

Project 3: *Malware Analysis*

CS 6262

Agenda

- Part 1: Analyzing Windows Malware
- Part 2: Analyzing Android Malware

Scenario

- Analyzing Windows Malware
 - You got a malware sample from the wild. Your task is to discover what the malware does by analyzing it
 - How do you discover the malware's behaviors?
 - Static Analysis
 - Manual Reverse Engineering
 - Programming binary analysis
 - Dynamic Analysis
 - Network behavioral tracing
 - Run-time system behavioral tracing(File/Process/Thread/Registry)
 - Symbolic Execution
 - Fuzzing

Scenario

- In our scenario, you are going to analyze the given malware with tools that we provide.
- These tools help you to analyze the malware with static and dynamic analysis.
- Objective
 1. Find which server controls the malware (the command and control (C2) server)
 2. Discover how the malware communicates with the command and control (C2) server
 - URL and Payload
 3. Discover what activities are done by the Linux malware
 - Attack activities

Scenario

- Requirement
 - Make sure that no malware traffic goes out from the virtual machine
 - But, updating the malware (stage 2), and downloading the Linux malware (stage 3) must be allowed for us to understand the malware's behavior
 - The command and control server is dead. You need to reconstruct it
 - Use tools to reconstruct the server, then reveal hidden behaviors of the malware
 - Analyze network traffic on the host, and figure out the list of available commands for the malware
 - Analyze network traffic trace of the host, and figure out what malware does
 - Write down your answer into assignment-questionnaire.txt

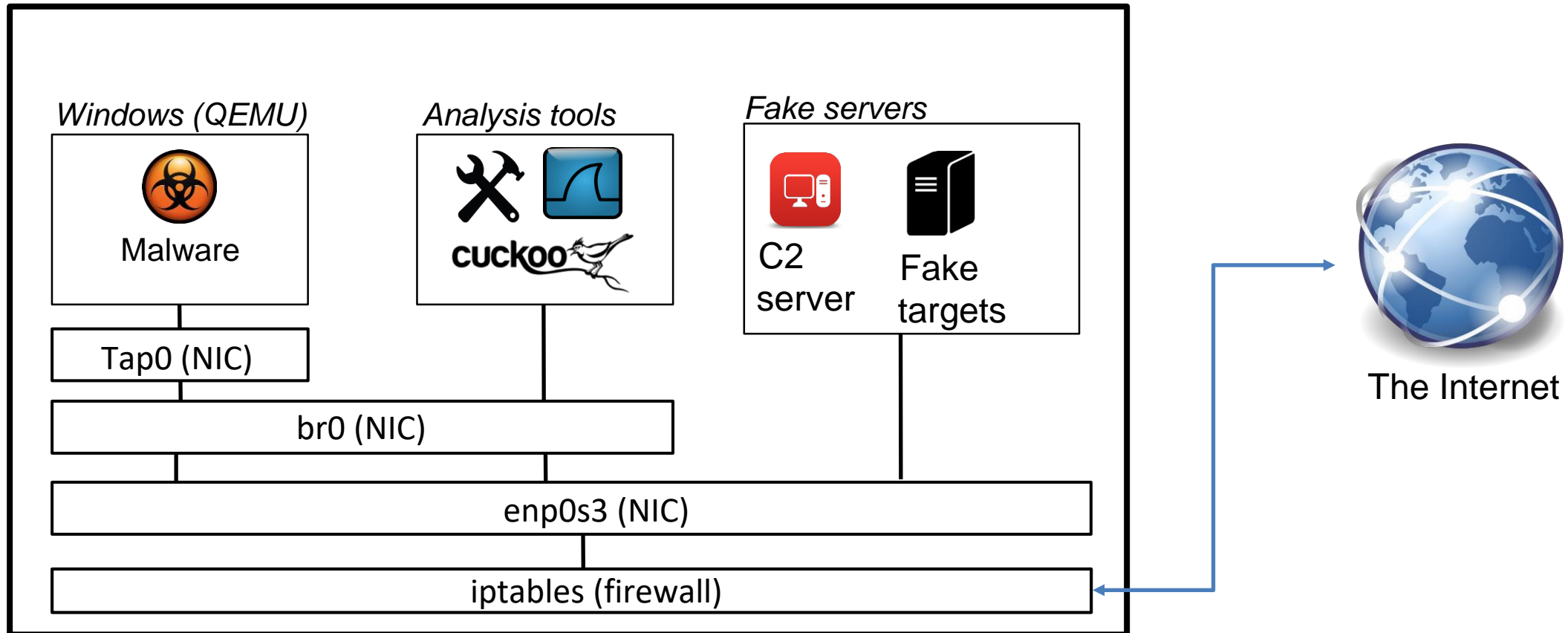
Project Structure

- A Virtual Machine for Malware analysis
 - Please install/update to the latest version of VirtualBox.
 - <https://www.virtualbox.org/wiki/Downloads>
- Download the VM
 - Download the project VM from one of the following links
 - https://gtvault-my.sharepoint.com/:u:/g/personal/vraymond6_gatech_edu/EeOzzyYd43FDrkx0sw8-xF0Bf4XWBPMHpdatP9gVayO59A?e=3l5kRw
 - MD5 Hash: 19E3E05638172762959C4C9E8D89E373

Project Structure

- Network Configurations

Ubuntu



Project Structure

- Network Configurations
 - tap0
 - Virtual network interface for Windows XP
 - IP Address: 192.168.133.101
 - br0
 - A network bridge between Windows XP and Ubuntu
 - IP Address: 192.168.133.1
 - enp0s3
 - A network that faces the Internet
 - IP Address: 10.0.2.15 (it varies with your VirtualBox settings)

Project Structure

- Open VirtualBox
 - Go to File->Import Appliance.
 - Select the ova file and import it.
 - For detailed information on how to import the VM, see:
 - https://docs.oracle.com/cd/E26217_01/E26796/html/qs-import-vm.html
- VM user credentials
 - Username: **analysis**
 - Password: **analysis**

Project Structure

- In the Virtual Machine (VM)
 - Files
 - `init.py`
 - This initializes the project environment
 - Type your Georgia Tech username (your Canvas login name) after running this
 - e.g \$./init.py
 - `update.sh`
 - This script updates the VM if any further update has been made by TAs
 - **Please run this script when you start the project! (If it says that you're already updated when you run it, that's fine)**
 - **If you have already completed stage 1 before running update.sh, you do NOT need to redo stage 1 – but you will need to run update.sh to complete stage 2**
 - `archive.sh`
 - This will archive the answer sheet for submission (create a zip file)

Project Structure

- In the Virtual Machine (VM)
 - Directories
 - `vm`
 - A directory that stores the Windows XP virtual machine (runs with QEMU)
 - We use the given VM for both Cuckoo and a testbed. Please see page 19.
 - `shared`
 - A shared directory between the Ubuntu host and Windows guest (XP is running on a VM within your project VM). You can copy/move files to or from this directory.
 - Please see page 22.
 - `report`
 - The answer sheet for project questionnaire.
 - `setup`
 - Required files for setting up the machine. You don't need to modify, nor use the files in this directory.

Project Structure

- In the Virtual Machine (VM)
 - Directories
 - tools
 - network
 - Configure your network firewall rules (iptables) by editing iptables-rules.
 - You can allow/disallow/redirect the traffic from the malware
 - './reset' command in this directory will apply the changes
 - **cfg-generation** (CFG stands for Control-Flow Graph)
 - An analysis tool that helps you to find interesting functions of malicious activity
 - You need to edit score.h to generate the control-flow graph
 - Use xdot to open the generated CFG.

Project Structure

- In the Virtual Machine (VM)
 - Directories
 - tools
 - sym-exec
 - A symbolic executor (based on angr: <https://github.com/angr>)
 - Helps you to figure out the commands that malware expects
 - Use cfg-generation tool to figure out the address of the function of interests
 - c2-command
 - A simplified tool for C2 server reconstruction
 - You can write down command in the *.txt file as a line
 - It will randomly choose one command at a time to send to the malware

Project Structure

- Malware
 - stage1.exe – stage 1 malware
 - It will download the stage 2 malware if this malware receives the correct command
 - stage2.exe – stage 2 malware
 - It will download the stage 3 malware if this malware receives the correct command
 - payload.exe – the linux malware attack payload
 - Analyze the dynamic instruction trace
 - Write a script to detect where the C&C communication happens – Find the loop entry point and function sequence in the loop
 - Add constraint to symbolic execution to limit the loop to one
 - Find the feasible attacks within given set of possible attacks.

Questionnaire

- **1) To get credit for the project, you have to answer the questionnaire, found at ~/report/assignment-questionnaire.txt !!!!!**
- **2) Please strictly follow the format or the example answer for each question in assignment-questionnaire.txt. TAs use a autograder for your submission.**
- Windows Part
 - Read ~/report/assignment-questionnaire.txt
 - Carefully read the questions, and answer them in ~/report/assignment-questionnaire.txt
 - For each stage, there are 4-6 questions regarding the behavior of the malware.
- Android Part
 - READ ~/Android/MaliciousMessenger/writeup.pdf
 - Carefully read the writeup, answer in ~/report/assignment-questionnaire.txt

Submitting Questionnaire

- Required files
 - Zip the following files and upload to Canvas
 - Running ~/archive.sh will automatically zip all of the files
 - ~/report/assignment-questionnaire.txt
 - Stage1.exe, stage2.exe, payload.exe (linux malware)
 - ~/tools/network/iptables_rules
 - ~/tools/cfg-generation/score.h
 - Running ~/archive.sh will create report.zip **automatically**
 - Please check the content of your zip file before submitting it to Canvas

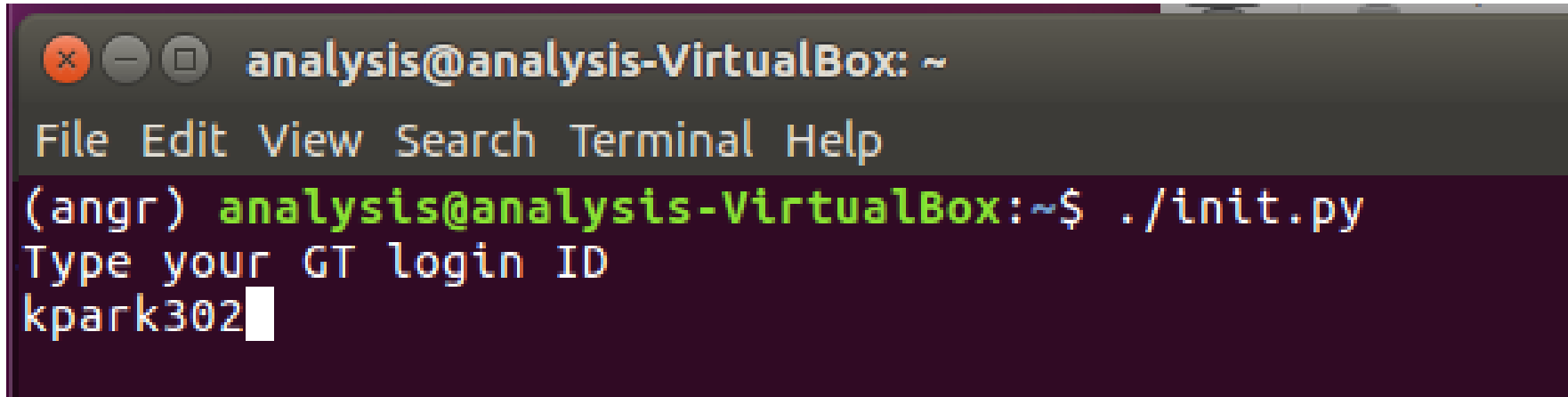
Tutorial (for stage1.exe malware)

- Update the project 3 before begin.
 - Open the terminal (Ctrl-Alt-T, or choose terminal from the menu)
 - Run, ./update.sh
 - It will update any necessary files that are required for this project.

```
(angr) analysis@analysis-VirtualBox:~$ ./update.sh
Already up-to-date.
UPDATE FIN
```

Tutorial (for stage1.exe malware)

- Initializing the project
 - Open the terminal (Ctrl-Alt-T, or choose terminal from the menu)
 - Run `./init.py`
 - Type your Georgia Tech username (the login name used for Canvas)
 - This will download the stage1 malware (stage1.exe) into the `~/shared` directory



```
analysis@analysis-VirtualBox: ~
File Edit View Search Terminal Help
(angr) analysis@analysis-VirtualBox:~$ ./init.py
Type your GT login ID
kpark302
```

Tutorial (for stage1.exe malware)

- **Special NOTE**

- These are malware samples hosted under the Goergia Tech Network
- It is likely that security measures would kick in and encrypt these files
 - That is, all the malware samples you will be downloading during this project

- **IMPORTANT**

- After each download, make sure to check the type of file.
- In the linux VM, execute
 - \$ file <path-to-exe>
- If the result of that is an archive of some sort then execute:
 - unzip <path-to-exe>
 - Password: infected

```
(angr) analysis@analysis-VirtualBox:~/shared$ file stage1.exe
stage1.exe: Zip archive data, at least v2.0 to extract
(angr) analysis@analysis-VirtualBox:~/shared$ unzip stage1.exe
Archive:  stage1.exe
[stage1.exe] stage1.exe password:
replace stage1.exe? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: stage1.exe
(angr) analysis@analysis-VirtualBox:~/shared$
```

Tutorial (for stage1.exe malware)

- **Special NOTE**

- For stage1 and stage2, the file format should be

```
(angr) analysis@analysis-VirtualBox:~/shared$ file stage1.exe
stage1.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

- For stage3, the file format should be

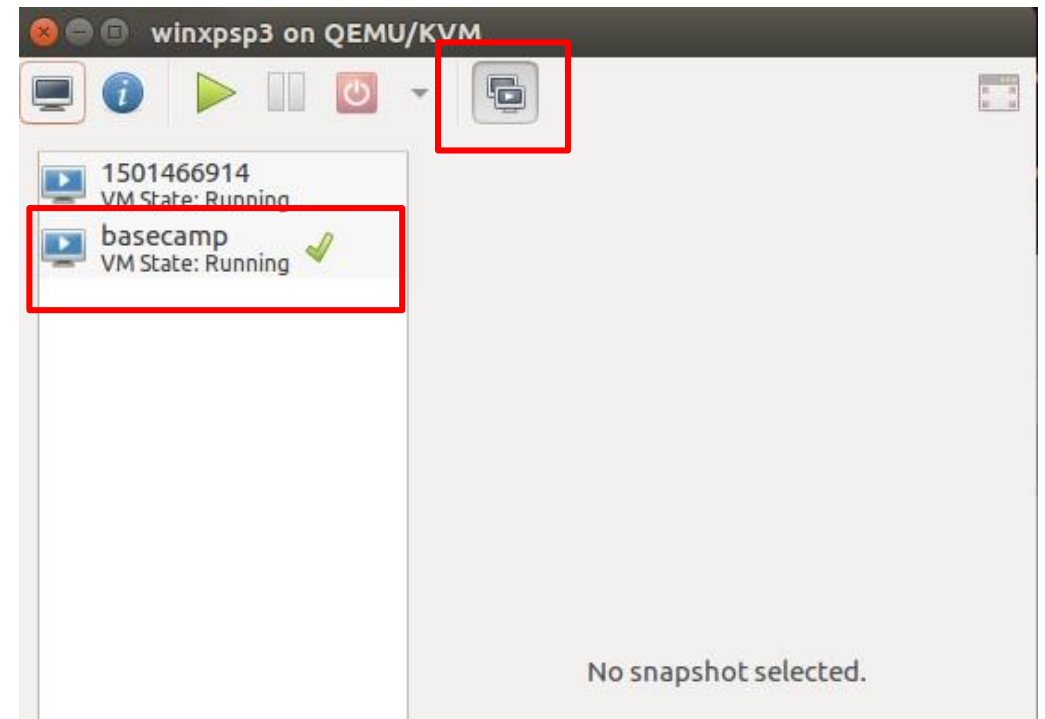
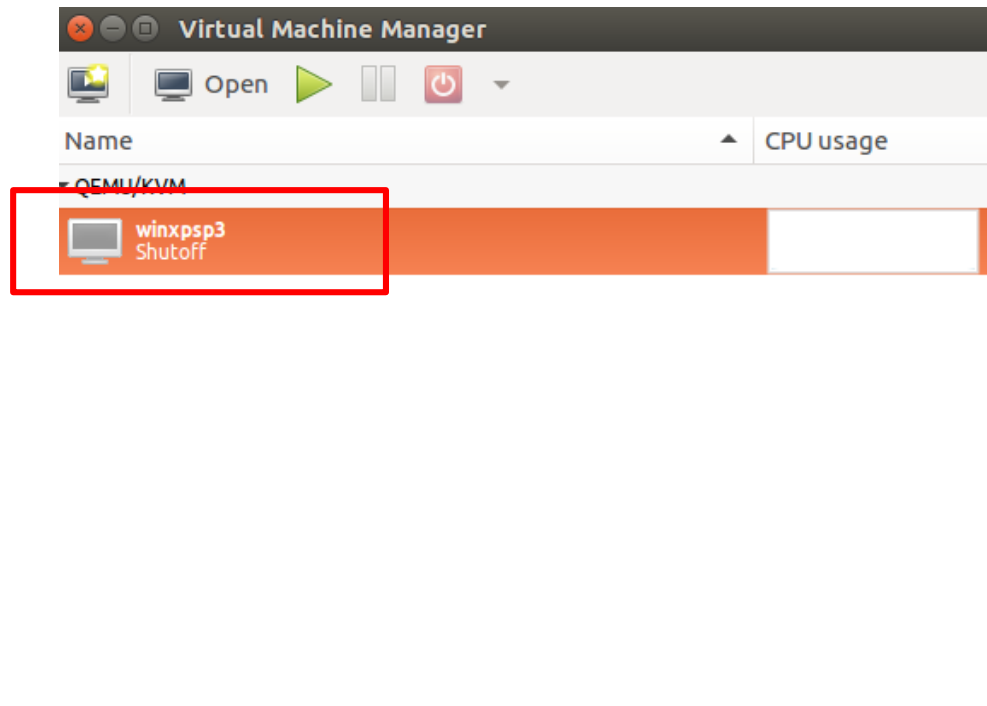
```
(angr) analysis@analysis-VirtualBox:~/shared$ file payload
payload: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, not stripped
```

Tutorial - Secure Experiment Environment

- We need a secure experiment environment to execute the malware.
- Why?
 - Insecure analysis environment could damage your system
 - You may not want:
 - Encrypting your file during a ransomware analysis
 - Infecting machines in your corporate network during a worm analysis
 - Creating a tons of infected bot client in your network during a bot/trojan analysis
- The solution:
 - Contain malware in a virtual environment
 - Virtual Machine
 - Virtual Network
 - Conservative rules(allow network traffic only if it is secure)
- We provide a Win XP VM as a testbed!

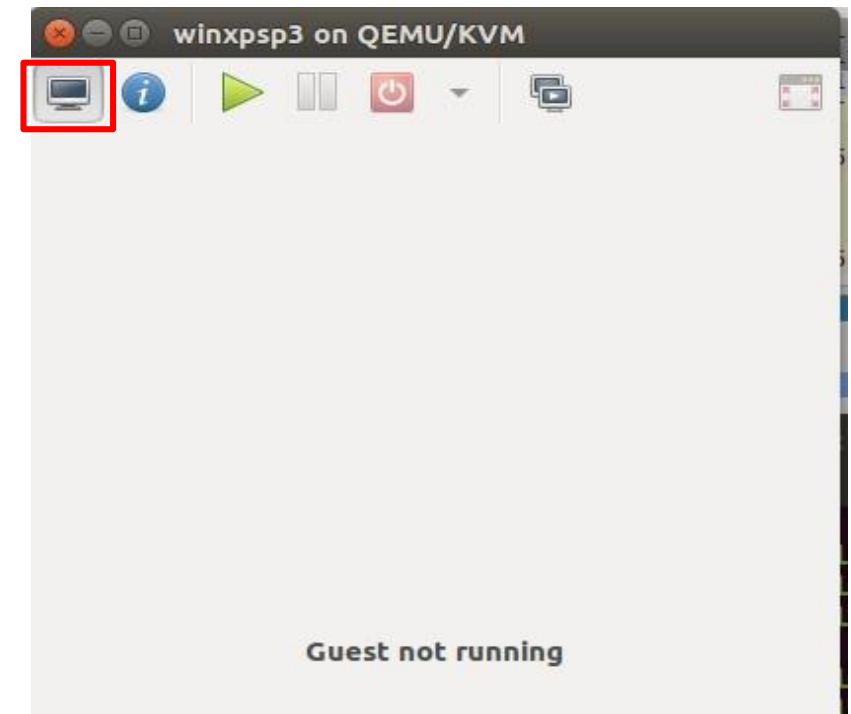
Tutorial - Run Win XP VM

- Run Windows XP Virtual Machine with virt-manager
 - Open a terminal
 - Type "virt-manager" and double click "winxpsp3"
 - Click the icon with the two monitors and click on "basecamp"



Tutorial - Run Win XP VM

- Run Windows XP Virtual Machine with virt-manager
 - Right click on basecamp, and click **"Start snapshot."** Click Yes if prompted.
 - Once, virt-manager successfully calls the snapshot, click **Show the graphical console.**
 - Click on the Windows Start Menu and Turn off Computer.
 - Then select Restart

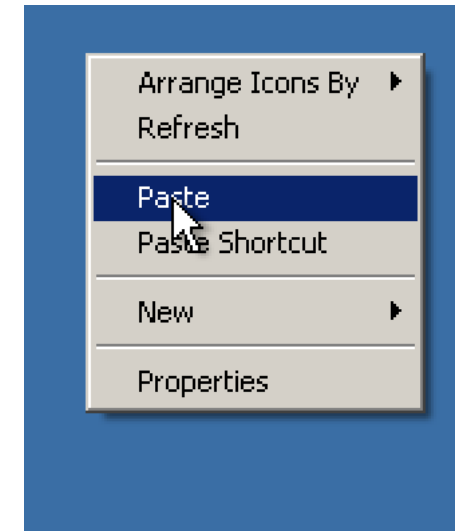
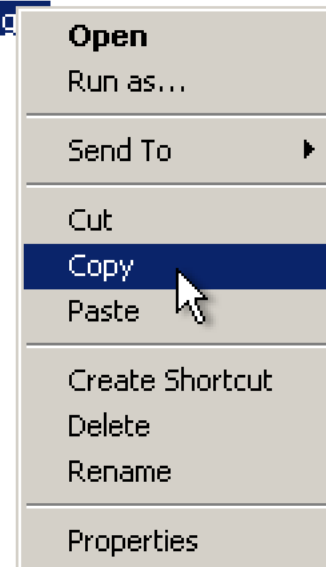
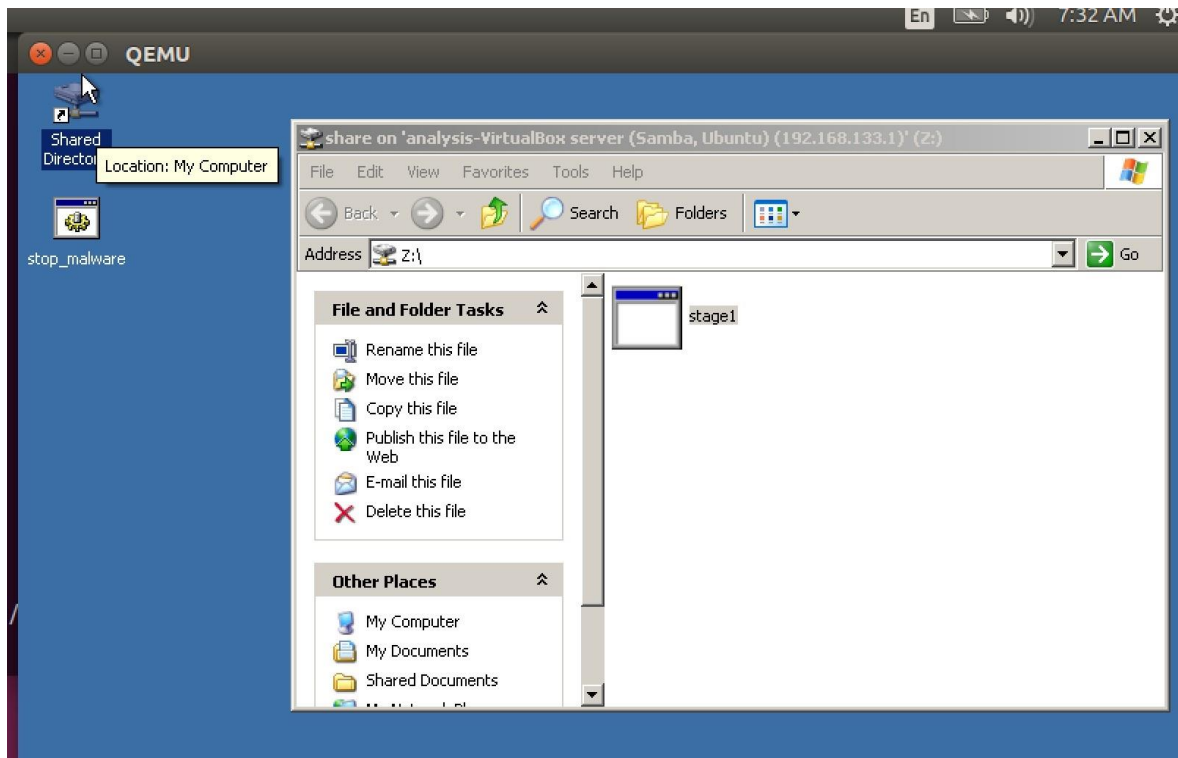


Tutorial - Run Win XP VM

- **DO NOT MODIFY OR DELETE THE GIVEN SNAPSHOTS!**
- The given snapshots are your backups for your analysis.
- If something bad happens on your testbed, always revert back to the basecamp snapshot.

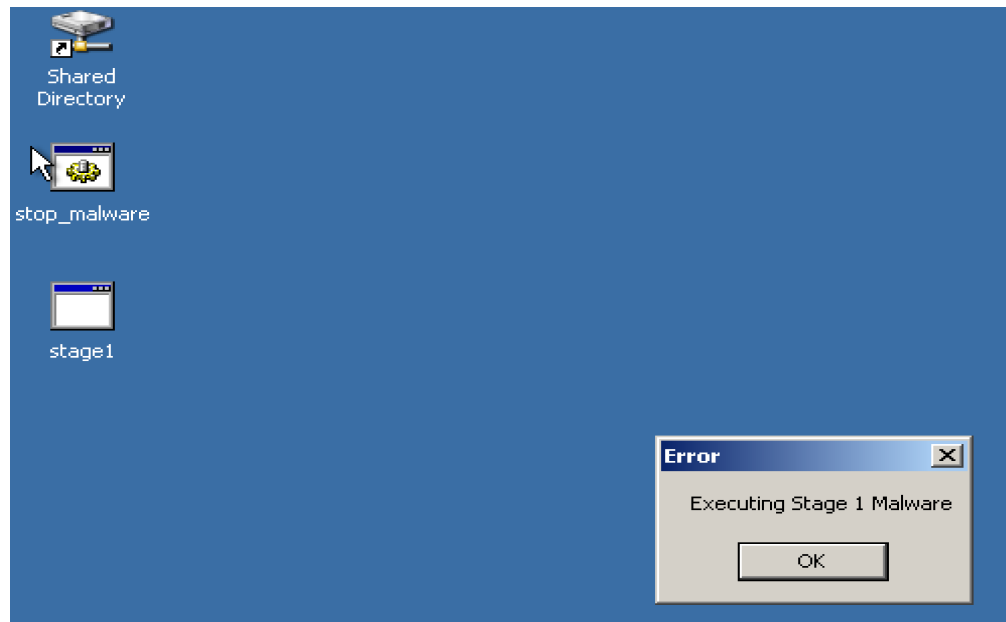
Tutorial - Copy from Shared Directory

- Go to the shared directory by clicking its icon (in Windows XP)
 - Copy stage1.exe into Desktop
 - If you execute it in the shared directory, the error message will pop up.
Please copy the file to Desktop.



Tutorial - Run the malware!

- Now we will run the malware
 - Execute stage1.exe (double click the icon)
 - It will say “Executing Stage 1 Malware”. Then, click OK.
 - You should click OK on each dialog to dismiss it
 - Otherwise, malware execution will be blocked



Tutorial - Run the malware!

- If you want to halt the malware that is running...
 - Execute stop_malware in the temp directory.
 - This will stop the currently running malware.
 - Please halt first before you execute another malware file.

Tutorial - Network behavioral analysis

- To analyze network behaviors, you need
 - Wireshark (<https://www.wireshark.org/>)
 - Network Protocol Analyzer
 - Cuckoo (<https://cuckoosandbox.org/>)
 - Capturing & Recording inbound/outbound network packets

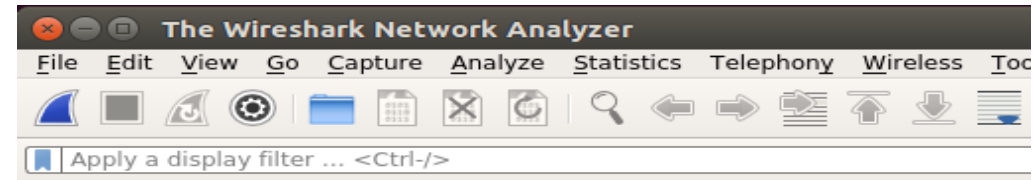
Tutorial - Observing Network Behavior

- By capturing and recording network packets through the tools,
 - Reveal C&C protocol
 - Attack Source & Destination
- **But, malware will not do anything. Why?**
 - The C2 server is dead!
 - Therefore, the malware(C2 client) will never unfold its behaviors.
 - Question?
 - If we know C&C dialog of malware, can we build a fake C2 server in order to unfold the malware behaviors?
 - Answer: Hack Yeah! That is your job for this project!

Tutorial - Wireshark

- Let's check it through network monitoring
 - Open wireshark (open a terminal. Type "sudo wireshark" – you can ignore the error message that pops up)
 - Choose br0 to capture the network traffic
 - Then start capture by clicking on the shark-fin on the top left

```
analysis@analysis-VirtualBox: ~  
(anгр) analysis@analysis-VirtualBox:~$ sudo wireshark
```



Welcome to Wireshark

Capture

...using this filter:

br0	
vnet0	
enp0s3	
any	
Loopback: lo	
tap0	
virbr0	
nflog	
nfqueue	
ushmon1	

Tutorial - Redirect Network Connection

- Redirecting Network Connection

- From WireShark, we can notice that the malware tries to connect to the host at 128.61.240.66, but it fails

1	0.000000000	192.168.133.101	128.61.240.66	TCP	62	1047 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
2	2.927100732	192.168.133.101	128.61.240.66	TCP	62	[TCP Retransmission] 1047 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
3	9.005808865	192.168.133.101	128.61.240.66	TCP	62	[TCP Retransmission] 1047 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1

- Let's make it redirect to our fake C2 server
 - Go to ~/tools/network
 - Edit iptables_rules to redirect the traffic to 128.61.240.66 to 192.168.133.1 (fake host)
 - Whenever you edit iptables_rules, always run reset. (type “./reset” from the ~/tools/network directory)
 - **IMPORTANT!** If you shut down your project VM, be sure to run reset again the next time you start it up.

```
(angr) analysis@analysis-VirtualBox:~/tools/network$ cd ~
(angr) analysis@analysis-VirtualBox:~$ cd tools/network/
(angr) analysis@analysis-VirtualBox:~/tools/network$ ls
iptables-rules  reset
(angr) analysis@analysis-VirtualBox:~/tools/network$ vim iptables-rules
```

```
analysis@analysis-VirtualBox: ~/tools/network
1 !/bin/sh
2 echo "Importing iptables rules"
3
4 # Do not forget to run ./reset after updating this.
5
6 # Uncomment following lines (or add rules) to adjust filtering rules
7 #sudo iptables -t nat -A PREROUTING -p tcp -s 192.168.133.101 -d 143.215.206.97
  --dport 80 -j DNAT --to 192.168.133.1:80
8 sudo iptables -t nat -A PREROUTING -p tcp -s 192.168.133.101 -d 128.61.240.66 --
  dport 80 -j DNAT --to 192.168.133.1:80
```

Tutorial - Reading C2 Traffic

- Observing C2 traffic
 - In WireShark, we can notice that now the malware can communicate with our fake C2 server
 - But there will not be further execution, because the command is wrong...

526	700.559453229	192.168.133.101	128.61.240.66	HTTP	849 POST /images/logo/header.php HTTP/1.1 (application/x-www-form-urlencoded)
527	700.560517510	128.61.240.66	192.168.133.101	HTTP	387 HTTP/1.1 200 OK (text/html)
528	700.722188696	192.168.133.101	128.61.240.66	TCP	60 1043 → 80 [ACK] Seq=12726 Ack=5329 Win=65202 Len=0
529	705.569407055	192.168.133.101	128.61.240.66	HTTP	849 POST /images/logo/header.php HTTP/1.1 (application/x-www-form-urlencoded)
530	705.570186210	128.61.240.66	192.168.133.101	HTTP	387 HTTP/1.1 200 OK (text/html)
531	705.640570547	192.168.133.101	128.61.240.66	TCP	60 1043 → 80 [ACK] Seq=12726 Ack=5329 Win=65202 Len=0

Tutorial - Reading C2 Traffic

- Observing C2 traffic
 - You can see the contents of the traffic by right-clicking on the line, then clicking Follow – TCP Stream

526	700.559453229	192.168.133.101	128.61.240.66	HTTP	849	POST /images/logo/header.php HTTP/1.1 (application/x-www-form-urlencoded)
527	700.560517510	128.61.240.66	192.168.133.101	HTTP	387	HTTP/1.1 200 OK (text/html)
528	700.722188696	192.168.133.101	128.61.240.66	TCP	60	1043 → 80 [ACK] Seq=12726 Ack=5329 Win=65202 Len=0
529	705.569407055	192.168.133.101	128.61.240.66	HTTP	849	POST /images/logo/header.php HTTP/1.1 (application/x-www-form-urlencoded)
530	705.570186210	128.61.240.66	192.168.133.101	HTTP	387	HTTP/1.1 200 OK (text/html)

528	700.722188696	192.168.133.101	128.61.240.66	TCP	60	1043
529	705.569407055	192.168.133.101	128.61.240.66	HTTP	849	POST
530	705.570186210	128.61.240.66	192.168.133.101	HTTP	387	HTTP
531	705.570186210	128.61.240.66	192.168.133.101	TCP	60	1043
532	705.570186210	128.61.240.66	192.168.133.101	HTTP	849	POST

Mark/Unmark Packet
Ctrl+M
Ignore/Unignore Packet
Ctrl+D
Set/Unset Time Reference
Ctrl+T
Time Shift...
Ctrl+Shift+T
Packet Comment...
Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window

Ctrl+M
Ctrl+D
Ctrl+T
Ctrl+Shift+T
Ctrl+Alt+C
TCP Stream
UDP Stream
SSL Stream
HTTP Stream

```
POST /images/logo/header.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Host: netscan.gtisc.gatech.edu
Content-Length: 509
Cache-Control: no-cache
Cookie: response=%24updatE%3Aexecute-command
```

```
page=ZjllYWNiZWUtNTJjZS00ZmVhLTkzMdAtOTUxOGVhYWQ5MjVl&unm=dA&cnm=R1Qt0DVGM0E5NDgwNjQ1&query=
V2luZG93cyBYUA==&spec=MzIgQm10&opt=MA&view=W1N5c3RlbSBQcm9jZXNzXQpTeXN0ZW0Kc21zcy5leGUKY3Nyc
3MuZXh1CndpbmxvZ29uLmV4ZQpzZXJ2aWwN1cy5leGUKbHNhc3MuZXh1CnN2Y2hvc3QuZXh1CnN2Y2hvc3QuZXh1CnN2Y
2hvc3QuZXh1CnN2Y2hvc3QuZXh1CnN2Y2hvc3QuZXh1CnNwb29sc3YuZXh1CnN2Y2hvc3QuZXh1Cnd1YXVjbHQuZXh1C
mV4cGxvcmVlV4ZQphbGcuZXh1CndzY250ZnkuZXh1CmN0Zm1vbi5leGUKd21pcHJ2c2UuZXh1CmllcHBsb3JlLmV4Z
QppZXhwbG9yZS5leGUK&var=TWJfJaGluZXN0Zm1vbi5leGUKd21pcHJ2c2UuZXh1CmllcHBsb3JlLmV4Z
Server: nginx/1.10.3 (Ubuntu)
Date: Sat, 17 Feb 2018 04:28:58 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: response=None%3Aexecute-command; expires=Sun, 18-Feb-2018 04:28:58 GMT; Max-
Age=86400; path=/; domain=netscan.gtisc.gatech.edu
```

```
4
None
0
```

Tutorial - Cuckoo

- Let's take a look at cuckoo. Cuckoo is **NOT necessarily required to complete this project**, but it is a useful tool to help you understand what your malware is doing, and therefore how you might want to modify your score.h file later in the project.
 - NOTE! **You can't run the testbed vm and cuckoo simultaneously.**
 - **Always turn off** the testbed vm, and follow the steps below to execute Cuckoo
 - Open two terminals.
 - \$workon cuckoo #Set virtualenv as cuckoo for both terminal1 and terminal2
 - \$cuckoo -d #To run cuckoo daemon for terminal1
 - \$cuckoo web #To run cuckoo webserver for terminal2

If you get an error when running cuckoo web because port 8000
Is already in use, run "sudo fuser -k 8000/tcp" and try again

```
(angr) analysis@analysis-VirtualBox:~$ workon cuckoo
(cuckoo) analysis@analysis-VirtualBox:~$ cuckoo -d
```



Cuckoo Sandbox 2.0.5
www.cuckoosandbox.org
Copyright (c) 2010-2017

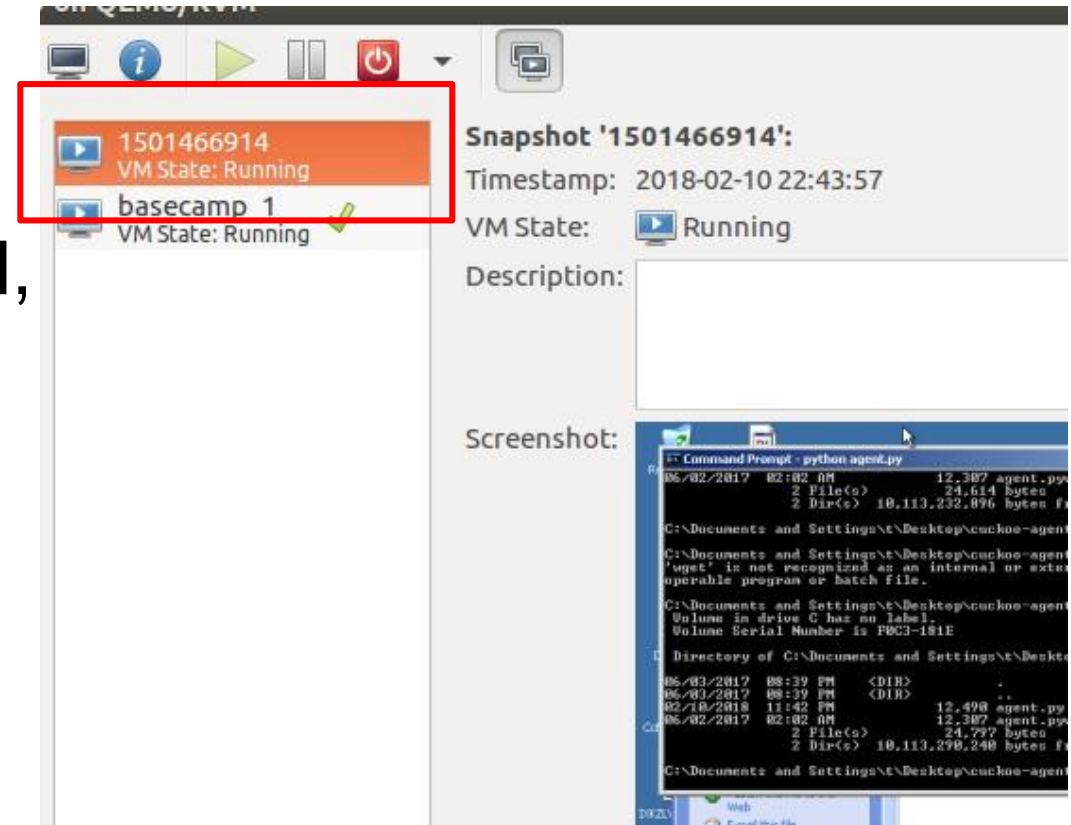
Checking for updates...

```
(cuckoo) [gtisc@gtisc-McSema ~]$ cuckoo web
Performing system checks...

System check identified no issues (0 silenced).
July 24, 2017 - 20:11:49
Django version 1.8.4, using settings 'cuckoo.web.web.se
Starting development server at http://localhost:8000/
Quit the server with CONTROL-C.
```

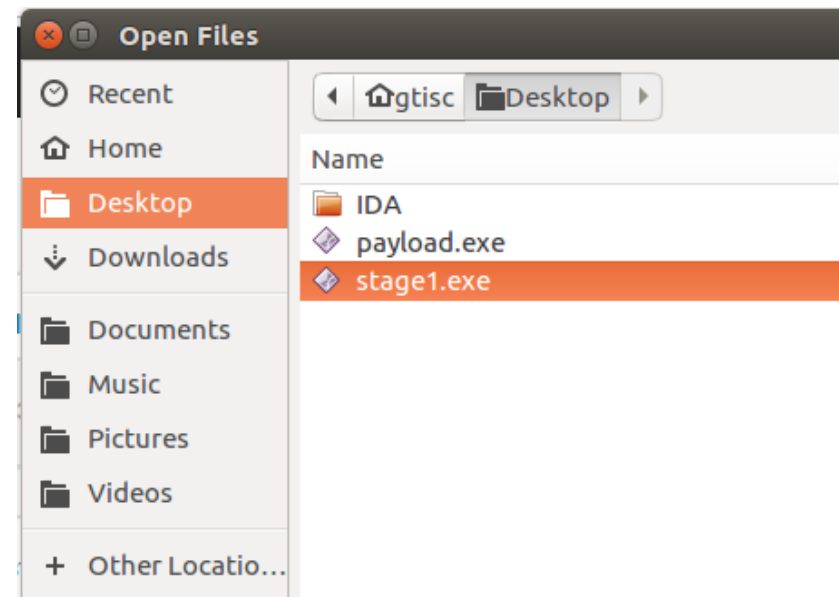
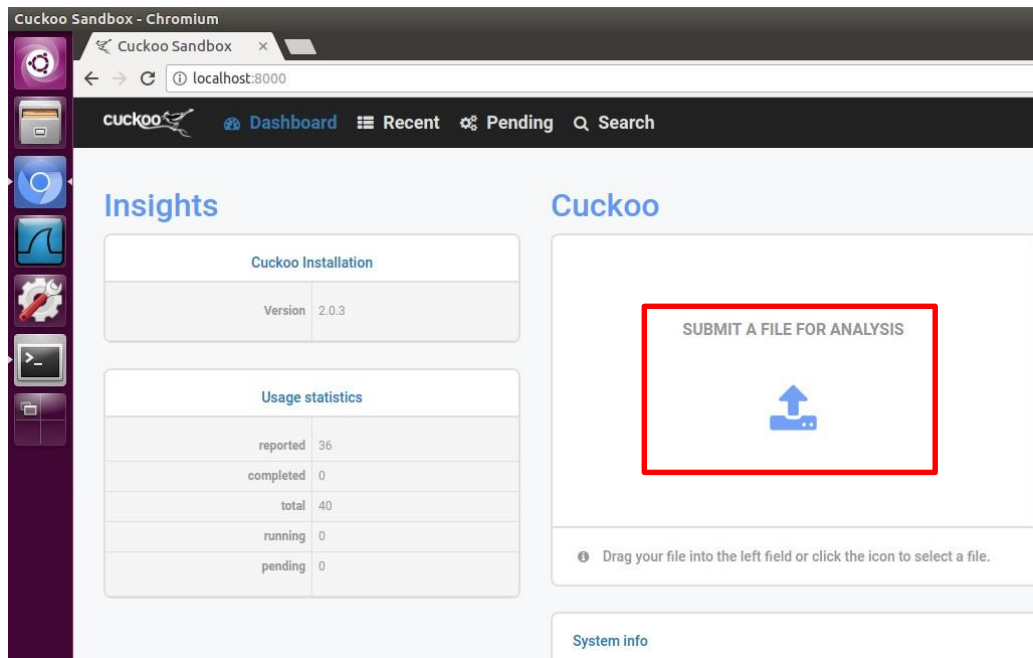
Tutorial - Cuckoo

- The Cuckoo uses a snapshot of the given testbed VM.
- The snapshot is 1501466914
- **DO NOT TOUCH** the snapshot!
- When you want to restore the test VM,
 - Refer to page 19.



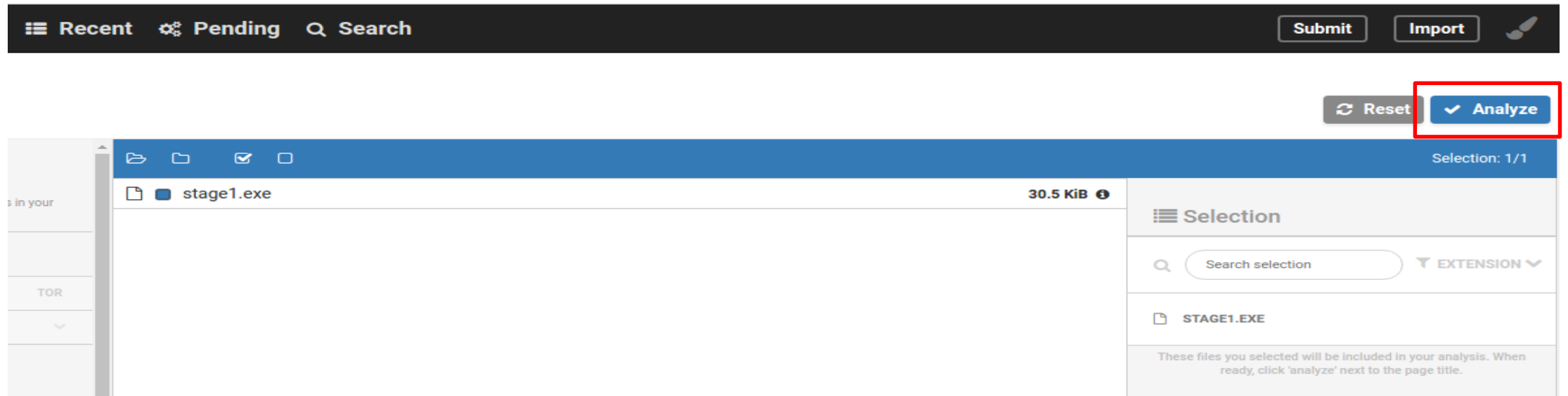
Tutorial - Upload a file to Cuckoo

- To open the cuckoo web server, type the following URL into Chromium
 - `http://localhost:8000`
- To upload a file, click the red box and choose a file.



Tutorial - Analysis with Cuckoo

- Once you click the Analyze button, it will take some time to run the malware.



Tutorial - Analysis on Cuckoo

- Once the pending job is completed, you can view the result
- Click the red box

Recent analyses				
#	Date	File	Package	Score
41	24/07/2017	stage1.exe	-	2.2 / 10

Tutorial - Analysis on Cuckoo(File Info)

Summary

File stage1.exe

Summary		Download	Resubmit sample
Size	30.5KB		
Type	PE32 executable (GUI) Intel 80386, for MS Windows		
MD5	7920522675e0a5a4519ee9730b039461		
SHA1	ecd27ba347668453a2cfb9eac79811e1477ae5bf		
SHA256	24c8eaccd8f9a345ebfd7b74846af849280bd9d5f927ab01eacd476c7ac54387		
SHA512	Show SHA512		
CRC32	78B434D7		
ssdeep	None		
Yara	None matched		

Information on Execution

Analysis				
Category	Started	Completed	Duration	Logs
FILE	July 24, 2017, 8:26 p.m.	July 24, 2017, 8:29 p.m.	161 seconds	Show Analyzer Log Show Cuckoo Log

Machine			
Name	Label	Started On	Shutdown On
winxpsp3_1	winxpsp3	2017-07-24 20:26:56	2017-07-24 20:29:35

Tutorial - Analysis on Cuckoo(Network Info)

- After redirecting, the result of cuckoo shows high-level information
- Observe the C2 traffic.
- Please compare this result with your Wireshark result.

Signatures

One or more potentially interesting buffers were extracted, these generally contain injected code, configuration data, etc.

Performs some HTTP requests (1 event)

request POST http://netscan.gtisc.gatech.edu/images/logo/header.php

Allocates read-write-execute memory (usually to unpack itself) (1 event)

Potentially malicious URLs were found in the process memory dump (50 out of 62 events)

url http://ironhide.gtisc.gatech.edu/

url http://www.time.gov/nist_time.lzx.js?

url http://dd799dce64a3ee847ffef300d41927f0.clo.footprintdns.com/apc/trans.gif

url http://www.time.gov/images/clock/sunshine.gif

url http://www.time.gov/images/rightnow3.gif

url http://9cbea18cfa132ccdaaf5ca8457cf18e6.clo.footprintdns.com/apc/trans.gif

Name	Response	Post-Analysis Lookup
netscan.gtisc.gatech.edu	A → 128.61.240.66	128.61.240.66

IP Address	Status	Action
128.61.240.66	Active	Moloch
8.8.8.8	Active	Moloch

Tutorial - Analysis on Cuckoo(Network Info)

- In the network analysis tab, cuckoo provides more detailed info: payload, HTTPs, etc.

The image shows the Cuckoo Network Analysis interface. On the left is a sidebar with navigation options: Summary, Static Analysis, Behavioral Analysis (1), Network Analysis (highlighted with a red box), Dropped Files (0), Dropped Buffers (1), Process Memory (2), Compare Analysis, Export Analysis, Reboot Analysis, Options, Feedback, and Lock sidebar. The main panel is titled 'Network Analysis' and features tabs for Hosts (2), DNS (1), TCP (1, highlighted), UDP (2), HTTP(S) (25), and ICMP. Under the 'TCP Requests' section, a request is shown from 192.168.122.3:1038 to 128.61.240.66:80, identified as netscan.gtisc.gatech.edu. To the right, a detailed view of the request shows the destination IP and a tabbed interface for viewing the data in plaintext or hex. The hex tab is selected, displaying 16 bytes of data. The data is shown in a table with offset, hex, and ASCII columns. The ASCII column shows an HTTP POST request to /images/logo/header.php.HTT with a Content-Type of application/x-www-form-urlencoded and a User-Agent of Mozilla/4.0 (compatible; MSI E.8.0; Windows.N T.5.1; Trident/4).

Network Analysis

Hosts 2 DNS 1 TCP 1 UDP 2 HTTP(S) 25 ICMP

TCP Requests

192.168.122.3:1038 → 128.61.240.66:80
netscan.gtisc.gatech.edu

192.168.122.3:1038 →

plaintext hex 16 bytes 32 bytes 48 bytes 64 bytes

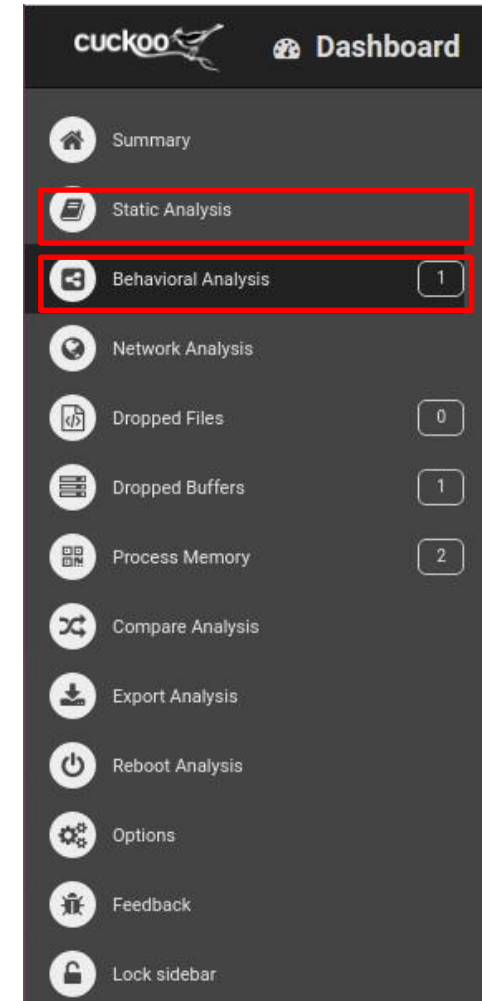
00000000:	504f 5354 202f 696d 6167 6573 2f6c 6f67	POST./images/log
00000010:	6f2f 6865 6164 6572 2e70 6870 2048 5454	o/header.php.HTT
00000020:	502f 312e 310d 0a43 6f6e 7465 6e74 2d54	P/1.1..Content-T
00000030:	7970 653a 2061 7070 6c69 6361 7469 6f6e	ype:.application
00000040:	2f78 2d77 7777 2d66 6f72 6d2d 7572 6c65	/x-www-form-urle
00000050:	6e63 6f64 6564 0d0a 5573 6572 2d41 6765	ncoded..User-Age
00000060:	6e74 3a20 4d6f 7a69 6c6c 612f 342e 3020	nt:.Mozilla/4.0.
00000070:	2863 6f6d 7061 7469 626c 653b 204d 5349	(compatible;.MSI
00000080:	4520 382e 303b 2057 696e 646f 7773 204e	E.8.0;.Windows.N
00000090:	5420 352e 313b 2054 7269 6465 6e74 2f34	T.5.1;.Trident/4

Tutorial - Figuring Out the List of Commands

- The **malware** does not exhibit its behavior because we did not send the correct command through our **fake C2 server**
- We will use
 - **File/Registry/Process tracing analysis** to guess the malware behavior.
 - **control-flow graph** (CFG) analysis and **symbolic execution** to figure out the list of the correct commands
- The purpose of **tracing analysis** is to draw a big picture of the malware
 - What kinds of System call/API does the malware use?
 - Does the malware create/read/write a file? How about a registry?
- The purpose of **CFG analysis** is to find the exact logic that involves the interpretation of the command and the execution of malicious behavior
- Then, **symbolic execution** finds the command that drives the malware into that execution path

Tutorial - Tracing Analysis on Cuckoo

- On the side bar, there are useful menus for tracing analysis.
 - We are focusing on:
 - Static Analysis
 - API/System Call.
 - Behavioral Analysis
 - Trace behaviors in time sequence.



Tutorial - Static Analysis on Cuckoo

- Static Analysis
 - Information of the malware.
 - Win32 PE format information
 - Windows binary uses the PE format
 - Complicated structure
 - Sections includes
 - .text
 - Strings, etc.
 - .data
 - .idata
 - .reloc
 - Virtual link, dynamic link, etc.

Sections	
Name	Virtual Address
.text	0x00001000
.data	0x00006000
.idata	0x00009000
.reloc	0x0000a000

- More info: <http://resources.infosecinstitute.com/2-malware-researchers-handbook-demystifying-pe-file/#gref>

Tutorial - Static Analysis on Cuckoo

- Interestingly three DLL(Dynamic Link Libraries) files are imported.
- In WININET.dll, we can see that the malware uses http protocol.
- In ADVAPI32.dll, we can check if the malware touches registry files
- In Kernel32.dll, we can check the malware waiting signal, also sleep.

Library WININET.dll:

- 0x409174 InternetOpenA
- 0x409178 InternetConnectA
- 0x40917c InternetOpenUrlA
- 0x409180 InternetReadFile
- 0x409184 HttpOpenRequestA
- 0x409188 HttpSendRequestA
- 0x40918c InternetGetCookieA
- 0x409190 InternetCloseHandle

Imports

Library KERNEL32.dll:

- 0x40902c lstrcatA
- 0x409030 strlenA
- 0x409034 GetModuleHandleA
- 0x409038 GetComputerNameA
- 0x40903c GetVersionExA
- 0x409040 CreateToolhelp32Snapshot
- 0x409044 Process32First
- 0x409048 Process32Next
- 0x40904c VirtualAlloc
- 0x409050 SetEvent
- 0x409054 WaitForSingleObject
- 0x409058 Sleep
- 0x40905c GetLastError
- 0x409060 SetLastError
- 0x409064

Library ADVAPI32.dll:

- 0x409000 OpenProcessToken
- 0x409004 LookupPrivilegeValueA
- 0x409008 GetUserNameA
- 0x40900c RegCloseKey
- 0x409010 RegCreateKeyExA
- 0x409014 RegOpenKeyExA
- 0x409018 RegQueryValueExA
- 0x40901c RegSetValueExA
- 0x409020 RegDeleteKeyA
- 0x409024 AdjustTokenPrivileges

Tutorial - Behavior Analysis on Cuckoo

- Tracing a behavior(file/process/thread/registry/network) in time sequence.
- Useful to figure out cause-and-effect in process/file/network.
- Malware creates a new file and runs the process, then writes it to memory.

July 24, 2017, 8:24 p.m. CreateProcessInternalW	thread_idenfifier: 844 thread_handle: 0x000000b0 process_idenfifier: 1300 current_directory: filepath: C:\Program Files\Internet Explorer\iexplore.exe track: 1 command_line: filepath_r: C:\Program Files\Internet Explorer\iexplore.exe creation_flags: 4 (CREATE_SUSPENDED) inherit_handles: 1 process_handle: 0x000000ac
---	--

July 24, 2017, 8:24 p.m. WriteProcessMemory	buffer: base_address: 0x00180000 process_idenfifier: 1300 process_handle: 0x000000ac
July 24, 2017, 8:24 p.m. CreateRemoteThread	thread_idenfifier: 0 process_idenfifier: 1300 function_address: 0x00183020 flags: 0 stack_size: 0 parameter: 0x00000000 process_handle: 0x000000ac

Tutorial - Cuckoo analysis result

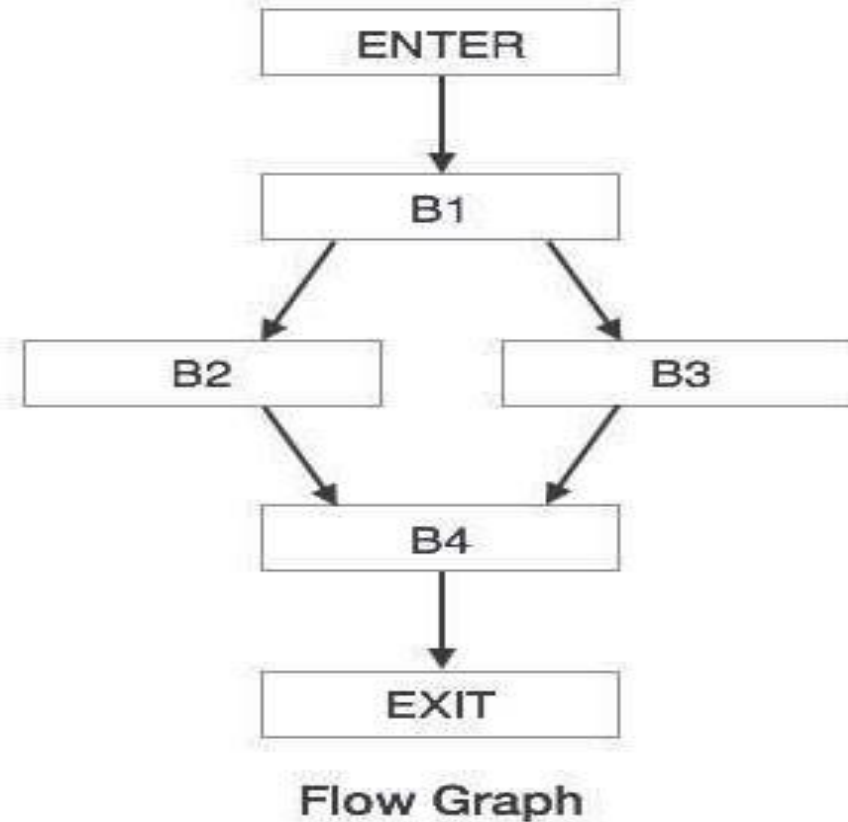
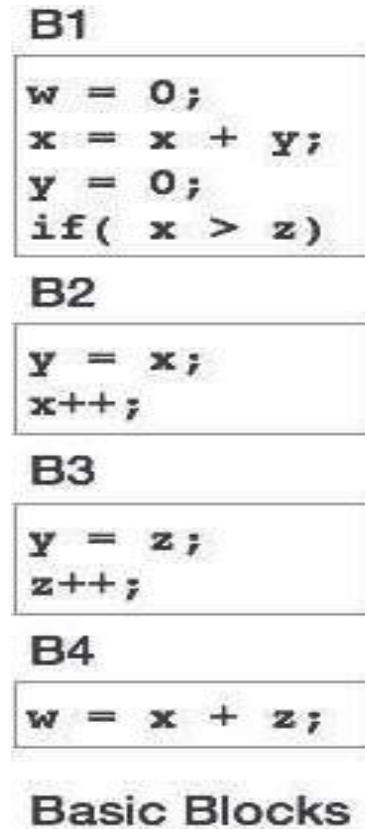
- Based on our analysis with Cuckoo, we can determine if...
 - The malware uses HTTP protocol to communicate
 - Communicate with whom? C&C?
 - Web server access? For checking if the C2 server is active?
 - Commands through http protocol? Cookies?
 - The malware touches(create/write/read) a file/registry/process
 - This might be a dropper? Or does it download a binary from the C2 server?
 - What is the purpose of creating processes? Modifying the registry?

Tutorial - Control Flow Graph Analysis

- Based on the pre-information that we collected from the previous step, we are going to perform CFG analysis & symbolic execution analysis
- CFG:
 - graph representation of computation and control flow in the program
 - Nodes are basic blocks
 - Edges represent possible flow of control from the end of one block to the beginning of the other.

Tutorial - Control Flow Graph Analysis

- CFG : An Example



- But, in malware analysis, we are analyzing CFG at the instruction level.

Tutorial - Control Flow Graph Analysis

- We provide a tool for you that helps to find command interpretation logic and malicious logic
 - We list the functions or system calls the malware uses internally
 - If you provide the score (how malicious it is, or how likely the malicious logic is to use such a function) for the functions, then the tool will find where the malicious logic is, based on its score
 - Example: if you set `StrCmpNIA` to have a score of **10**, then the function that calls `StrCmpNIA` **5** times within itself will have the score **50**.
 - A higher score implies that more functions related to the malicious activity are used within the malware.
 - Your job is to write the score value per each function
- More info:
 - <http://www.cs.cornell.edu/courses/cs412/2008sp/lectures/lec24.pdf>

Tutorial - Control Flow Graph Analysis

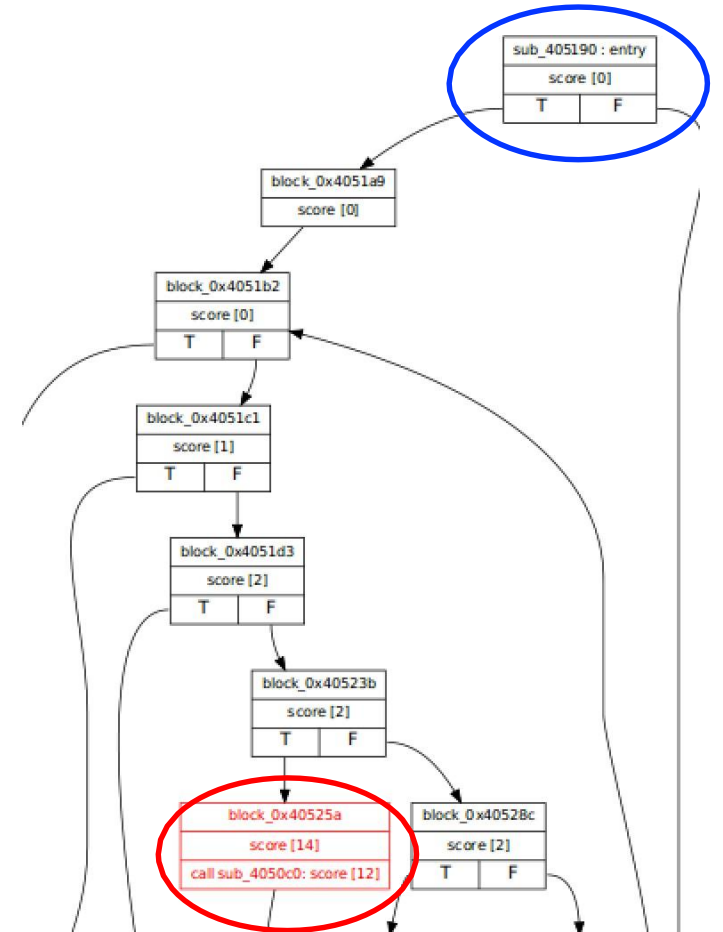
- From our network analysis, we know that the malware uses an Internet connection to 128.61.240.66
- From our cuckoo-based analysis, we know that the malware uses the HTTP protocol.
 - Let's make the Internet related functions to have higher score
 - Open score.h, and edit the score of all of the Internet related functions
 - The score is the value at the end (all others are set to 1)

```
(angr) analysis@analysis-VirtualBox:~/tools$ cd ~
(angr) analysis@analysis-VirtualBox:~$ cd tools/cfg-generation/
(angr) analysis@analysis-VirtualBox:~/tools/cfg-generation$ ls
default_header.h  generate.py  reset.sh  score.h  score.h.bak
(angr) analysis@analysis-VirtualBox:~/tools/cfg-generation$ vim score.h
```

```
45     {"HttpSendRequestA", 1, 1},
46     {"InitializeCriticalSection", 1, 1},
47     {"InternetCloseHandle", 1, 5},
48     {"InternetConnectA", 1, 5},
49     {"InternetGetCookieA", 1, 5},
50     {"InternetOpenA", 1, 5},
51     {"InternetOpenUrlA", 1, 5},
52     {"InternetReadFile", 1, 5},
53     {"IsBadReadPtr", 1, 1},
54     {"LeaveCriticalSection", 1, 1},
```

Tutorial - Control Flow Graph Analysis

- Build control flow graph
 - By executing `./generate.py stage1`, the tool gives you the CFG
 - This finds the function with higher score
 - Implies that this calls high score functions on its execution
 - For stage2
 - Use 'stage2' as argument
 - **Note: your graph and its memory addresses will vary from this example**
 - The function entry is at the address of **405190**
 - And, there is a function (marked as sub) of score **12**
 - At the address **40525a** (marked in red)
 - Use the block_address, not the call sub_address
 - This implies that
 - `sub_4050c0` calls some internet related functions.
 - We need to find out what this command is
 - Run from **405190** to **40525a**



Tutorial - Finding Command

- Finding Commands with Symbolic Execution
 - We want to find a command that drives malware from 405190 to 40525a
 - Let's do symbolic execution to figure that out
 - What is symbolic execution?
 - Rather than executing the program with some input, symbolic execution treats the input data as a symbolic variable, then tries to calculate expressions for the input along the execution.
 - Let's take an example

Example - Symbolic Execution

- What is Symbolic Execution?

Symbolic execution moves along the path of conditional statements, and combines all conditions until it reaches the target function. At the end, it solves the expression to get an input that satisfies all of the conditions

- Path explosion
- Modeling statements and environments
- Constraint solving

Example1 - Symbolic Execution

Code Example

```
1 int main() {  
2   int i, j;  
3   printf("Give me two integers\n");  
4   scanf("%d %d", &i, &j);  
5   if(i+5<j) {  
6       if(i%2 == 0) {  
7           if(j%3 == 0) {  
8               printf("Correct!\n");  
9               return 0;  
10          }  
11      }  
12  }  
13  printf("Incorrect!\n");  
14  return 1;  
15 }
```

Expressions

$i+5 < j$

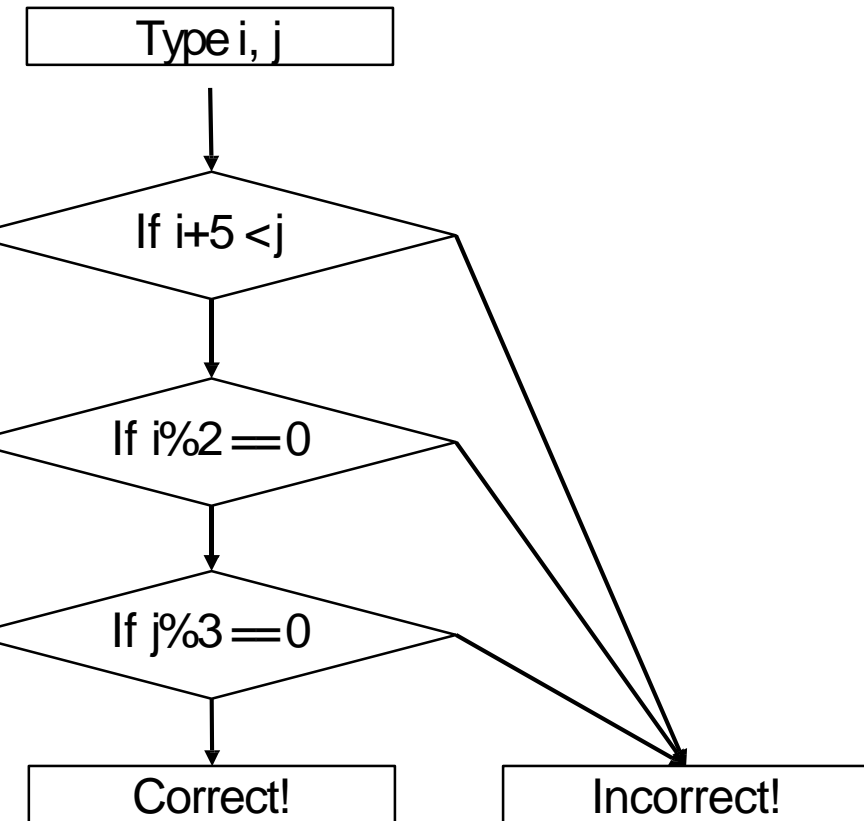
$i+5 < j; i\%2 == 0$

$i+5 < j; i\%2 == 0; j\%3 == 0$

Solve the expression

$i = 2$

$j > 7$, but multiple of 3 so
 $j = 9$



$i=2, j=9$ will lead the program to print "Correct!"

Example1 - Symbolic Execution

In this example, ONLY $i=2, j=9$ conditions will lead the program to print **“Correct!”**

Symbolic execution is available to solve the expression in order to reach a target, in this case "Correct".

Let's apply it into Malware Command & Control logic.
A C&C bot(malware) is expecting inputs(**solve the expressions**) to trigger behaviors(**targets**).

Example2 - Symbolic Execution

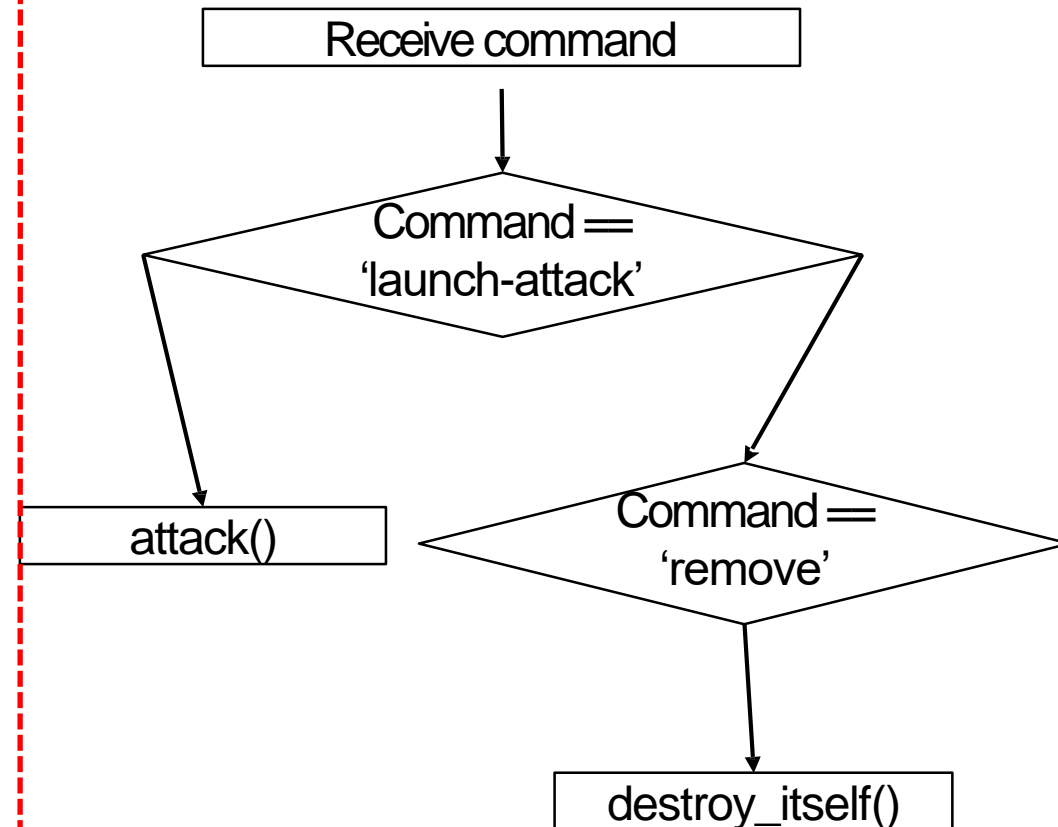
Code Example

```
1 int main() {  
2     char command[512];  
3     // receive the command  
4     recv(socket, command, 512, 0);  
5  
6     // compare the input with 'launch-attack'  
7     if(strcmp(command, "launch-attack") == 0) {  
8         attack();  
9     }  
10    else if (strcmp(command, "remove") == 0) {  
11        // when the command is 'remove'  
12        destroy_itself();  
13    }  
14 }
```

Expressions

Command =
'launch-attack'

Command =
'remove'



This executes `attack()` on command 'launch-attack', and `destroy_itself()` on 'remove' command

Example2 - Symbolic Execution

In this example, ONLY `'launch-attack'` and `'remove'` commands(inputs) triggers `attack()` and `destroy_itself()`.

Symbolic execution is able to find "launch-attack" as an input to trigger `attack()`, which is a malicious behavior. Plus, "remove" will lead to `destroy_itself()`, which is another behavior.

Our job in this project with Symbolic execution is to find inputs, and then feed the inputs to trigger behaviors.

Symbolic execution engine

- Symbolic Execution Engine: Klee, Angr, Mayhem, etc.
 - Loading a binary into the analysis program
 - Translating a binary into an intermediate representation (IR).
 - Translating that IR into a semantic representation
 - Performing the actual analysis with symbolic execution.

For more information:

<https://www.cs.umd.edu/~mwh/se-tutorial/symbolic-exec.pdf>

Tutorial - Finding Commands with Angr

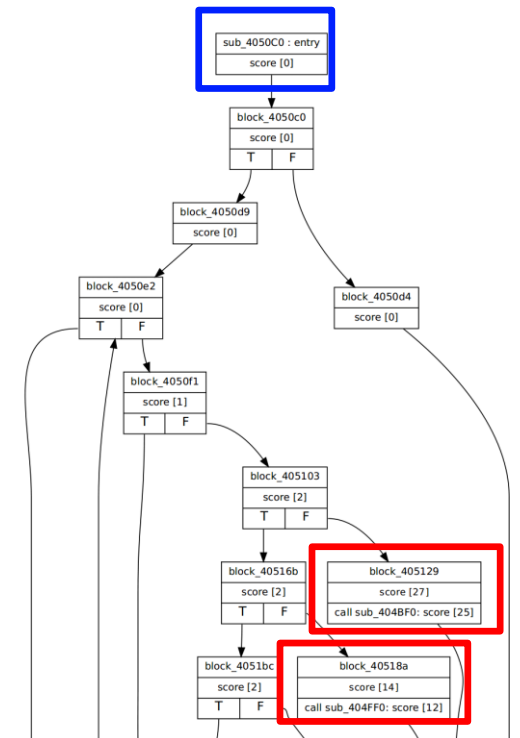
- We prepared a symbolic executor and a solver for you
 - Your job is to find the starting point of the function which interprets the command, and find the end point where malware actually executes some function that does malicious operations
 - Use a Control-flow Graph (CFG) analysis tool!
- The symbolic executor is called angr. (<http://angr.io/index.html>)

Tutorial - Finding Command on Angr

- We prepared a symbolic executor and a solver for you
 - How do you run it?
 - Go to ~/tools/sym-exec
 - Run it as
 - `python ./sym-exec.py [program_path] [start_address] [end_address]`
 - `python ./sym-exec.py ~/shared/stage1.exe 4050c0 40518a`
 - The command will be printed at the end (if found)

Replace these with start and end addresses from your graph

```
(angr) analysis@analysis-VirtualBox:~/tools/sym-exec$ python sym_exec.py ~/shared/stage1.exe 4050c0 40518a
WARNING | 2019-09-27 16:59:52,414 | angr.analyses.disassembly_utils | Your version of capstone does not support MIPS instruction groups.
WARNING | 2019-09-27 16:59:52,689 | angr.project | Address is already hooked, during hook(0x1000008, <class 'angr.procedures.libc.strlen.strlen'>). Re-hooking.
CRITICAL | 2019-09-27 16:59:52,689 | angr.project | Hooking with a SimProcedure class is deprecated! Please hook with an instance.
WARNING | 2019-09-27 16:59:52,689 | angr.project | Address is already hooked, during hook(0x10002a8, <class 'angr.procedures.libc.strncmp.strncmp'>). Re-hooking.
Input found: $uninstall
(angr) analysis@analysis-VirtualBox:~/tools/sym-exec$
```

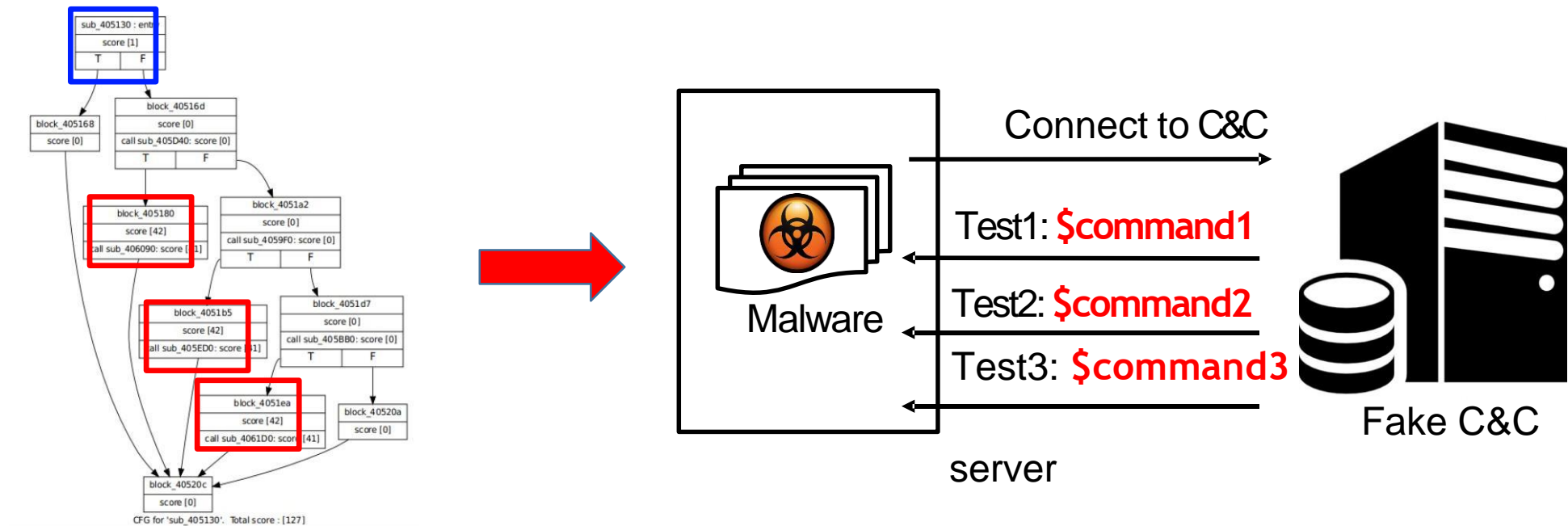


Symbolic Execution - Special Note for stage2.exe

- sys-exec for stage2 takes a lot of time to resolve (up to 20 minutes)
 - you are welcome to modify the VM performance settings (memory, cores) based on your hardware to speed this up
- If you get a single error message, keep trying again – sym-exec will occasionally fail for stage2
- If your screen is filling up with error messages, then you have the wrong start and/or end address

Tutorial - Reconstructing C2 server

- After CFG analysis + symbolic execution, reconstruct the C2 server



Tutorial - Reconstructing C2 server

- The tool for reconstructing the C2 server is already on the VM
 - It runs nginx and php script
 - This will look like `~/tools/c2-command/stage*-command.txt`
 - Your job is to add your commands to the relevant `*.txt` file
 - The command that leads the execution from 405190 to 40525a is “\$uninstall” (note: the name of the command you see may vary)
 - Then, type “\$uninstall” and save the file.
 - Important: be sure to put the ‘\$’ character before your commands, even if `stage*-command.txt` says that it’s optional
 - The order of commands in the file does not matter – they’ll run in a random order
 - Note: This means that if you want to run only a particular command, you’ll need to remove, or comment out the other commands in your file

```
analysis@analysis-VirtualBox: ~/tools/c2-command
(angr) analysis@analysis-VirtualBox:~$ cd ~/tools/c2-command/
(angr) analysis@analysis-VirtualBox:~/tools/c2-command$ ls
command-sample.txt  stage1-command.txt  stage2-command.txt  stage3-command.txt
(angr) analysis@analysis-VirtualBox:~/tools/c2-command$ vim stage1-command.txt
```

```
analysis@analysis-VirtualBox: ~/tools/c2-command
1 # write down commands in the following lines
2 $uninstall
3
```


After that...

- If you find all of the commands for stage1.exe malware, the malware will download stage2.exe by updating itself.
- Now you've found the commands from running sym-exec.py
- Add those commands to stage1-commands.txt. Remember to put \$<command>.
- Start up the windows VM again, then copy stage1.exe to the desktop. Then double click on it and continue.
- Note if stage1 fails to download stage2, your firewall might be blocking it
 - This is actual malware so some IDS have signatures that match it.

After that...

- For stage2.exe, please follow the same steps in the tutorial
 - Check its network access with Wireshark
 - Redirect network traffic to if required (if the connection fails)
 - Try to identify malicious functions by editing score.h and using the cfg-generation tool
 - Discover the list of commands using the symbolic execution tool
 - Fill the commands in ~/tools/c2-command/stage2-command.txt
 - Run it as mentioned before.

Linux Malware

- Stage2.exe will download stage3 malware, which is payload.exe. This is a linux malware.
- We need to handle the linux malware differently unlike windows malware, and will use different tools and methods to analyze this malware

Linux Malware Tools

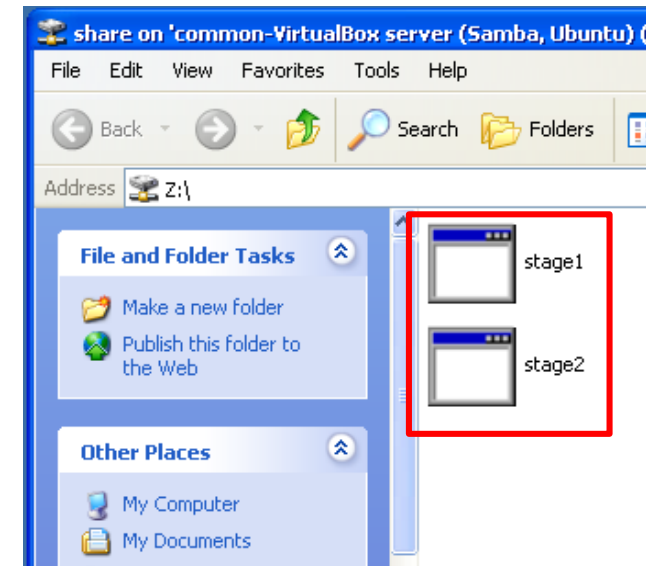
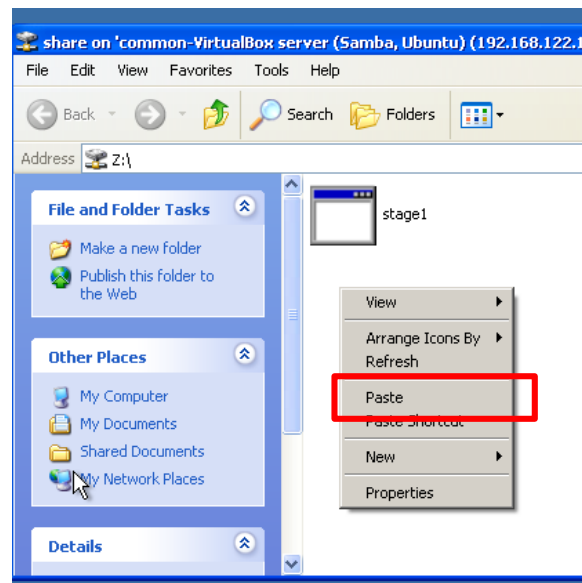
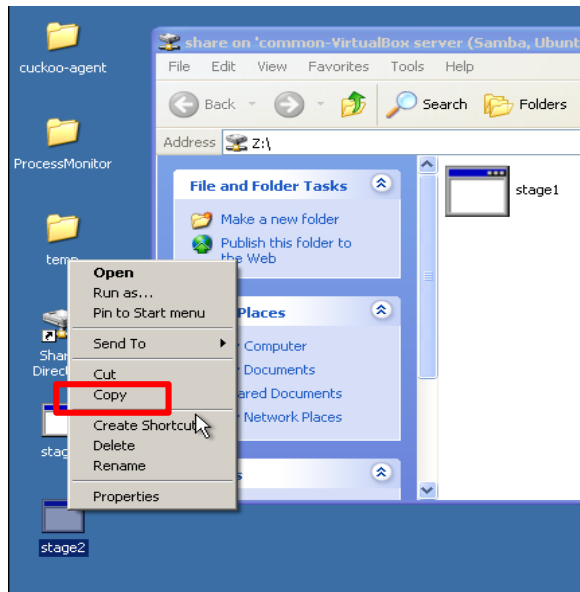
- First copy the linux malware into shared folder. The tools which you will use are installed inside the Linux host.
- `~/tools/linux_sym_exe.py`
 - for linux malware symbolic execution
 - `python linux_sym_exec.py path_to_linux_mw start target`
 - To make it work, you need to modify two `linux_sym_exec.py` functions
 - `targs_len_before` and `opts_len_before`
- `~/tools/dynamicanalysis/`
 - `instrace.linux.log` : the dynamic instruction trace for the linux malware
 - `detect_loop.py` : you have to modify this file to find the loop in the given trace
 - Usage: `python detect_loop.py <path-to-debug-file>`
- **Reverse Engineering Resource:**
 - Please check **cs6262proj3tutorial.pdf** in Canvas

Linux Malware

- Run 'python linux_sym_exec.py path_to_linux start target'.
- It won't be able to find any input because of path explosion. You need to add constraints to make symbolic execution targeted
- Follow the steps in '~/.report/assignment-questionnaire.txt' and find the inputs.
- Analyze the dynamic instruction trace and locate the C&C communication

Tutorial - Copy to Shared Directory

- Once you have followed the previous instructions, you will see that a new malware file has been downloaded.
- You need to copy the malware into the Linux host to analyze it.
 - Right-click the downloaded malware on the Desktop, then click "Copy".
 - Open Shared Directory and right-click, then click "Paste".



Tutorial - Copy to Shared Directory

- If you're having trouble with file permissions on the XP VM, open a terminal on the Linux host, navigate to the "~/shared" folder, and follow the steps below:

```
analysis@analysis-VirtualBox: ~/shared
(angr) analysis@analysis-VirtualBox:~$ cd shared/
(angr) analysis@analysis-VirtualBox:~/shared$ ls -al
total 76
drwxrwxrwx  2 analysis analysis  4096 Feb 12 11:45 .
drwxr-xr-x 29 analysis analysis  4096 Feb 12 10:41 ..
-rw-rw-r--  1 analysis analysis 31232 Feb 12 10:49 stage1.exe
-rwxr--r--  1 nobody  nogroup  33280 Feb 12 11:34 stage2.exe
(angr) analysis@analysis-VirtualBox:~/shared$ sudo chmod 664 stage2.exe
(angr) analysis@analysis-VirtualBox:~/shared$ sudo chown analysis:analysis stage2.exe
(angr) analysis@analysis-VirtualBox:~/shared$ ls -al
total 76
drwxrwxrwx  2 analysis analysis  4096 Feb 12 11:45 .
drwxr-xr-x 29 analysis analysis  4096 Feb 12 10:41 ..
-rw-rw-r--  1 analysis analysis 31232 Feb 12 10:49 stage1.exe
-rw-rw-r--  1 analysis analysis 33280 Feb 12 11:34 stage2.exe
(angr) analysis@analysis-VirtualBox:~/shared$
```


Tutorial - Copy to Shared Directory

```
analysis@analysis-VirtualBox: ~/shared
(angr) analysis@analysis-VirtualBox:~/tools/network$ ./reset
Reset iptables rules
Importing iptables rules
(angr) analysis@analysis-VirtualBox:~/tools/network$ cd ..
(angr) analysis@analysis-VirtualBox:~/tools$ ls
c2-command  cfg-generation  dynamicanalysis  network  sym-exec
(angr) analysis@analysis-VirtualBox:~/tools$ cd ~
(angr) analysis@analysis-VirtualBox:~$ ls
Android  bin  Downloads  report  shared  update.sh
archive.sh  Desktop  init.py  setup  tools  vm
(angr) analysis@analysis-VirtualBox:~$ cd shared/
(angr) analysis@analysis-VirtualBox:~/shared$ ls
payload.exe  stage1.exe  stage2.exe
(angr) analysis@analysis-VirtualBox:~/shared$ file payload.exe
payload.exe: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically
linked, not stripped
(angr) analysis@analysis-VirtualBox:~/shared$ ls -al
total 264
drwxrwxrwx  2 analysis analysis  4096 Sep 23 17:30 .
drwxr-xr-x 29 analysis analysis  4096 Sep 23 16:43 ..
-rwxr--r--  1 nobody  nogroup 189796 Sep 23 17:29 payload.exe
-rw-rw-r--  1 analysis analysis 31232 Sep 23 17:26 stage1.exe
-rwxr--r--  1 nobody  nogroup 33280 Sep 23 17:27 stage2.exe
(angr) analysis@analysis-VirtualBox:~/shared$
```


Tips for assignment-questionnaire.txt

- Complete the questionnaire as you go; try to avoid backtracking as this wastes time
- The URL example in the questionnaire is “http://scouter.cc.gatech.edu/a/b/c”, but some URLs may not include a path (a/b/c after the domain) – this is fine, just be sure to include the path in your answer for the URLs that include it
- The grading script will ignore “http://”, “https://” and “www.” for your convenience, but try to be thorough and match what you see exactly
- Commands and memory addresses are NOT case sensitive, but be sure you don’t mix up 0 (zero) and O – the zero should have a dot in it in the VM

Tips for assignment-questionnaire.txt

Please use the latest version of VirtualBox when you import the VM. Please do not modify anything related to network settings in the VM.

1. Domain name

On the questionnaire sheet, there are entries for writing domain names. Please follow the following rules on getting answers for those questions.

- You should write FQDN, which means, if the full domain name is canof.gtisc.gatech.edu then write canof.gtisc.gatech.edu, not just gatech.edu or gtisc.gatech.edu
- For the others (connections check, DDoS, sending info, etc.), you should get the exact domain name that the malware uses. For example, the IP address 130.207.188.35 belongs to both coe.gatech.edu and web-plesk5.gatech.edu. Because there are multiple mappings, you cannot be sure about which domain that the malware used by just using **nslookup**. In this case, please go through the other way of getting domain names from ***DNS Packets in Wireshark***.

Please, all Domains should be based on Wireshark DNS packets

(e.g., get it from DNS query packet or redirecting HTTP traffic into local VM and examine Host header).

If you get see the log in the Wireshark, You will find DNS query(Standard query) and DNS response(Standard query response)

In Domain Name System section, there is Query section, like below

Queries:

x.y.z: type A, class IN.

Answers:

x.y.z: type CNAME, class IN, cname a.b.c

You should use x.y.z

Tips for assignment-questionnaire.txt

2. URL

For all URLs, you do not have to specify the protocol (http:// or https://, etc.). However, if HTTP traffic is like the following:

```
POST /a/b/c/d?asdf=1234 HTTP/1.1 Host: www.zzz.com
```

then please write this as

```
www.zzz.com/a/b/c/d?asdf=1234
```

3. Writing commands in *.txt files under c2-command directory

There are pre-installed PHP scripts in the VM locally that read the *.txt file for each stage, these scripts send the command to the malware after reading them from the TXT files. One caveat of these scripts is that they are written to send the commands in random order (i.e., if there are commands a, b, c, then the script will randomly choose one command and send it to the malware). So if you want to test ONE command at a time, then please write only that command in the TXT file. For example, if you just want to run the command \$uninstall, then please write only that command in stage1-command.txt.

4. linux_sym_exec and detect_loop for linux malware

You could use free IDA-Pro, objdump or radare2 for this task to find out called attack functions, and the target addresses. Look for some angr examples on the github, which adds constraints to the state. For the loop detection, focus on function sequence that called repetitive.

Tips for assignment-questionnaire.txt

5. When you think that you found the correct command but malware is not working...

Note that some commands for stage 2 are different per each student, by having 4 digit hexadecimal numbers at the end of the command. For example, a command for stage 2 is formatted like

\$COMMANDa1b4

(NOTE: two commands in stage 2 have the 4 digit hexadecimal tail. And, all commands in stage 3 have the 4 digit hexadecimal tail on the command.)

However, there could be a case that only gets the front part of the command like

\$COMMAND

if the end point address of symbolic execution is not correctly set. In such a case, please set the correct end point that you can get the entire command.

6. Cuckoo

- In the VM, we provide cuckoo, which is a dynamic malware analysis framework. It is very convenient and easy to use. While you are running cuckoo, you might meet some warnings and errors "critical time blah blah~" and "YARA signature.... blah blah". Please ignore them.

Because you are executing a malware in the QEMU Windows VM, the framework needs to set a time. Cuckoo will check the malware is terminated or not. However, the three malware you will meet are never going to be terminated(Intentionally, modified by me in educational purpose.) So, please ignore "critical time blah blah~, terminating. In our case, the malware is never going to unfold even though you give an infinite time to be executing the malware unless you feed the right inputs(The malware expects C2 commands.)

-Iptable setting.

If you check `/home/analysis/.cuckoo/conf/kvm.conf` you will find how we set the QEMU windows host VM. You will find the IP of the host VM is "192.168.133.101". If you want to see network behaviors in Cuckoo, you want to forward the IP in `/home/analysis/tools/network/iptables-rules`.

For example, open iptables-rules, you want to add

```
sudo iptables -t nat -A PREROUTING -p tcp -s 192.168.133.101 -d [DEST-IP] --dport 80 -j DNAT --to 192.168.133.1:80
```

Tips

- Getting the domain name from an IP address (if the packet is encrypted)
 - Use nslookup (IP -> domain, and domain name -> IP vice versa)

```
analysis@analysis-VirtualBox: ~  
(anгр) analysis@analysis-VirtualBox:~$ nslookup 128.61.240.66  
Server:      8.8.8.8  
Address:     8.8.8.8#53  
  
Non-authoritative answer:  
66.240.61.128.in-addr.arpa      name = netscan.gtisc.gatech.edu.  
  
Authoritative answers can be found from:  
  
(anгр) analysis@analysis-VirtualBox:~$ nslookup netscan.gtisc.gatech.edu  
Server:      8.8.8.8  
Address:     8.8.8.8#53  
  
Non-authoritative answer:  
Name:   netscan.gtisc.gatech.edu  
Address: 128.61.240.66
```

Tips

- Getting the exact domain name from an IP address
 - Establish a fake connection (redirect to 192.168.133.1)
 - Then look at the TCP stream data
 - The HTTP header will contain the answer
 - Host: netscan.gtisc.gatech.edu

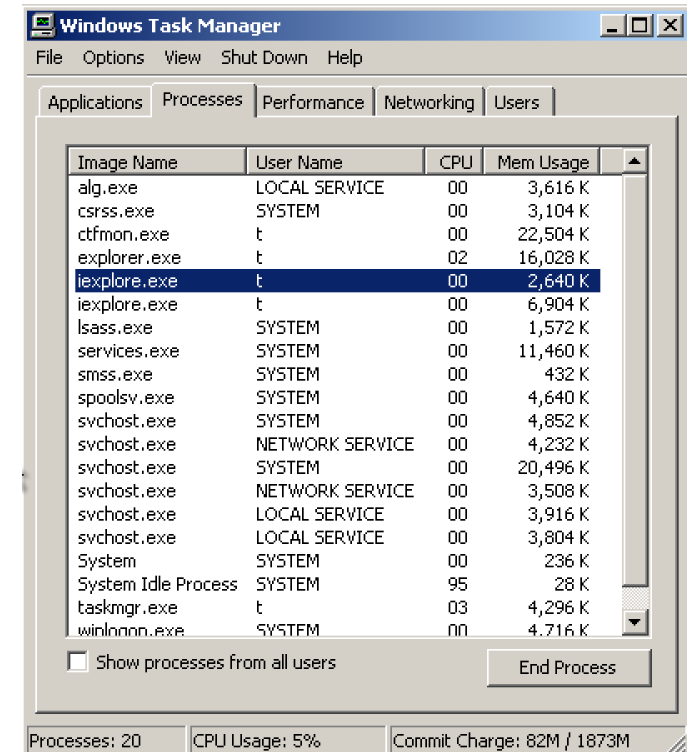
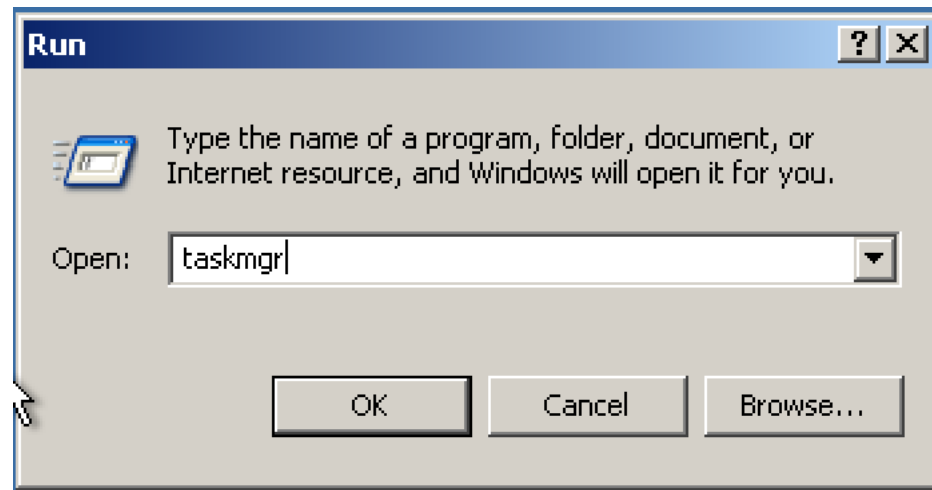
```
POST /images/logo/header.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Host: netscan.gtisc.gatech.edu
Content-Length: 477
Cache-Control: no-cache

page=0GJiYTNhMTMtYjF10C00NTJiLWE3NzctNjRkMmE4MjM5NThi&unm=dA&cnm=R1Qt0DVGM0E5NDgw
NjQ1&query=V2luZG93cyBYUA==&spec=MzIgQm10&opt=MA&view=W1N5c3R1bSBQcm9jZXNzXQpTeXN
0ZW0Kc21zcy5leGUKY3Nyc3MuZXhlCndpbmxvZ29uLmV4ZQp3ZXJ2aWN1cy5leGUKbHNhc3MuZXhlCnN2
Y2hvc3QuZXhlCnN2Y2hvc3QuZXhlCnN2Y2hvc3QuZXhlCnN2Y2hvc3QuZXhlCnN2Y2hvc3QuZXhlCnNwb
29sc3YuZXhlCnN2Y2hvc3QuZXhlCmV4cGxvcmV4ZQp3dWFiY2x0LmV4ZQpjdGZtb24uZXhlCmFsZy
5leGUKaWV4cGxvcmUuZXhlCmllcHBsb3JlLmV4ZQo=&var=TWfJaGluZXM=&val=Z2hhcGo=HTTP/1.1
200 OK
Server: nginx/1.10.0 (Ubuntu)
Date: Sun, 25 Sep 2016 11:46:29 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: response=None%3Aexecute-command; expires=Mon, 26-Sep-2016 11:46:29
GMT; Max-Age=86400; path=/; domain=netscan.gtisc.gatech.edu

4
None
0
```

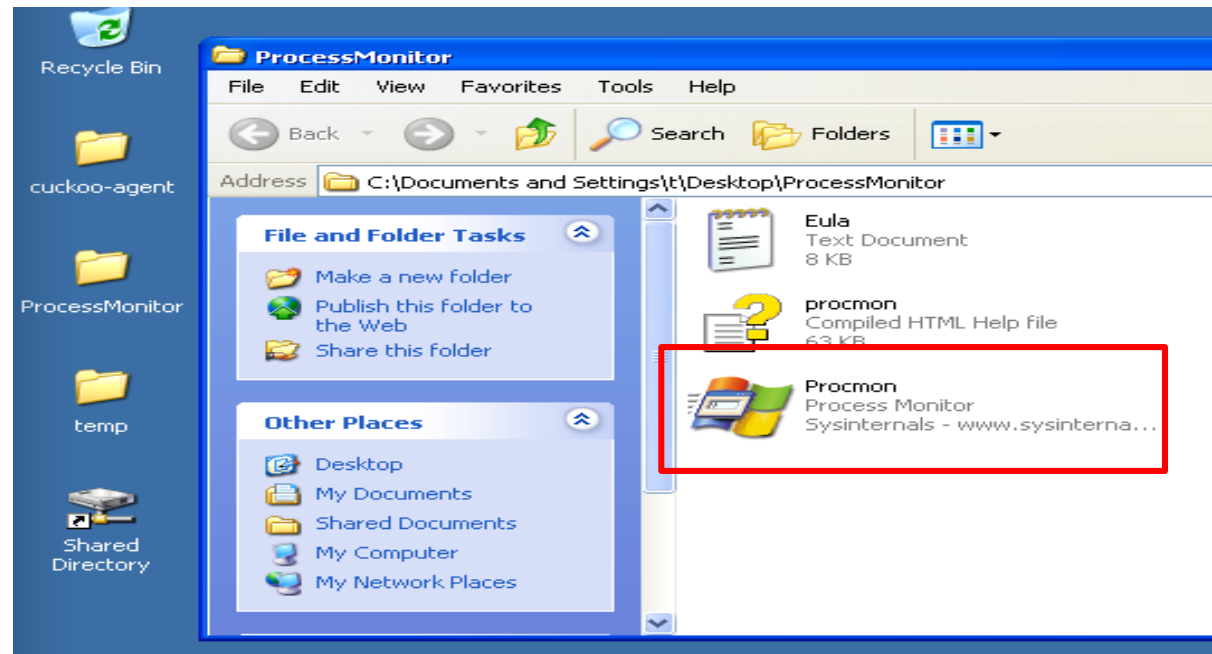
Tips

- Getting the process name of the malware
 - Use taskmgr in Windows
 - Start menu -> run -> taskmgr; or, press Ctrl-Shift-Esc on Windows.
 - Click on the 'Processes' tab to see the list of processes
 - Or use Cuckoo's behavioral analysis



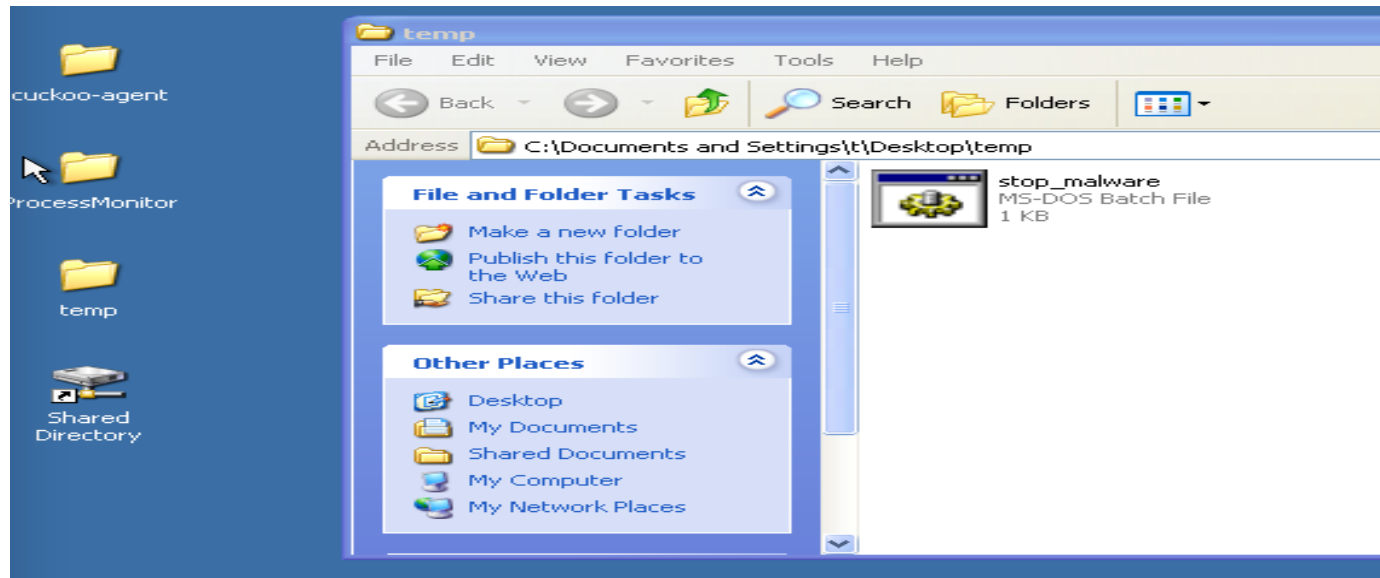
Tips

- To get the process name of the malware and the registry key that was created by the malware
 - Use the Procmon in ProcessMonitor on the testbed VM



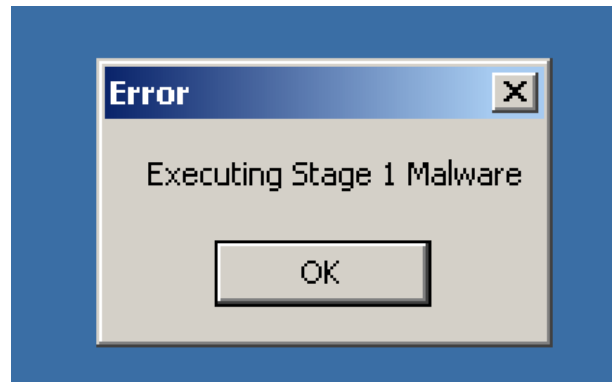
Tips

- If the malware does not run
 - E.g., not displaying the dialog box with “Starting Stage X malware” on startup
 - Try to run stop_malware on the desktop
 - This will stop all malware activity, and you can try again with a clean state



Tips

- Click OK to proceed with malware execution
 - The dialog box pauses execution of the malware
 - Click OK whenever this dialog pops-up from the malware
 - Otherwise, the malware will not execute further and show its behavior



Tips

- Iptables rules
 - Edit `~/tools/network/iptables_rules`
 - Make sure you've written your rules correctly (follow the format and double-check your IPs)
 - **Make sure you execute `./reset` on that directory**
 - This command will update the current iptables rules...
 - NAT Redirect Syntax
 - `iptables -t nat -A PREROUTING -p tcp -s [source-ip-address] -d [destination-ip-address] -- dport 80 -j DNAT --to 192.168.133.1:80`
 - Insert the rule in the **PREROUTING** table of **NAT**,
 - And if **the protocol is tcp**, source ip is matched with **[source-ip-address]**,
 - **Destination IP is matched with [destination-ip-address]**, and destination port is **80**
 - Then **redirect this traffic to 192.168.133.1, port 80.**

Miscellaneous VM Performance Tips (taken from Piazza)

Part 1 : Windows Malware / Generic VM Issues

- **Try lowering your screen resolution**
- **Save often!**
- Avoid using a resource heavy IDE like IntelliJ, Eclipse etc. Lightweight alternatives include gedit, vim, emacs, Sublime Text, Visual Studio Code, nano, etc

Most importantly, do / run only 1 task at a time. That means

- Run the Windows VM only when:
 1. Sending commands to malware
 2. Analyzing network traffic via Wireshark
 3. Once done with those tasks, turn off the Windows VM.
- Avoid running the windows VM when:
 1. Running cuckoo analysis
 2. Generating CFGs
 3. Running Symbolic Execution - This is quite resource intensive, avoid doing other stuff to get this done quickly. (TIP: If this seems to be taking infinite memory/time, your mostly trying to reach a unreachable / invalid address ! check your addresses !)
- Try running the VM at a lower resolution (recommend at-least 1280x800, for legibility) - If you have a very high resolution on your host machine (I had 2560x1440, this may impact the VM performance). You can do this in 2 ways
 1. VirtualBox Menu - View > Virtual Screen 1 > Resize to a x b
 2. Ubuntu Menu - Type "Displays" > Change it there
- Restart after a task / stage. This is mostly a last resort but restarting the VM after finishing a task/stage made everything feel really smooth for me, instead of trying to free memory etc. Just be sure to run `./reset` in `~/tools/networks` after each VM restart!

Part 2 : Android

Some of the above stuff applies here (VM Settings, resolution, etc). Restarting after working on Part 1, helps a lot.

If you still really feel your android emulator is slow you can add the following flags to the **emulator** command flags in `~/bin/run-emulator`

`-memory 2048 -gpu swiftshader`

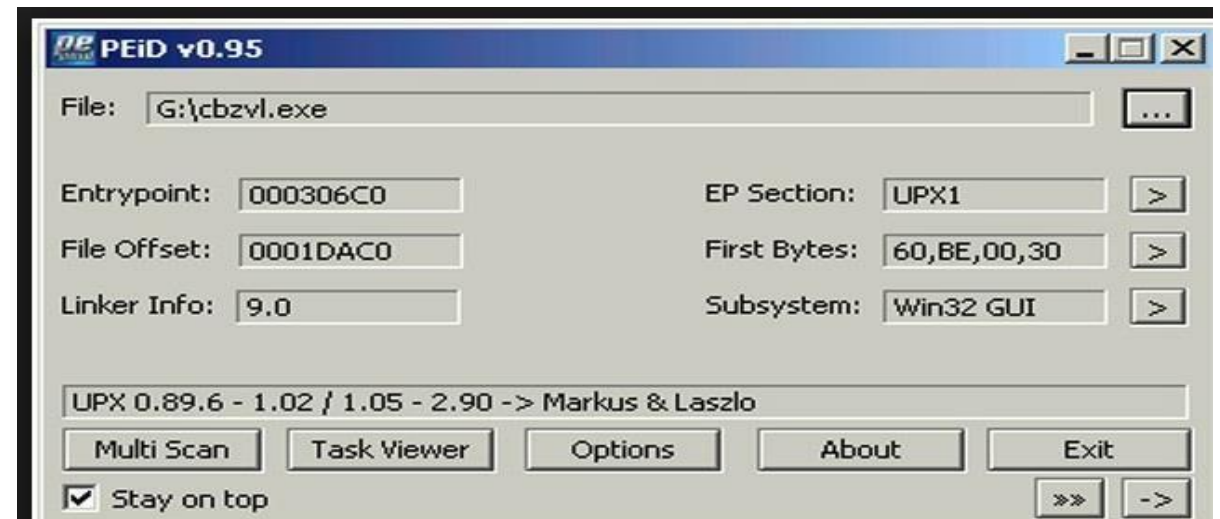
You can experiment with RAM allocation and CPU usage based on your machine – but keep in mind that the project VM has only been tested at 4 GB and with 2 or 3 CPUs.

Advanced Tips

- For those of you who are interested in Reverse Engineering, these slides cover fundamental material for you to study.
- Disassembler/Debugger
 - IDA Pro, binary ninja, radare2, x64 dbg, GDB, immunity debugger, etc.
- Packer/Obfuscation
 - Ether, VMIUnpacker, xorunpacker, etc.
- PE/ELF binary format
- Memory snapshot
- More...

Advanced Tips

- Most malware are packed or obfuscated by a known/unknown packer or obfuscator.
- For Win32 binaries, by checking the PE32 format, we can see whether the binary is packed.
- For obfuscation, we usually need to reverse engineer to determine if the binary is obfuscated.



Advanced Tips

- Assembly code & OS architecture
 - X86, x86-64, arm64, etc.
 - Stack, heap, canary, guardian, etc.
 - An example:

```
.text:004032CF
.text:004032D4
.text:004032DA
.text:004032DC
.text:004032DE
.text:004032E1
.text:004032E2
.text:004032E7
.text:004032EA
.text:004032EC
.text:004032EF
.text:004032F0
.text:004032F5
.text:004032FB
.text:004032FE
.text:00403303
.text:00403303 loc_403303:
.text:00403303
.text:00403308
.text:0040330A
.text:0040330C
.text:00403311
.text:00403311 loc_403311:
.text:00403311
.text:00403316
.text:0040331B
.text:00403320
.text:00403326
.text:00403327
.text:0040332D
.text:00403332

call ds:LoadLibraryA
offset aRpcrt4_dll ; "rpcrt4.dll"
push 1Ch
push 0
lea ecx, [ebp+Buffer]
push ecx
call sub_4018A0
add esp, 0Ch
push 1Ch ; dwLength
lea edx, [ebp+Buffer]
push edx ; lpBuffer
push offset start ; lpAddress
call ds:VirtualQuery
mov eax, [ebp+Buffer.AllocationBase]
mov dword_408C50, eax

; CODE XREF: start+26Bfj
call sub_402A30
test eax, eax
jnz short loc_403311
call sub_402B30

; CODE XREF: start+2EAfj
call sub_402B50
push offset aGetlastinputin ; "GetLastInputInfo"
push offset aUser32_dll_0 ; "user32.dll"
call ds:GetModuleHandleA
push eax ; hModule
call ds:GetProcAddress
mov dword_408CBC, eax
call sub_4013D0
```

Advanced Tips

- Anti debugging/Anti VM techniques
 - Malware is becoming more advanced.
 - Malware authors know that:
 - Malware analysts use debugging/disassembler tools
 - Malware analysts use VM environments
 - Malware authors embed evasive techniques to thwart debugging software and VM environments.
 - Detection of software/hardware breakpoints
 - Detection of memory/conditional breakpoints
 - Timing/Artifact based VM detection

Android Malware Analysis

- Manifest Analysis
 - Identifying suspicious components
- Static Analysis
 - Search for C&C commands and trigger conditions
 - Vet the app for any anti-analysis techniques that need to be removed.
- Dynamic analysis
 - Leverage the information found via static analysis to trigger the malicious behavior.

Manifest Analysis

- Identify suspicious components
 - Broadcast receivers registering for suspicious actions.
 - Background services
- Narrow the scope of analysis
 - Malicious apps are repackaged in benign apps with thousands of classes.

```
<receiver android:name="com.android.SMSReceiver">  
  <intent-filter android:priority="10000">  
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />  
    <action android:name="android.provider.Telephony.SMS_SENT" />  
  </intent-filter>  
</receiver>
```

Broadcast receiver from CoinPirate's malware family.

Static Analysis

- Search for C&C commands and trigger conditions

```
private int checkMessage(String body) {  
    String message1 = getResources().getString(C0197R.string.message_1);  
    String message2 = getResources().getString(C0197R.string.message_2);  
    String message3 = getResources().getString(C0197R.string.message_3);  
    if (body.equals(message1)) {  
        return 1;  
    }  
    if (body.equals(message2)) {  
        return 2;  
    }  
    if (body.equals(message3)) {  
        return 3;  
    }  
    return 0;  
}
```

```
<string name="message_1">hey whats up?</string>  
<string name="message_2">Is this Tim?</string>  
<string name="message_3">Oh, sorry wrong number! =( </string>
```

Static Analysis

- Identifying Anti-analysis techniques

```
public static boolean checkID(Context context) {  
    TelephonyManager telephonyManager = (TelephonyManager) context.getSystemService("phone");  
    if (telephonyManager == null || telephonyManager.getDeviceId().equals(context.getResources().getString(C0197R.string.avalue))) {  
        return false;  
    }  
    return true;  
}
```

Scenario

Analyzing Android Malware

- You have received a malware sample **sms.apk**.
- You need to identify communication with the C&C server
- Identify anti-analysis techniques being used by the app.
- Identify commands that trigger any malicious behavior.

Project Structure

- Android emulator
 - An emulator for Android 4.4 is pre-installed
 - Run 'run-emulator'
 - This will start the Android emulator (this takes a long time, especially the first time you start it)
- Jadx
 - Disassembles apk files into Java source code.
- Apktool
 - Disassembles apk file into Smali.
 - Rebuilds apk files.
- Write-up (~/.Android/MaliciousMessenger/writeup.pdf)
 - Detailed guide on how to complete the Android section of the lab.

Project Structure

- Android App
 - ~/Android/MaliciousMessenger/tutorialApps
 - emu-check.apk
 - A tutorial example (Shown as 'My application' in the emulator)
 - CoinPirate.apk
 - Another tutorial example
 - ~/Android/MaliciousMessenger/sms.apk
 - Target app to analyze to answer the questionnaire
- **READ ~/Android/MaliciousMessenger/writeup.pdf**

Starting C&C Server

- Starting C&C Server
 - Run `start_server`

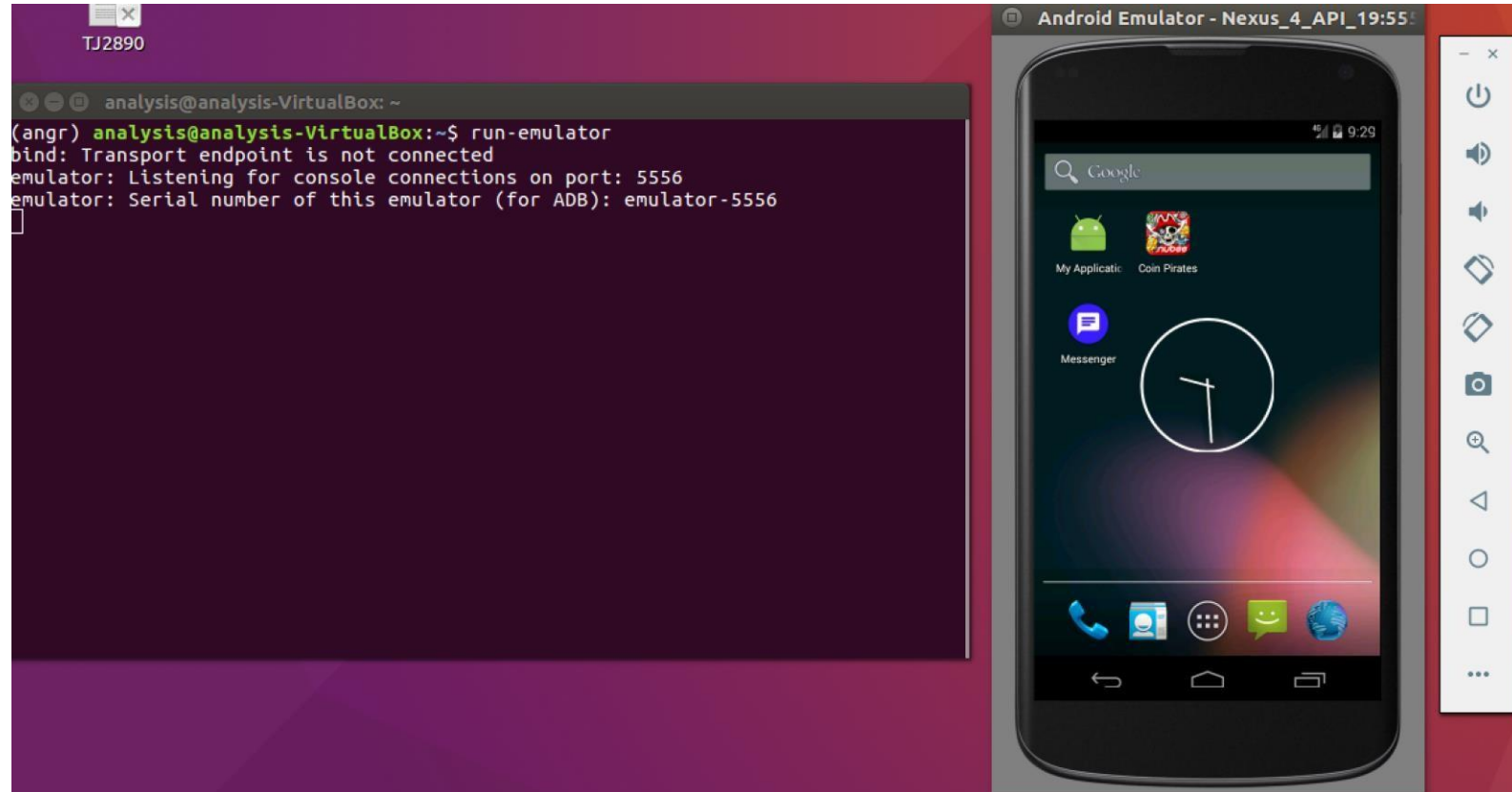
```
(angr) analysis@analysis-VirtualBox:~/Android/MaliciousMessenger$ ./start_server
starting server
* Running on http://192.168.133.1:8080/ (Press CTRL+C to quit)
```


Things To Take Note of...

- If something goes wrong and you don't find the emulator already setup, run the following commands to handle it:
 - `run-emulator adb emulator-5554 install tutorialApps/emu-check.apk`
 - `run-emulator adb emulator-5554 install tutorialApps/CoinPirate.apk`
 - `run-emulator adb emulator-5554 install sms.apk`

How to

- Emulator
 - Run with 'run-emulator'



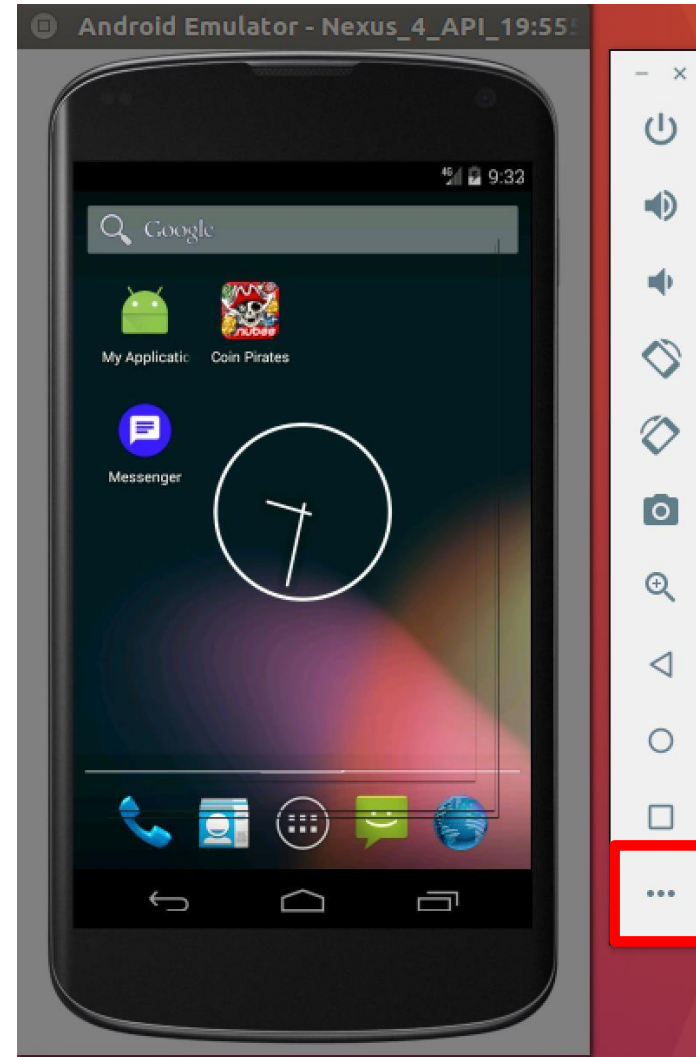
How to

- Emulator
 - Run Application
 - My Application (tutorial, not required)
 - emu-check.apk
 - Coin Pirates (tutorial, not required)
 - CoinPirates.apk
 - Messenger
 - Sms.apk (analysis target)



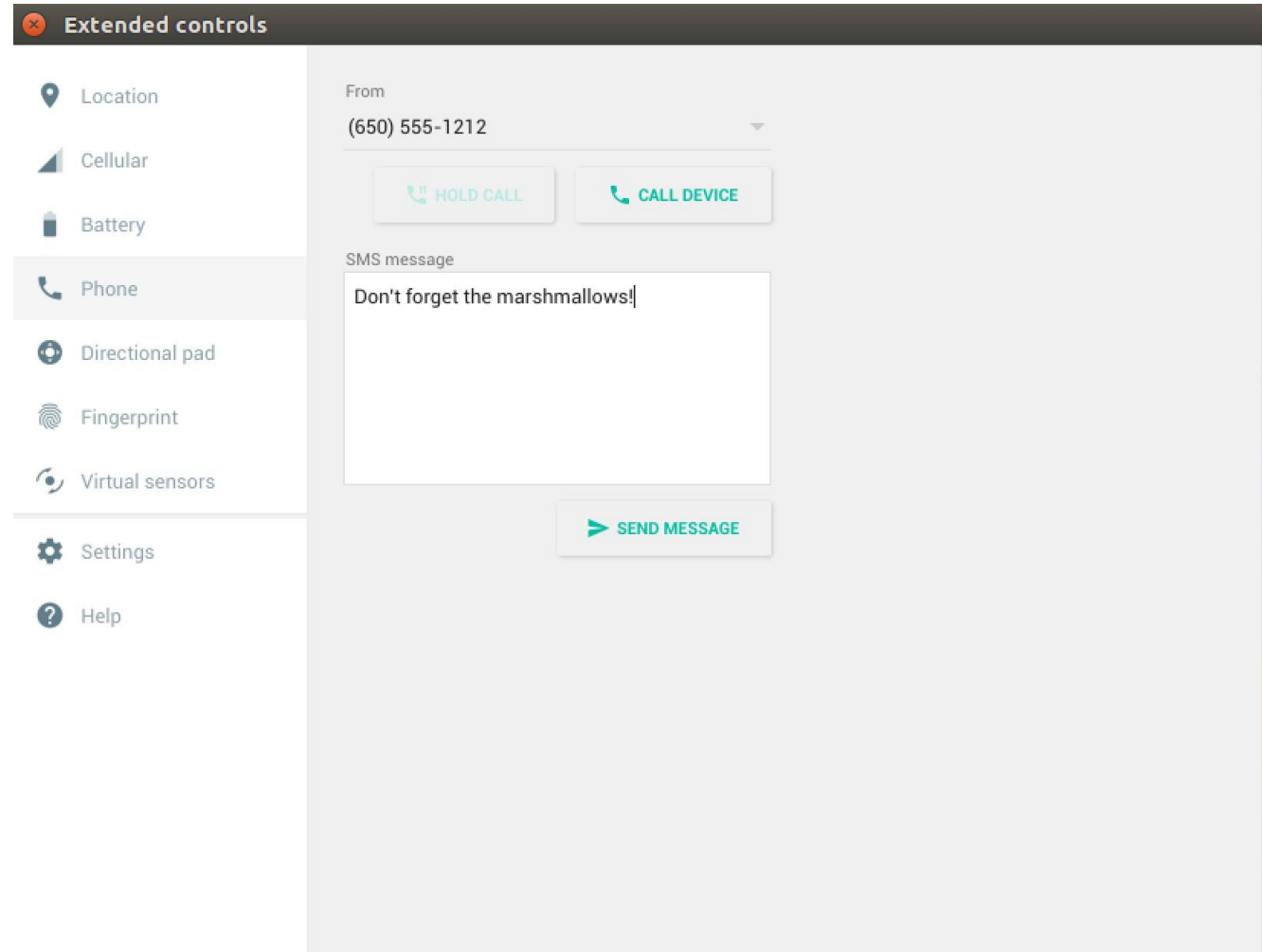
How to

- Emulator
 - Click '...' to control the emulator



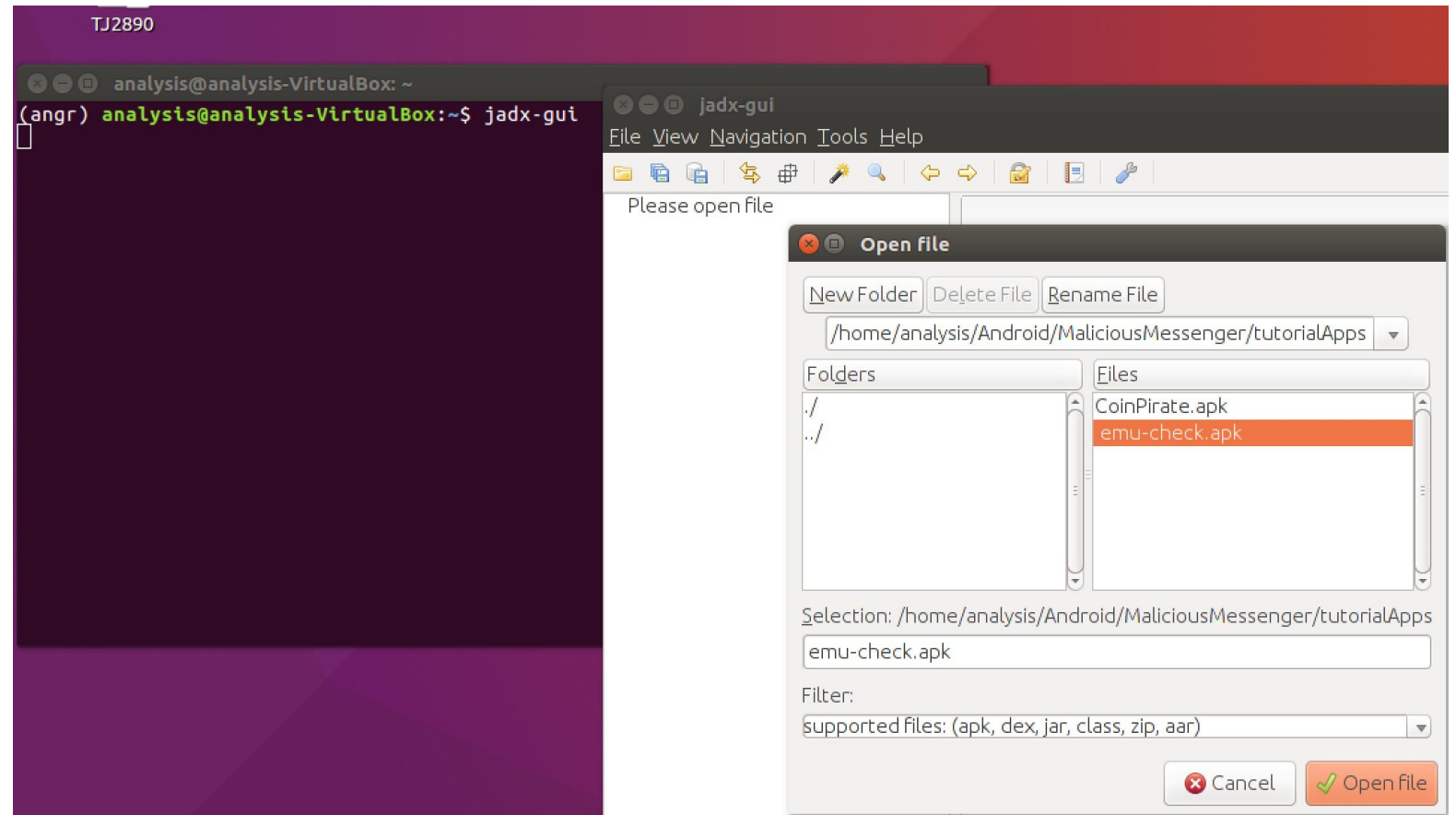
How to

- Emulator
 - Send SMS
 - Can change sender ID
 - Can change content



How to

- Decompile
 - Run jadx-gui



How to

- Disassemble

- Run apktool

- apktool d -f -r sms.apk
 - This command generates decompiled *.smali files
 - Save a copy of the APK file before doing this.

- Repackage (requires signing)

- apktool b sms -o sms.apk

- This command will re-assemble *.smali files into an apk file (as sms.apk, you can change this)

- Sign

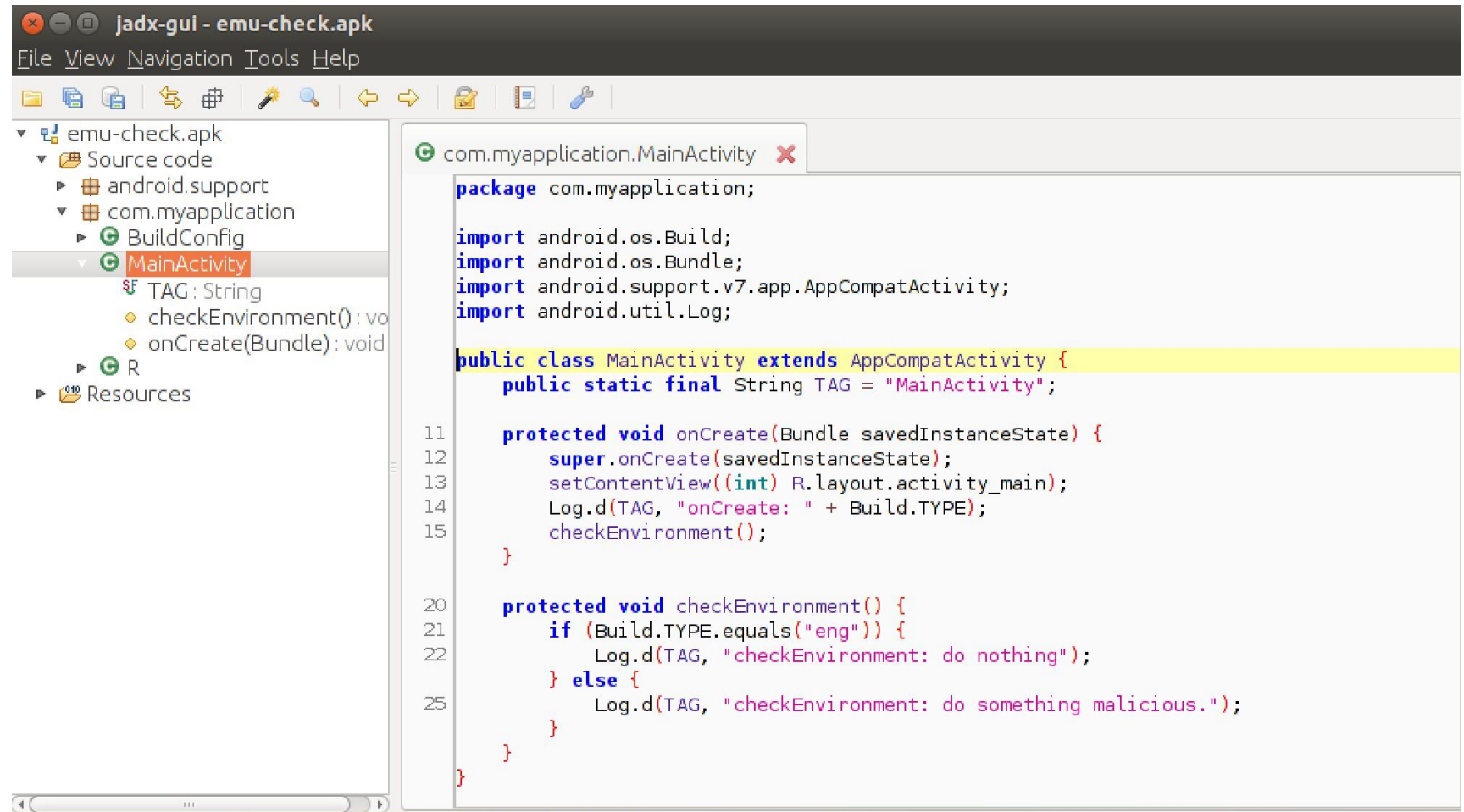
- You should sign the app to install the app to emulator
 - Run 'apksigner sign --ks ~/.android/debug.keystore sms.apk'
 - Password is 'android'

How to

- Install / uninstall (you should uninstall first to re-install the app)
 - Install
 - adb install sms.apk
 - This command will install sms.apk into the emulator
 - Make sure to turn on the emulator first
 - adb uninstall com.smsmessenger
 - This command will uninstall sms.apk from the emulator

How to

- Decompile
 - Run jadx-gui
 - Open apk file
 - Open class...



Android Tip

- Be sure that you do NOT include the “+” character as part of your phone number, even before the country code
 - For example, use “5 123 456 7890”, NOT “+5 123 456 7890”

Android Cheatsheet (thanks to Joey Allen)

•**Start Emulator** `~/bin/run-emulator`

•**Add Contact**

The sleeps are needed to allow a slow emulator time to process

`adb shell "am start -a android.intent.action.INSERT -t vnd.android.cursor.dir/contact -e name 'GatechID'" sleep 1 adb shell input keyevent 4 sleep 1 adb shell input keyevent 4`

•**Android Log** `adb logcat` Reference: [abd-logcat](#)

•**Filtered log**

The adb tool has no way to filter by app, fortunately there's a script that'll do just that. colors!

Get the script and make it executable (review it before running something off the internet :|)

`wget -O ~/bin/pidcat.py https://raw.githubusercontent.com/JakeWharton/pidcat/master/pidcat.py chmod +x ~/bin/pidcat.py`

Monitor the malware log `~/bin/pidcat.py com.smsmessenger` Reference: [pidcat](#)

•**Decompile APK**

Note: Omitting the `-r,--no-res` option allows it to decode the resources as well as the smali code.

`apktool decode ~/Android/MaliciousMessenger/sms.apk --output ~/Android/MaliciousMessenger/sms`

•**Build Modified APK**

`apktool build ~/Android/MaliciousMessenger/sms --output ~/Android/MaliciousMessenger/sms_modded.apk`

•**Sign Modified APK** `~/bin/signer.py ~/Android/MaliciousMessenger/sms_modded.apk`

•**Uninstall APK** `adb uninstall com.smsmessenger`

•**Install Modified APK** `adb install ~/Android/MaliciousMessenger/sms_modded.apk`

•**Launch the app**

The app will not be active until you run it at least once after re-installation 😓, spent a bunch of time banging my head against the wall until i figured this one out. `adb shell monkey -p com.smsmessenger -c android.intent.category.LAUNCHER 1`

•**Send an SMS**

Use single quotes or you'll need to escape the message contents.

Note: I didn't test with emojis!

`adb emu sms send 8675309 '🎵 Jenny Ive called your number...🎵'`

•**Enable Ubuntu Workspaces**

This is a personal preference but it makes it easier to separate the work into different contexts

`gsettings set org.compiz.core:/org/compiz/profiles/unity-lowgfx/plugins/core/ hsize 2` `gsettings set org.compiz.core:/org/compiz/profiles/unity-lowgfx/plugins/core/ vsize 2`

Or if you prefer using the mouse change the settings: Start -> Appearance -> Behavior -> Enable Workspaces

Questionnaire

- **1) To get credit for the project, you have to answer the questionnaire, found at ~/report/assignment-questionnaire.txt !!!!!**
- **2) Please strictly follow the format or the example answer for each question in assignment-questionnaire.txt. TAs use a autograder for your submission.**
- Windows Part
 - Read ~/report/assignment-questionnaire.txt
 - Carefully read the questions, and answer them in ~/report/assignment-questionnaire.txt
 - For each stage, there are 4-6 questions regarding the behavior of the malware.
- Android Part
 - READ ~/Android/MaliciousMessenger/writeup.pdf
 - Carefully read the writeup, answer in ~/report/assignment-questionnaire.txt
 - Make sure you overwrite ANSWER_HERE

Submitting Questionnaire

- Required files
 - Zip the following files and upload report.zip to Canvas
 - Running ~/archive.sh will automatically zip all of the files
 - ~/report/assignment-questionnaire.txt
 - Stage1.exe, stage2.exe, payload.exe (linux malware)
 - ~/tools/network/iptables_rules
 - ~/tools/cfg-generation/score.h
 - Running ~/archive.sh will create report.zip **automatically**
 - Please check the content of your zip file before submitting it to Canvas

Typos in Questionnaire

Line number	What	Action
Line 194	extra Example in 3.1 answer section	Safe to ignore, if you removed it, that is fine too
Line 252	Misspelled Funtion List	Safe to ignore, if you correct it, that is fine. But, if you completely removed it, there will be a penalty
Line 122	Typo "three commands"	Safe to ignore, make sure you have four commands in the answer section

Project 3 Rubric

* - The value for each max score is within its particular section – Windows has 110 possible points, and Android has 100. As each section is worth an equal amount of your overall P2 grade, we normalized the Windows score by dividing by 1.1 (and rounded up), then averaged it with the Android score to get your final grade. So effectively, each point in the table above is worth half a point of your final project grade (slightly less for Windows).

** - If Partial Credit column is blank, there is no partial credit for the question. “Ratio” refers to Levenshtein ratio, it’s a metric of similarity between strings.

Question Number	Subquestion	Max Score*	Partial Credit**	Penalty
1.1 (Stage 1)	IP	1		
	URL	1	0.5 pts if ratio > 0.9	
1.2	Process Name	2	1 pt if ratio > 0.8	
1.3	Command 1	4		
	Command 2	4		
	Command 3	4		
1.4	Command	2	1 pt if correct answer in student answer/vice versa	
1.5	IP	1		
	Domain	1	0.5 pts if ratio > 0.9	
2.1 (Stage 2)	IP	2		
	URL	2	1 pt if ratio > 0.9	
2.2	Process Name	2	1 pt if ratio > 0.8	
2.3	Command 1	3		
	Command 2	3		
	Command 3	3		
	Command 4	3		
2.4	Command	2	1 pt if correct answer in student answer/vice versa	
2.5	Path	6	5 pts if student answer in correct answer, 4 pts if ratio > 0.8	
2.6	Command	4		
3.1 (Linux Malware)	Function Name 1	4		
	Function Name 2	4		
3.2	Constraint	4		
3.3	Constraint	4		
3.4	Command/Target 1	10		
	Command/Target 2	10		
3.5	Function Addresses	24	No partial credit per address, you get a fraction of the max point per correct address	If you provide more addresses than the correct answer, a -4 pt penalty will be applied
4.5.1 (Android)	Question 0	5		
4.5.2	Question 1	10		
4.5.3	Question 2	20		
4.5.4	Question 3	20		
4.6.1	Question 4	15		
4.6.2	Question 5	30		