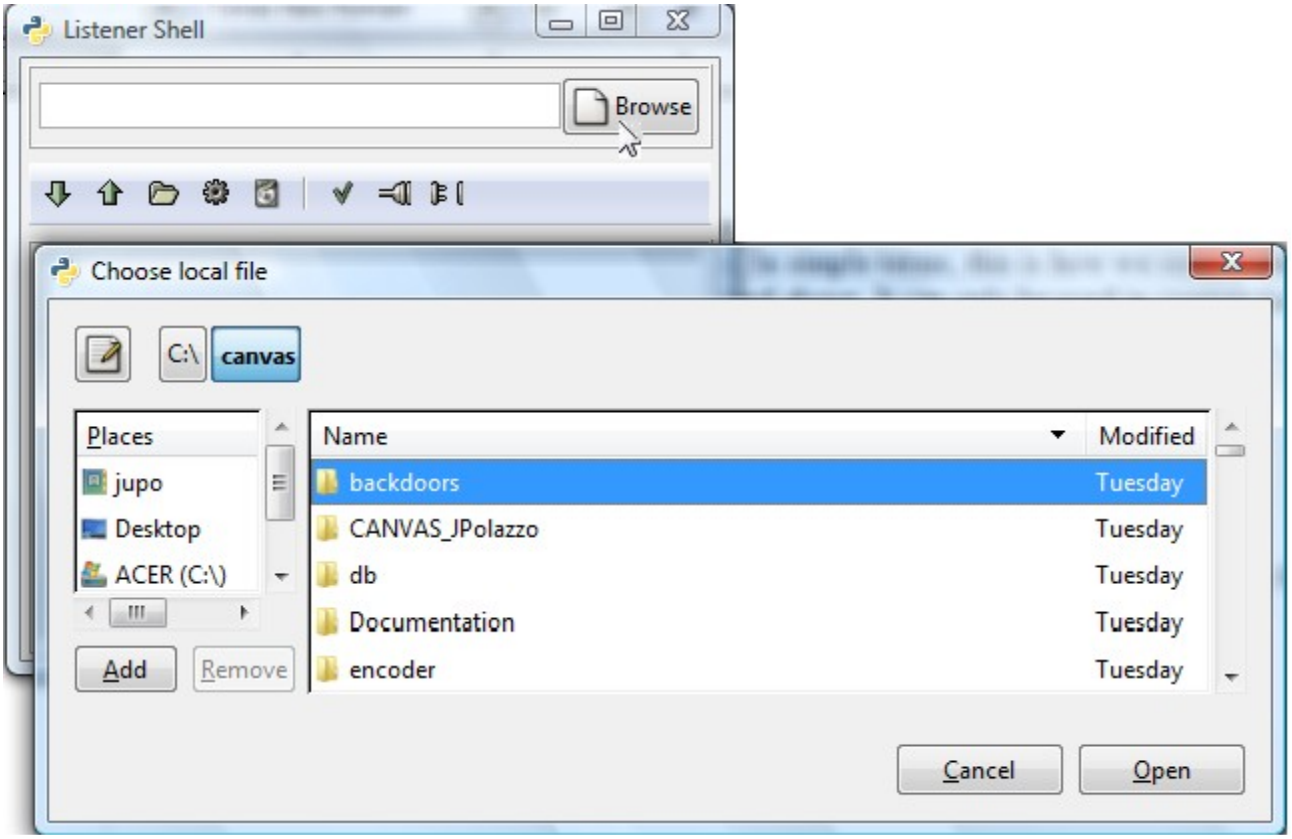
The Directory Icon will send a “dir” or “ls” command and return a listing of the files in the current working directory of the compromised system

The Checkmark Icon will tell you your current working directory on the compromised host

The Connected Icon will pipe any command into the command shell of a system (i.e: cmd.exe or /bin/sh). You should be very careful when using this Icon. If you were to type “calc.exe” and pipe that to the compromised system, the MOSDEF listener shell would stop responding. This Icon should only be used for piping commands that are not interactive. For example, if I was to pipe in “nslookup” I would lose responsiveness from the MOSDEF shell. If I was to type in “nslookup bob.com”, it would return the data from my nslookup command and then return control to the MOSDEF shell.

The Command Frame is where we type variables and commands we want the Action Frame to execute or send/receive to/from the compromised host. In simple terms, this is how we control the various options available from the Action Frame described above. It can only be used in conjunction with the functions available within the Action Frame.

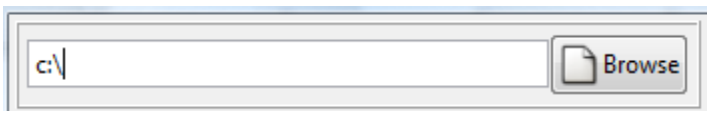
The Browse Icon is only used for selecting a file to upload to the compromised host. This button will open an explorer shell for you to browse:



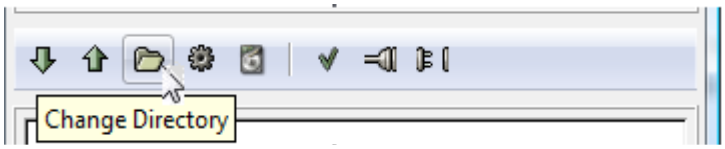
The Logging Frame will return the results of your commands, inputed into the Command Frame, and executed from the Action Frame. For a review, lets go over a few actions step-by-step.

I will change the working directory to be c:\. I will enter "c:\" into the Command Frame and then click on the Folder Icon. The results of this should be displayed in the Logging Frame:

This:



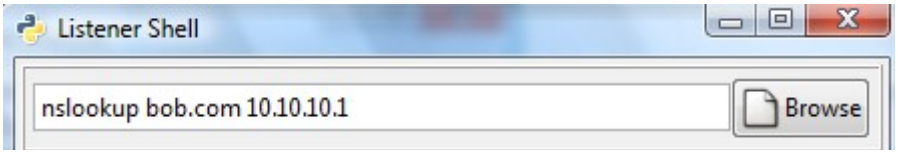
Plus This:



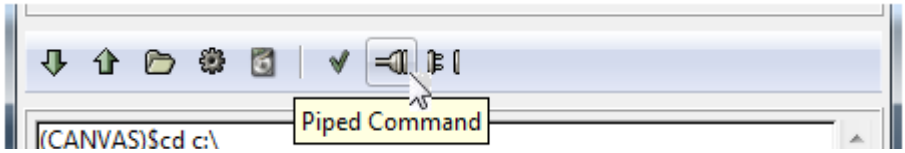
Returns This:

Lets do one more before moving on to next section

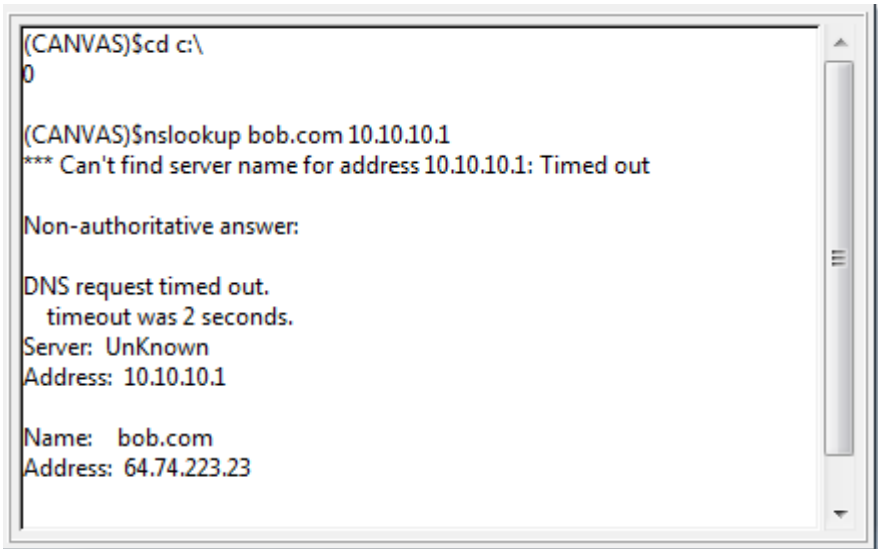
This:



Plus This:



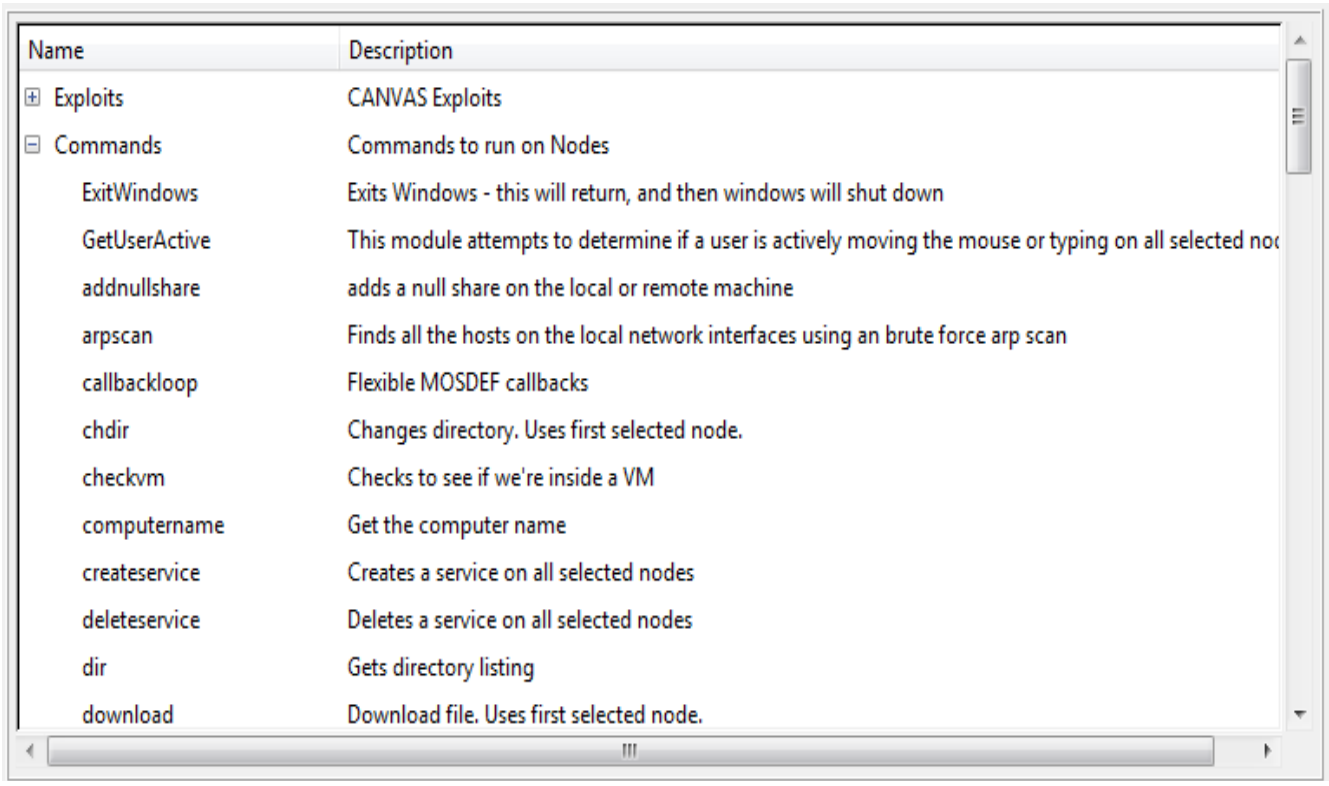
Returns This:



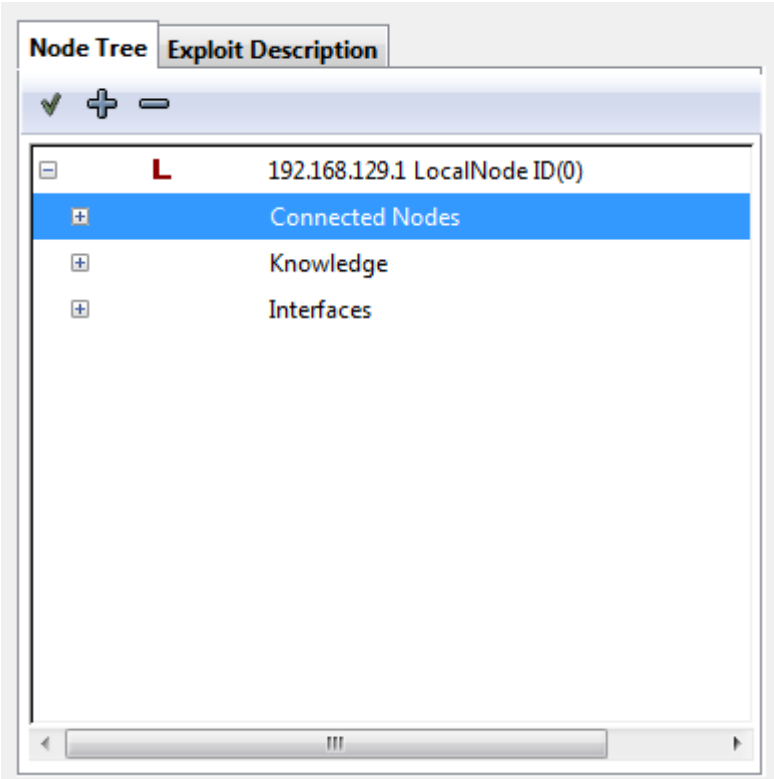
If I was to only pipe the command “nslookup” the MOSDEF shell would quit responding until some nice admin killed the process on the compromised host, so once again, only spawn processes that will do something, then exit; returning control to the MOSDEF Shell

Running Commands on Penetrated Machines

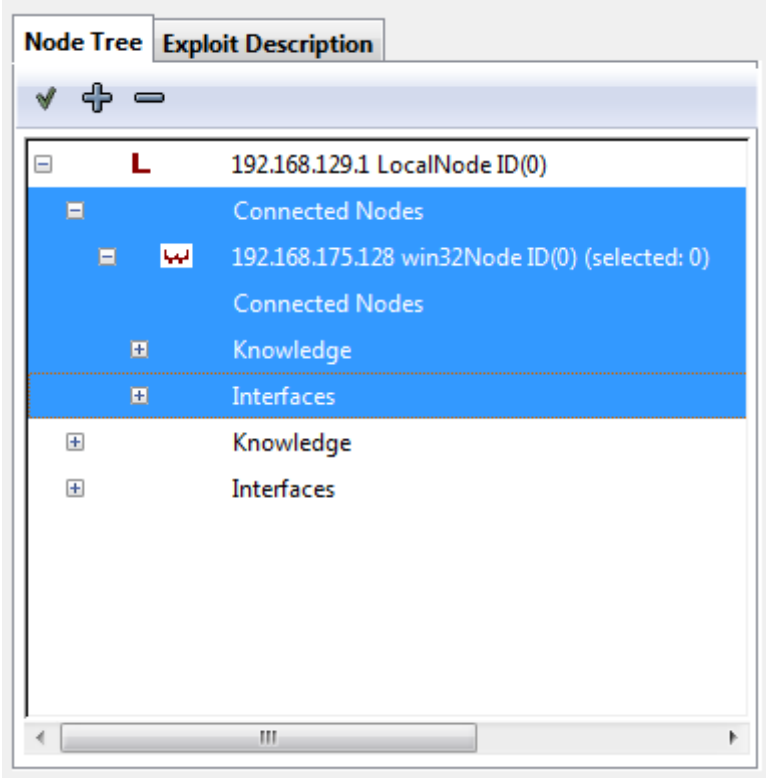
While the MOSDEF Shell GUI interface is a pretty nifty tool, you gain a lot more control and some of the full functionality of MOSDEF by piping in modules via the Commands tree in the Control window, pictured below:



First we need to tell the CANVAS GUI who to pipe the commands to. If you will take a look at the Knowledge Window, you may notice that there is a plus(+) sign next to the Connected Nodes tree



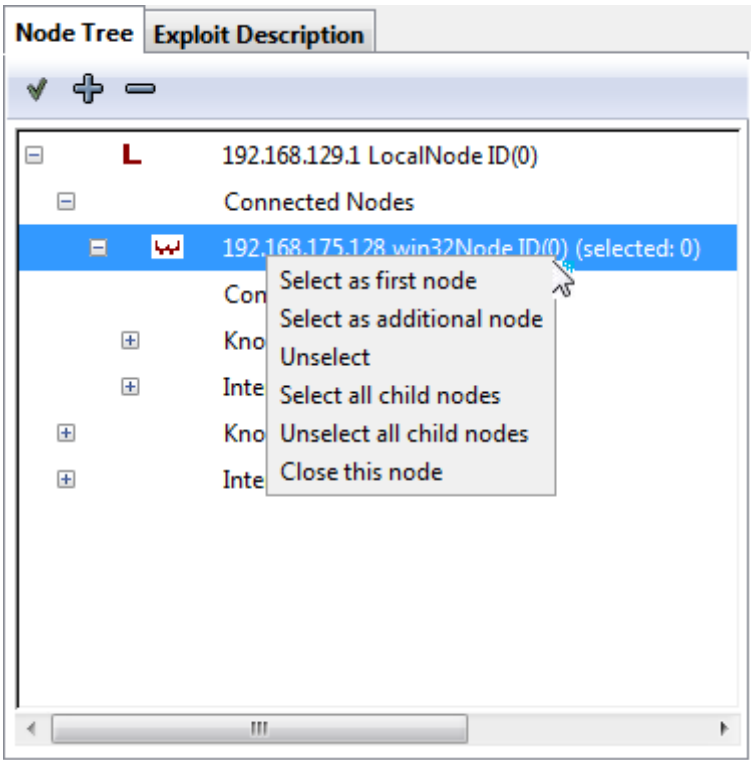
If we expand this, we will see our compromised host, 192.168.175.128



The more observant reader will notice that under the entry for 192.168.175.128 in the Connected Nodes tree there are also Connected, Interface, and Knowledge trees. Since we compromised 192.168.175.128 using CANVAS, that host is now completely under our control. Anything we can do from our CANVAS workstation can now be piped through the Connected Node. This is useful because there are many more exploits available to you if you are in the same subnet, or broadcast domain, as your victim. This repeating pattern of Connected Node to Connected Node will continue to grow as we compromise new hosts using our first conquest as a proxy for further attacks.

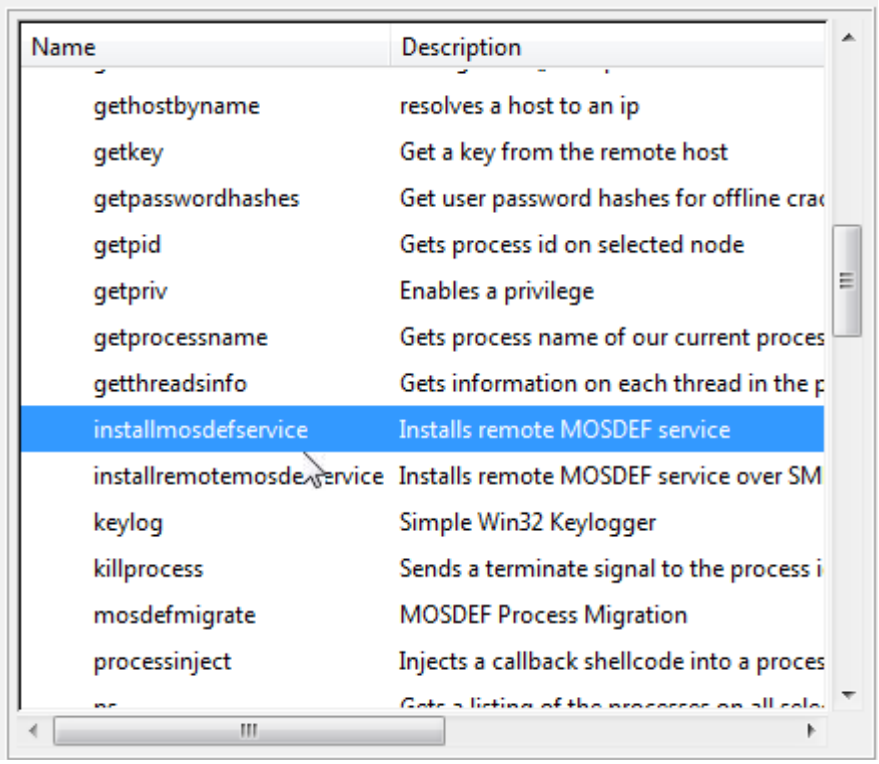
In order to do this, we need to make sure we are not discovered on our first conquest 192.168.175.128. over the next chapter, we will cover installing MOSDEF as a service on this Connected Node, hiding that service, and connecting to the newly installed and hidden MOSDEF service.

We begin by highlighting 192.168.175.128 under the Connected Nodes tree, right clicking on it to bring up the context menu, and then selecting “Set as first node”

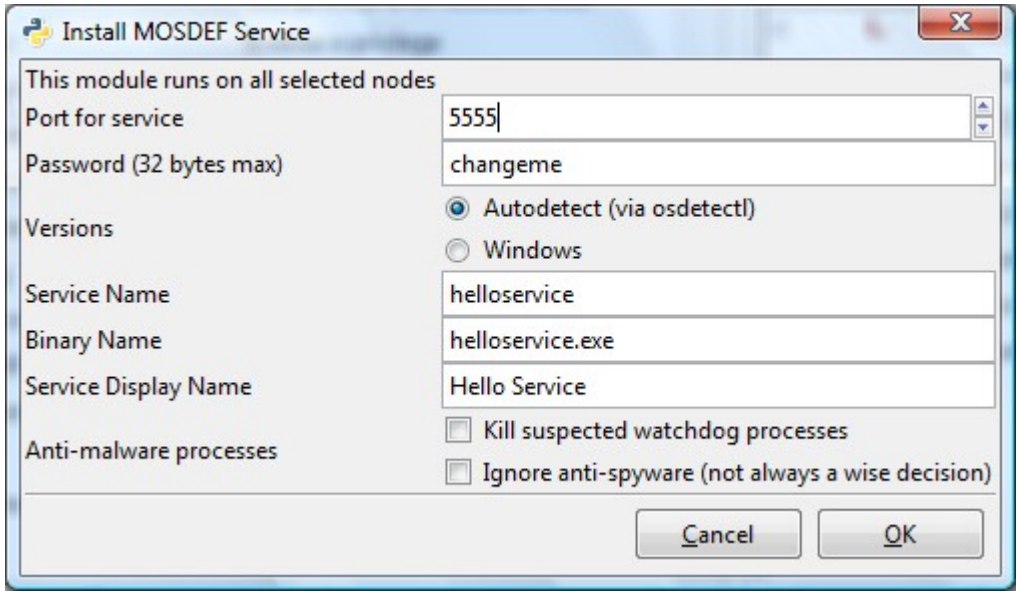


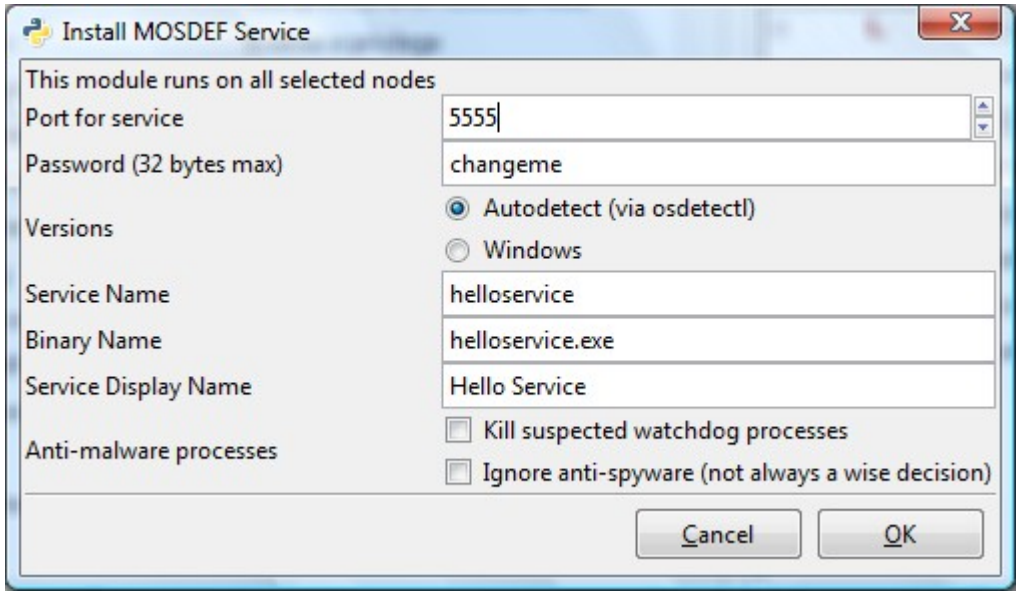
This tells CANVAS to send all MOSDEF commands to this host, much like when we set this option from the knowledge tree.

Then we select the installmosdefservice module from the Control Window

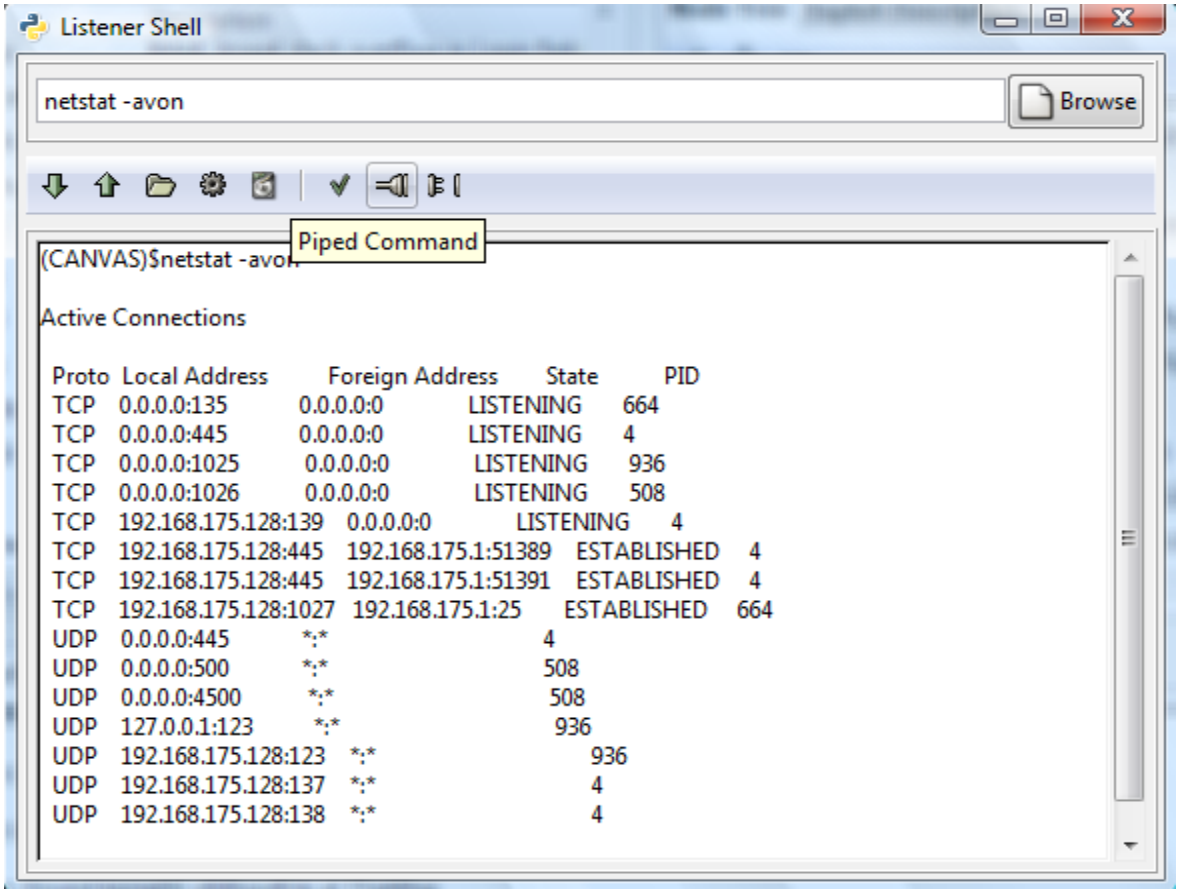


This will bring up the installation options for the MOSDEF service module:



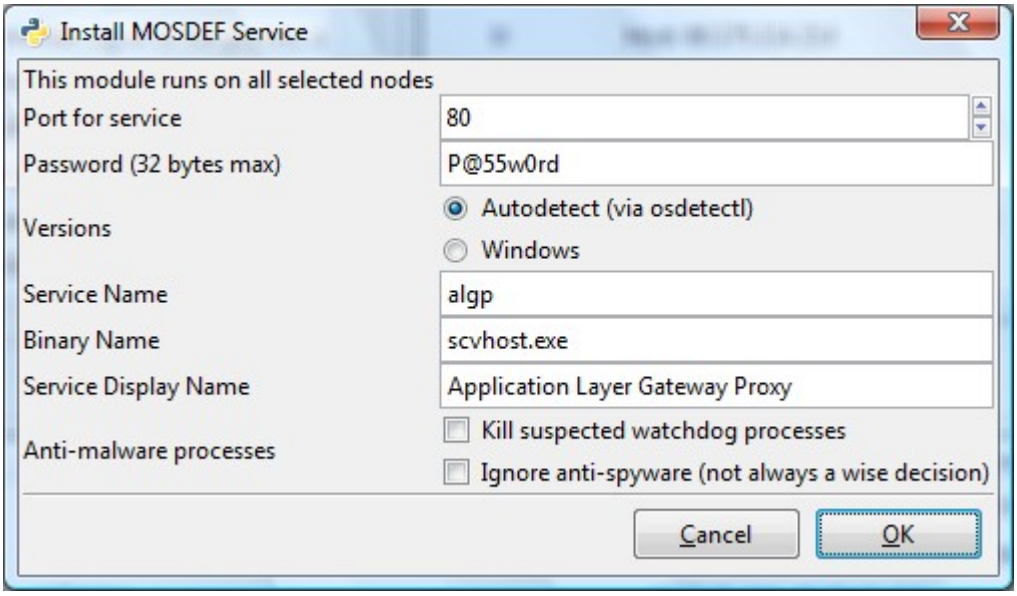


For this new service to work, a few things need to happen. The port we select has to be reachable from our CANVAS workstation. This port also has to not already be in use on the windows host. I recommend running a "netstat -avon" on the compromised host to see what ports are already in use:

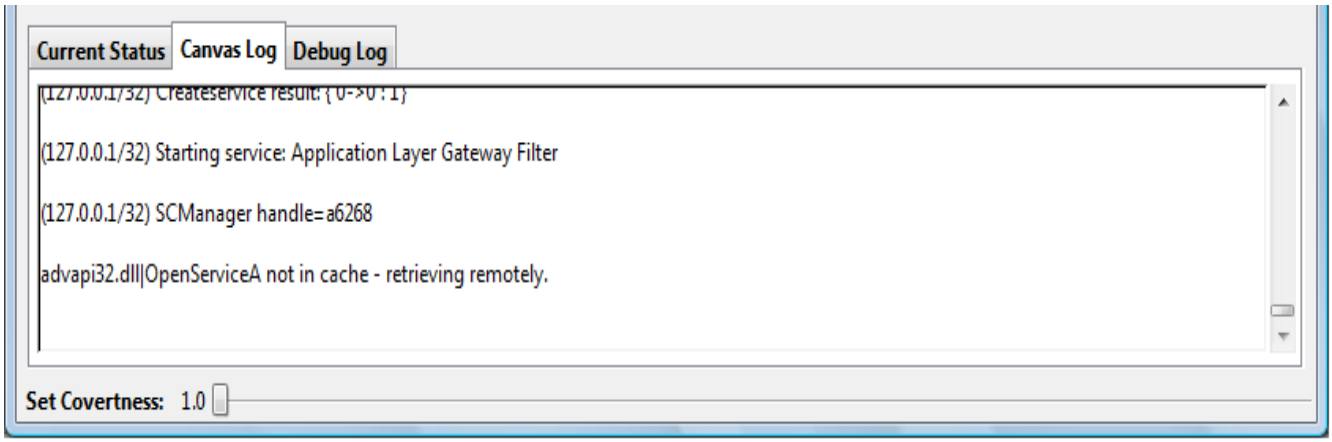


As we can see from the Logging Frame, there is not much running on this server. Since we have a plethora of ports to choose from, I recommend choosing a port that will not arouse suspicion, such as 53 or 80. Since we will be sending a lot of data to and from the compromised machine, I am going to choose 80, since its not that unusual for an admin to see a lot of traffic to this port. If all non-firewalled ports are in use, there are ways around it, but they are outside the scope of this tutorial.

Next we need to choose a password and also name our service. Make sure when you fill in the Service Name that you don't include any spaces, I will name this one algp. I recommend naming the Service Display Name something that will not arouse suspicion like “Application Layer Gateway Proxy” Likewise, name the executable something the average admin will glance at and then move on, such as “scvhost.exe”. Be careful not to name it “svchost.exe as that name will conflict with an already running windows process.

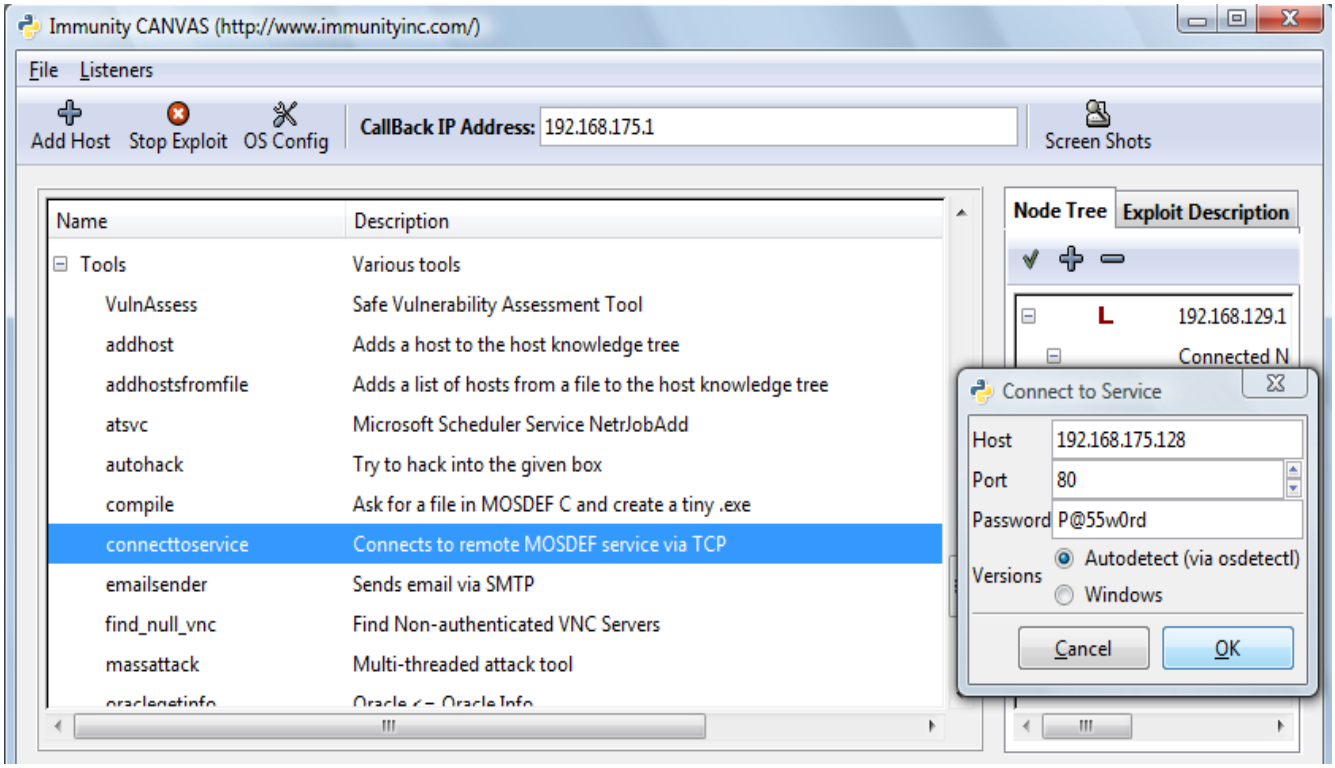


After pressing Ok, you can verify that CANVAS is installing the MOSDEF service by watching the Canvas Log tab in the Information Window:



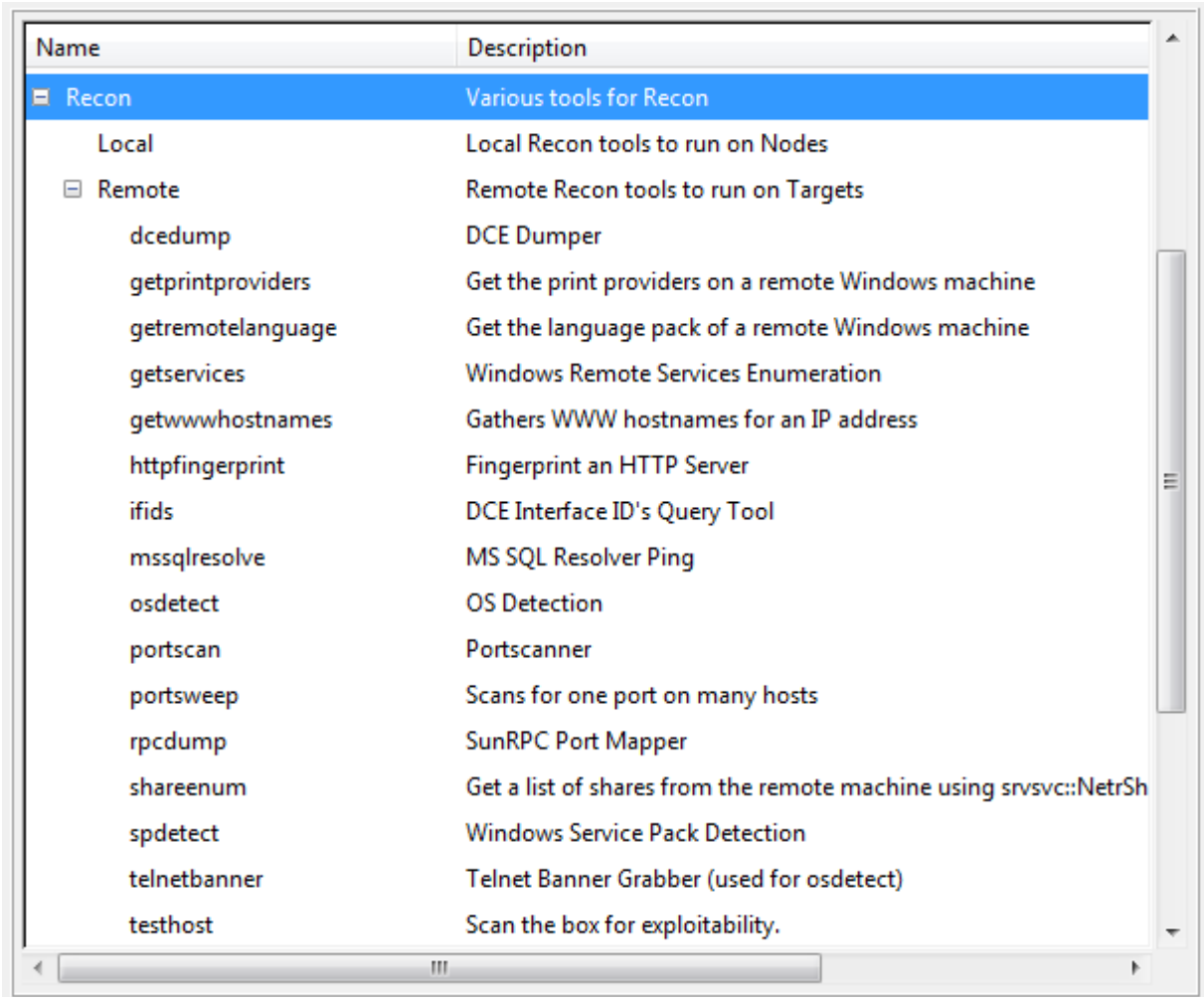
You can verify the serivce is running by piping in another “netstat -avon” via the MOSDEF shell GUI and looking for a service listening on port 80.

After we have verified that our service installed correctly and is listening on the proper port, select the connecttoservice module from the Tools tree in the Control Window. As you may have guessed by now, this will connect us to our newly installed MOSDEF service, after filling in the relevant fields.

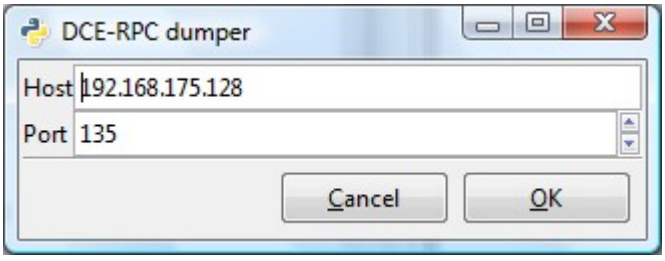


Recon and Exploit Modules

Over the next few chapters, we are going to give you a breif synopsis of some of the CANVAS modules available through the GUI interface, starting with the Recon and Exploit modules listed in the so named trees in the Control Window. Let us start with the Recon Tools (pictured below).



dcedump



Supported Architectures: Windows: NT, 2000, XP, and 2003

This module will dump the windows endpoint mapper and return the data in the Canvas Log window, the canvas.log file and also display in the canvas.bat shell. The returned data will look similar to this:

192.168.175.128/32) 378e52b0-c0a9-11cf-822d-00aa0051e40f:1:ncalrpc:wzcsvc:

(192.168.175.128/32) 378e52b0-c0a9-11cf-822d-00aa0051e40f:
1:ncalrpc:OLE2BCCD633C4364A2DB761675F4FDF:

(192.168.175.128/32) 378e52b0-c0a9-11cf-822d-00aa0051e40f:1:tcp:1025:ip:192.168.175.128:

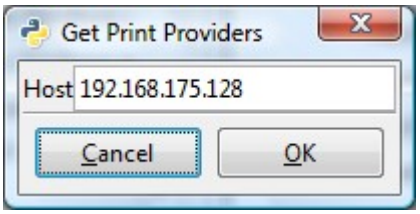
(192.168.175.128/32) 378e52b0-c0a9-11cf-822d-00aa0051e40f:
1:netbios:\\DUDE:namedpipe:\PIPE\atsvc:

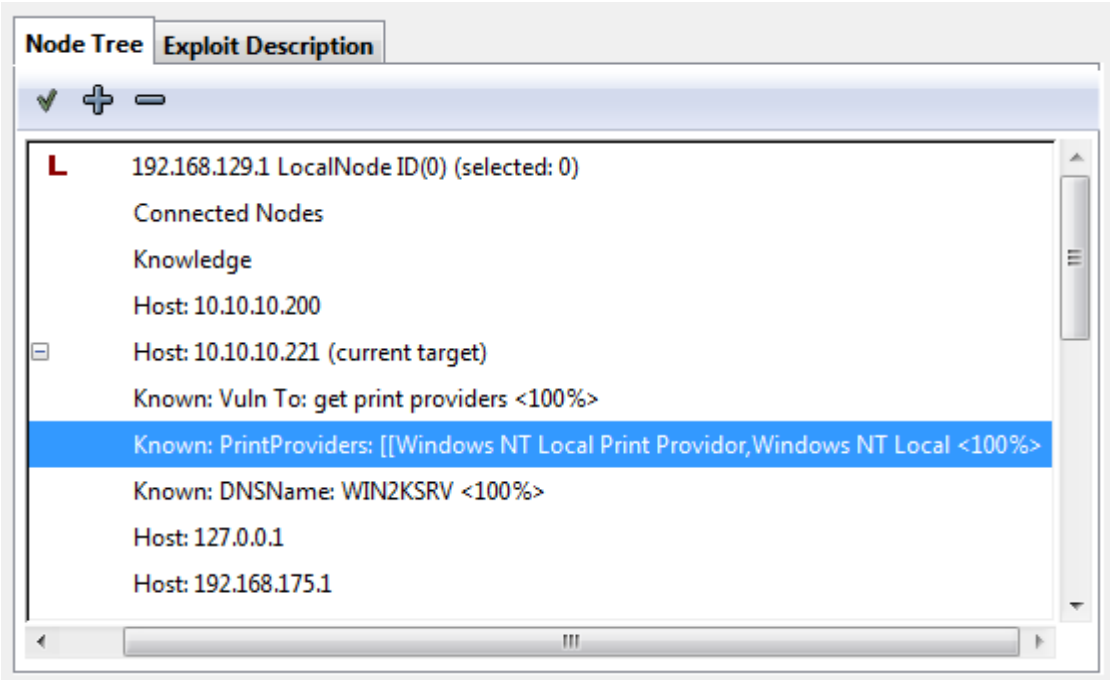
(192.168.175.128/32) 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53:1:ncalrpc:wzcsvc:

(192.168.175.128/32) 0a74ef1c-41a4-4e06-83ae-
dc74fb1cdd53:1:ncalrpc:OLE2BCCD633C4364A2DB761675F4FDF:

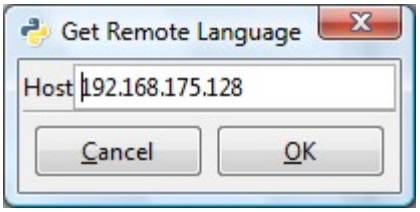
(192.168.175.128/32) 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53:1:tcp:1025:ip:192.168.175.128:

getprintproviders





getremotelanguage



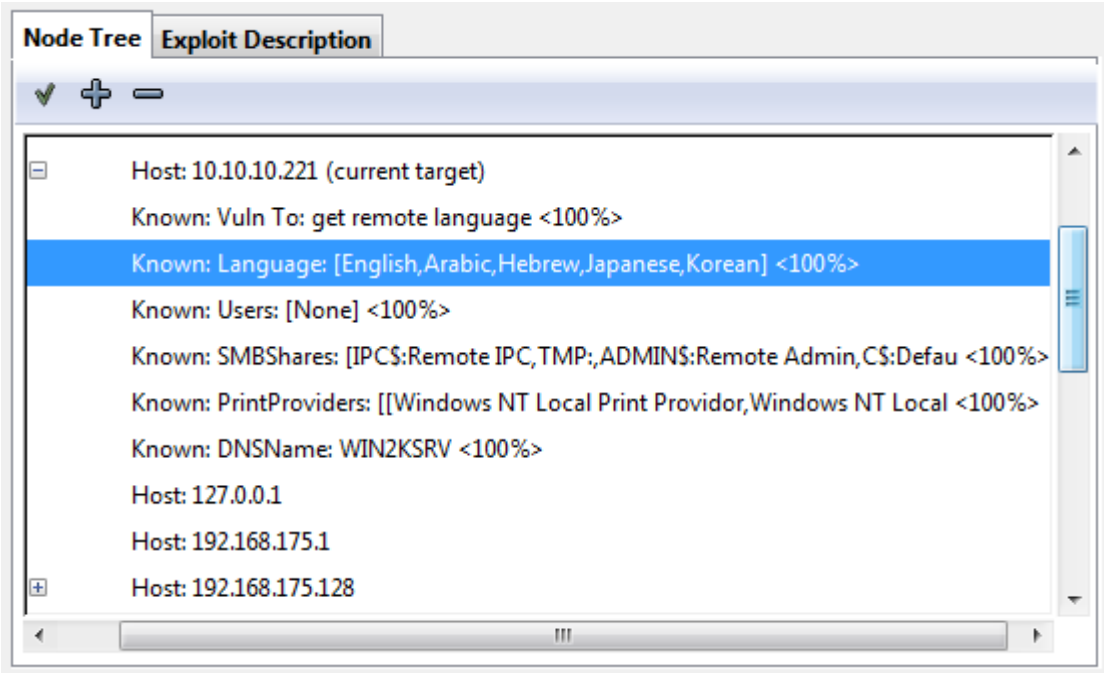
Supported Architectures: Windows: NT, 2000, XP, and 2003

This module gets the remote language pack of a Windows system. It does this by requesting a list of shares from that system, and then comparing the comment fields with known comment fields. By default a Windows system will have ADMIN$, C$, and IPC$ shared out, so we can look at all of them to narrow it down.

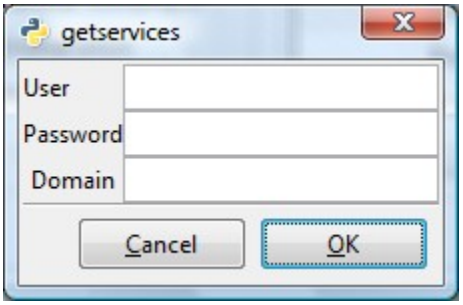
Other methods currently used include:

- o 404 handlers
- o Print providers (from getprintprovider)
- o Usernames (from userenum)

This also adds an entry in the knowledge tree under the appropriate Host: entry

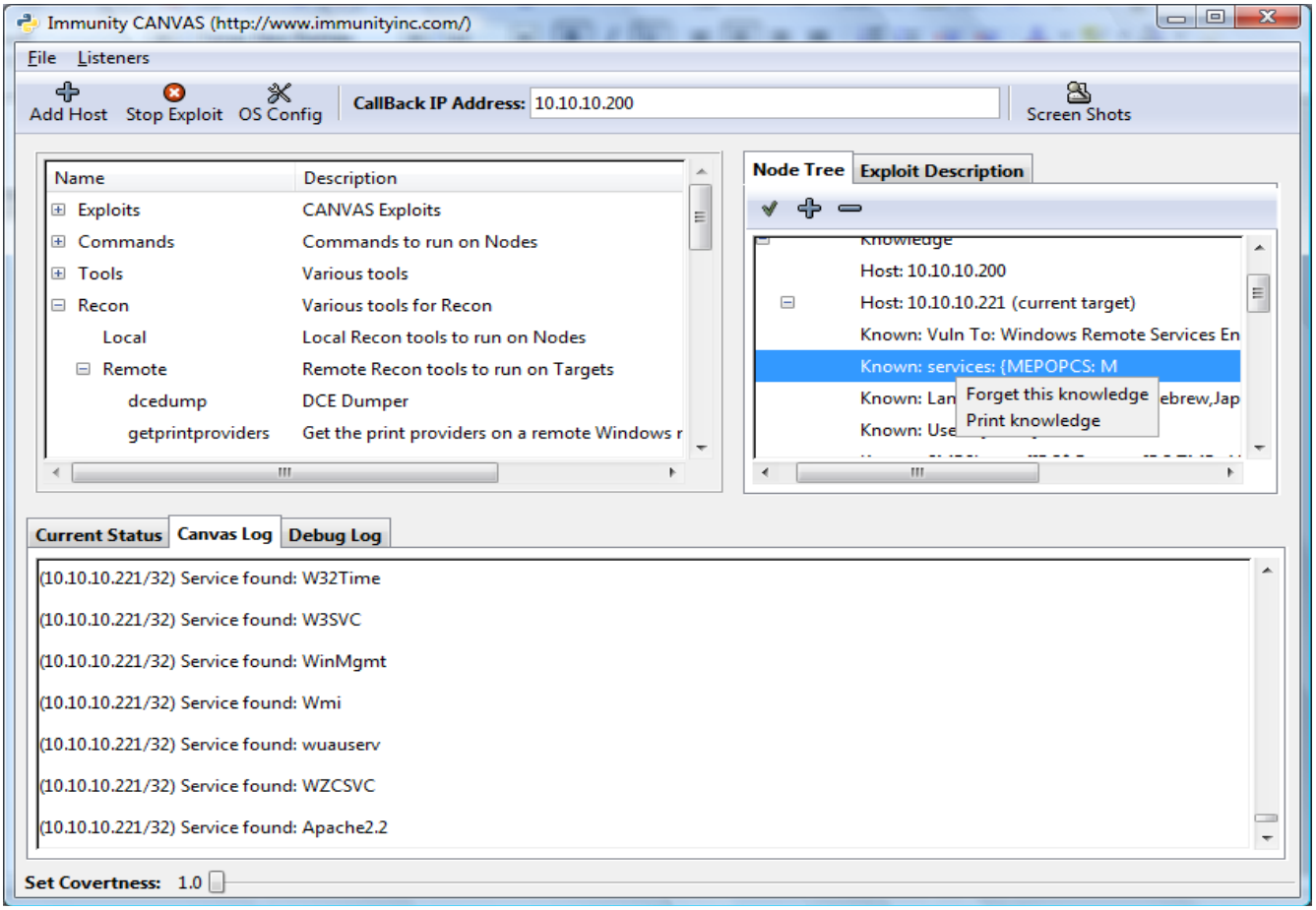


getservices

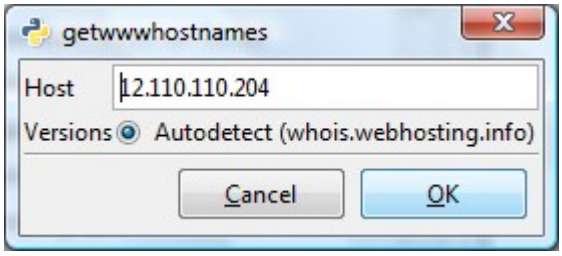


Supported Architectures: Windows: NT, XP, and 2000 up to service pack 4.
it should be noted that this module does not work against windows 2000 with the post SP4 roll-up.
It should also be noted that you do not have to enter a user, password or domain for this to work.

You may notice that the services entry under the appropriate Host: entry may be cut off. You can always review the Canvas Log tab in the Information Window, or, just highlight the Known services entry, then right-click and select “Print Knowledge” to re-print this information in the Canvas Log tab.

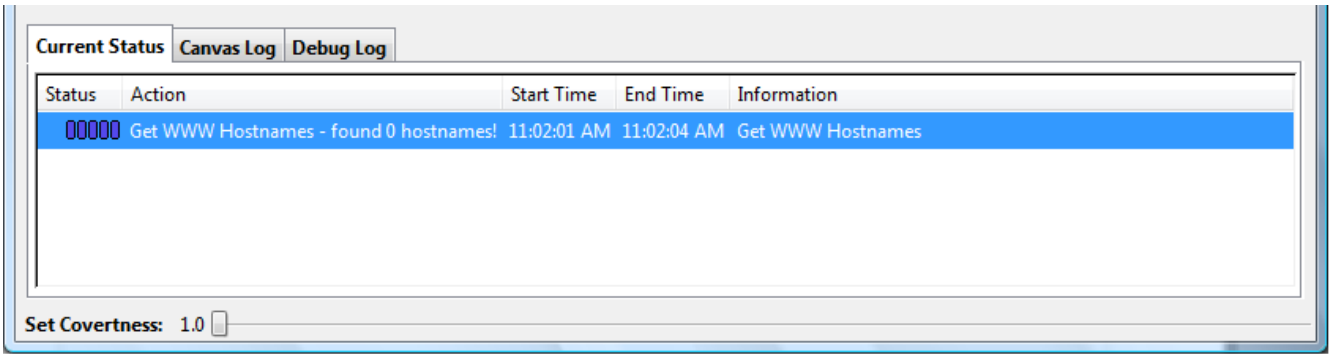


getwwwhostnames



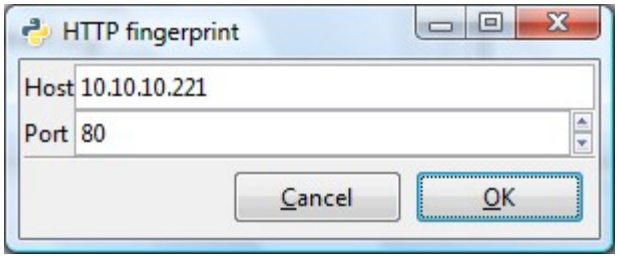
Supported Architectures: N/A

This tool can only be used three times in one calendar day. It does a hostname lookup against whois.webhosting.info which will add anything it finds in the appropriate Host: entry in the knowledge tree:



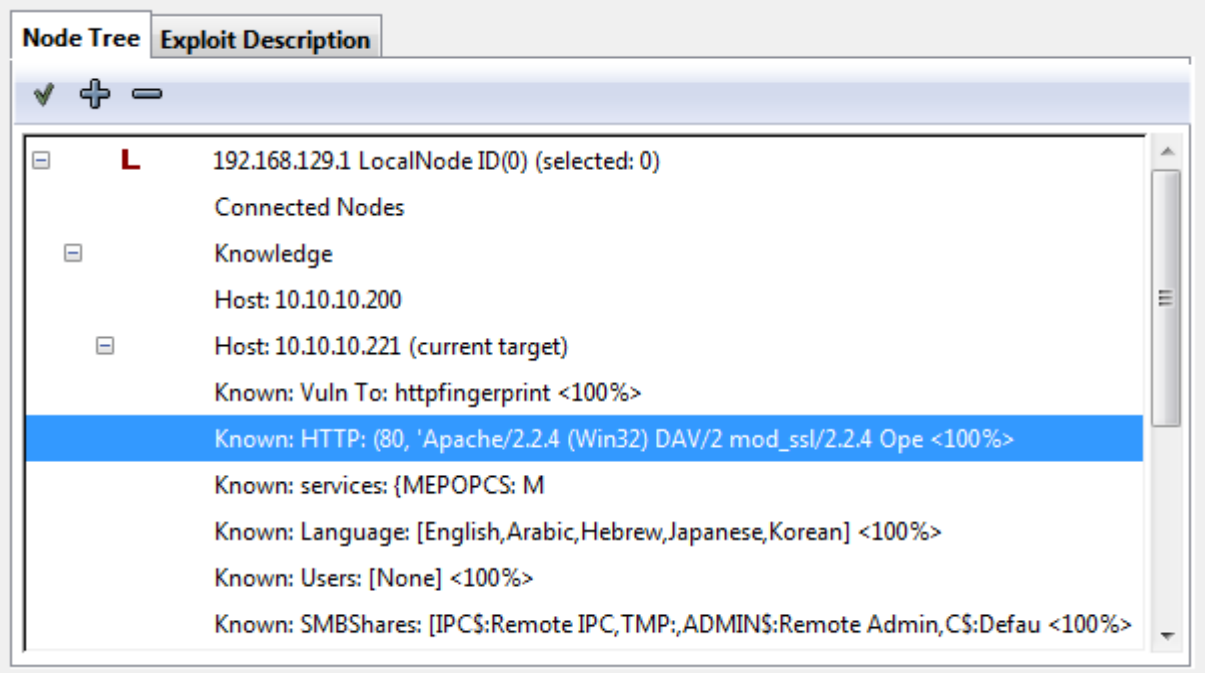
As you can see, this tool is not always successful if the whois database does not have the appropriate entry.

httpfingerprint

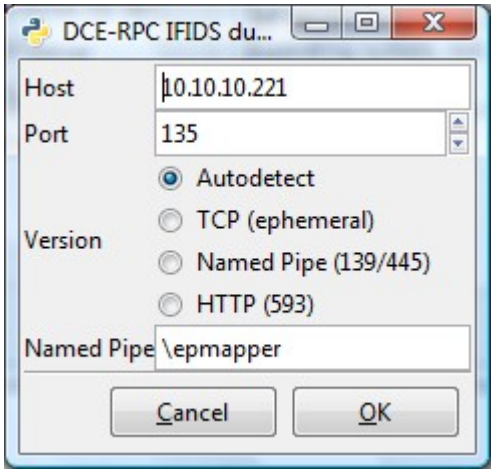


Supported Architectures: Any with the caveat that the host is running a Web Server

This module will grab the banner from a specified web server, if one is running on the host, and add the appropriate data under the Host: entry if any is found.

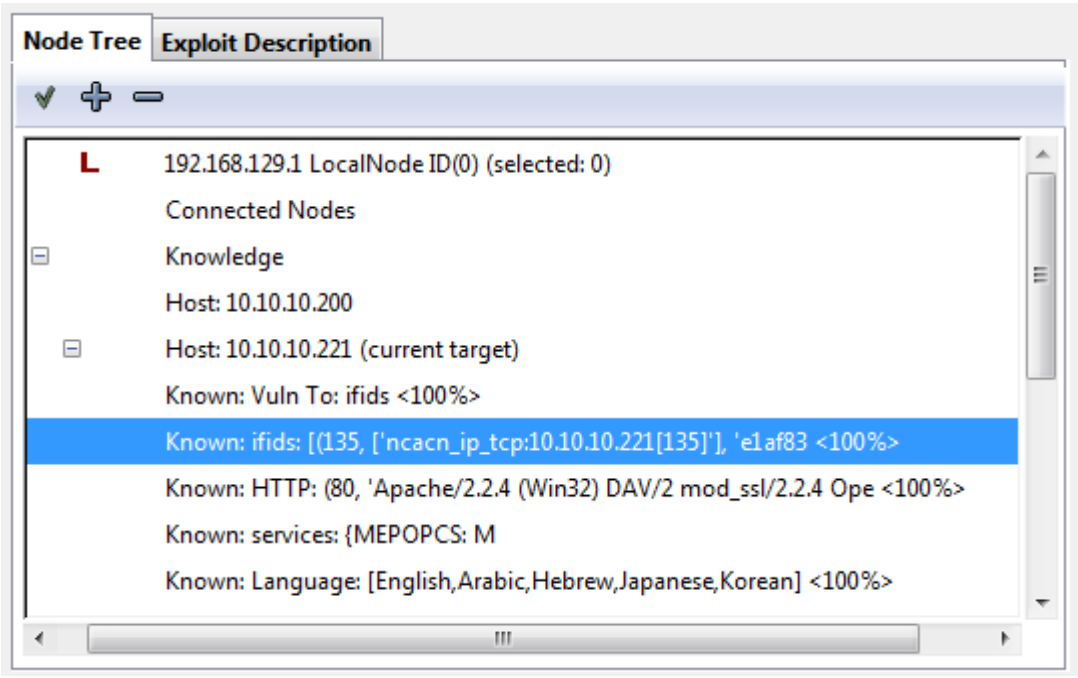


ifids (Interface ID's)

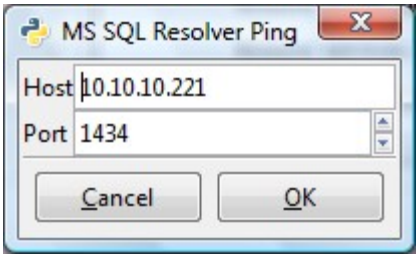


Supported Architectures: Any with port 135 open and an RPC Endpoint mapper listening, except XPSP2

ifids stands for InterFace ID's. If a port or named pipe is running DCE-RPC, then you can send a query to it asking it which endpoints it supports. This tool sends that query and returns the results. It also will brute force an endpoint that appears to be a DCE-RPC endpoint, but does not respond directly to the query by trying to bind to a list of interfaces.



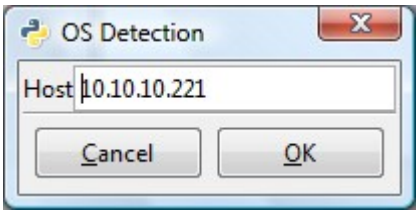
mssqlserver



Supported Architectures: Most versions of MSSQL Server

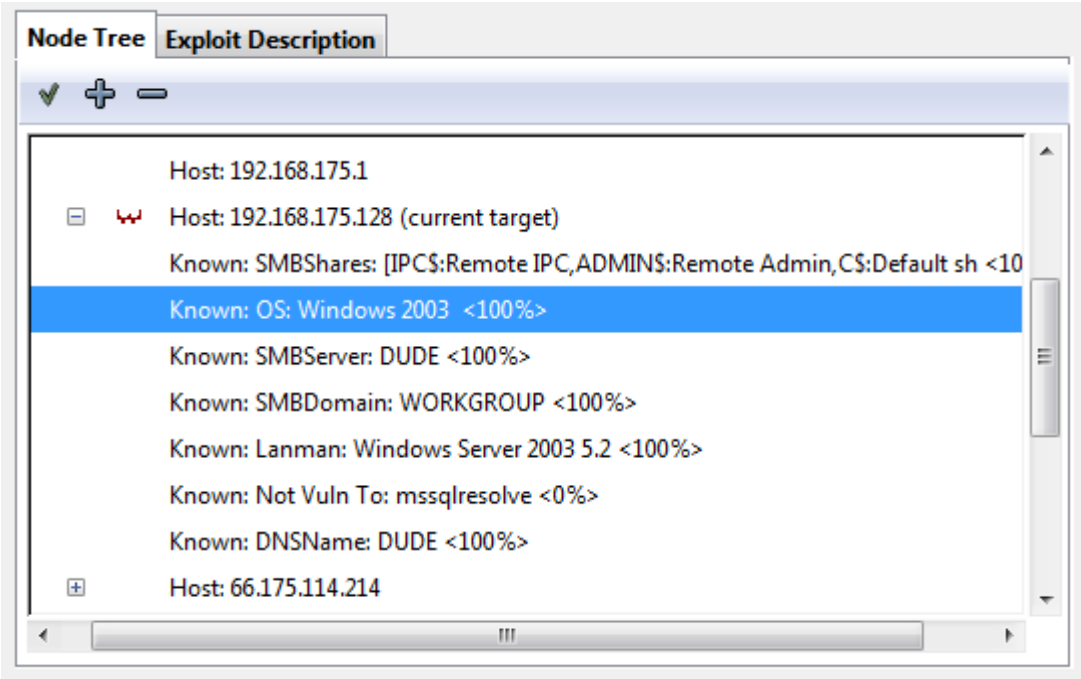
This utility sends a MS SQL Resolver Ping to the default 1434 port in order to try and detect the version running

osdetection

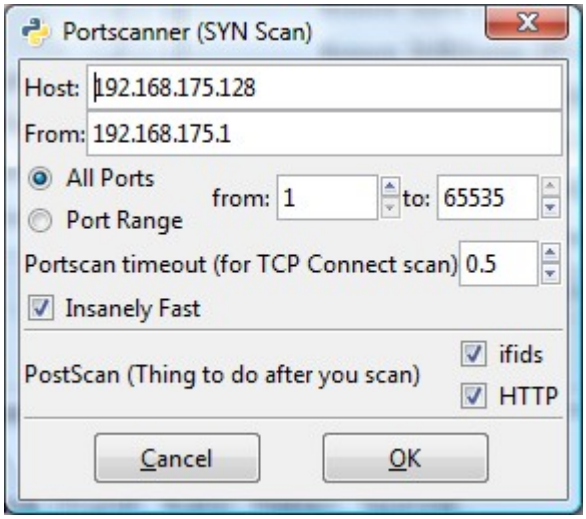


Supported Architectures: N/A

This module attempts to detect the architecture of the current target through various methods. If it is successful, it will update the Host: entry in the knowledge tree.

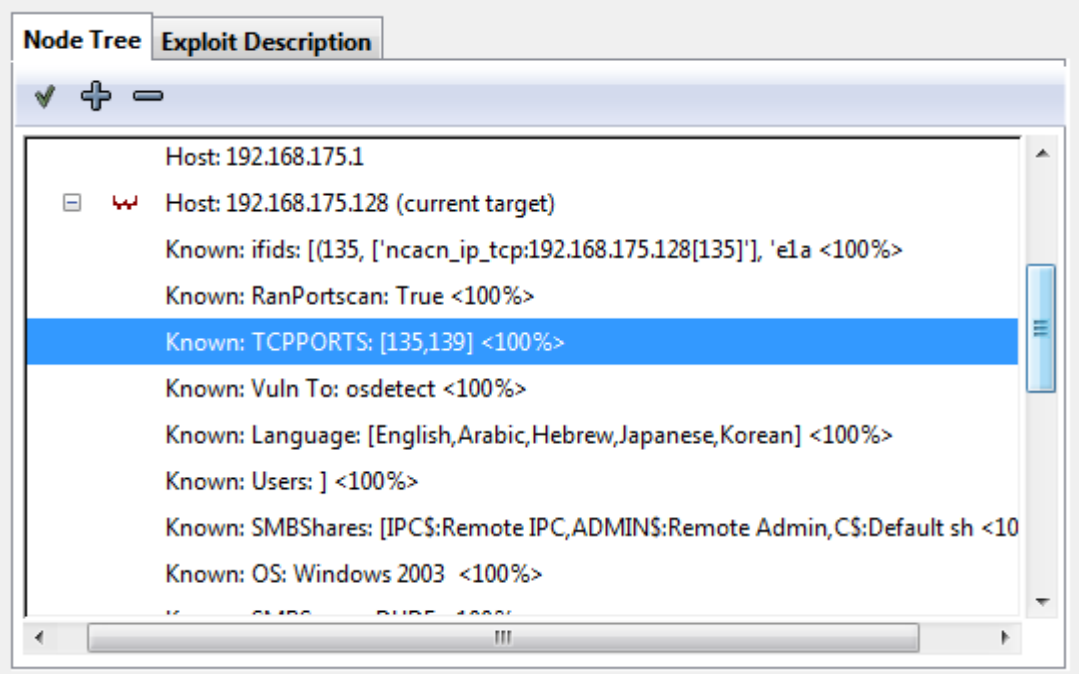


portscan

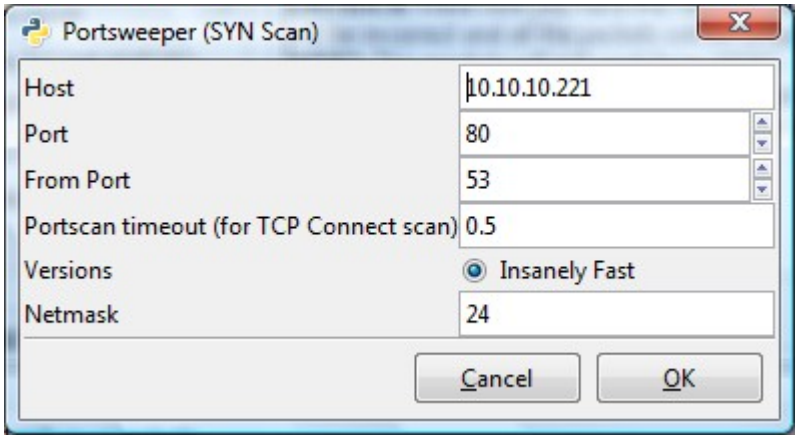


Supported Architectures: N/A

This module will run a TCP port scan on the range you specify by sending the first part of the TCP three-way-handshake (SYN) and listening for any port open responses (SYN/ACK) It will not report any closed or firewalled ports.



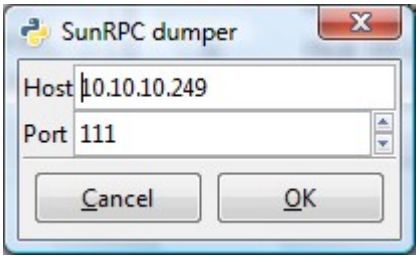
portsweeper



Supported Architectures: N/A

This module will scan a single port across many IP addresses. This is useful for telling us if any hosts are alive on victim subnet. The from port is set to 53 by default, because a lot of firewalls will allow this packet in, thinking it is a DNS query. If it finds any hosts it will create a new Host: entry for the associated IP address in the knowledge tree. Like the postscanner module, this will only create a Host: entry if a port is open,

rpcdump



Supported Architectures: Any architecture running SunRPC

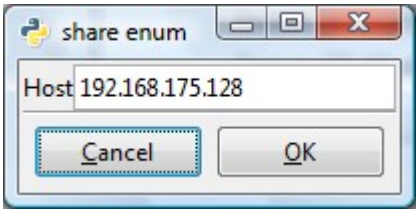
This utility is the Unix varient of the dcedump utility. The output does not put any data into the Host:

entry, but rather displays this often changing information into the Logging Windows, canvas.log file and displays it in the canvas.bat shell.

Here is an example of the output:

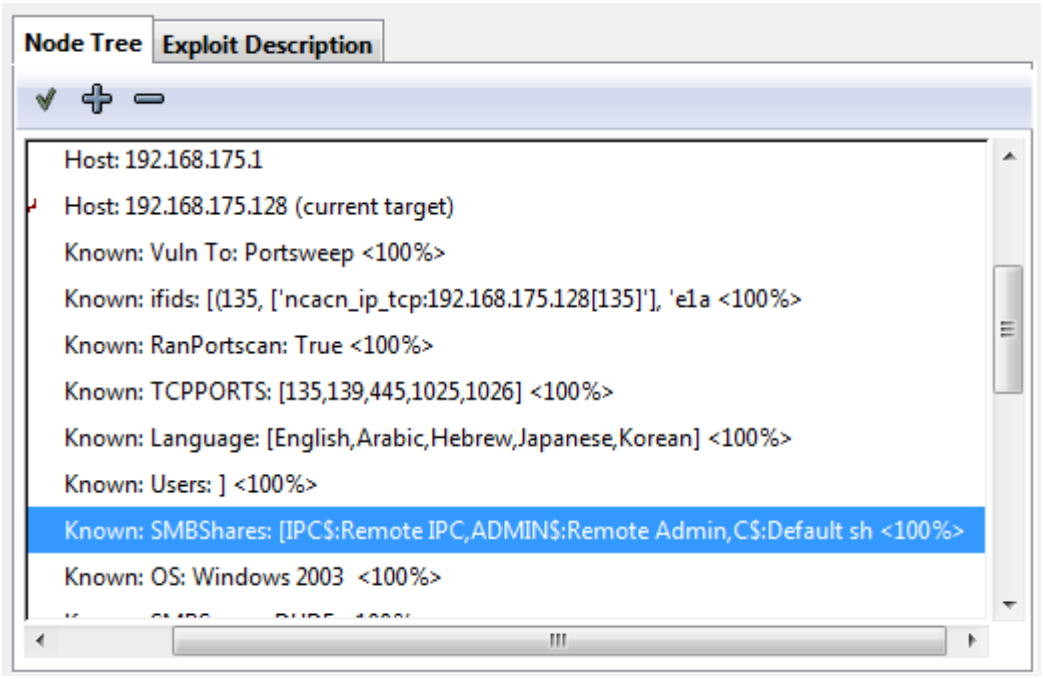
(10.10.10.249/32)	100242	rpc.metamedd	1	tcp	32788
(10.10.10.249/32)	100230	metamhd	1	tcp	32790
(10.10.10.249/32)	100001	rstatd	2	udp	32801
(10.10.10.249/32)	100001	rstatd	3	udp	32801
(10.10.10.249/32)	100001	rstatd	4	udp	32801
(10.10.10.249/32)	100002	rusersd	2	tcp	32792
(10.10.10.249/32)	100002	rusersd	3	tcp	32792
(10.10.10.249/32)	100002	rusersd	2	udp	32803
(10.10.10.249/32)	100002	rusersd	3	udp	32803
(10.10.10.249/32)	100011	rquotad	1	udp	32805

shareenum

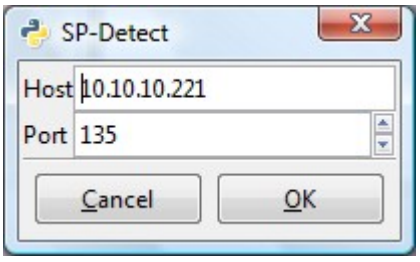


Supported Architectures: All Windows platforms with the file sharing ports open.

This module will find open shares on a windows server, including the administrative and hidden shares.

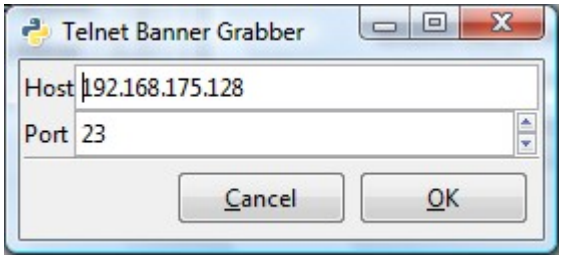


spdetect



Supported Architectures: All Windows

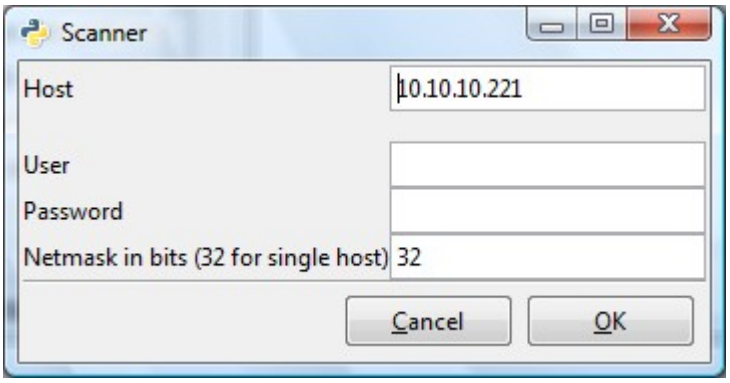
telnetbanner



Supported Architectures: N/A

This module is the telnet equivalent to the gethttpbanner module, except that it will grab a telnet servers banner instead.

testhost (One of the most important recon modules!)



Supported Architectures: All

This noisy, yet handy utility runs every CANVAS exploit against a host, with one caveat: rather than compromising the host my installing MOSDEF via the functional exploits, it will just let you know which exploits a host may be vulnerable to by printing a list in the Canvas Log screen in the information Window, the canvas.log file, and in the output of the canvas.bat shell. This is very handy if your rules of engagement states you are not allowed to actually compromise the the host. This exhaustive scan may take a few hours to run, and may produce a few false positives, but you can always get some lunch while its running, and you will have a short list of exploits that may work.

Below is an example of the output:

(192.168.175.128/32) Printing all vulnerabilities found!

(192.168.175.128/32) Host: 192.168.175.128 Vuln: MSRPC Crash

(192.168.175.128/32) Host: 192.168.175.128 Vuln: Microsoft Windows RPC Interface Overflow

(192.168.175.128/32) Host: 192.168.175.128 Vuln: SAMBA api_lsa_lookup_sids

(192.168.175.128/32) Host: 192.168.175.128 Vuln: Start Remote Service

(192.168.175.128/32) Host: 192.168.175.128 Vuln: AtSvc

(192.168.175.128/32) Host: 192.168.175.128 Vuln: MSSQL (Null) Auth Connect

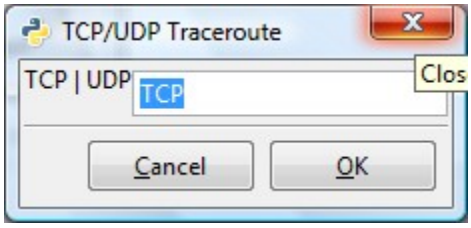
(192.168.175.128/32) Host: 192.168.175.128 Vuln: MSSQL Resolver Stack Overflow

(192.168.175.128/32) Host: 192.168.175.128 Vuln: MSRPC MESSENGER Heap Overflow

(192.168.175.128/32) Host: 192.168.175.128 Vuln: mailenable_imap

In this case you will notice that the SAMBA api_lsa module clearly produced a false positive. Immunity tries to keep false positives to a minimum, but the only way to completely remove them is to run the attack and get a MOSDEF Node!

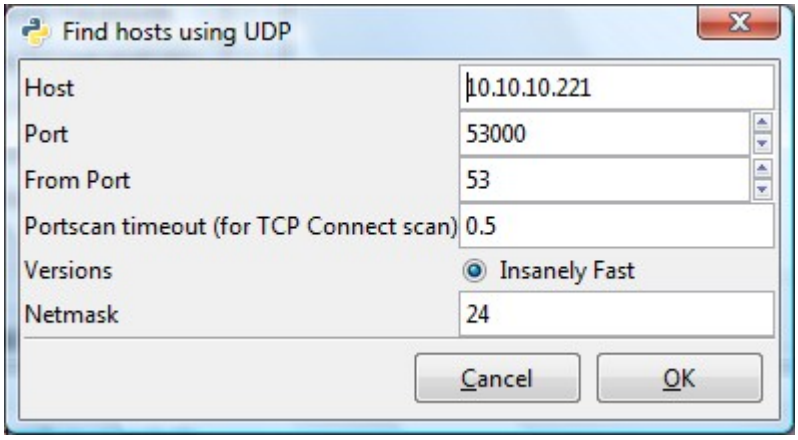
traceroute



Supported Architectures: N/A

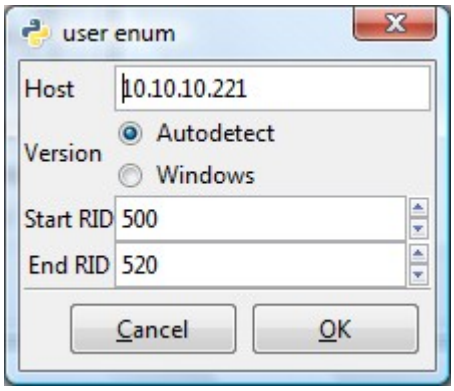
This is an improved version of the standard traceroute utility, if you're running as root under Linux. Under Windows, this will use the standard Windows tracert command to get a traceroute between yourself an the remote target.

udpsweep



Supported Architectures: N/A

userenum



This module will use MSRPC calls to get the remote usernames and groups from a Windows machine. These can then be brute forced for bad passwords.