Configuration Manual

MSc Research Project
Cyber Security

Kapil Patil
Student ID: X18127126

School of Computing
National College of Ireland

Supervisor: Mr. Christos Grecos

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Kapil Patil

Student ID: X18127126

Programme: Cyber Security **Year:** 2019

Module: MSc Research Project

Lecturer: Mr. Christos Grecos

Submission Due Date: 12/12/2019

Project Title: Securing Remote Access communications using Deep packet Inspection

Word Count: 681 **Page Count:** 9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature: ……

Date: 12/12/2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use	
Only Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Securing Remote Access communications using Deep packet Inspection

Kapil Patil
X18127126

1 Introduction

The Configuration manual illustrates the hardware and software requirements for implementing the Research project. It also contains screenshots that shows the step-by-step implementation procedure of this research project "Securing remote access communications using Deep packet Inspection". The aim of this research to detect SSH-Tunnelling or port-forwarding using deep packet inspection where open-source firewall that is integrated with a customized script. This script is written in Python and various Python libraries are used which are namely: Pyshark, Pcapy, Scapy, dpkt to inspect the traffic inside the tunnel.

2 System Configuration

For implementing any projects system configuration is the most important step. We must know the hardware and software requirements for the implementation of this project.

2.1 Hardware Requirements

This research is conducted on a local system with the following hardware configuration:

- **Processor:** Intel i5- 3230M CPU @2.60 GHz
- **RAM:** 8.00 GB
- **System Type:** 64-bit operating System

2.2 Software Configuration

Below are the softwares which are used, during the implementation of this project

- **Operating System:** Windows 7
- **Tools Used:**

1. **VMware Workstation 15 Pro:** This is used for the virtualization and integration.
2. **Python 3.8 (64-bit):** The entire customize script for the detection of the nested tunnel is written in Python. All the parameters to inspect traffic such as Source-IP, Destination-IP, Source-port, Destination-port, Payload and Cipher suite and results of the same are done using Python (Python, 2019).
3. **PyCharm:** PyCharm is an integrated development environment that is used for the script. (Brains, 2019)
4. **PfSense firewall version 2.4.4:** This is the open-source firewall that is used inside a virtualized environment where the customized script is integrated with this to detect the malicious activities inside the SSH tunnels (PfSense, 2019).
5. **GNS3:** This is a network simulator that is used to simulate the overall project implementation.
6. **Wireshark version 3.0.7:** This is a packet sniffing tool that is intercepting the traffic between client and server, can create a .pcap file for further inspection.

3 Steps for Implementation

- Download VMware Workstation.
- Download GNS3
- Install various python libraries as mentioned above.
- Install Wireshark
- Install PfSense firewall
- Execute the python file that contains the project code and after that respective output will be displayed accordingly.

4 Inspection of SSH tunnelling traffic using Python code

4.1 Shows the code to detect the client-server handshake first

This will basically check SSH version that is send from server to client and vice-versa

```
# First SSH server will sends its version number before the client does. Because
# here we require to index the handshakes dictionary by (clientaddress, clientport, serveraddress, serverport)
# (which is fully arbitrary), we will build an 'index tuple' and the reverse of it, to detect
# the client responding to the server
index_tuple = ip.dst, tcp.dport, ip.src, tcp.sport
reverse_tuple = ip.src, tcp.sport, ip.dst, tcp.dport
# If we see any evidences of an SSH handshake
if re.search('^SSH-2.0-', str(tcp.data)):
    # If it is first SSH packet, we have seen (from this pair of hosts+ports),
    # and this will be  server sending its version number details to the client (tcp connection
    # is already established at this point of time

    # This is true only when client sends its version number details to the server, after
    #  server has sent its own version number
    if (reverse_tuple in self.handshakes) and type(self.handshakes[reverse_tuple]) == bool and self.handshakes[reverse_tuple] == False:
        self.handshakes[reverse_tuple] = True
        # It will returns a Conversation object.
        # As of now, we don't detect the cipher suite,
        # we are just assume that the cipher suite is "aes128-ctr hmac-md5 none"
        return Conversation(ip.src, tcp.sport, ip.dst, tcp.dport, Ciphersuite("aes128-ctr", "hmac-md5", False))
```

4.2 Check the conversation between client and server

Below screenshot shows that, if any SSH tunnel traffic comes at edge and this script will check the client-address, client-port, source-address, source-port, and cipher suite.

```
class Conversation:
    def __init__(self, caddr, cport, saddr, sport, csuite):
        # Client address and port
        self.caddr = caddr
        self.cport = cport
        # Server address and port
        self.saddr = saddr
        self.sport = sport
        # Cipher suite - we assume here that the same cipher suite is in use in both

        # Needs to type cipher
        self.csuite = csuite
        # Client stats
        self.cstat = StatsEntity()
        # Server stats
```

This script is integrated with open-source firewall (PfSense) to inspect the tunnelled traffic and decide the action whether this is pure-SSH or non-SSH traffic.

4.3 PfSense firewall console

```
FreeBSD/amd64 (pfSense.localdomain) (ttyv0)

VMware Virtual Machine - Netgate Device ID: e4dc138d9fc6f4e3867b

*** Welcome to pfSense 2.4.4-RELEASE (amd64) on pfSense ***

 WAN (wan)       -> em0        -> v4: 10.1.56.10/24

 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) PHP shell + pfSense tools
 4) Reset to factory defaults         13) Update from console
 5) Reboot system                     14) Enable Secure Shell (sshd)
 6) Halt system                       15) Restore recent configuration
 7) Ping host                         16) Restart PHP-FPM
 8) Shell

Enter an option:
```

After logging into firewall, create a rule from client to server where only port 22 (SSH) is enabled.

5 Evaluation / Experimental Results

5.1 Flow between 10.1.56.45 and 10.1.56.85

The figure shows the connection initiation process between the SSH client and server, where 10.1.56.45 is an SSH client and 10.1.56.85 is a server. After connection established, web-service is accessed on 10.1.56.219 via SSH-tunneling using port 8080 on localhost.

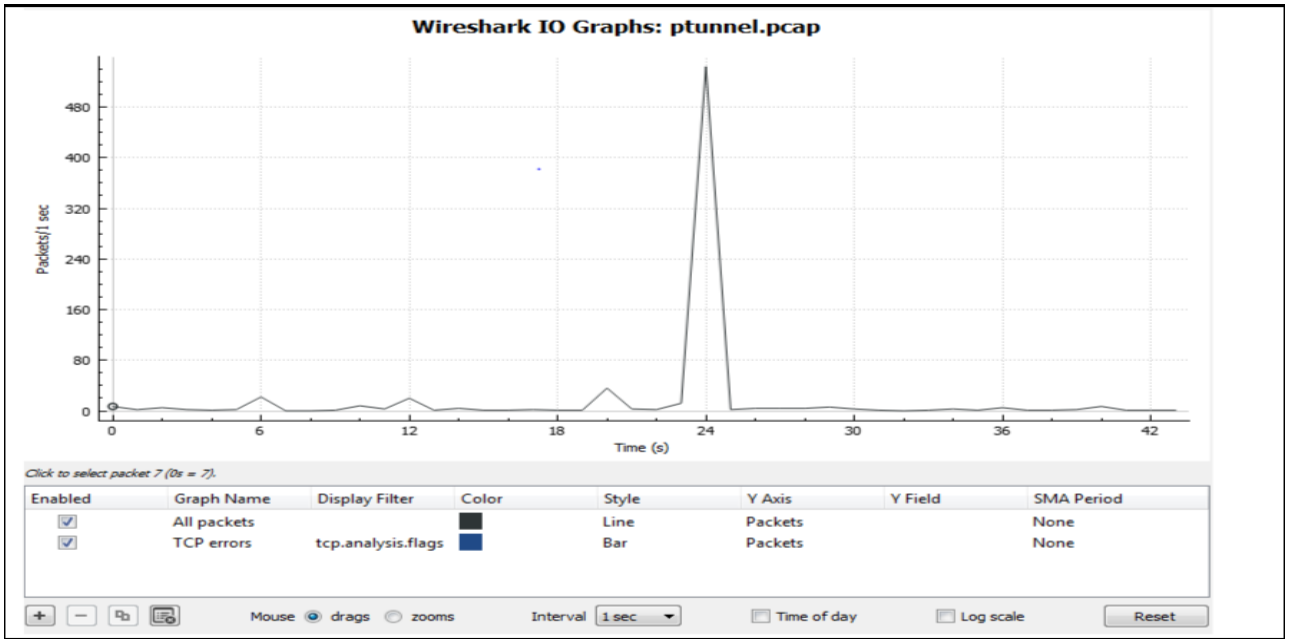
The evaluation is carried out in two possible ways such as below:

1. SSH tunnel detected: If any crafted or nested traffic is shared between two communication mediums then the results will show as SSH tunnel detected as shown in below snapshot.

2. SSH tunnel not detected: If the traffic between two communications medium is genuine and authentic then it will show as SSH tunnel not detected as shown in below snapshot.

5.2 SSH tunnelling traffic detected

In below diagram, packets will be checked against the Python script if the tunnel contains any nested tunnel within it or not.



5.3 Screenshots of Tunnel traffic detected and not detected

Below screenshots, shows that whether port-forwarding is done or not by analyzing the network traffic between client and server. Also It checks how many packets are being transferred inside this tunnel.

SSH Tunnelling detected (NON-SSH)

```
sshdetection
C:\Users\kapil\AppData\Local\Programs\Python\Python38\python.exe "D:/MS study material/RIC work/Thesis work/Thesis Script/sshtraffic_detection-master/sshtrafficdetection.py"
| ssh traffic detection flow
Analyzing....
  crafted_tunnels
  detection of x11_tunneling
  general_statistics
  jabber_session_info
  Layer_for_nested_tunnel_detected
  session_interaction
generating statistics from pcap file, please wait...

processed 729 packets, SSH Tunneling Detected

Process finished with exit code 2
```

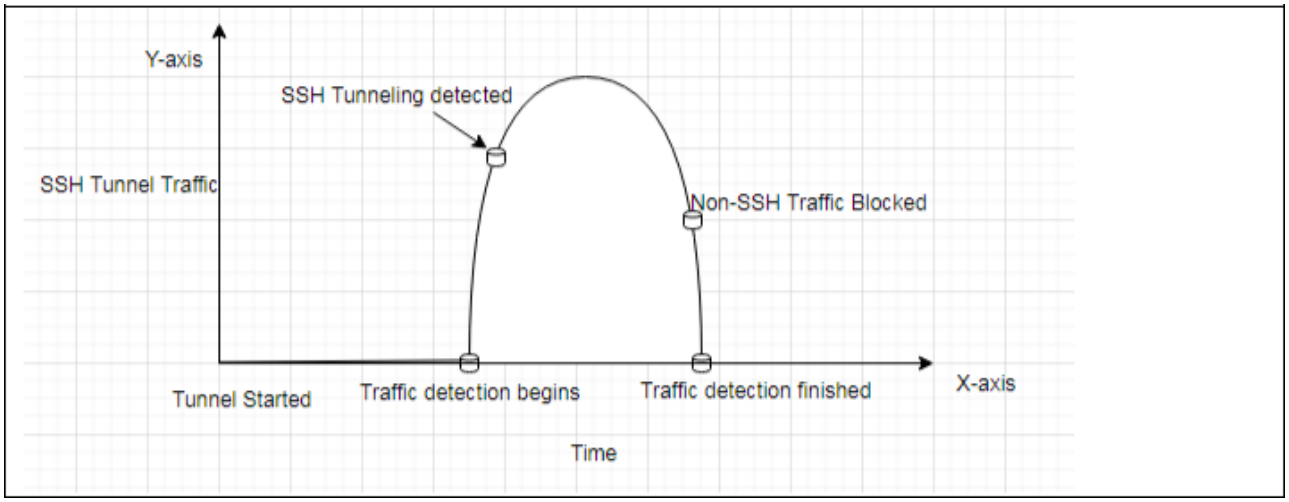
SSH Tunnelling is not happening (Pure-SSH)

```
sshdetection
C:\Users\kapil\AppData\Local\Programs\Python\Python38\python.exe "D:/MS study material/RIC work/Thesis work/Thesis Script/sshtraffic_detection-master/sshtrafficdetection.py"
| ssh traffic detection flow
Analyzing....
  crafted_tunnels
  detection of x11_tunneling
  general_statistics
  jabber_session_info
  Layer_for_nested_tunnel_detected
  session_interaction
generating statistics from pcap file, please wait...

processed 407 packets, SSH Tunneling not Detected

Process finished with exit code 2
```

On basis of the traffic, this solution will block the traffic, if it contains non-SSH traffic inside SSH as shown with below graph.



If a traffic will contains other than SSH traffic then it will detect as shown above or else it will allow the traffic at firewall end.

6 Conclusion

Thus, the deep packet inspection is achieved by implementing customized python script with the PfSense firewall. The solution is able to efficiently differentiate between normal SSH traffic and nested tunneling.

References

Brains, J. (2019, December 2). *PyCharm*. Retrieved from jetbrain.com: https://www.jetbrains.com/pycharm/download/

PfSense. (2019, December 12). *Latest Stable Version (Community Edition)*. Retrieved from pfsense.org: https://www.pfsense.org/download/

Python. (2019, December 12). *Download the latest version for Windows*. Retrieved from Python.org: https://www.python.org/downloads/