ONR-MURI N0014-16-1-2710
DIFT implementation transition to Naval Research Laboratory, July 30th 2021

MURI Title:

ADAPT: AN ANALYTICAL FRAMEWORK FOR ACTIONABLE DEFENSE AGAINST ADVANCED PERSISTENT THREATS

ONR MURI ADAPT Website: https://adapt.ece.uw.edu/

Program Manager
Dr. Daniel Koller, Email: daniel.koller1@navy.mil

Principal Investigator and Technical Point of Contact:

Prof. Radha Poovendran

Network Security Lab
BOX 352500, Dept. of Electrical Engineering
University of Washington, Seattle, WA 98195-2500

Email: rp3@uw.edu
Phone Number: (206) 605-8540
Fax: (206) 543-3842

TEAM:
Georgia Institute of Technology: W. Lee
University of California, Berkeley: S. Sastry, A. Joseph
University of California, Santa Barbara: J. Hespanha
University of Illinois, Urbana-Champaign: T. Başar
University of Washington: R. Poovendran, M. Fazel, L. Bushnell

Github Link: https://github.com/kalanaS95/Light-Weight-DIFT

For additional information, contact: Kalana Sahabandu,
email: ksahaban@uw.edu

Acknowledgement: This work was supported by ONR grant N00014-16-1-2710 P00002 and DARPA TC grant DARPA FA8650-15-C-7556.

Light weight DIFT on Linux 32 bit

View from the attacker's perspective Light weight DIFT implementation

View from the attacker's perspective Light weight DIFT was implemented to give security analysts an clear picture of the processes created by a malicious program during its execution. This tool helps to visualize process trace of an malicious program using graphs through the logs generated while the malicious program was running. Attacker-side Light weight DIFT implementation mainly consists of two components.

1. **Intel PIN based logging tool**

Logging tool is a C/C++ program built using Intel Pin tool which allows to bind to a malicious program and detect all the processes spawned during its run time. During spawning of new processes, we are capturing above following information

- Commands executed by the process along with the arguments passed into the command.
- User of the process
- Group which the use belongs to

and log them into a log file. This file then fed into the below mentioned visualization component for post processing

2. **Visualization tool**

   Visualization component is a python script that acts as a post processing tool which takes log file produced by the logging tool and produce in-memory python dictionary objects for each unique process along with their children processes spawned during the program run time. Ultimately, these in-memory JSON objects are fed to the Neo4j graphing database management system through Neo4j python driver for visualization.

View from the victim's perspective Light weight DIFT implementation

View from the victim's perspective Light weight DIFT was implemented to capture the behavior of the processes during an execution of an malicious remote attack happens through SSH and SCP. Victim-side light weight DIFT helps to visualize process trace of a victim system during the execution of a SSH and SCP based attack. Implementation of the victim-side light weight DIFT consists of three components.

1. **Start and termination script**
   This component will first start the logging program on the victim side just right before it starts the attack. This is achieved by sending a signal to the victim computer to start the process logging program.

   Once the attack is finished this script will send another signal to the victim computer to stop the process logging in 20 seconds once the signal is received (This 20 second padding time is added to give processes related to the attack created on the victim side to finish their tasks in case they are still executing during the time the signal is received.)

   This synchronization mechanism (start and stop signals) added to only to clearly capture processes created during the time frame of the attack.

2. **Victim-side logger**
   This component is written using C/C++ to directly connect with Linux kernel using Netlink kernel module which will allow us to get process notifications along with their PID (Process ID), PPID (Parent Process ID) and executable. At the end of the execution of the victim-side logger, it will produce a log of all the processes executed in the victim system during the time frame of the attack.

3. **Visualization tool**
   The process log produced by the above-mentioned C/C++ is then fed into the python-script. Python script will read through the log file and first find the line where the stop signal was received. Then it will take all the process log information up-to that time frame and create relationships between children and parent processes and draw the graph

What is screengrab ?

Screengrab is a malicious program written in C language which has the ability to takescreenshot(s) of the host computer without the user’s permission. Screengrab has the abilityto occasionally take screenshot(s) of the victim system and send selected screenshot(s) to an attacker specified server. For the purpose of analysis scenarios presented, screengrab was configured to send the captured screenshot(s) to the local-host’s(http://127.0.0.1) ”/home/kalana/Desktop” directory aliong with the file name ”attack”appended with the time stamp when the screenshot was taken.

Example:http://127.0.0.1/home/kalana/Desktop/attack2021-04-2011:28:06

What is Multi-host screengrab ?

Multi-host screengrab is an extension of the screengrab attack. Where screengrab infect all the computers within a computer network by sending screengrab attack as an payload to all the computers and remotely running the attack on all the computers in the network without user’s knowledge. We assume, attacker will first grant access to one of the computers in the network through SSH and inject all the files needed for the attack. Once this step is completed multi-host screengrab will replicate itself within the network infecting all the computers

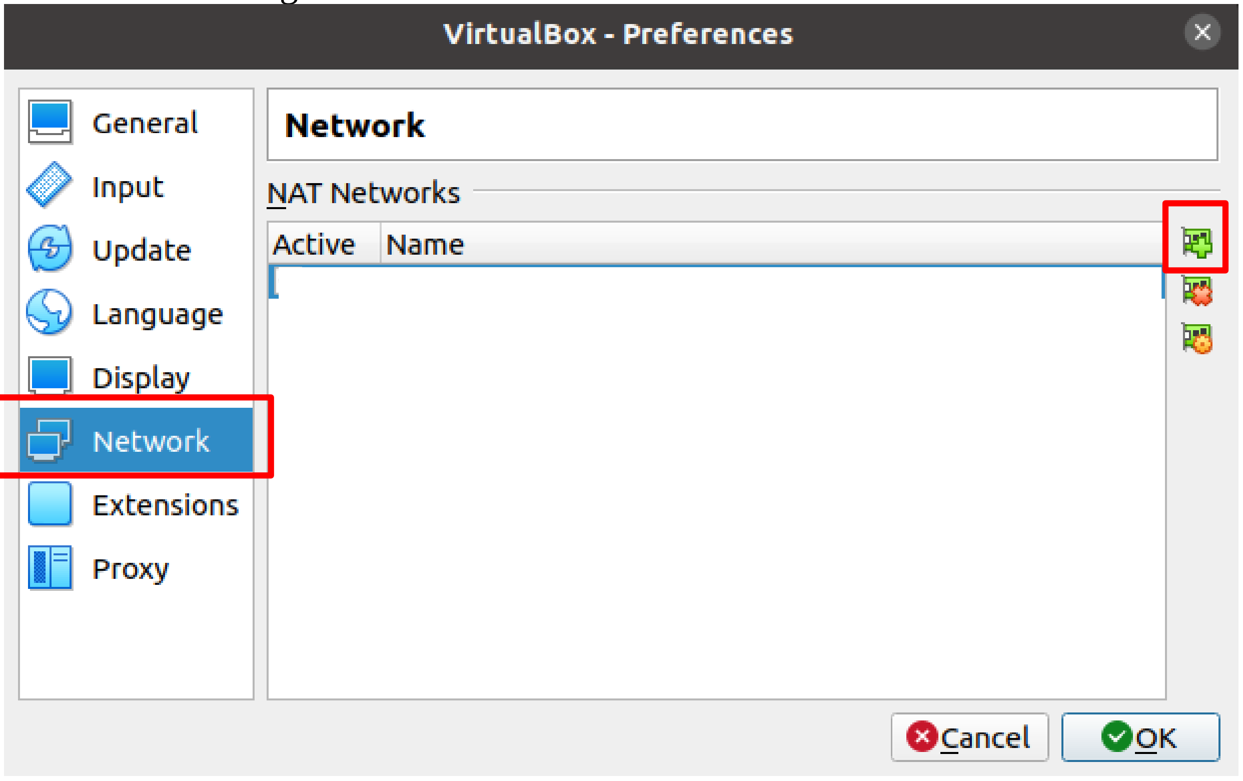
Installation guide

Note: Following installation guide was tested on Ubuntu 20.04 LTS

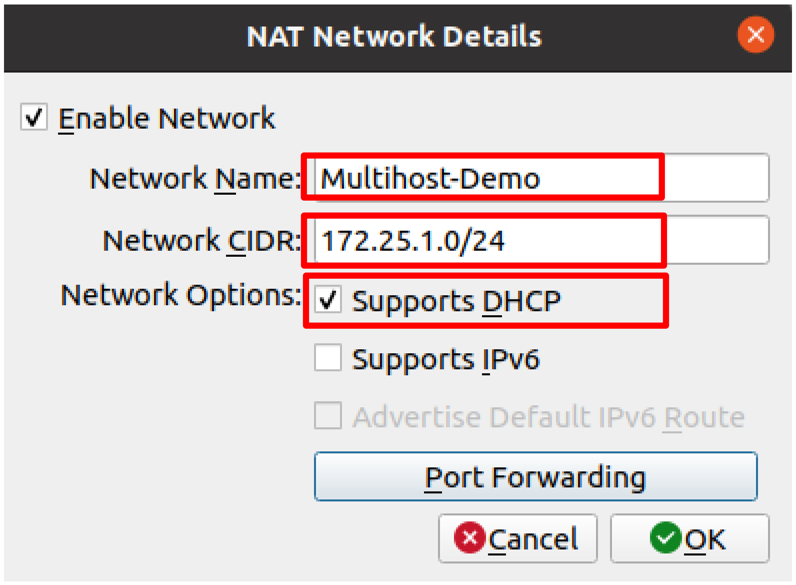
Installation of VirtualBox and setting up a NAT network

1. First download the latest version of virtual box from the following link: https://www.virtualbox.org/wiki/Downloads

2. Once installation is completed, we need to create a NAT network within the virtual box, such that we can network attacker-side virtual computer and the victim-side computer. Following steps will discuss about, how create a NAT network.
   1. First navigate to File → Preferences (shortcut: Ctrl + G).
   2. Then click on the Network tab on the left side bar and then click on the add button on the right hand side of the window.



   3. Then you will be presented with a pop-up window. In that pop-up window type the following details as show in the screenshot below.



[✓] Enable Network

Network Name: Multihost-Demo

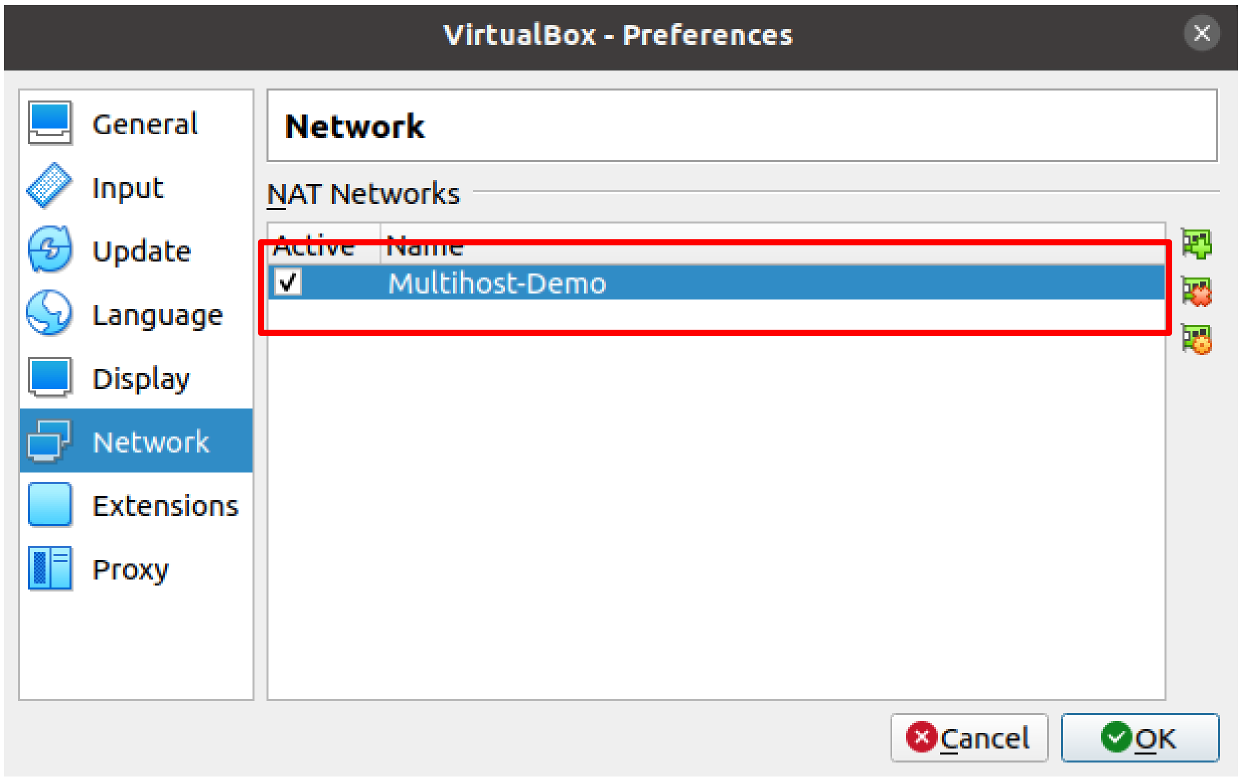
Network CIDR: 172.25.1.0/24

Network Options: [✓] Supports DHCP

[ ] Supports IPv6

[ ] Advertise Default IPv6 Route

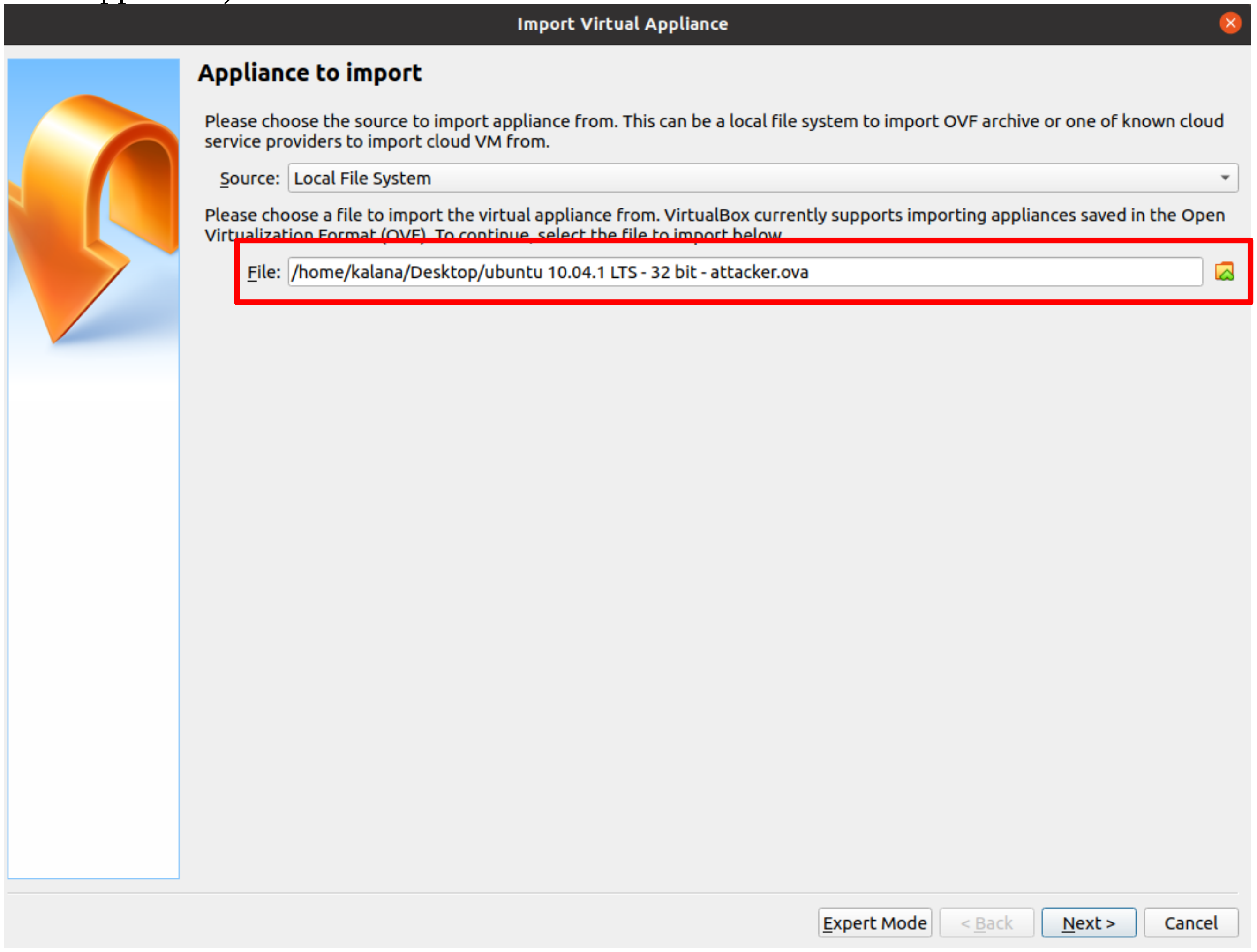
4. Then click OK. Then you will see a NAT under NAT networks called “Multihost-Demo”



Active	Name
✓	Multihost-Demo

Installing attacker-side VirtualBox disk image.

1. First download the "ubuntu 10.04.1 LTS - 32 bit - attacker.ova" file
   Download Link to the disk image:
   https://www.dropbox.com/sh/4yc3uhyczi0hjzm/AAC21rVdqkaXlP4nvWILZuUQa?dl=0
2. Navigate to File → Import appliance (Shortcut Ctrl + I).
3. Then choose "ubuntu 10.04.1 LTS – 32 bit – attacker.ova" from the file explorer (VirtualBox images can be found under "VirtualBox Images" directory in the unzipped file) window and click next.


Import Virtual Appliance

Appliance to import

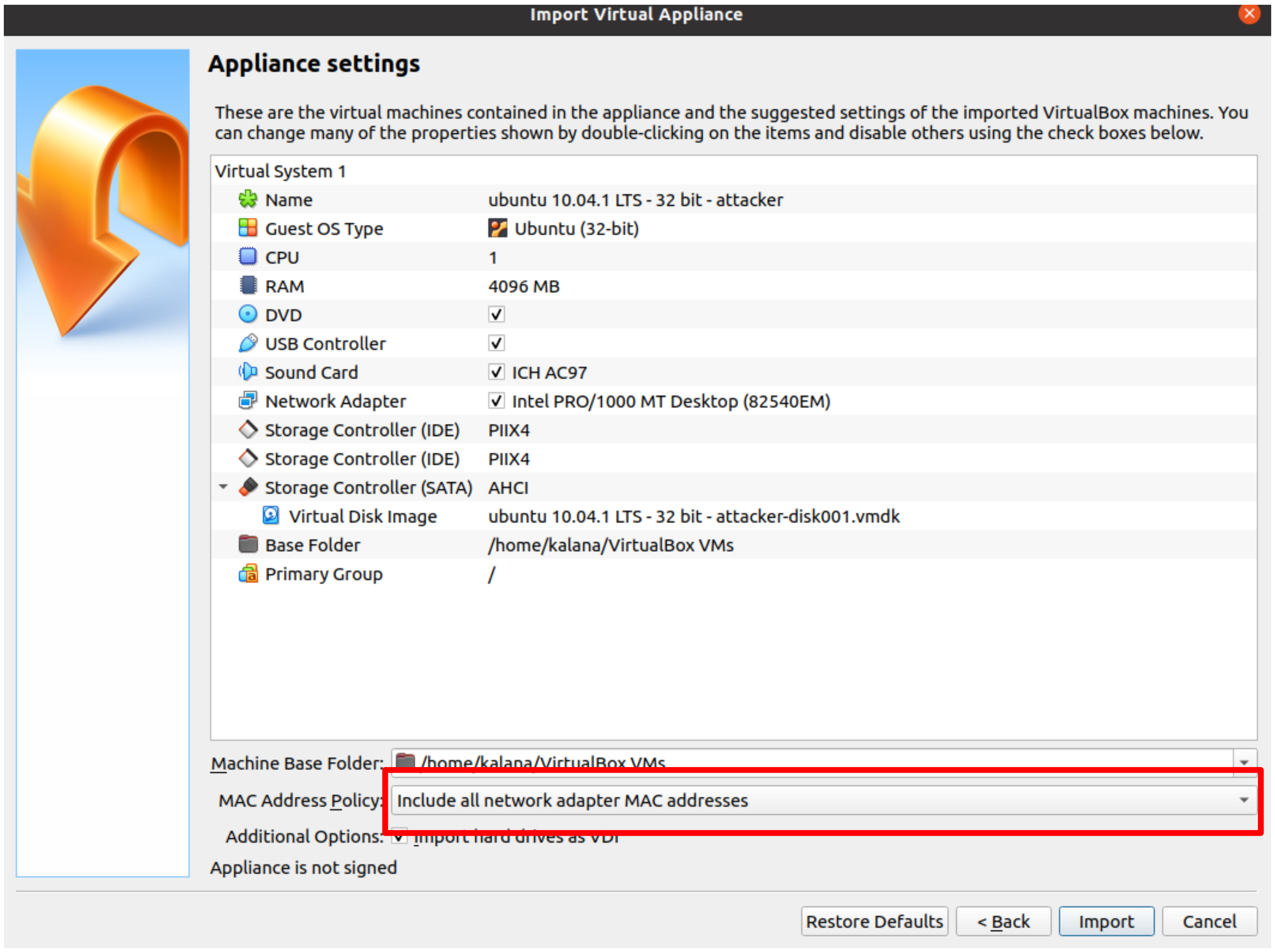
Please choose the source to import appliance from. This can be a local file system to import OVF archive or one of known cloud service providers to import cloud VM from.

Source: Local File System

Please choose a file to import the virtual appliance from. VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

File: /home/kalana/Desktop/ubuntu 10.04.1 LTS - 32 bit - attacker.ova

4. From the next window, make sure to choose "include all network adapter MAC addresses" from the MAC Address Policy drop-down menu. And click import. Once finished you will see this virtual machine added to under virtual machines section of the VirtualBox window



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Virtual System 1

Name	ubuntu 10.04.1 LTS - 32 bit - attacker
Guest OS Type	Ubuntu (32-bit)
CPU	1
RAM	4096 MB
DVD	✓
USB Controller	✓
Sound Card	✓ ICH AC97
Network Adapter	✓ Intel PRO/1000 MT Desktop (82540EM)
Storage Controller (IDE)	PIIX4
Storage Controller (IDE)	PIIX4
Storage Controller (SATA)	AHCI
Virtual Disk Image	ubuntu 10.04.1 LTS - 32 bit - attacker-disk001.vmdk
Base Folder	/home/kalana/VirtualBox VMs
Primary Group	/

Machine Base Folder: /home/kalana/VirtualBox VMs

MAC Address Policy: Include all network adapter MAC addresses

Additional Options: ✓ Import hard drives as VDI

Appliance is not signed

Note: Passwords for both virtual computers are 131kalana

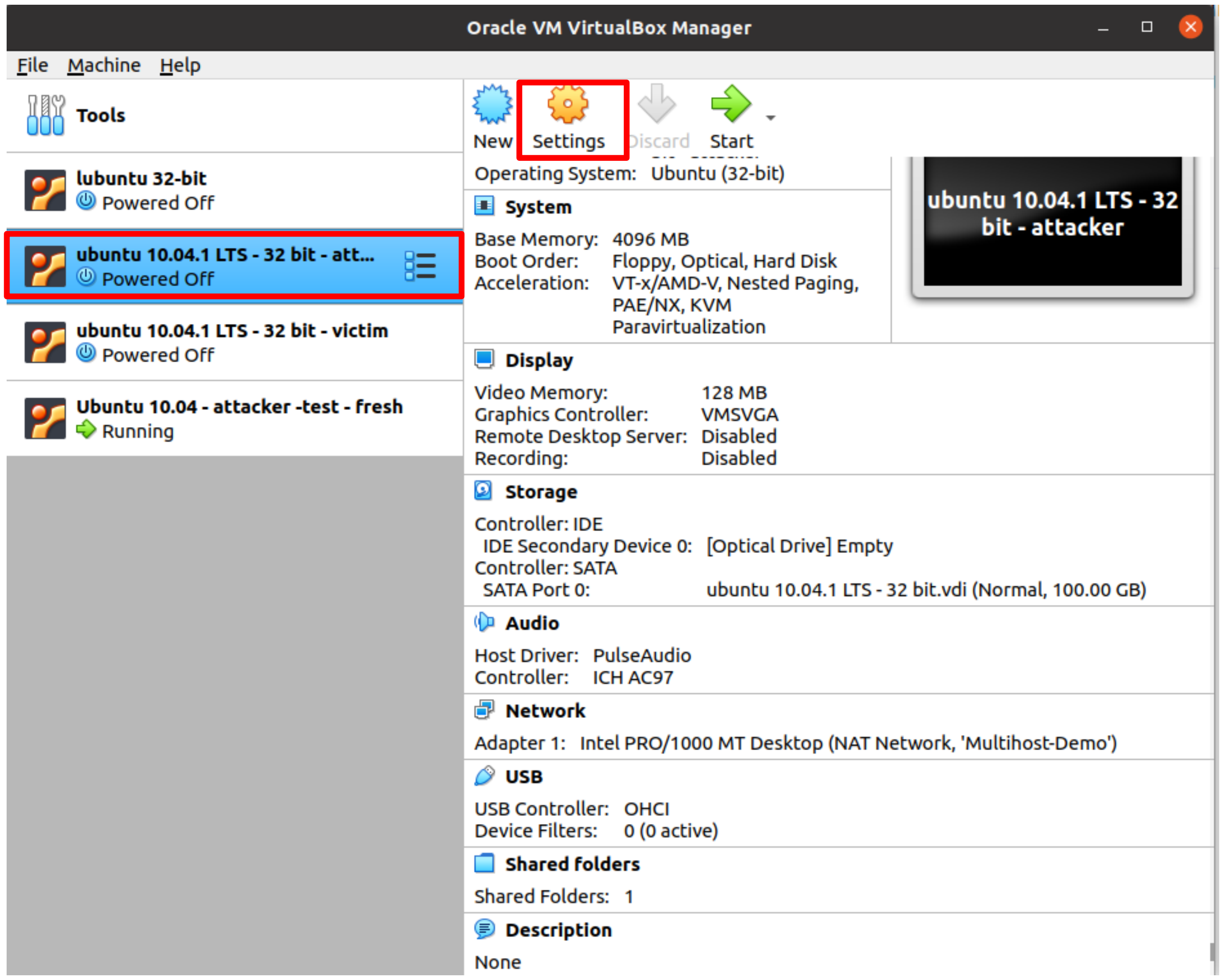
Installing victim-side VirtualBox disk image.

Follow the steps described in "Installing attacker-side VirtualBox disk image." section to install the victim-side VirtualBox disk image. This time select "ubuntu 10.04.1 LTS – 32 bit – victim.ova" (make sure to select "include all network adapter MAC addresses")

Download Link to the disk image:
https://www.dropbox.com/sh/4yc3uhyczi0hjzm/AAC21rVdqkaXlP4nvWILZuUQa?dl=0

Networking both computers through VirtualBox

1. First click on the attacker-side virtual computer and then click on the setting button on the top of the window


Oracle VM VirtualBox Manager

File Machine Help

Tools

lubuntu 32-bit
Powered Off

ubuntu 10.04.1 LTS - 32 bit - att...
Powered Off

ubuntu 10.04.1 LTS - 32 bit - victim
Powered Off

Ubuntu 10.04 - attacker -test - fresh
Running

New Settings Discard Start

ubuntu 10.04.1 LTS - 32 bit - attacker

Operating System: Ubuntu (32-bit)

System

Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display

Video Memory: 128 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: IDE
IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
SATA Port 0: ubuntu 10.04.1 LTS - 32 bit.vdi (Normal, 100.00 GB)

Audio

Host Driver: PulseAudio
Controller: ICH AC97

Network

Adapter 1: Intel PRO/1000 MT Desktop (NAT Network, 'Multihost-Demo')

USB

USB Controller: OHCI
Device Filters: 0 (0 active)

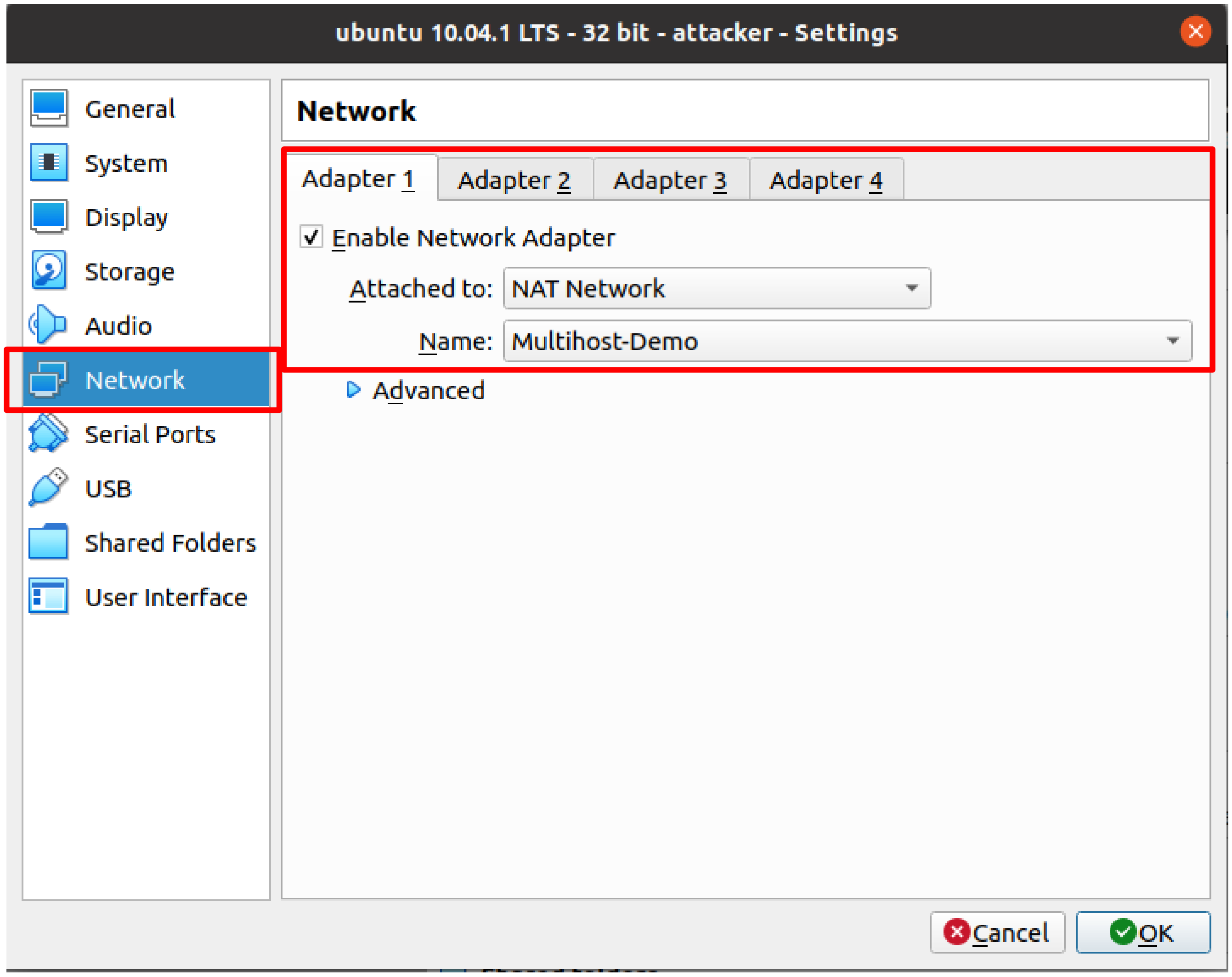
Shared folders

Shared Folders: 1

Description

None

2. Then click on the Network tab on the left hand side and select Attached to "NAT Network" from the drop down as well as Name parameter as "Multi-host" demo from the drop down


ubuntu 10.04.1 LTS - 32 bit - attacker - Settings

Network

Adapter 1

☑ Enable Network Adapter

Attached to: NAT Network

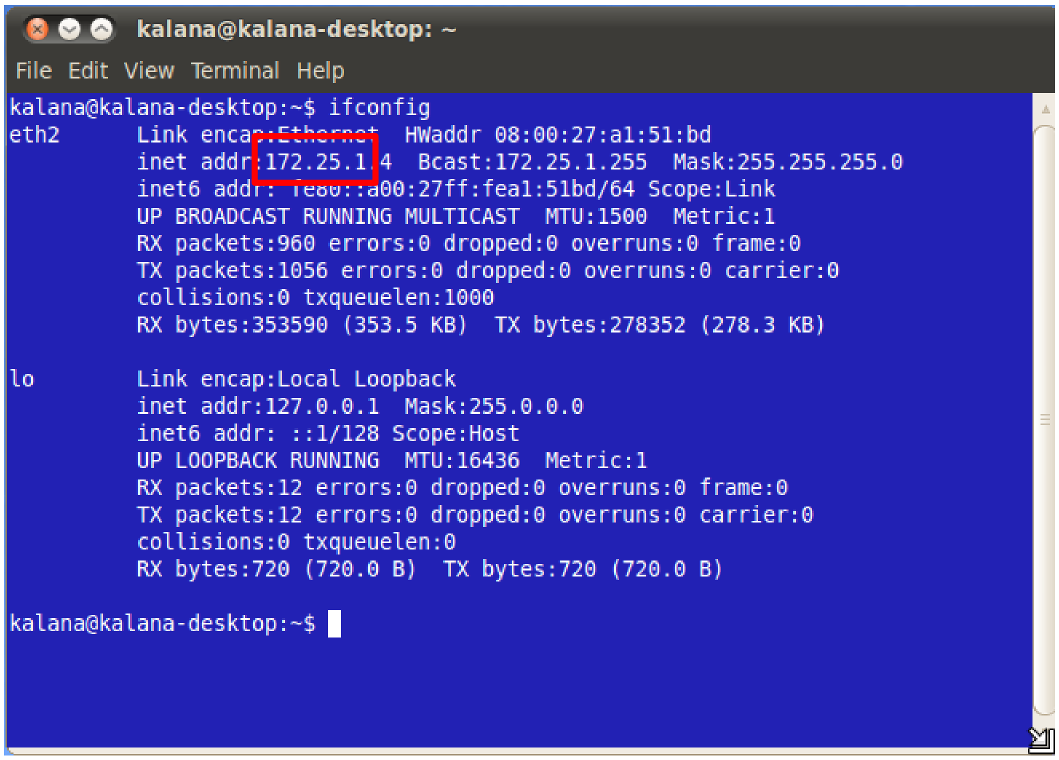
Name: Multihost-Demo

3. Repeat step 1- 2 to add victim-side virtual computer to the same network

Now let’s make sure both computers are on the same network by running following command on both attacker and victim virtual computers. (New terminal can be opened by clicking Applications → Accessories → Terminal)

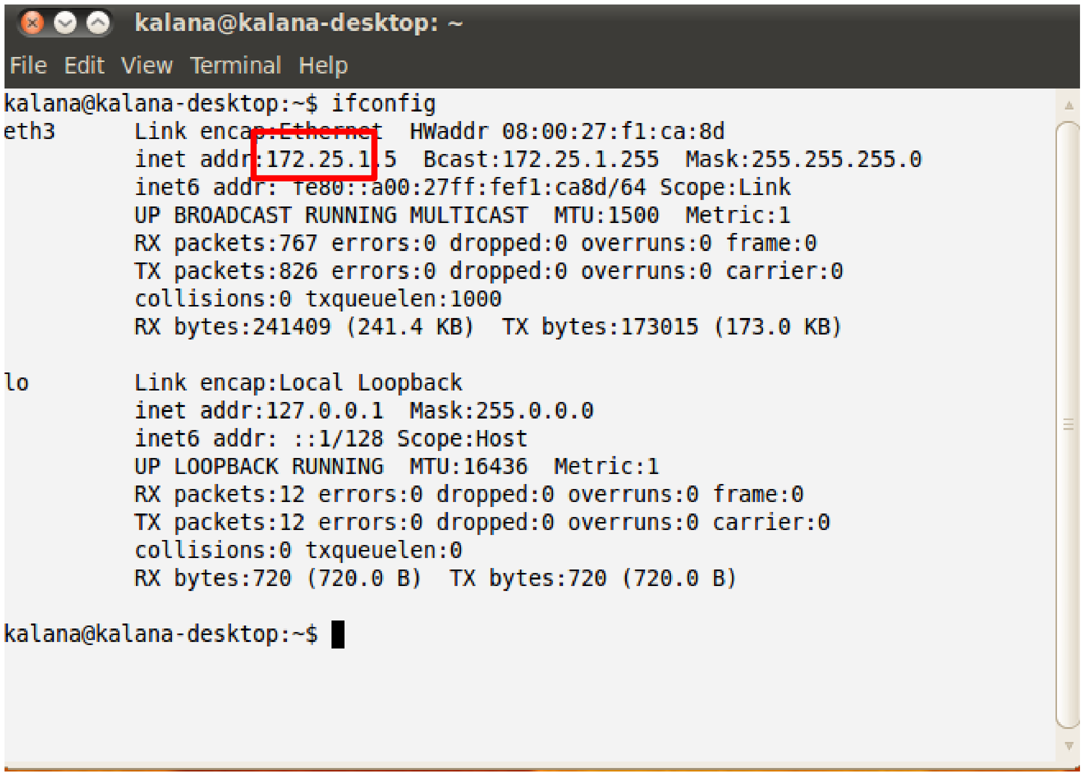
- ifconfig

if both computers are on the same network. First three parts of their IP address should be similar



```
kalana@kalana-desktop:~$ ifconfig
eth2      Link encap:Ethernet  HWaddr 08:00:27:a1:51:bd
          inet addr:172.25.1.14  Bcast:172.25.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea1:51bd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:960 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1056 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:353590 (353.5 KB)  TX bytes:278352 (278.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:720 (720.0 B)  TX bytes:720 (720.0 B)

kalana@kalana-desktop:~$
```

Attacker side



```
kalana@kalana-desktop:~$ ifconfig
eth3      Link encap:Ethernet  HWaddr 08:00:27:f1:ca:8d
          inet addr:172.25.1.15  Bcast:172.25.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef1:ca8d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:767 errors:0 dropped:0 overruns:0 frame:0
          TX packets:826 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:241409 (241.4 KB)  TX bytes:173015 (173.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:720 (720.0 B)  TX bytes:720 (720.0 B)

kalana@kalana-desktop:~$
```

Victim side

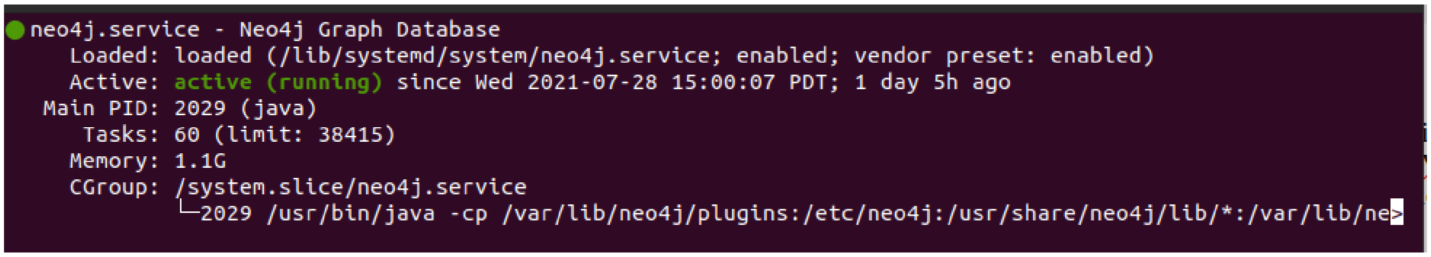
Installation of Neo4j graphing database system.

Note: Neo4j will be installed on the host computer (i.e. computer that we installed VirtualBox.)

1. First install neo4j server and command line interface (CLI) using following Linux commands
   - sudo apt update
   - sudo apt install apt-transport-https ca-certificates curl software-properties-common
   - curl -fsSL https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add -
   - sudo add-apt-repository "deb https://debian.neo4j.com stable 4.1"
   - sudo apt install neo4j
   - sudo systemctl enable neo4j.service

2. Finally check if neo4j is successfully installed on the system by running following Linux command
   1. sudo systemctl status neo4j.service

   If neo4j is successfully installed on the system, above command will display an output similar to the following output



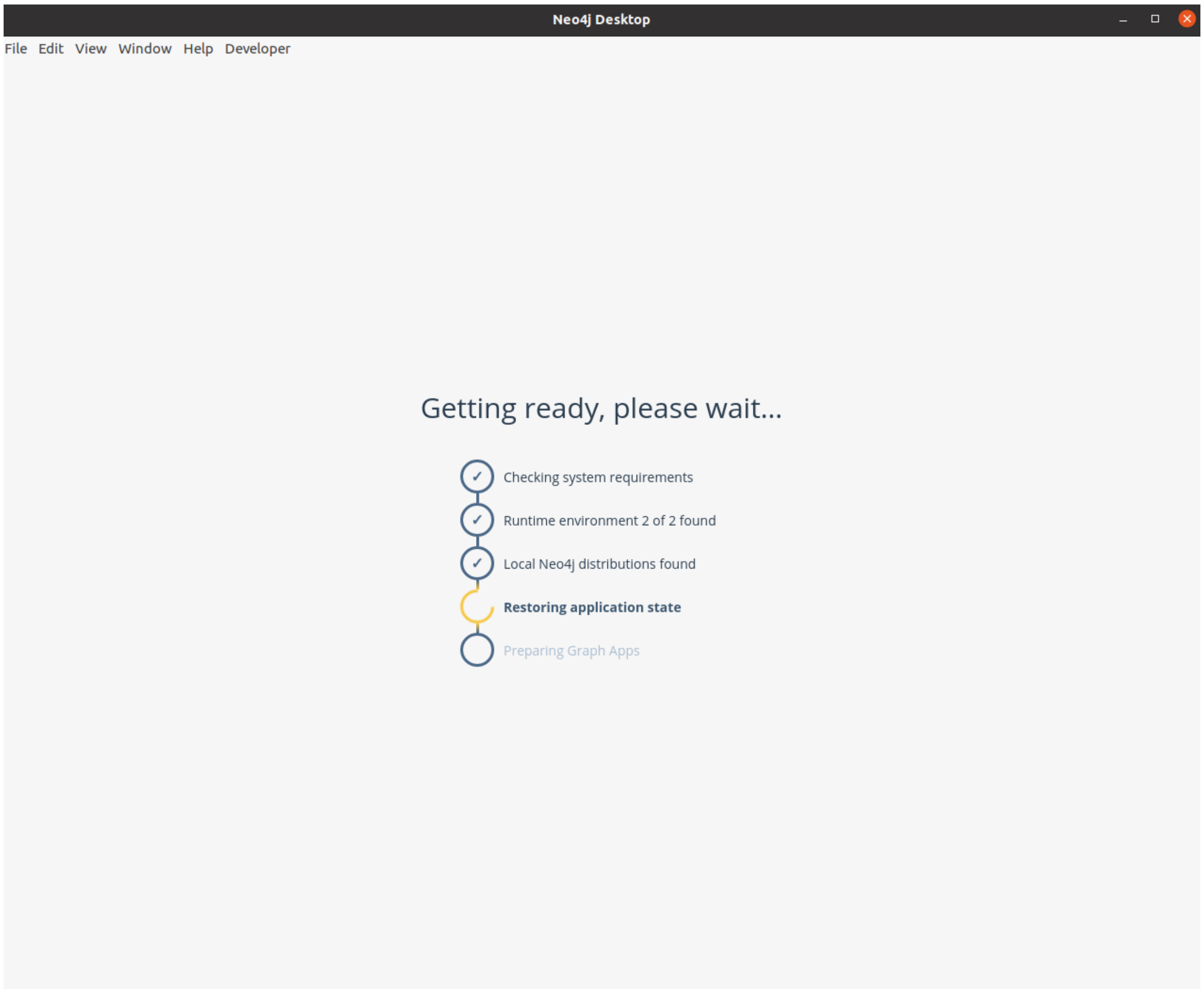
```
● neo4j.service - Neo4j Graph Database
     Loaded: loaded (/lib/systemd/system/neo4j.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2021-07-28 15:00:07 PDT; 1 day 5h ago
   Main PID: 2029 (java)
      Tasks: 60 (limit: 38415)
     Memory: 1.1G
     CGroup: /system.slice/neo4j.service
             └─2029 /usr/bin/java -cp /var/lib/neo4j/plugins:/etc/neo4j:/usr/share/neo4j/lib/*:/var/lib/ne>
```

3. Once neo4j server and CLI is installed successfully, now we can install the Neo4j front-end. This will allow us to easily create new graphing database through the GUI. To install Neo4j front-end, first locate the **“neo4j-desktop-1.4.5-x86_64.AppImage”** bundled with the .zip file.

4. Then execute the following linux command to grant necessary permission to execute the APPImage file
   - chmod +X neo4j-desktop-1.4.5-x86_64.AppImage

5. Now let’s make sure, Neo4j front-end works by executing following command.
   - ./neo4j-desktop-1.4.5-x86_64.AppImage

Upon successful execution, you will be greeted with a window similar to following



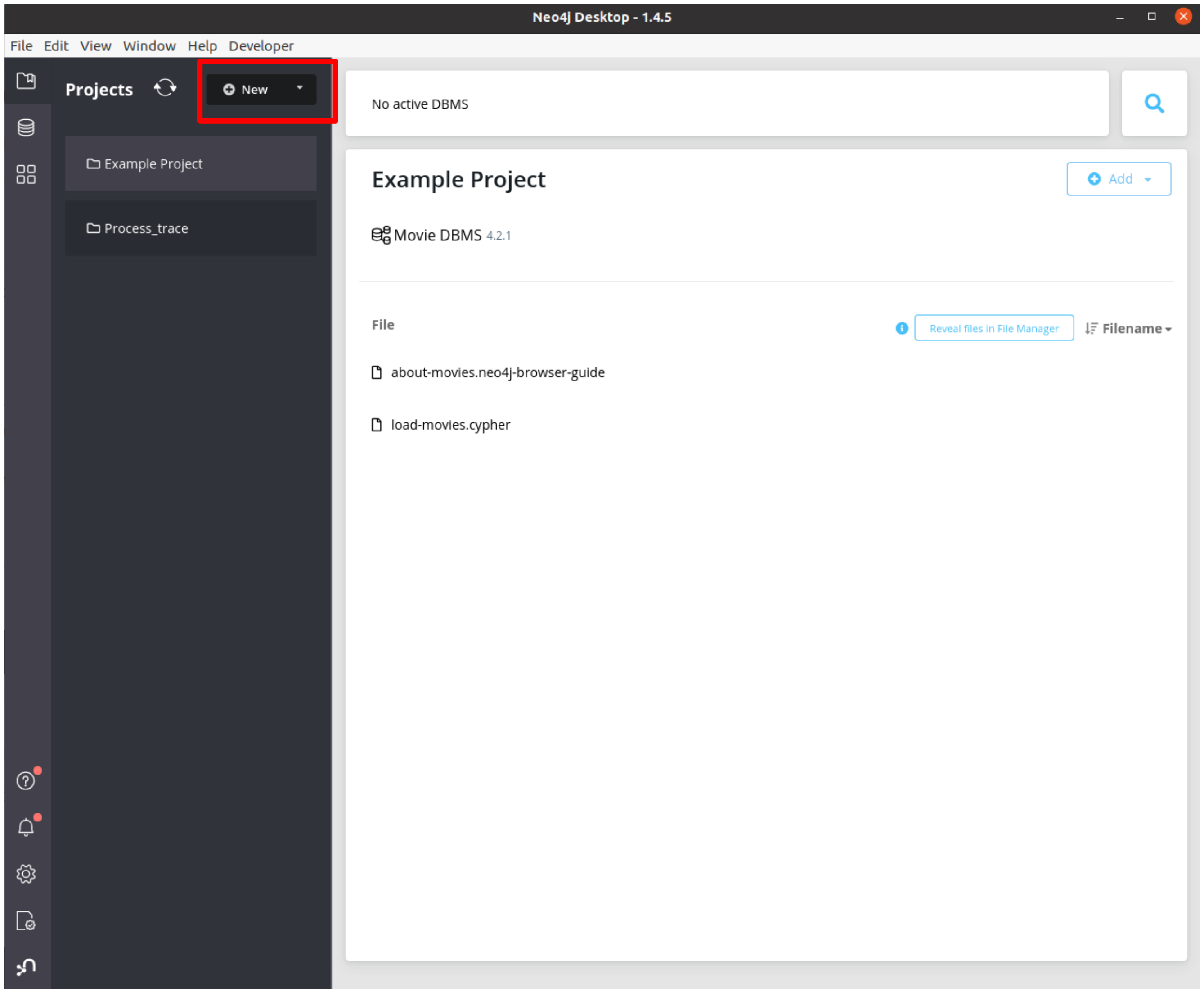
Getting ready, please wait...

✓ Checking system requirements
✓ Runtime environment 2 of 2 found
✓ Local Neo4j distributions found
Restoring application state
Preparing Graph Apps

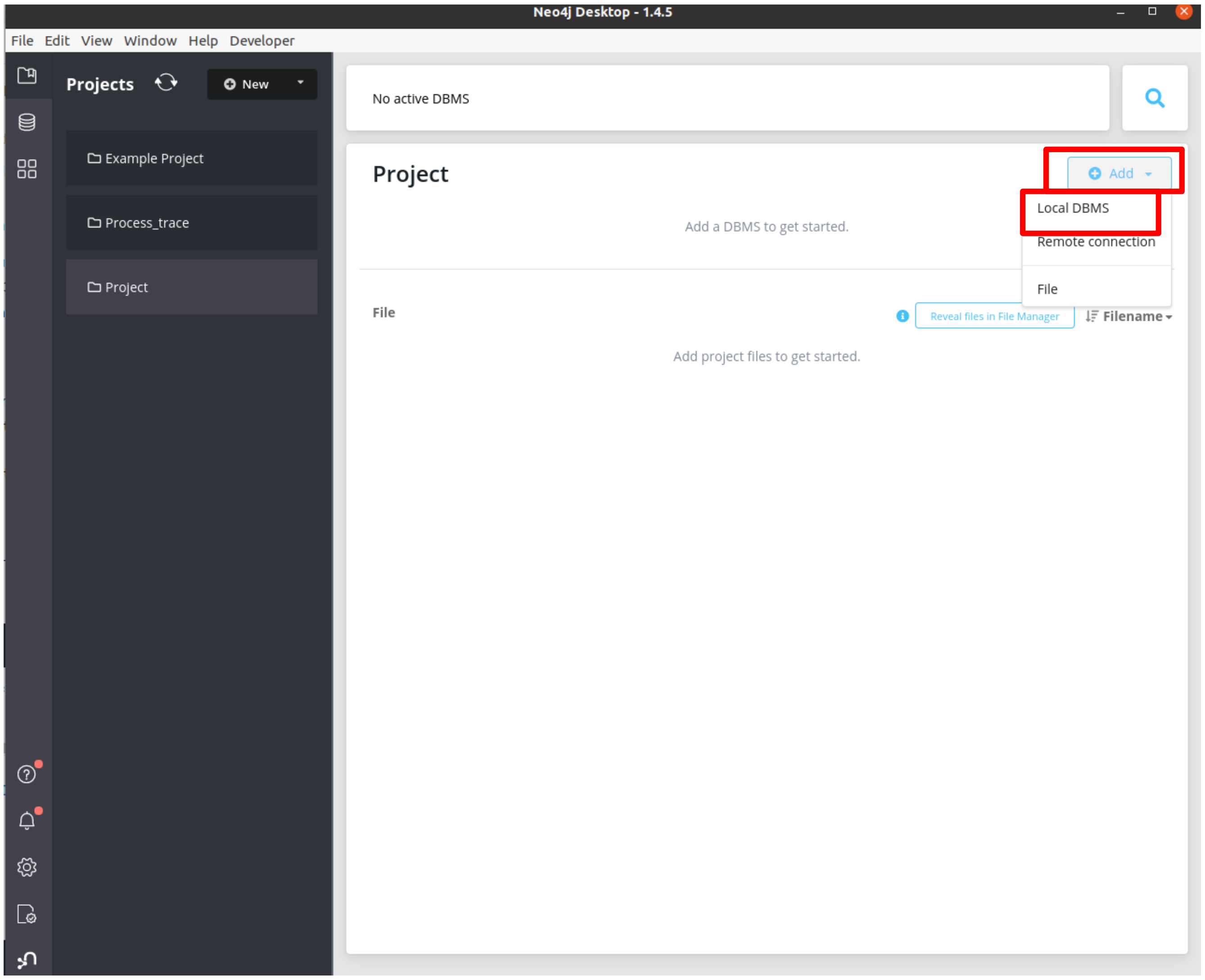
Setting-up Neo4j for victim and attacker-side visualization tools

1. First launch the Neo4j front-end by executing following command
   - ./neo4j-desktop-1.4.5-x86_64.AppImage

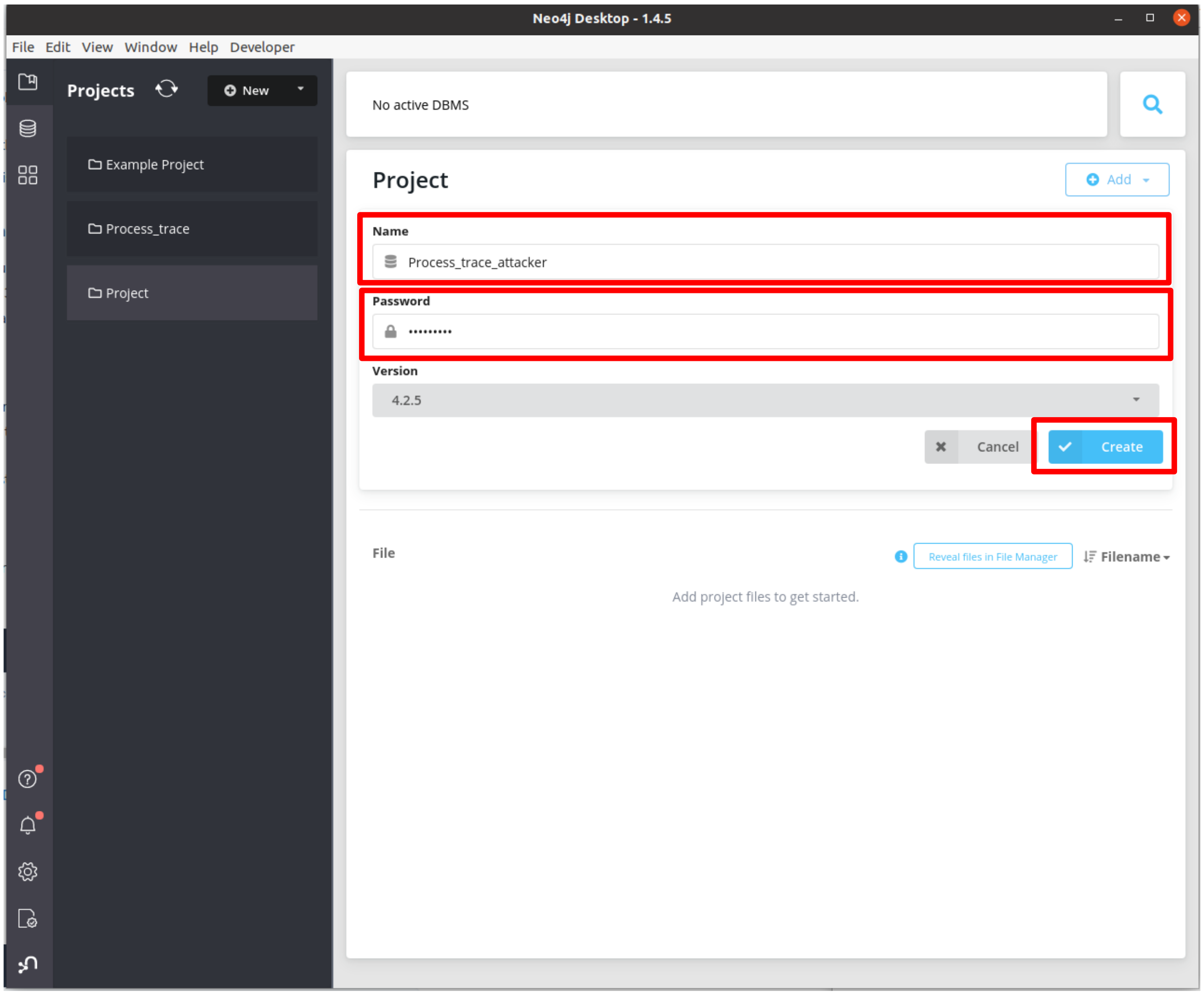
2. Now lets create a new project folder by clicking the “New” button on the top left hand side ribbon



3. Then press on the “Add” button on the top right hand side of the window and click on the “Local DBMS” button.



4. Then give the Database a name and a password. For the purpose of the guide. I will name it “Process_trace_attacker” and password as “131kalana” and press create button
    - Note: make sure to store the password somewhere because we need this password to connect to the database for both the Visualization tools

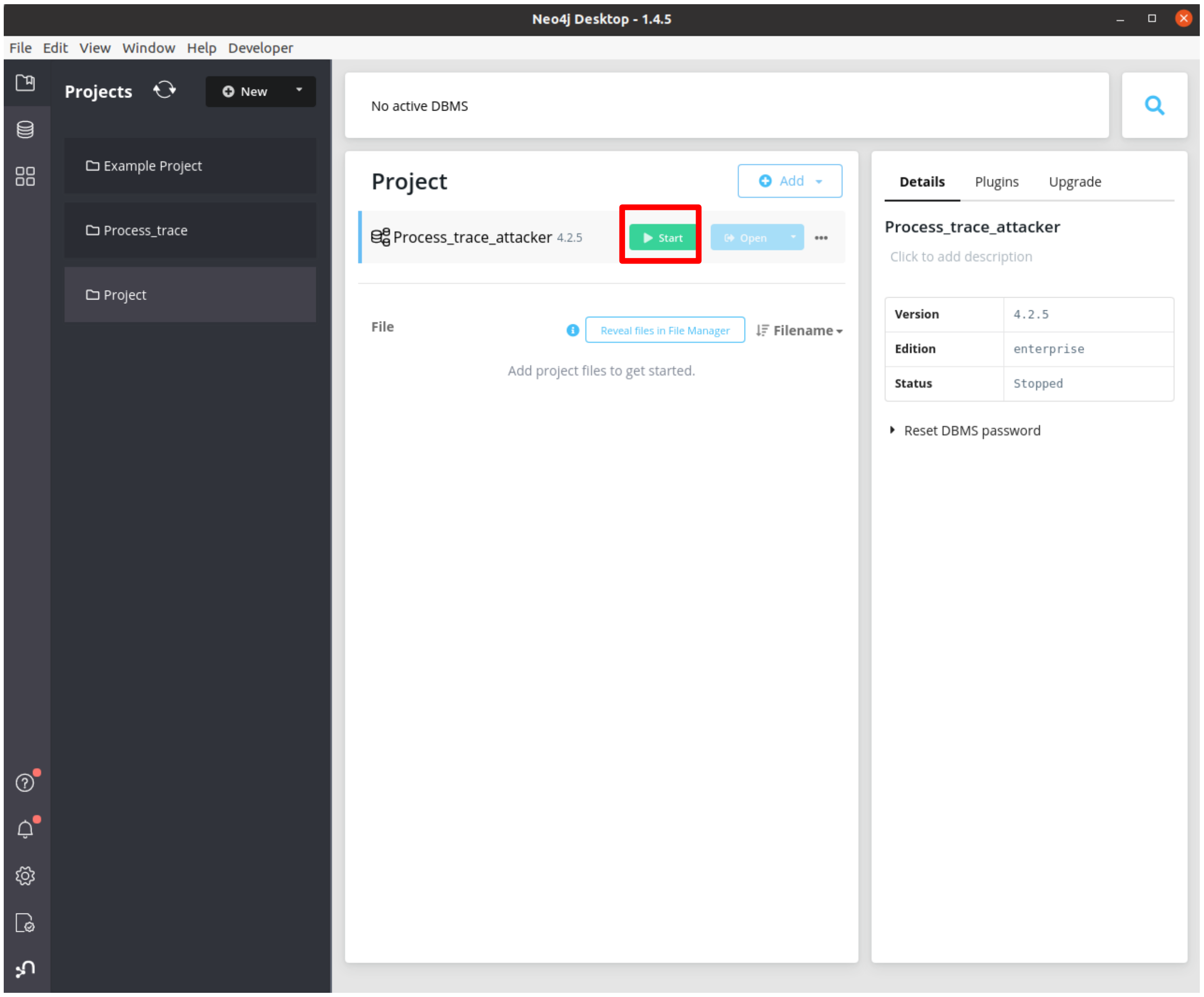


Name: Process_trace_attacker

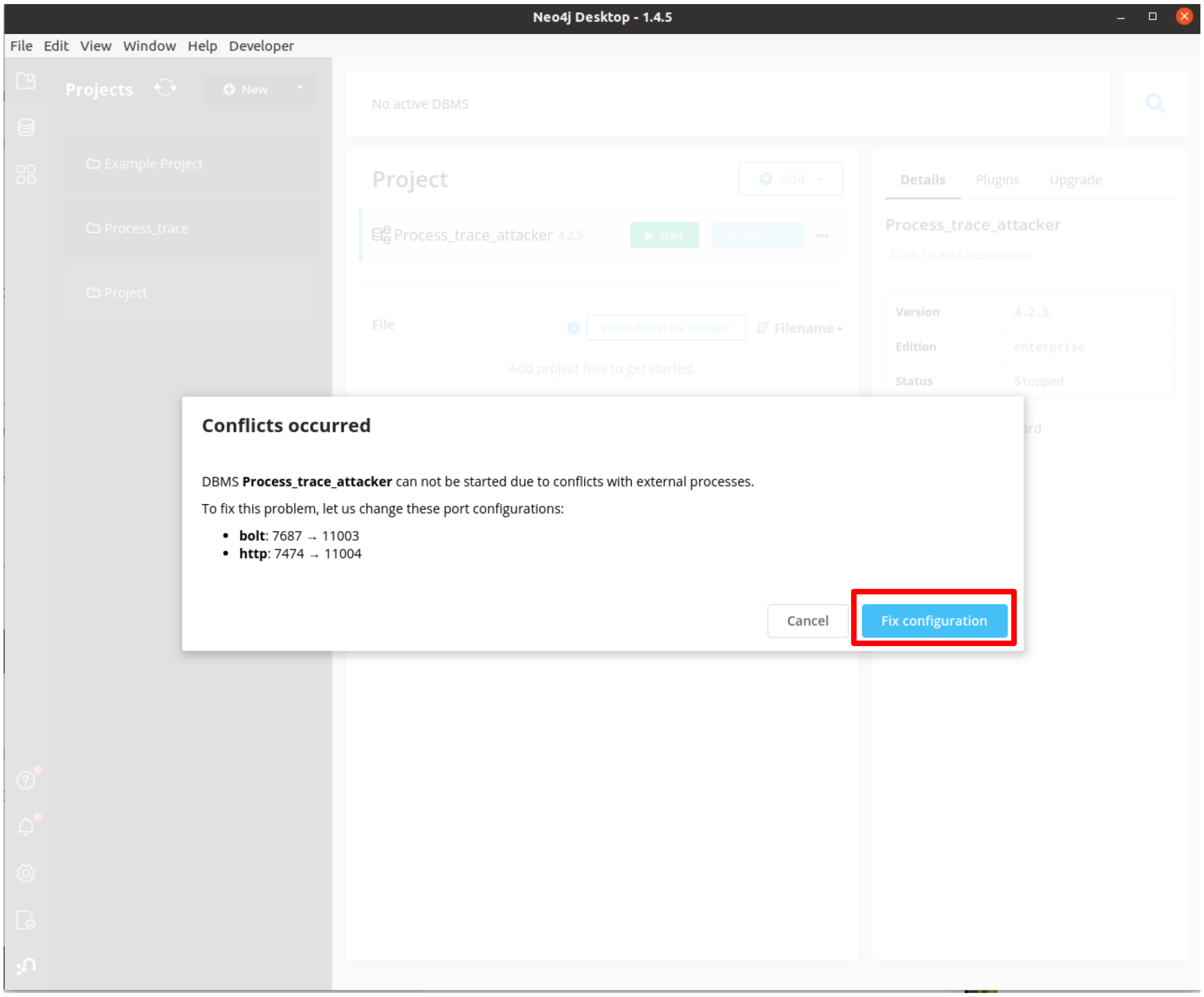
Password: •••••••••

Version: 4.2.5

5. To make sure everything is working as expected click on the start button to start the database



- Note: in case if you get a Conflicts encountered warning press “Fix configuration” on the message. This should fix the port conflict issue by the Neo4j program.



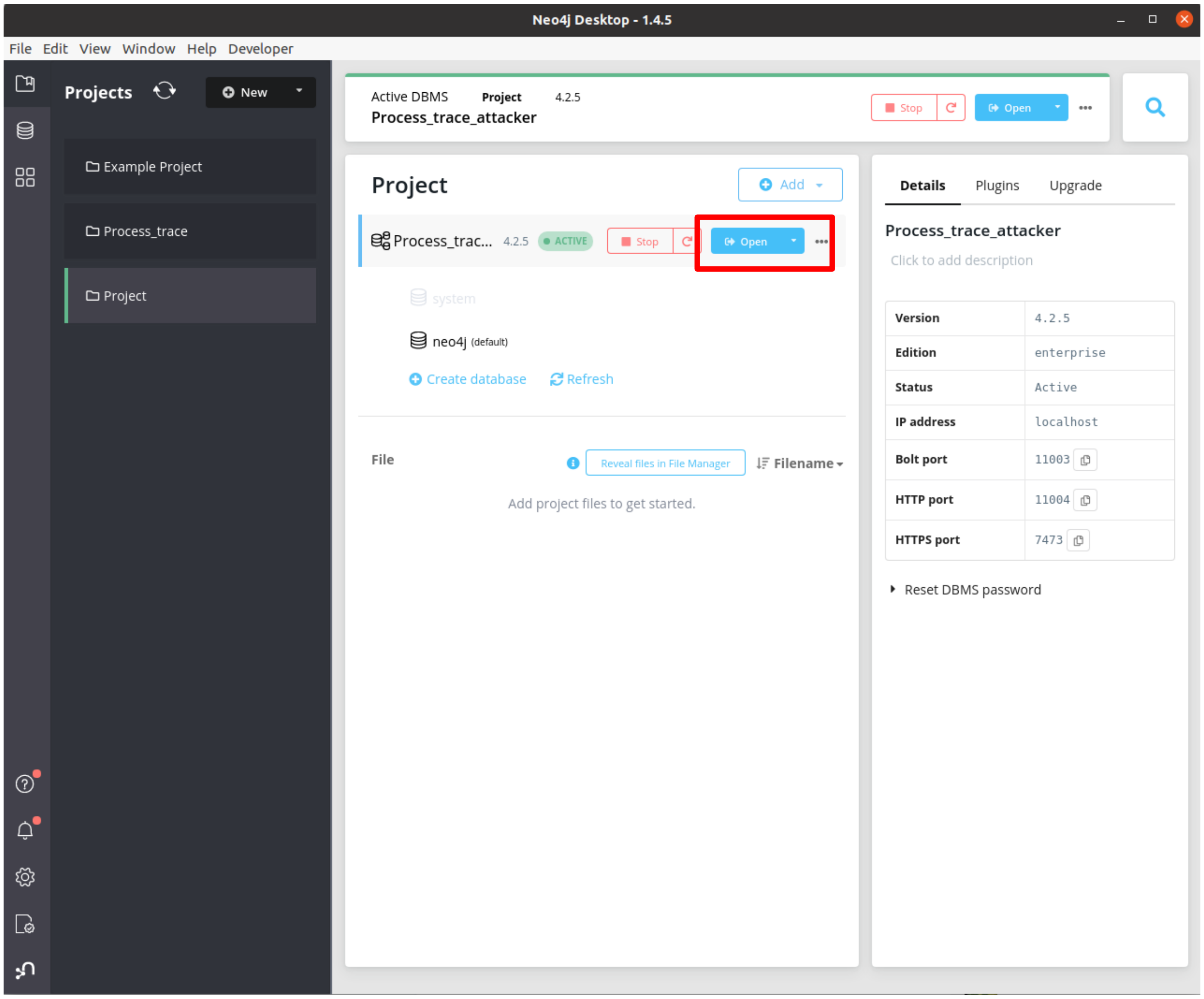
Conflicts occurred

DBMS **Process_trace_attacker** can not be started due to conflicts with external processes.

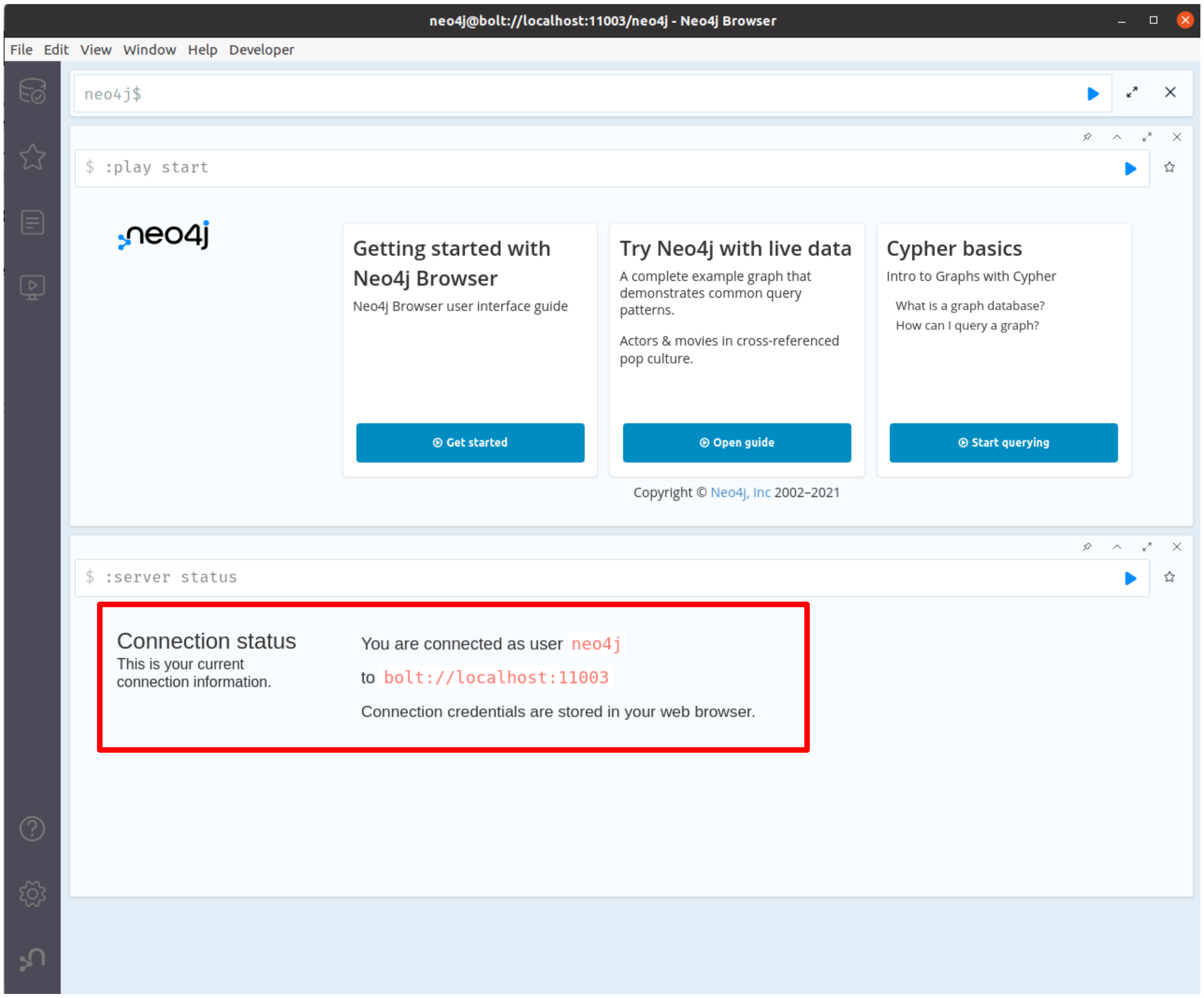
To fix this problem, let us change these port configurations:

- **bolt**: 7687 → 11003
- **http**: 7474 → 11004

6. Then click on the database name and press “open” button to make sure we can open the database. Once you click “open” you will be presented with another window where you can run Neo4j queries as well as information on how to connect to the database. Make sure to store connection details since this will be needed to configure both visualization tools.



Version	4.2.5
Edition	enterprise
Status	Active
IP address	localhost
Bolt port	11003
HTTP port	11004
HTTPS port	7473


neo4j@bolt://localhost:11003/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$
$:play start
neo4j
Getting started with Neo4j Browser
Neo4j Browser user interface guide
Get started
Try Neo4j with live data
A complete example graph that demonstrates common query patterns.
Actors & movies in cross-referenced pop culture.
Open guide
Cypher basics
Intro to Graphs with Cypher
What is a graph database?
How can I query a graph?
Start querying
Copyright © Neo4j, Inc 2002–2021
$:server status

Connection status
This is your current connection information.

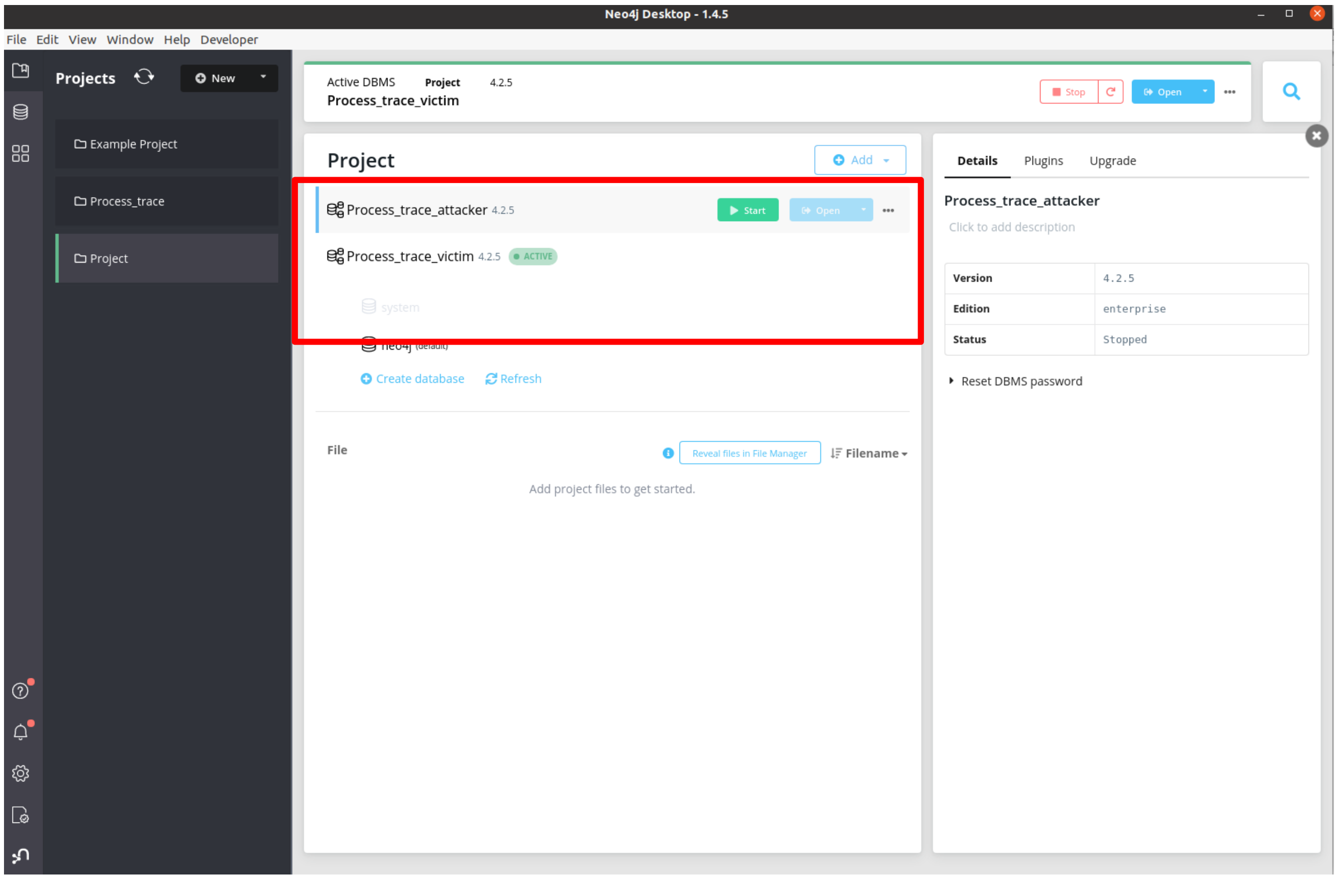
You are connected as user `neo4j`

to `bolt://localhost:11003`

Connection credentials are stored in your web browser.

7. Configuring Neo4j for the attacker-side Visualization tool is done!

8. To configure Neo4j for the victim-side. Repeat steps 3 – 7 mentioned above. Also, make sure to give victim-side database a distinct name. For the purpose of this guide I have used "Process_trace_victim" and same password (131kalana).



9. Now lets open the “Neo4j-Graphing-tool” directory.

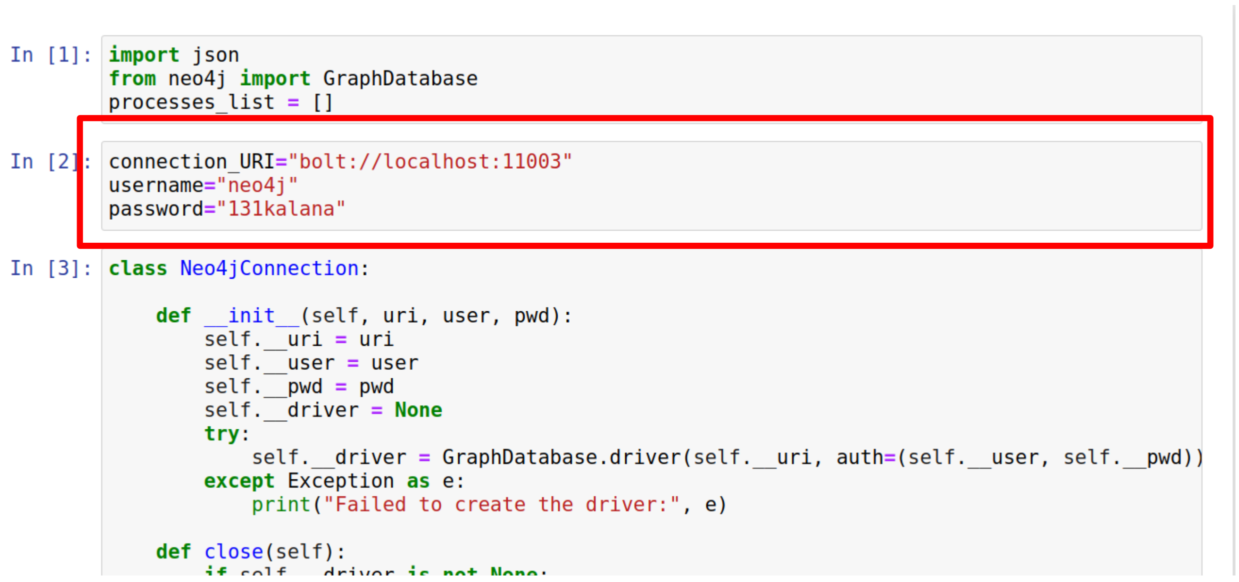
10. Open a new terminal and install the neo4j python driver. To install neo4j driver, type following command on the Linux terminal.
    - pip install neo4j

    - Note: I assume, python 3.8, pip and jupyter notebook is already installed on the system.

11. On another terminal launch the neo4j front-end and start the “Process_trace_attacker” database.

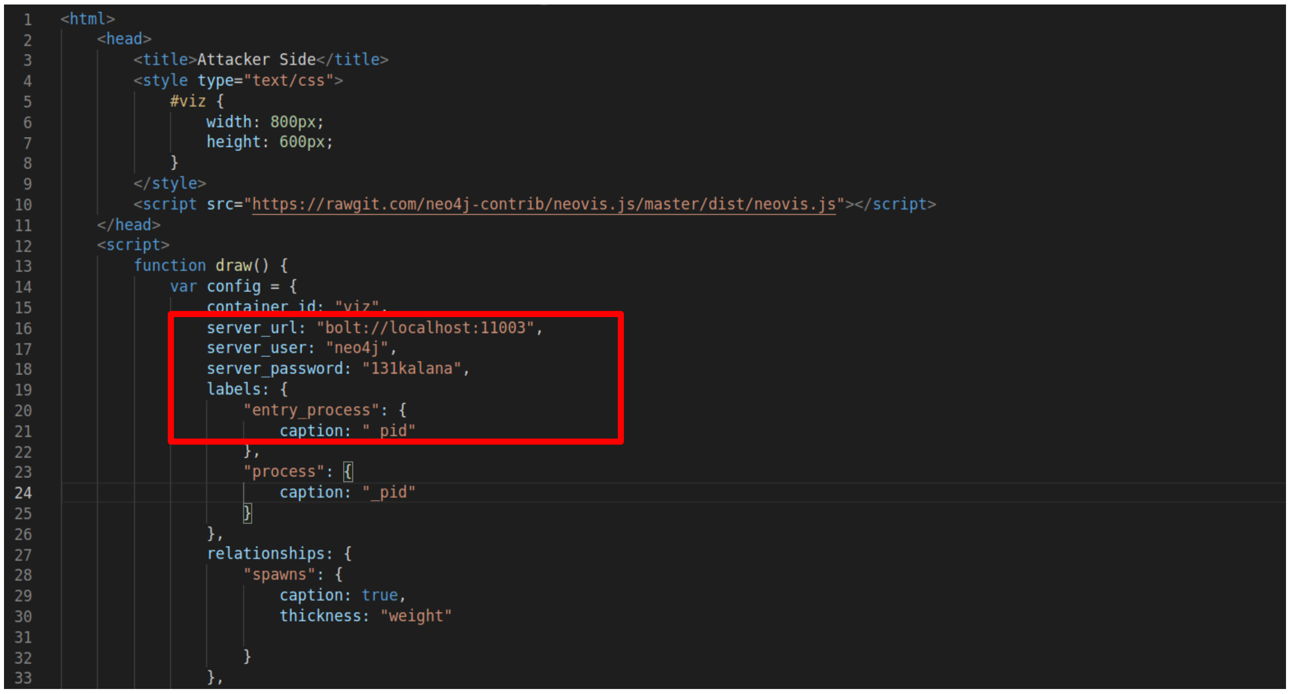
12. Now launch a jupyter notebook instance on the “Neo4j-Graphing-tool” directory and open “attacker side.ipynb” file.

13. On the second cell of the python file, replace the **connection_URI**, **username** and **password** variables with the values we have obtained when creating the attacker-side neo4j database (i.e. process_trace_attacker database in my case)



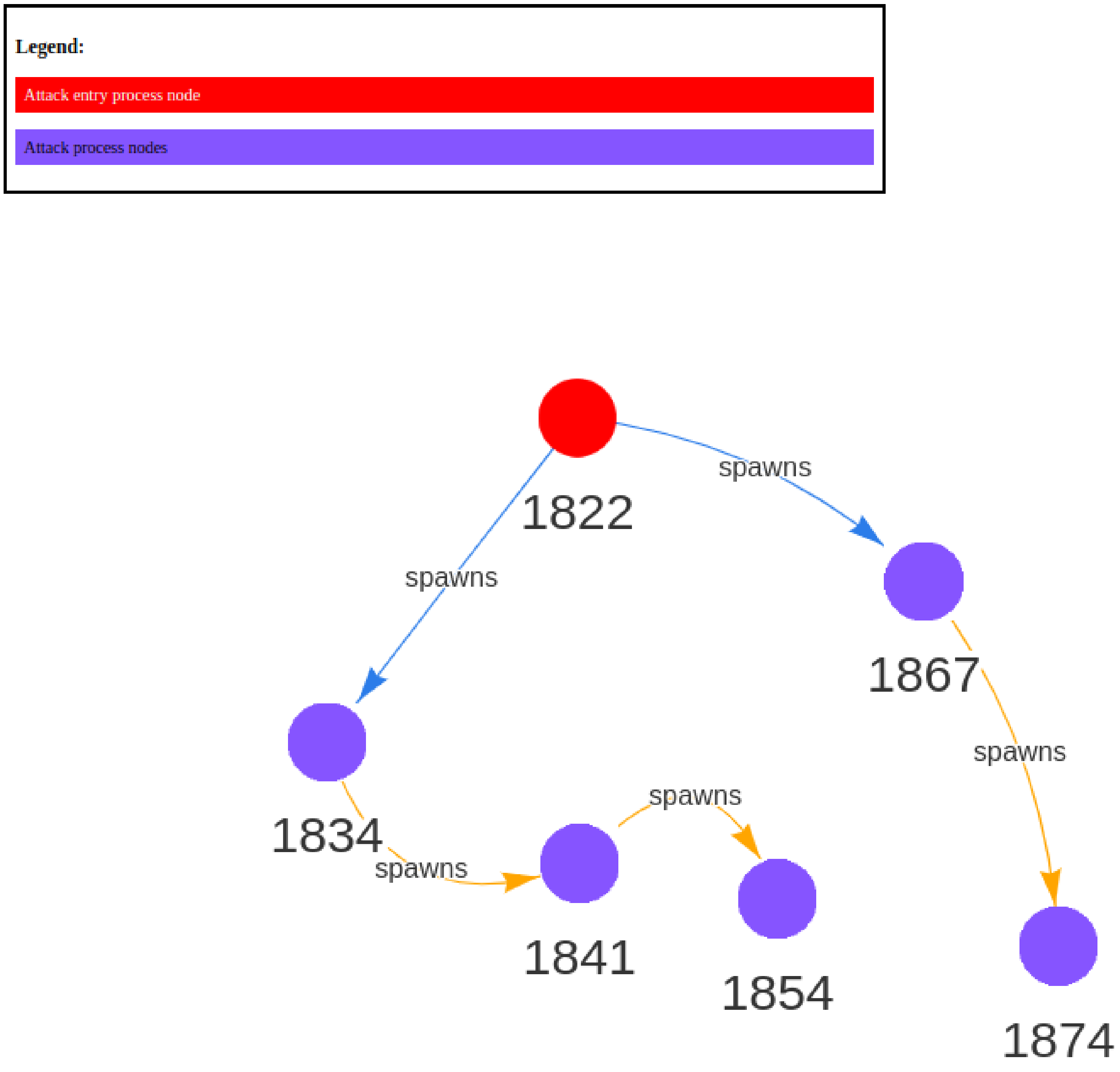
```
In [1]: import json
        from neo4j import GraphDatabase
        processes_list = []

In [2]: connection_URI="bolt://localhost:11003"
        username="neo4j"
        password="131kalana"

In [3]: class Neo4jConnection:

            def __init__(self, uri, user, pwd):
                self.__uri = uri
                self.__user = user
                self.__pwd = pwd
                self.__driver = None
                try:
                    self.__driver = GraphDatabase.driver(self.__uri, auth=(self.__user, self.__pwd))
                except Exception as e:
                    print("Failed to create the driver:", e)

            def close(self):
```

14. Also, in the “Neo4j-Graphing-tool” directory, open the “attacker_side.html” file with your favorite IDE and change the **server_URL**, **server_user** and **server_password** JavaScript variables inside the draw function to the values we obtained when creating the attacker-side neo4j database (i.e. process_trace_attacker database in my case)



```
<html>
    <head>
        <title>Attacker Side</title>
        <style type="text/css">
            #viz {
                width: 800px;
                height: 600px;
            }
        </style>
        <script src="https://rawgit.com/neo4j-contrib/neovis.js/master/dist/neovis.js"></script>
    </head>
    <script>
        function draw() {
            var config = {
                container_id: "viz",
                server_url: "bolt://localhost:11003",
                server_user: "neo4j",
                server_password: "131kalana",
                labels: {
                    "entry_process": {
                        caption: "_pid"
                    },
                    "process": {
                        caption: "_pid"
                    }
                },
                relationships: {
                    "spawns": {
                        caption: true,
                        thickness: "weight"

                    }
                },
```

15. Once configuring these parameter, you could run the whole notebook file to make sure everything works correctly by clicking kernel → Restart and Run all. (I have included logs files from previous experiments for both attacker and victim-side visualization tools in the “Neo4j-Graphing-tool”). Once execution is completed, you should see an graph similar to following in a new browser window:

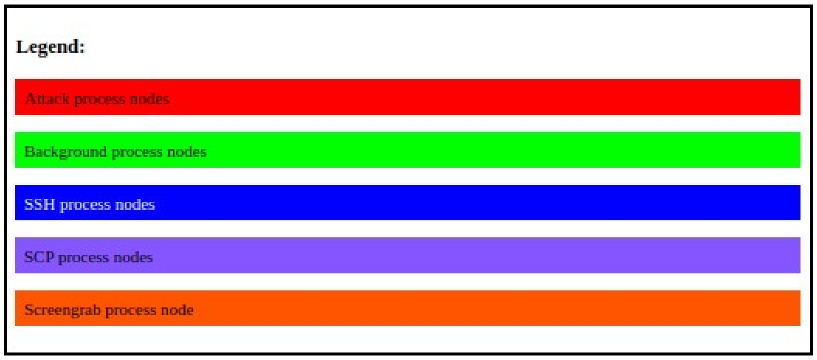


Legend:

🟥 Attack entry process node

🟪 Attack process nodes

16. To configure victim-side visualization tool, first stop the Neo4j database related to the attacker-side and start the victim-side database (i.e. Process_trace_victim database in my case). Then open “Victim side.ipynb” on Jupyter notebook and repeat steps 13 – 15 (During step 14, change the values in “victim_side.html” file). Once you run this file you will be able to see a graph similar to following in a new browser window:



Legend:

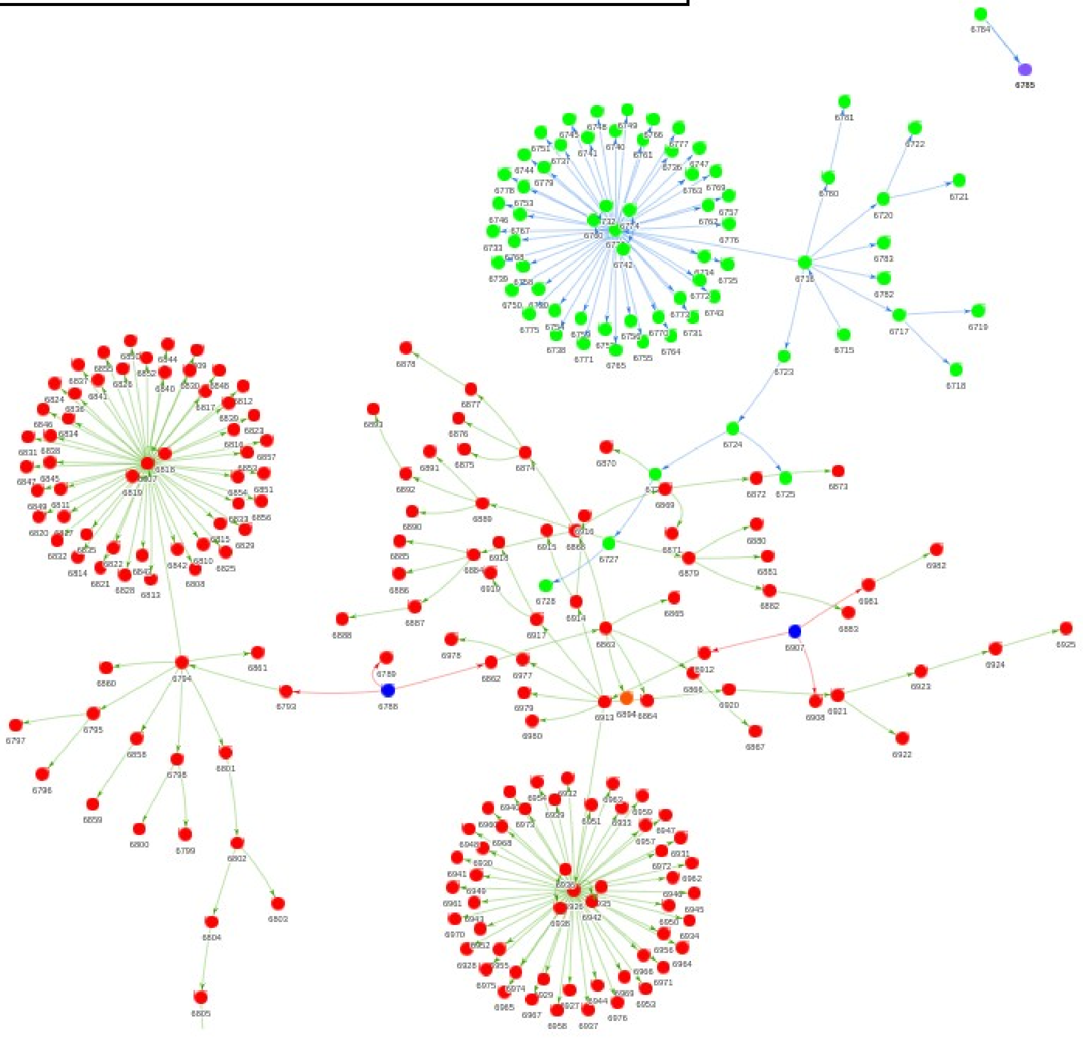
🟥 Attack process nodes

🟩 Background process nodes

🟦 SSH process nodes

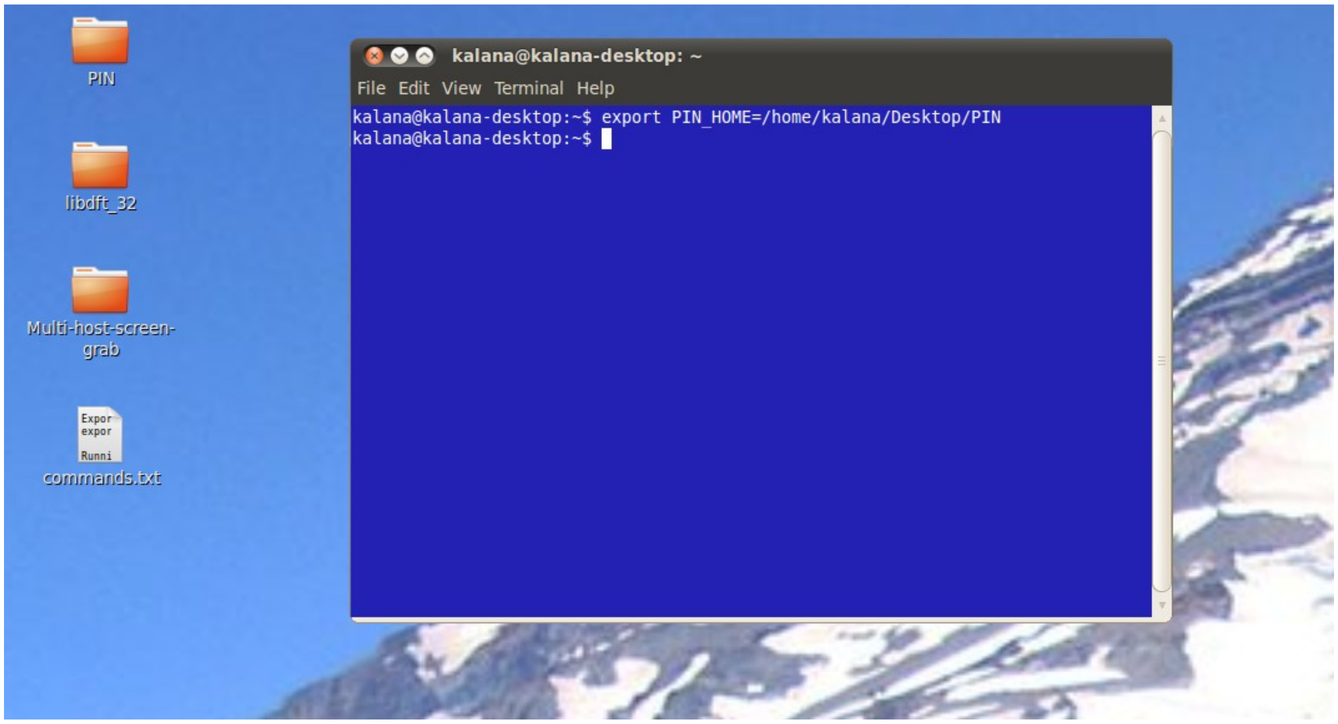
🟪 SCP process nodes

🟧 Screengrab process node



Analysis of Multi-host screengrab attack on the attacker side

1. First, fire up both virtual computers.
2. Then open a terminal in the attacker-side virtual computer and type the following command to export environmental variable to setup the Intel Pin executable location
   - export PIN_HOME=/home/kalana/Desktop/PIN
   - If you want to double check path is correctly setup, run "echo $PIN_HOME", it should print out "/home/kalana/Desktop/PIN".

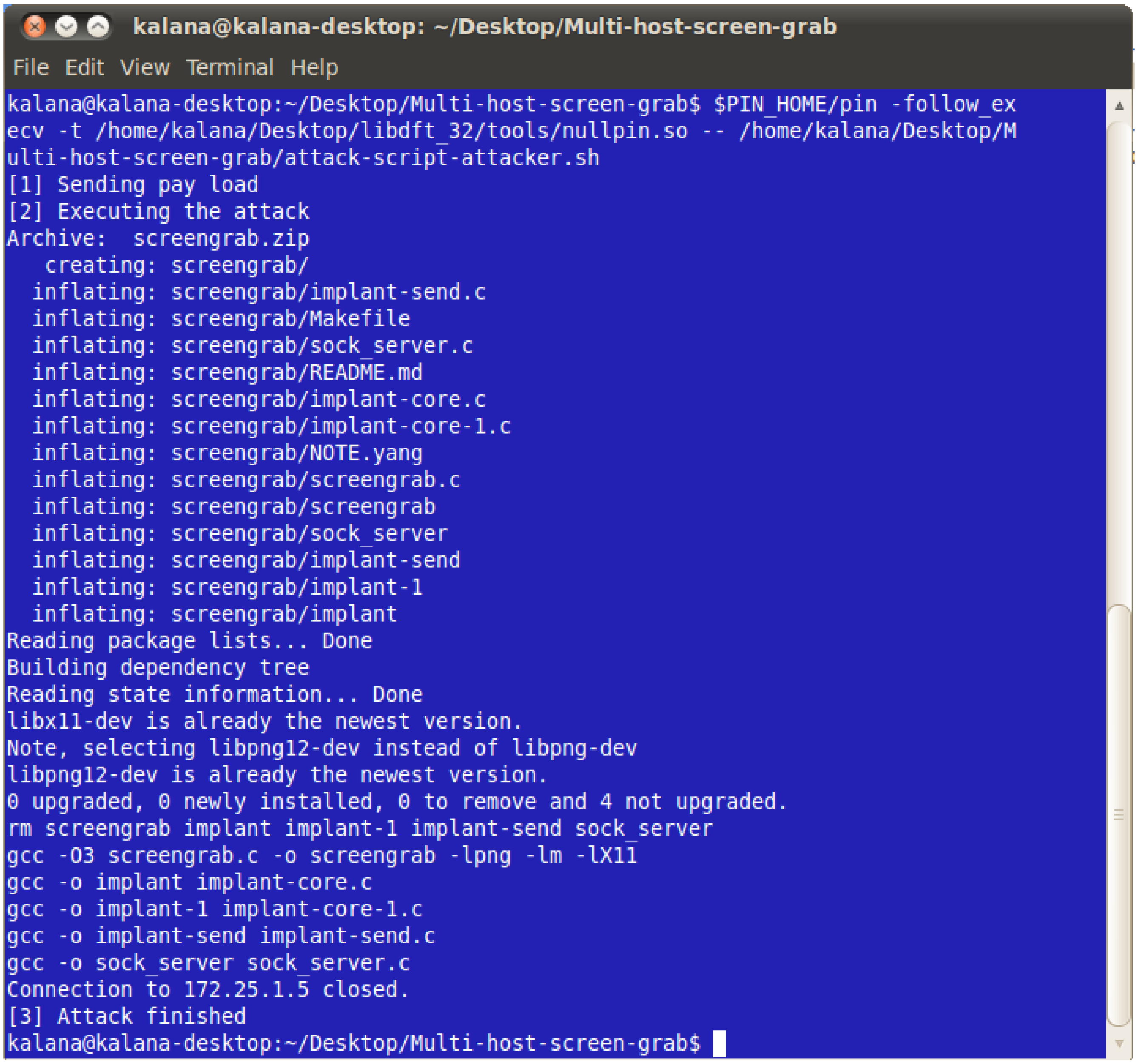


```
kalana@kalana-desktop:~$ export PIN_HOME=/home/kalana/Desktop/PIN
kalana@kalana-desktop:~$
```

3. To run the attack first change directory to the "Multi-host-screengrab" directory on the Desktop. To do so, type the following command on the terminal
   - cd Desktop/Multi-host-screengrab

4. Now lets run the attack by typing following command:
   - $PIN_HOME/pin -follow_execv -t /home/kalana/Desktop/libdft_32/tools/nullpin.so -- /home/kalana/Desktop/Multi-host-screengrab/attack-script-attacker.sh

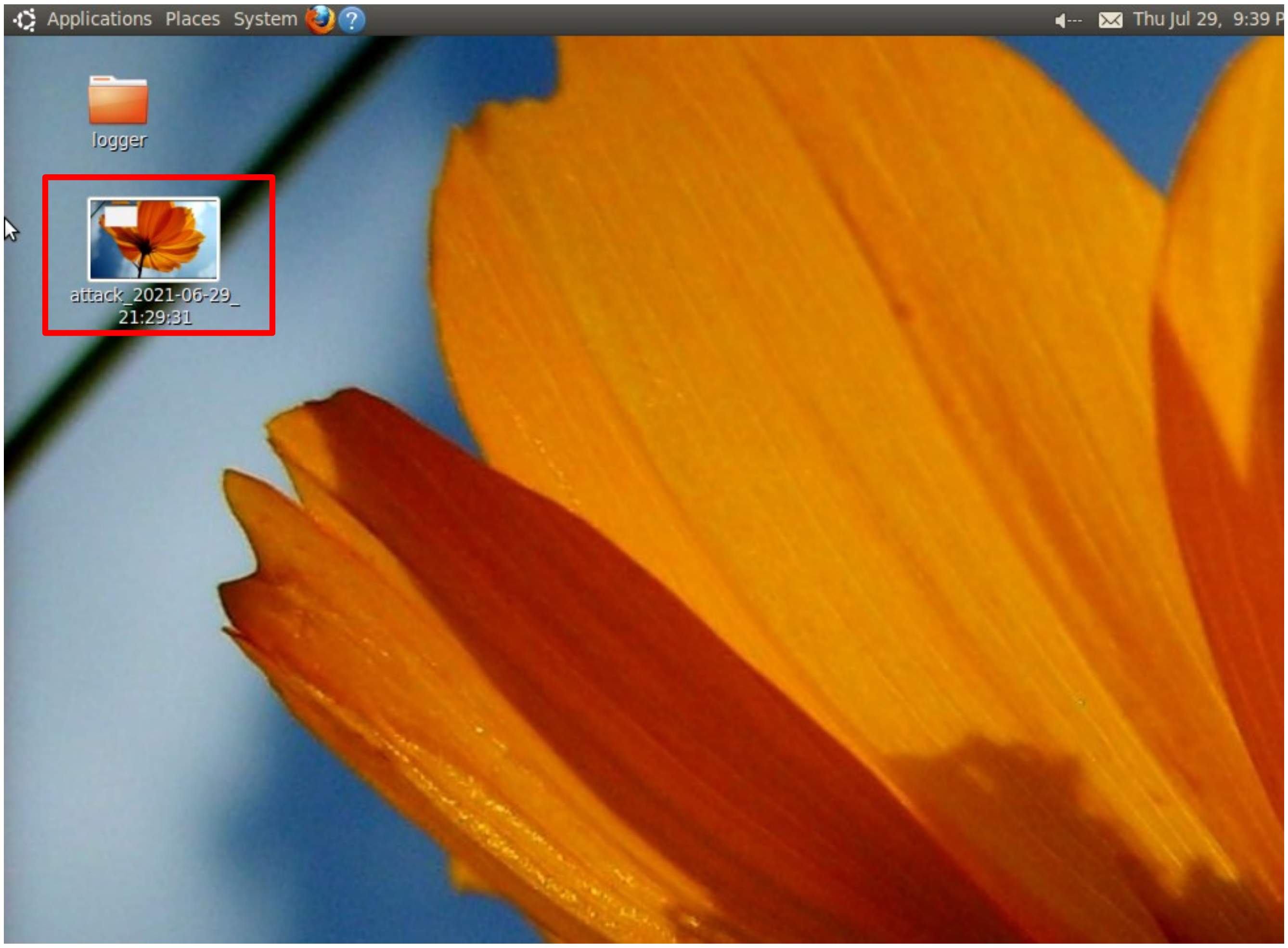
- **This command can be found on the “command.txt” file stored in the desktop**
- **In case, during the execution script prompts for password, password is 131kalana**
- **During the execution of the attack a similar output to the following will be displayed.**


kalana@kalana-desktop: ~/Desktop/Multi-host-screen-grab
File Edit View Terminal Help

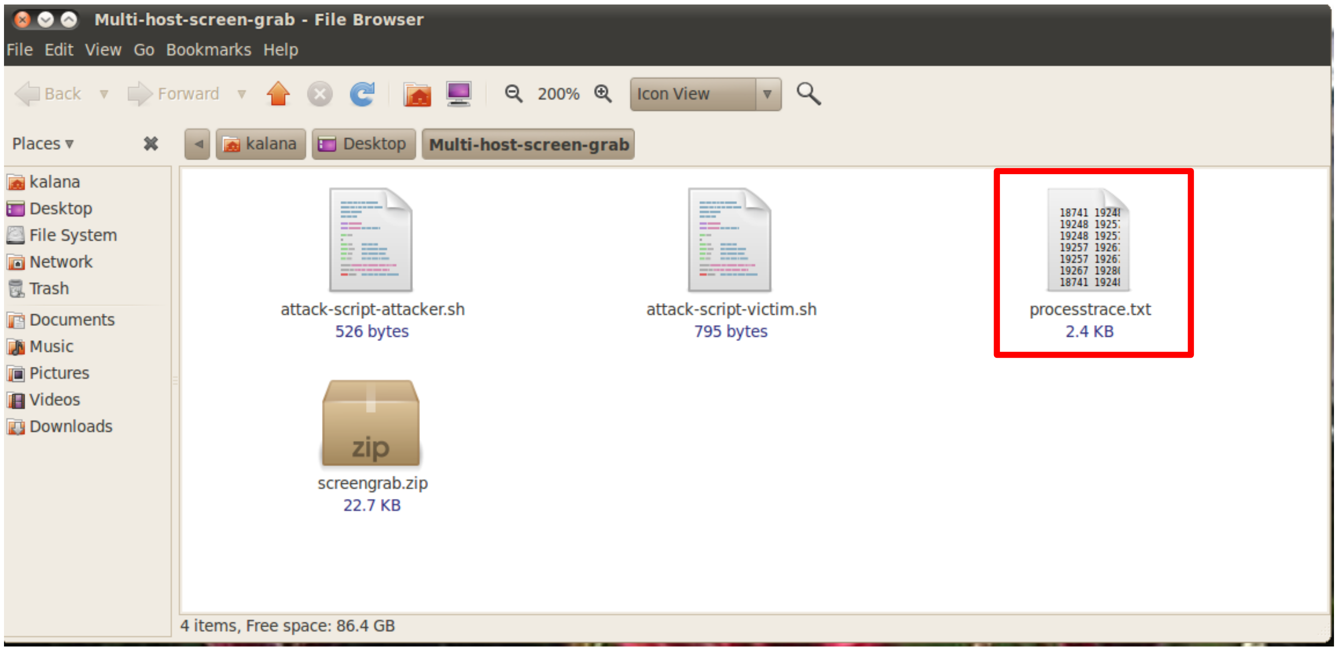
```
kalana@kalana-desktop:~/Desktop/Multi-host-screen-grab$ $PIN_HOME/pin -follow_ex
ecv -t /home/kalana/Desktop/libdft_32/tools/nullpin.so -- /home/kalana/Desktop/M
ulti-host-screen-grab/attack-script-attacker.sh
[1] Sending pay load
[2] Executing the attack
Archive:  screengrab.zip
   creating: screengrab/
  inflating: screengrab/implant-send.c
  inflating: screengrab/Makefile
  inflating: screengrab/sock_server.c
  inflating: screengrab/README.md
  inflating: screengrab/implant-core.c
  inflating: screengrab/implant-core-1.c
  inflating: screengrab/NOTE.yang
  inflating: screengrab/screengrab.c
  inflating: screengrab/screengrab
  inflating: screengrab/sock_server
  inflating: screengrab/implant-send
  inflating: screengrab/implant-1
  inflating: screengrab/implant
Reading package lists... Done
Building dependency tree
Reading state information... Done
libx11-dev is already the newest version.
Note, selecting libpng12-dev instead of libpng-dev
libpng12-dev is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
rm screengrab implant implant-1 implant-send sock_server
gcc -O3 screengrab.c -o screengrab -lpng -lm -lX11
gcc -o implant implant-core.c
gcc -o implant-1 implant-core-1.c
gcc -o implant-send implant-send.c
gcc -o sock_server sock_server.c
Connection to 172.25.1.5 closed.
[3] Attack finished
kalana@kalana-desktop:~/Desktop/Multi-host-screen-grab$
```

5. Upon completion of the attack. On the victim side Desktop you will be able to see a screenshot taken by the attack. On the attacker-side, in “Multi-host-screengrab”

directory you will be able to see the log file (“processtrace.txt”) produced by the analysis tool.



Victim-side



Attacker-side

6. Now lets transfer processtrace.txt to our host machine using your preferred method (i.e. computer that we installed VirtualBox on). When transferring the file, make sure to transfer the whole file (not the copying and pasting the content. This is just to be safe). You can copy this file to a USB stick or upload it to google drive and download it back from the host computer (emailing this file works too)

7. Then place the downloaded log file (i.e. “processtrace.txt” file) in to the “Neo4j-Graphing-tool” directory.

8. Open a terminal window and launch Neo4j front-end. Then start the attacker-side database (I.e Process_trace_attacker database in my case).

9. Launch jupyter notebook instance and run the “attacker side.ipynb” file (clicking kernel → Restart and Run all)

Note (optional): if you want to run this attack again. On the victim side, open a terminal and run the following commands:

- **cd /tmp**
- **rm -r screengrab**
- **rm screengrab.zip**

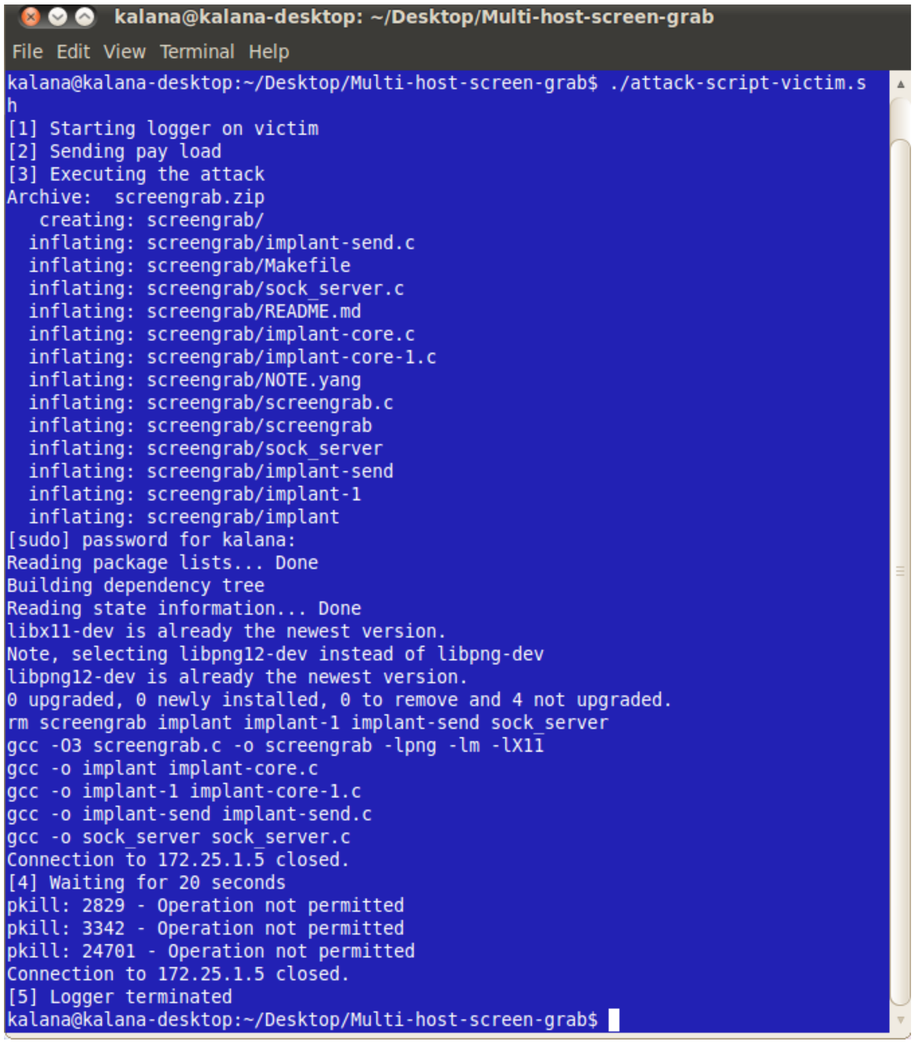
This is to remove the payload injected by the attacker on the victim computer. It is not necessary to remove the payload. But it will avoid attacker script asking to replace the files from the first attack when re running the attack.

Analysis of Multi-host screengrab attack on the victim side

1. First, fire up both virtual computers
2. Then, to run the attack first change directory to the "Multi-host-screengrab" directory on the Desktop. To do so, type the following command on the terminal
   - cd Desktop/Multi-host-screengrab

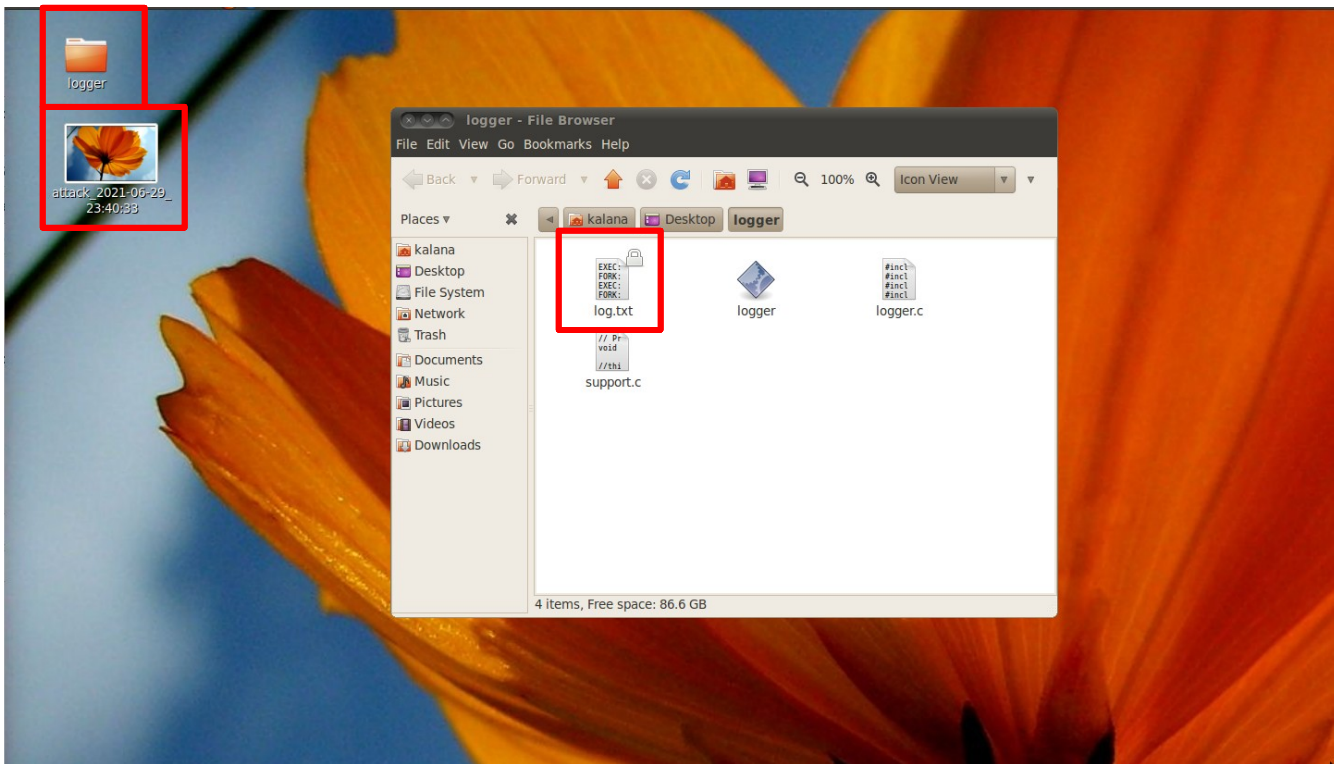
3. Now lets run the attack by typing following command:
   - ./attack-script-victim.sh

   - **This command can also be found on the "command.txt" file stored in the desktop**
   - **In case, during the execution script prompts for password, password is 131kalana**
   - During the execution of the attack a similar output to the following will be displayed.


kalana@kalana-desktop: ~/Desktop/Multi-host-screen-grab
File Edit View Terminal Help

```
kalana@kalana-desktop:~/Desktop/Multi-host-screen-grab$ ./attack-script-victim.s
h
[1] Starting logger on victim
[2] Sending pay load
[3] Executing the attack
Archive:  screengrab.zip
   creating: screengrab/
  inflating: screengrab/implant-send.c
  inflating: screengrab/Makefile
  inflating: screengrab/sock_server.c
  inflating: screengrab/README.md
  inflating: screengrab/implant-core.c
  inflating: screengrab/implant-core-1.c
  inflating: screengrab/NOTE.yang
  inflating: screengrab/screengrab.c
  inflating: screengrab/screengrab
  inflating: screengrab/sock_server
  inflating: screengrab/implant-send
  inflating: screengrab/implant-1
  inflating: screengrab/implant
[sudo] password for kalana:
Reading package lists... Done
Building dependency tree
Reading state information... Done
libx11-dev is already the newest version.
Note, selecting libpng12-dev instead of libpng-dev
libpng12-dev is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
rm screengrab implant implant-1 implant-send sock_server
gcc -O3 screengrab.c -o screengrab -lpng -lm -lX11
gcc -o implant implant-core.c
gcc -o implant-1 implant-core-1.c
gcc -o implant-send implant-send.c
gcc -o sock_server sock_server.c
Connection to 172.25.1.5 closed.
[4] Waiting for 20 seconds
pkill: 2829 - Operation not permitted
pkill: 3342 - Operation not permitted
pkill: 24701 - Operation not permitted
Connection to 172.25.1.5 closed.
[5] Logger terminated
kalana@kalana-desktop:~/Desktop/Multi-host-screen-grab$
```

4. Upon completion of the attack, on the victim side virtual computer log file (named “log.txt”) can be found under directory name “logger” that resides on the desktop. Moreover, a screenshot will be created in the victims desktop



5. Similarly to the attacker-side analysis scenario, transfer this log file to the host computer and store it in the “Neo4j-Graphing-tool” directory.

6. Open a terminal window and launch Neo4j front-end. Then start the attacker-side database (I.e Process_trace_victim database in my case).

7. Launch jupyter notebook instance and run the “victim side.ipynb” file (clicking kernel → Restart and Run all)

Note (optional): if you want to run this attack again. On the victim side, open a terminal and run the following commands:

- **cd /tmp**
- **rm -r screengrab**
- **rm screengrab.zip**

This is to remove the payload injected by the attacker on the victim computer. It is not necessary to remove the payload. But it will avoid attacker script asking to replace the files from the first attack when re running the attack.