

# MSTICPy and Jupyter Notebooks for Repeatable Threat Hunting

---

 [topomojo.ini.cmu.edu/mojo/17a5aef13e7f427c8c8a4df4700a463f/msticpy-and-jupyter-notebooks-for-repeatable-threat-hunting](https://topomojo.ini.cmu.edu/mojo/17a5aef13e7f427c8c8a4df4700a463f/msticpy-and-jupyter-notebooks-for-repeatable-threat-hunting)

 disconnected

## Introduction

---

Threat hunting refers to the process of proactively searching for threats that cannot otherwise be located with passive threat detection methods, e.g., firewalls and Intrusion Detection Systems (IDS). As modern-day attackers engage in devising more novel, complex, and covert exploitations that can evade traditional cyber defense systems, it becomes increasingly imperative for cybersecurity professionals to incorporate routine threat hunting as part of their defense strategies. However, there are circumstances where actions taken by professionals alone turn out insufficient, that even engineers without strong security or coding backgrounds should preferably participate in the threat hunting process; that is where Jupyter Notebook and MSTICPy come in handy.

Jupyter Notebook is a web-based computational environment that supports plain-text, markdown, and programming languages, which enables users to combine them to create a interactive computing experience. It has become one of the most popular data analysis tools since its invention due to its exceptional repeatability, maintainability, and user-friendliness, and the vast range of libraries it supports. The MSTICPy library that we are interested in is a Python-based, infosec library created by Microsoft, addressing three central needs for threat hunting: data acquisition and enrichment, analysis, and visualization.

In this lab, we will be using Jupyter Notebook and the MSTICPy library to carry out threat hunting. Students will familiarize themselves with employing Jupyter and MSTICPy to conduct threat detection, analysis, and visualization on local data as well as real-time log data generated by Splunk when the victim machine had experienced suspicious security events.

## Learning Objectives

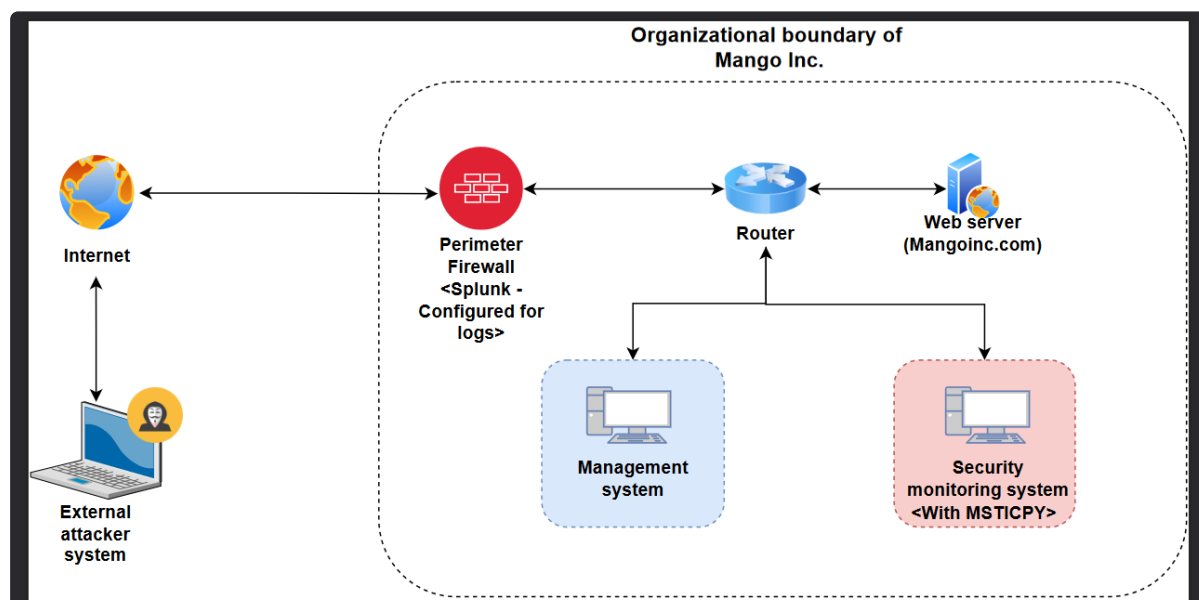
---

By the end of the lab, students will be able to:

1. Import local log data to Jupyter Notebook and conduct threat analysis and visualization with MSTICPy
2. Query real-time log data from network traffic monitoring tools, e.g., Splunk to Jupyter Notebook and utilize MSTICPy to conduct threat hunting
3. Explore and implement other tools from MSTICPy

## Network Diagram

---



IP address: Pfsense - WAN - 12.0.0.34 LAN - 10.0.40.100

**Start all the VMs before you start to work on the lab.**

## Credentials

1. sec-monitor - username: student password: tartans
2. kali-attacker - username: student password: tartans

Some of credentials are intentionally hidden as they will serve as the targets of the attack simulation in Section Two. Students will gain full access to them as they work through this lab.

## Lab

### Section One: Learn about MSTICPy

In section one, we will get familiar with MSTICPy and Jupyter Notebooks by completing an introduction lab on the sec-monitor machine. The lab will show us some core features of the MSTICPy package and also let us have a better idea of why notebooks is an ideal tool for doing threat hunting activities. The main features of MSTICPy introduced are described here.

1. Querying and Importing Data
2. Enriching Data
3. Analyzing Data (Base64 Decoding and Unpacking, IoC Extraction)
4. Displaying/Visualizing Data (FoliumMap plotting, Entity Graph, Process Tree, Event Timeline)

### Access the sec-monitor machine

Login into the sec-monitor machine with its credentials (student,tartans).

### Access the grading script

The python based grading scrips are in the desktop of the Kali-attacker machine. Run python gs1.py and complete the quiz as you proceed with section 1.

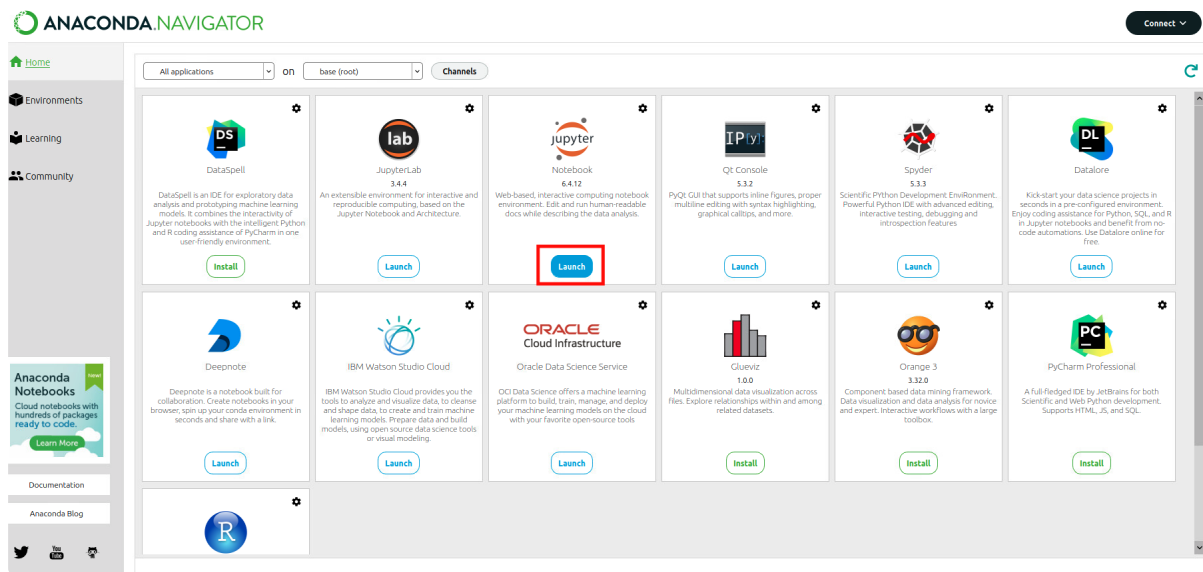
## Launch Jupyter Notebooks

Open a terminal by right-clicking on the desktop and choosing Open in Terminal.

Use the following command to open the conda navigator:

```
anaconda-navigator
```

In the anaconda navigator open Jupyter Notebooks by clicking on the Launch button of the Notebook item.



A notebook instance will be lauched in the firefox browser. To open the Lab, click on the file MSTICPy\_Lab.ipynb.



We can see that the jupyter notebook file has been successfully loaded. To run a cell in the notebook file, simply click on the Run button on the menu at the top after selecting the cell by clicking on it.

Now follow the instructions in the notebook file to complete the lab. Run the code cells one by one in sequential order throughout the lab. **Note that do not modify any code or text cells while working through the lab, except for the #TODO sections in the exercises.**

## Section Two: Attack Simulation on Mango Inc.

In Section Two, you will put on the attacker hat and perform the attack against Mango Inc, a start-up company that surprisingly has valuable customer information stored on their management machine. The scenario and the attack is briefly described here:

1. Goal - Retrieve the customer information file from the Management System.

2. Defense - The Management System is protected by firewall (pfSense) and not accessible from external network.
3. Vulnerabilities -
  1. The web framework (Web2py Ver. 2.14.5) utilized by the corporation to build their web application has some inherent vulnerabilities (brute force and reverse file extraction vulnerabilities).
  2. Mango Inc. has not conducted sufficient information security training among its employees; some store critical system credential memos in .txt documents on their web server machine.

### Access and Configure the attacker machine

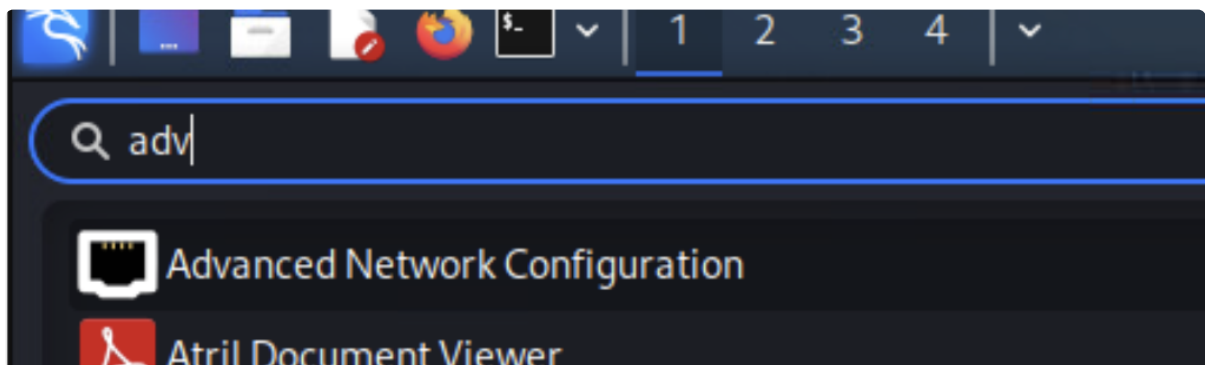
---

Login into the Kali machine with its credentials (student,tartans).

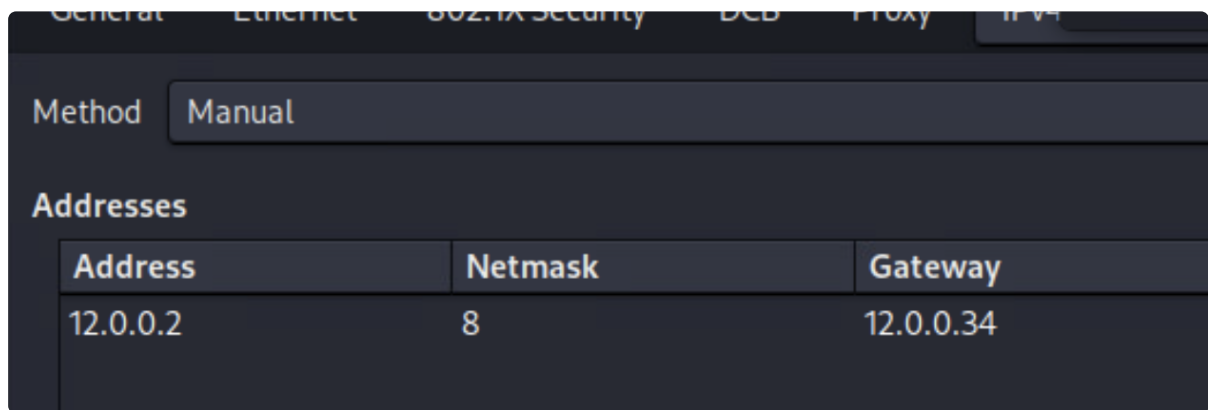
You will also need to configure the network settings of Kali. First look up **Advanced Network Configuration in the computer**:



Then change the network settings such that it is identical to the screenshot below:



Finally, disconnect and reconnect the network to apply the configuration:

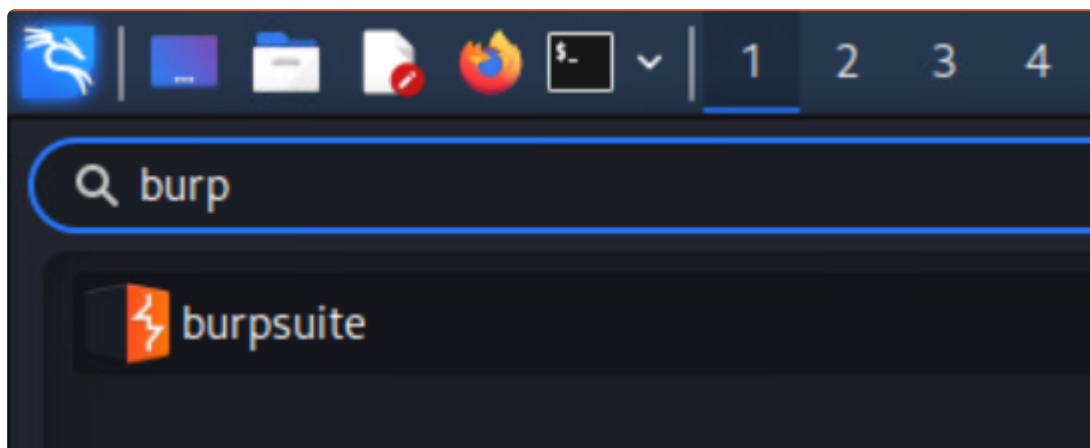


### Access the grading script

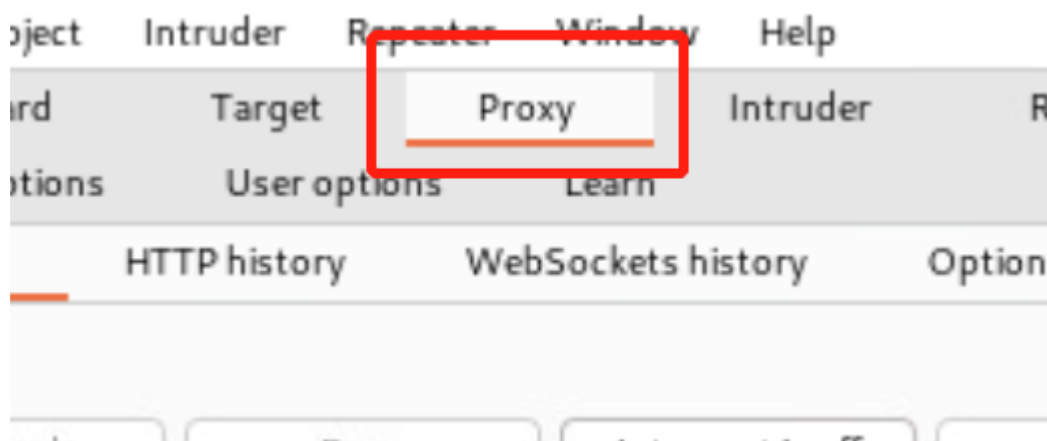
The python based grading scripts are on the desktop of the Kali-attacker machine. Run `python gs2.py` and complete the quiz as you proceed in Section Two.

### Exploit the web server

The first step that you, the attacker, perform is to exploit the brute force vulnerability of the webserver application. We are going to utilize Burp Suite for the first few steps of the attack. Open Burp Suite from the application panel of the machine.



We are going to utilize Burp Suite's inbuilt browser to perform the attack. Inside Burp Suite, create a new temporary project with default settings, and then navigate to the **Proxy** tab on the top of the screen.



Click the orange **Open browser** button on the same page to get a Chrome-like browser, namely, Chromium, preconfigured by Burp Proxy.

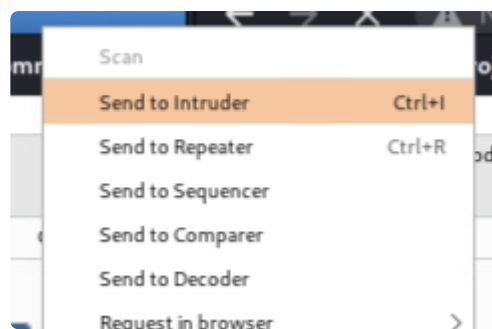
Navigate to the web server on the browser (NAT has been set up to firewall's external IP (12.0.0.34)). In the webpage, we can see an link to the admin panel. Open the admin panel. You can also access the admin panel by directly opening the url: `http://12.0.0.34/admin/default/index`.

Now, as the attacker, we can try to brute force the admin password. Burp Suite helps us do this with its inbuilt intruder mode. Toggle on the **Intercept** button:



Enter any random password and try to login, and the outgoing HTTP request will be intercepted and come up in Burp Suite. **Do not forward the request before performing the below steps.**

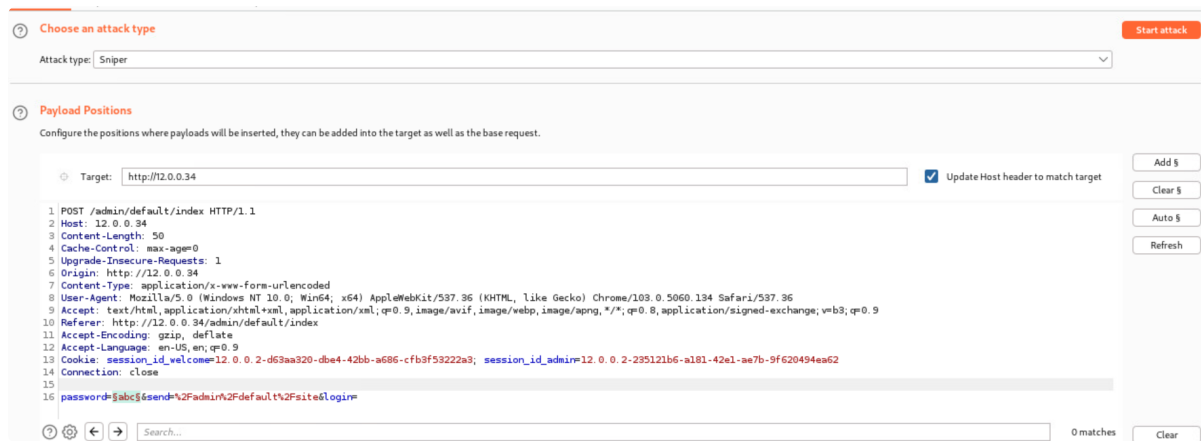
Click on the **Action** button and forward the intercepted HTTP request to **Intruder**.



We can see that the HTTP request we sent has been intercepted, and the password that we entered is included in the request. Click on actions tab on the Burp window and select send to intruder. Click on intruder on the top window.

In the intruder window, we can see the same request is displayed. Select the attack type as sniper.

The \$'s and highlighted fields in blue denote the places that burpsuite will be brute forcing. Burpsuite will by default identify multiple places to bruteforce in the request, we just need it to bruteforce the password. Ensure that the \$'s are removed in the other places except the password as per the screenshot shown below.



Move to the next tab on the top "Payloads". Set the payload type "Brute Forcer". Generally, the minimum length and maximum length of the passwords are set over a larger range; but for this exercise, we will set both to the value 3. Change the character set to 0-5 to make it faster. Start the Brute Forcer by click on start attack on the top right corner. *Click ok on the prompt to continue.*

Do not let the Brute Forcer run completely, as it is not only time consuming, but the server might also crash, since Mango Inc. is a small-scale company that operate their service on rather cheap servers. Keep checking the status on the window, and the whole process should take approximately 2 minutes to complete. The first entry with the **status 303** is the password that we are looking for. Pause the attack by clicking on attack -> pause on the top. The payload column will give you the password.

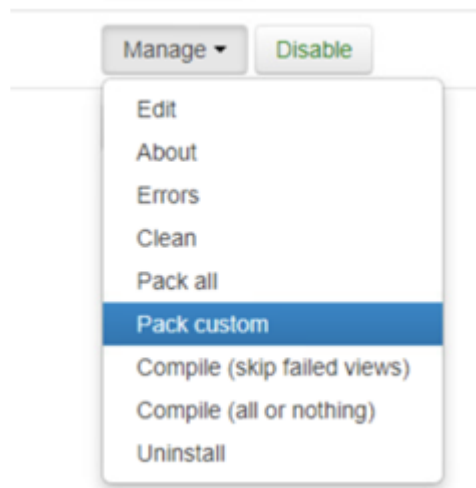
*Now you should be able to answer Question 1 of the grading quiz.*

Now you are able to login to the admin portal using the password you found. Open the Burp browser again and go to the admin login page, and you will be prompted to enter the password. When you have logged in, you will see a list of application we installed.

## Installed applications

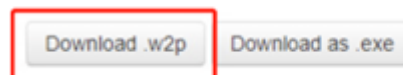
	admin (currently running)	Manage ▾	
	examples	Manage ▾	Disable
	welcome	Manage ▾	Disable

With admin privileges, Web2py allows us to download the web applications to make modifications. You can click on one of the 'Manage' buttons of your choice, and then choose 'Pack custom' in the drop menu.

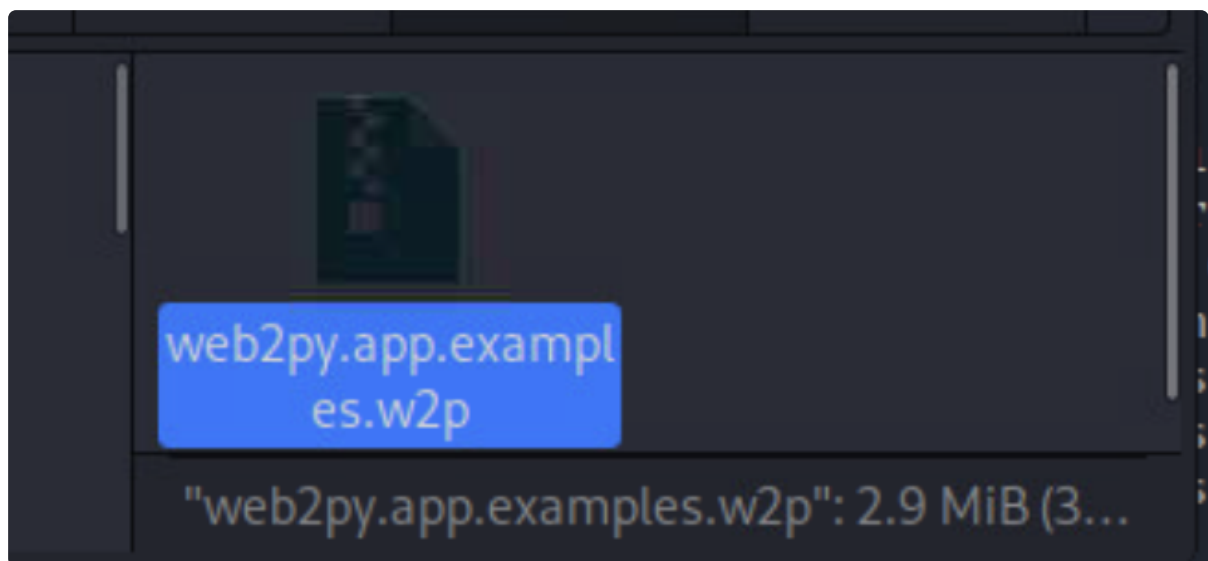


Try to click on 'Download .w2p' in the next page.

## Select Files to Package



The normal downloading process will start, and you can see in the File Manager that the actual .w2p source document of our 'examples' project has a size of 2.9 MB.



The 2.14.5 version of Web2py has a Reflected File Download Vulnerability, which enables the attacker to gain access to any file on the victim's machine upon which the web application is served. Similar to how we intercepted the outgoing HTTP request to do the brute forcing, we can tamper with the HTTP payload to acquire a secret file stored on the victim's Documents folder.

As you have familiarized with how to intercept outgoing HTTP requests, try to intercept the download request and you will be seeing an HTTP request as shown in the picture below. The 'file' field is what we are interested in.



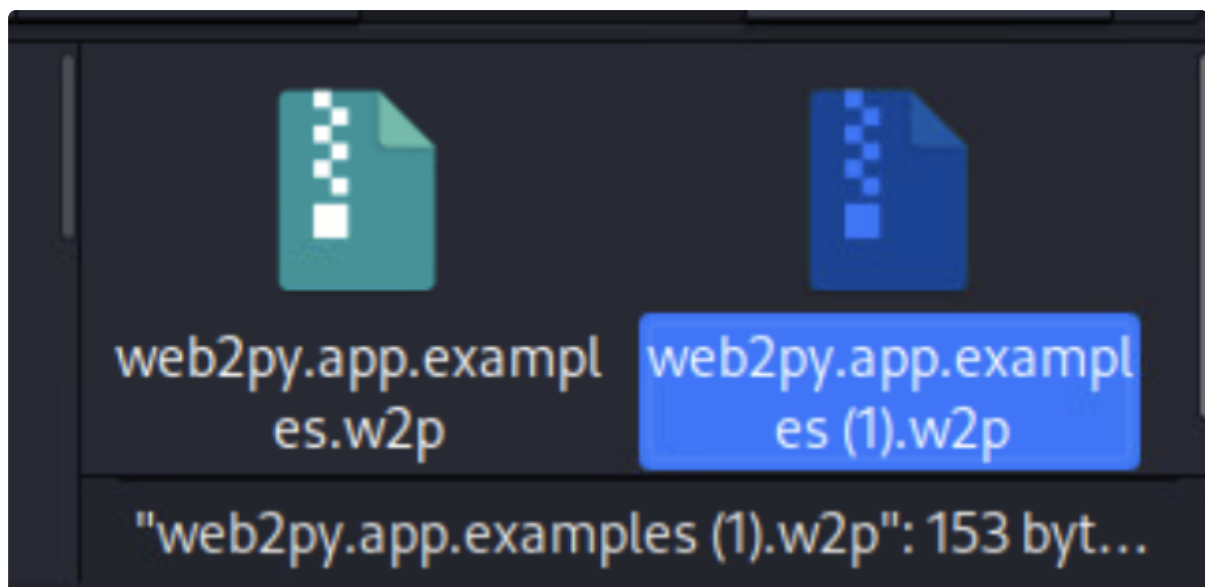
```

POST /admin/default/pack_custom/examples HTTP/1.1
Host: 127.0.0.1
Content-Length: 4687
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,i
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/admin/default/pack_custom/examples
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: session_id_welcome=127.0.0.1-6652fc4e-7d73-405f-a504-8c
Connection: close

file=ABOUT&file=LICENSE&file=__init__.py&file=controllers%5Caja
controllers%5Cdefault.py&file=controllers%5Cform_examples.py&fi
controllers%5Csession_examples.wv&file=controllers%5Csimple_exp

```

Change the value of the 'file' variable to '/Users/Student/Documents/secret1.txt', and then turn off the interception. The tampered downloading request will start automatically, and you will see that the content of it has been changed by examining the size of the file.



The .w2p file we have acquired from the victim's machine is a type of compressed file intended to be loaded by Web2py with admin privilege, which is, in essence, a renamed .tgz file that can be decompressed with standard zip/unzip softwares. It was designed this way to prevent the browser from automatically unzipping it. For this lab, we have written a python script to make it more convenient for you.

Rename the file we just downloaded to 'secret1.tgz', and then you should be able to unzip it and see the secret .txt file that contains credentials for the web server.

Now that we have the credentials to SSH into the web server. In the terminal, run

```
ssh student@12.0.0.34.
```

Enter the password you found in the file when prompted for password. There is another interesting file on the web server machine. Find this file. You can use the command `type filename` to display the contents of the file. *Remember that the server is a windows machine so the commands used should be windows cmd commands* (Hint: The locations where this file maybe located are Desktop, Downloads and Documents.)

*Now you should be able to answer the second and third questions of the quiz.*

Connect using FTP to the MGMT machine.

```
ftp 10.0.40.1
```

There is another interesting file on the system. Again, the locations to search for are *Desktop, Downloads and Documents*. Download this file to the web server machine using the get command.

Finally, get the file from the web server to the attacker's system using the file download vulnerability in the website which we already tried previously; or you can use `type` to just read the file.

*Now you should be able to answer questions 4 and 5 of the quiz.*

### Section Three: Analyze the Attack on Mango Inc. with MSTICPy

---

As the attack has been completed, now you will again play the role of the victimized Mango Inc. employee who has been instructed to conduct routine threat hunting with Jupyter and MSTICPy by the information security technician of your company.

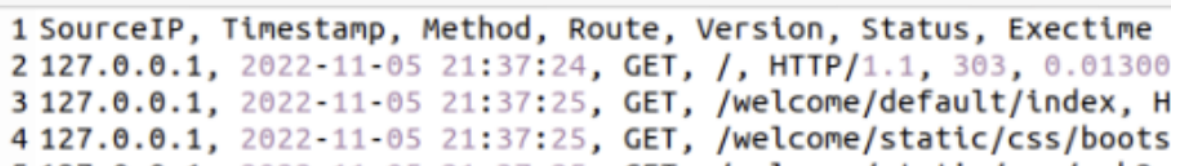
Navigate to the `web2py` folder on the Desktop of the *Web Server Machine*, scroll down and you will find the `httpserver.log` file, the default log file generated by the web2py server. We will combine this with the traffic log of Splunk for more comprehensive analysis. Copy it to the `Documents` directory for easier access.

Now you can move onto the *Security Monitor Machine*. Open a terminal, establish a ftp connection with *Web Server Machine* by entering:

```
ftp 10.0.40.2
```

Enter the password you acquired in Section Two when prompted. Use the `get` command to download the `httpserver.log` file we just looked at.

For it to be uploaded to Splunk, you will need to modify it to a comma-separated `.csv` file. First, open it with Text Editor, and enter the corresponding heading for each column, such that it looks identical to this screenshot below:



```
1 SourceIP, Timestamp, Method, Route, Version, Status, Exectime
2 127.0.0.1, 2022-11-05 21:37:24, GET, /, HTTP/1.1, 303, 0.01300
3 127.0.0.1, 2022-11-05 21:37:25, GET, /welcome/default/index, H
4 127.0.0.1, 2022-11-05 21:37:25, GET, /welcome/static/css/boots
```

You can then upload this file to Splunk to make it automatically combined with the log data of Splunk. Open a browser and navigate to <http://localhost:8000>, choose **Add Data**, and then **Upload (files from my computer)**. Set the Host field value to "web\_server" when prompted, with.

Once this is done, you can launch the Jupyter Notebook, but open **Splunk.ipynb** this time. Follow the instructions in the notebook to complete the last part of lab like you did in Section One.

## Conclusion

---

Upon the completion of the three sections of this lab, you should now be familiar with Jupyter Notebook and MSTICPy and can utilize them to conduct basic repeatable threat hunting.

Note that we disabled network connection when configuring the MSTICPy library for this lab's purpose, which serves as a tutorial and demonstration of this library's functionalities. MSTICPy includes a larger collection of analysis and visualization tools that are powerful and convenient to use, which should become more accessible for you to implement after you completed this lab.

## Appendix: Grading Scripts

---

### Grading Script: Section One

---

```

import hashlib

marks=0

ans1 =
"c2b37e4037631aaa4809e9a0dc82ad5ce7a04fa98a6b6de280d16181dc88de0b3e337a96a7aac19619ac65d68537dbe1
71b3857a72344a1a9d74bd3923460854"

ans2 =
"86f9b43a9bc2dee8342d2a780c1ef9fc2cd5d83a8fc792859143298a0bba80a4155313e575c419f775570a49246223fd
33191e3ecfd04c702312d61a644eae91"

ans3 =
"983d12f02c30e27f02e428b761aab9865412db91b6ab1eb77998d87f221f58d7acea28140e606aeb86d16032807b2fef
8db0861fddc5ad541345e97d642ad973"

ans4 =
"5cf8ba1f568f7aa57a9fb519554b8e09d7079e0ce4b2138719ea3b68190ec81858126413669442e4b70bc8c0cab2f6d3
06efcb1f27e68bb1d70f570ada755fca"

ans5 =
"ed371216b8048a8138e5404eeb4dd6872963d632647e479879896989b02b46b609cf1adf6add7a96c97102e2f69eec77
428edf4e3ef5bc28c0921e875bbda90a"

ans6 =
"eb6a1d1e3971e93a31d7c47cc8b10a3a5936b8742f21808a1467939b4fd9a4999554b1c117d4f7688e2ea9cb357316d1
c9e8bc5962969efa8715b593f9595e7b"

Q1 = input("Scroll through the data frame output, what is the destination port of these network
flows ?")

result = hashlib.sha512(Q1.encode())

if result.hexdigest() == ans1:
    marks = marks + 10
    print(" Congratulations Question 1 correct: The Name is Bond, James Bond!!")
else:
    print("Houston, we have a problem!. Try again, incorrect answer")
    exit()

Q2 = input("How many windows security host logon logs are there for the account
'NTAUTHORITY\SYSTEM'?")

result = hashlib.sha512(Q2.encode())

if result.hexdigest() == ans2:
    marks = marks + 10
    print(" Congratulations Question 2 correct: Po, There is no secret ingredient!! ")
else:
    print("Houston, we have a problem!. Try again, incorrect answer")
    exit()

Q3 = input("What is the URL of the first index in the extracted indicator array?")

result = hashlib.sha512(Q3.encode())

```

```

if result.hexdigest() == ans3:
    marks = marks + 10
    print(" Congratulations Question 3 correct: E.T phone home")
else:
    print("Houston, we have a problem!. Try again, incorrect answer")
    exit()

Q4 = input("What is the secret URI?")

result = hashlib.sha512(Q4.encode())

if result.hexdigest() == ans4:
    marks = marks + 10
    print(" Congratulations Question 4 correct: Luke, I am your father")
else:
    print("Houston, we have a problem!. Try again, incorrect answer")
    exit()

Q5 = input("What is the observable DNS?")

result = hashlib.sha512(Q5.encode())

if result.hexdigest() == ans5:
    marks = marks + 10
    print(" Congratulations Question 5 correct: Hasta La vista, baby!!")
else:
    print("Houston, we have a problem!. Try again, incorrect answer")
    exit()

Q6 = input("There is an IP that has the most host logon events, can you find it?")

result = hashlib.sha512(Q5.encode())

if result.hexdigest() == ans6:
    marks = marks + 10
    print(" Congratulations Question 6 correct: What is your deepest Fear Mr.Cruz??")
else:
    print("Houston, we have a problem!. Try again, incorrect answer")
    exit()

print('You have finished the Quiz, score = ',marks)

```

## Grading Script: Section Two

---

```

import hashlib

marks=0
ans1 =
"bb4770d066049ebfb000aec863d8f994919c278039c7048e8944b4b459ab39f3db49a694c6d8514ac0090956fb3c5d71
0ab01a4ab7c30e1f6e0f18563b1b8d9c"
ans2 =
"22f7bf3bd40ba4bbb385a4506633aac576c2e99d7215c6d9e98267a0353dd8d166c836f82797e650100bf32fb42a8fc3
e4c648dfaa5a2beea697c6590e2b0e9d"
ans3 =
"6327cb4201e38f7190ca2394b17f3766d391eb08e4dea0b2daaa874fd403354dab72d3102c219a85cf64766c2f7e0fee
5cec24282c8e6f82c625af6c08286016"
ans4
="079b40e41372b83e6310cc31bc2bd08428d4c2010780519695e919b5ba8a7e07d53aebd46bae17ee9bbe101126233a5
d6985e78510a111de2c310d5e77e26559"
ans5 =
"3bafbf08882a2d10133093a1b8433f50563b93c14acd05b79028eb1d12799027241450980651994501423a66c276ae26
c43b739bc65c4e16b10c3af6c202aebb"

Q1 = input("What is the admin portal password of the website:")

result = hashlib.sha512(Q1.encode())

if result.hexdigest() == ans1:
    marks = marks + 10
    print(" Congratulations Question 1 correct: This is Sparta!!!")
else:
    print("You are not gonna get away with this, try harder.")
    exit()

Q2 = input("What is the name of the file you found on the webserver:")

result = hashlib.sha512(Q2.encode())

if result.hexdigest() == ans2:
    marks = marks + 10
    print(" Congratulations Question 2 correct: All work and no play makes Jack a dull boy")
else:
    print("You are not gonna get away with this, try harder.")

Q3 = input("What is the password present inside the file:")

result = hashlib.sha512(Q3.encode())

if result.hexdigest() == ans3:
    marks = marks + 10
    print(" Congratulations Question 3 correct: Bazinga!!!")
else:

```

```
print("You are not gonna get away with this, try harder.")

Q4 = input("What is the name of the file found on MGMT system:")

result = hashlib.sha512(Q4.encode())

if result.hexdigest() == ans4:
    marks = marks + 10
    print(" Congratulations Question 4 correct: This is Sparta!!!")
else:
    print("You are not gonna get away with this, try harder.")

Q5 = input("The number of customer entries that you found on the file:")

result = hashlib.sha512(Q5.encode())

if result.hexdigest() == ans5:
    marks = marks + 10
    print(" Congratulations Question 5 correct: This is Sparta!!!")
else:
    print("You are not gonna get away with this, try harder.")

print('You have finished the Quiz, score = ',marks)
```

### **Grading Script: Section Three**

---

```

import hashlib
import sys

marks=0

hashans=["aeb6d07fa73b64058060ba0a9efb4bf97d16b28067db67c9d8fe5e43b70a429d"
, "9fd7e522f8ee5854a3de30ed15cee3463f62ed957f979b9724863bbf95a56a11",
"5d5e63ffd5ef09f044d6d8a1128d3fb8fb0d4a84b617e4510f02d13cc515b1c5",
"7011791a5f3d49a8f6e5e0f134f915a7d430826ed215c4df44c95d58c6c68568",
"8a798890fe93817163b10b5f7bd2ca4d25d84c52739a645a889c173eee7d9d3d",
"b896db9af1cfb35a4d2151ce816f01e53a7b3189b2c62fecf9b3c71846356a22",
"5d5e63ffd5ef09f044d6d8a1128d3fb8fb0d4a84b617e4510f02d13cc515b1c5"]
questions=["Provide the part after | in the splunk query construction to get the SourceIPs in
tabular format",
"Provide the pandas dataframe function that helps you to get the unique values in a column.
Answer in the format .function()", "What is the IP address of the system from the external network
that has connected to the web server?",
"There is a request that is happening in a malicious frequency, what is the web route being
queried that request?",
"Is both UDP and TCP used? (Answer yes/no)",
"What is the destination IP of the FTP services?",
"What is the source IP of the TCP services?"]

i = 0

for i in range(7):
    ans = input("Q"+str(i+1)+": "+questions[i]+"\\n")
    res = hashlib.sha256(ans.encode())
    if res.hexdigest() == hashans[i]:
        marks = marks + 10
    else:
        sys.exit("quiz 3 failed with "+ str(marks) + " out of 70 marks")
print("quiz 3 passed with 70 marks");

```