Securing Remote access communications using Deep packet Inspection

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:	MSc Cyber Security **Year:** 2019
Module:	Academic Internship
Lecturer:	Mr. Christos Grecos
Submission Due Date:	12/12/2019
Project Title:	Securing Remote Access communications using Deep Packet Inspection
Word Count:	**6608** **Page Count:** 20

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):	

Securing Remote Access communications using Deep packet Inspection

Kapil Patil

X18127126

MSc Research Project in Cyber Security

December 2019

Abstract

Securing organizational or individual data is critical in today's digitized world. Each day comes with a new attack and depending on the attack; countermeasures are set in organizations to mitigate the attacks. As technology is evolving various security solutions are proposed to be a hack-proof organization. Where some organizations are focusing on Deep-packet inspection that can be added to the Firewalls while others are believing in the Host-based IDS (Intrusion detection system) and also the organizations which are dealing with web services are working on the WAF (Web application firewall).For remote access, SSH (Secure-shell) and Telnet are being used to get remote access to servers or applications. Moreover, SSH uses encrypted mode to access resources remotely and also, sends its traffic by using an encrypted tunnel. There is another method called "SSH Tunneling" by which an attacker or any internal user from any organization can breach the data by spoofing the firewall. In this work, deep packet inspection is done to detect the SSH Tunnel by using an open-source firewall which is integrated with customized script.

Keywords: ***SSH Tunneling, Remote access security, Deep packet Inspection, Data Integrity, Open-source technology.***

1 Introduction

In today's world, every business is more focused on data security as well as keeping all the assets secure from an attacker. There are several methods that are implemented throughout the organization network, by implementing various policies and rules. Basically, these implementations are done at three layers namely, access layer, distribution layer, and core layer depending upon the requirement. In every layer, there are different types of networking devices which can be placed, such as Layer-3 switches, routers, firewalls, intrusion detection system or intrusion prevention system and so on. These techniques are capable of tracking and detecting threats in traffic. There are several ways in which security can be implemented on network devices so that Local Area Network (LAN) network can communicate with Wide Area Network (WAN) (also commonly known as the internet) securely without providing any entry point for an attacker to leak the critical data which could cause major losses.

Allowing required ports to access web-hosted servers is a basic requirement for companies. However, these ports may be used for spoofing. The data can be leaked bypassing crafted traffic through an open port, such as the client sitting inside a corporate network and using putty for accessing web servers through SSH port which is allowed at firewall end. Basically, putty can be used to establish an SSH connection from the client to the server. The network administrator opens the port on the firewall based on the request made by users (connected through LAN) who wants to access company web servers. Taking advantage of this connection, an internal user can send the company's critical data through this tunnel. Hybrid traffic can be crafted into the single tunnel and edge base firewalls will see this traffic as genuine traffic, for example, SSH (port 22) leading to data breaches [1].

The communication between a company's internal users to the outside world over the internet is the main concern. There are many methods that can be used to connect through remote login to access hosted servers using Telnet, Secure Shell (SSH), and other third-party tools. Telnet is deprecated because of its insecure communication and to overcome this issue SSH enables extra encryption and decryption at the client as well as server-side. Although SSH is used for a secure connection from any client to server, SSH tunneling mechanisms can also be misused to steal critical data by spoofing corporate firewalls. A malicious user masquerades harmful traffic to look like genuine traffic. Making use of such spoofing, company user or some attacker can easily steal the company data. In order to mitigate such attacks, it is very important to inspect all traffic (ingress/egress) to and fro an organization's endpoint. This will help to track and restrict malicious activity at the perimeter level and ensure the protection of critical data [2].

There are many Original equipment manufacturers (OEM) who are claiming that their devices are capable of inspecting the network base traffic, but these devices are mostly into the traffic pertaining to target service port such as HTTP (TCP-80), HTTPS (TCP-443), DNS (UDP-53), SSH (TCP-22).In a world where cyber-attacks are rampant, one cannot trust traffic which is passing over the network because it may or may not be genuine. Man-in-the-middle attacks are commonly used to intercept traffic flowing through a company's network.

In this paper, the process of deep packet inspection (DPI) is achieved by implementing a script for inspection and decryption of SSH traffic at the open-source firewall (PfSense) so that platform-dependency is eliminated. By implementing this mechanism at the

firewall, malicious or crafted traffic hidden within the SSH tunnel can be inspected to mitigate the risk of breaches. It is very important to protect the data from an attacker by implementing countermeasures to track all the malicious as well as genuine traffic so that anybody can review the logs and can prevent malicious activities inside any network. This script is configured using various python models such as Wireshark, Pcapy, and Scapy among others. The proposed solution detects SSH tunneling, also known as Port forwarding, by inspecting the SSH payload and if there is any traffic other than SSH, it will be blocked. Using deep packet inspection at the perimeter level, data loss prevention can be achieved from inside the organization or the outside world while accessing a company's resources.

Research Question: "How deep packet inspection technique can be used to detect and prevent malicious client-server communication traffic via Secure Shell (SSH) tunnel?"

The first goal is to intercept network traffic with the help of the Wireshark packet snipping tool. Then, a comparison is made between the payload of normal SSH traffic vs. tunneled traffic inside the SSH Tunneling. Following this, a mechanism is built which can detect the tunneled traffic and gives us detailed information. A suitable model that has minimum complexity for differentiating amongst crafted traffic or pure SSH Traffic is chosen. In order to achieve the best results or outcomes, increase the interpretability of the solution as much as possible. Integrate the customized script with an open-source firewall (PfSense) to detect or inspect the traffic. The final objective is to improve accuracy by integrating this solution with a lab-based scenario.
The flow of this document is as follows. Section 2 contains the literature review. Section 3 gives the methodology used for the research. The fourth section specifies the design of the solution. The subsequent sections give the actual implementation followed by the results and conclusion along with a future scope.

2 Literature review

An extensive literature review has been conducted to understand the current state of the art when it comes to securing remote access communication. The literature review has been logically divided into the following four sections, namely Pattern matching, method of filtering, machine learning methodologies, and Deep packet inspection.

2.1 Pattern Matching:

In [3], secure shell (SSH) tunnels are inspected on the basis of pattern matching or recognition by using a Gaussian mixture model (GMM). Further, the mechanism for detection of other than SSH traffic in tunnel traffic is being shared within the SSH secure tunnel. In this paper, the author's main focus was to detect the crafted application other than pure SSH traffic, which checks the packet size and their direction. The GMM method used has some presumptions. Experimental results are based on statistical traffic classification techniques. The authors used the detection of applications that are being crafted inside the SSH tunnel by training a model so that whenever the same kind of incidences occur, they can be captured and monitored. Whereas, in [4], the authors have built upon their existing work, analyzing the traffic based on basic IP-level information

which is nothing but the length of the packets. Also, the author introduced three research contributions in which the first one is on statistical analysis. Second, the author considers the traffic on the basis of behavior. In this stage, the author used the two most popular techniques Gaussian Mixture Models (GMM) and Support Vector Machines (SVM). These two models are characterized by SSH-encrypted sessions. Finally, the author implements a software tool called "SSHgate", which is basically used to collect SSH-encrypted sessions and also the corresponding clear-text traffic.

In [5], authors consider the length, arrival order and the direction of packets. By using this accurate boundary, the detection of whether an actual tunnel happens inside the network or not can be detected. As per the author, the identification of tunneled traffic is limited due to its specific boundary. Also in this paper, the author uses the statistical pattern recognition method. This SSH tunnel's boundary plays a very crucial role in the classification of tunneled traffic, but this method is more dependent on packet-size. If the packets are compressed or some type of padding happens then the size can be varied. So results will inaccurate.

In this paper [6], the author implemented a mechanism for early traffic classification. In this work, Support vector machines (SVM) and Naive Bayes algorithms are used. The author's main focus is on binary classification. In this normal case, the author investigates the traffic based on the number of first data packets collected from TCP packets and secondly, some other data packets being injected so that abnormal behavior can be checked.

In this research [7], the author used the graph-comparison approach which is used to establish a profile of several protocols and also to classify unknown encrypted traffic which is being used inside tunneled. Basically, this method is focused on the behavior of the protocol which is injected inside tunneled- direction of packets, size, and timing. In this paper, multiple streams which are nothing but having multiple protocols present inside one stream and on the basis of this author did the research for encrypted network traffic.

2.2 Filtering Method:

This paper [8] compares traditional network filtering methods i.e. port numbers and IP based the author introduced the two-phased method for classifying SSH tunneled applications on a real-time basis. This two-phase classification involves the identification of SSH connection and others to inspect tunneled applications. The author used the K-Means clustering algorithm for classification. An evaluation was performed using a custom method and results were dependent only upon the traces.

In this paper [9] The author uses a statistical approach for the detection of anomalies inside the tunnels. A chi-square test is used for the analysis of tunneled traffic inspection. As per the author detection of encrypted tunnel applications can be traced using packet size distribution, in this distribution signature is explored for application layer traffic detection. Furthermore, many different trace files were collected.

In this paper [10], the author used Bloom filters for the detection of predefined signatures inside packet payload. Bloom filter is a data structure; this is a hardware-based filter that is used to isolate all the packets which contain predefined signatures. The filter eliminates false positive logs. Furthermore, the approach is used for string matching at line speeds and presents the performance analysis.

This paper [11] discussed the data security and any users can use their own customize cryptographic algorithms such as DES, AES, and RSA inside OpenSSH application. As per the author, any third party or an attacker will not be tracing the data due to its customization while initiating any type of communication.

2.3 Machine Learning Method:

In this paper [12], a machine learning-based method is used for the detection of application-layer traffic. This approach is a holistic one, and this will not check the content or static attributes. This research focuses on forensic analysis. This machine learning method mainly uses two techniques, namely clustering, and classification techniques. According to the author, this method may be used for the detection of malicious traffic.

In this paper [13], the author uses a machine learning method for analyzing network traffic for the use case of intrusion detection systems. Further, an extreme learning method is used for the detection of malicious content inside network traffic.

This paper [14] explains the mechanism to identify accurate network traffic. In this paper, the author used two methods to deal with inspection, one using machine learning and other by using deep packet inspection. As per the author, these two methods are very useful for the detection of malicious traffic. Moreover, machine learning methods are used to identify network traffic based on various encryption methodologies. Basically, in this paper, the main focus of the author is to identify the actual traffic from malicious traffic inside any network.

In [15] is focused on cloud architecture. Due to its dynamic data transaction behavior, a deep packet inspection mechanism is needed. This DPI protects malicious attackers from entering into the organization and also to mitigate data breaches. In this paper, Blitzdump is a packet capturing utility developed so that the inspection over the network can be achieved. Also, this utility improves security by improving the performance of intrusion detection.

In [16] the author explains why the improvement of quality to user service is necessary while network traffic monitoring and data analysis. The Author proposed two methods for the identification of network traffic - Machine learning and deep packet inspection. By using this combination accuracy of identification (malicious traffic) is achieved. Machine learning is used to assist in detecting encrypted traffic whereas deep packet inspection is not suitable for this type of identification.

In this paper [17], the author explains how content-based network traffic inspection is achieved by using the content-based traffic analyzer for software-defined networks. Also, the author suggests two possible improvements, nDPI (ntop deep packet inspection) and ATCA (Advanced Telecom Computing Architecture). These improved methods can be used for a real-time high volume of traffic as well as IoT applications.

2.4 Deep packet Inspection

This [18] work states that there are two categories of IDS 1) Signature-based which can only work if any signature match if that attack has happened before and 2) Anomaly-based which means this will see the behavior of traffic and if any abnormality occurs, it will be triggered, The author introduces a solution based on the second method that is anomaly based on encrypted protocols, such as SSH and SSL. To achieve this detection of abnormality, a non-parametric CUSUM algorithm is used for the detection of

deviation from a normal profile. If a detected profile contains any type of abnormality then their solution will generate an alert with delay to the protocol response, which can be traced back to the attacker.

In [19] This paper explains how deep packet inspection (DPI) is important in a modern network where there are plenty of applications and software are changing on a daily basis. Deep packet inspection depends on predefined signatures in the payload which is a matching process that results in elevated consumption of memory and CPU resources. The approach given by the author yields a faster process by minimizing the load and other factors responsible for consuming memory and CPU. To achieve an accelerated Deep packet inspection, engine quotient filter is used.

In this paper [20], deep packet inspection is performed in an efficient two-level IDS, which is applied with statistical patterns using sequential, differential methods for the abnormal packet. As per the author, the two-level system converts high-faceted character space into low-faceted space. This will reduce the computational cost and integrates groups of patterns into a uniform pattern. The final evaluation is done using DARPA 1999 IDS dataset for the detection of unauthorized traffic.

In this paper [21], the author's main target is TOR (The Onion Router) anonymous proxy service. Tor is widely used by botnets, malware while performing distributed denial of service attacks, online frauds, spams, and etcetera. This work explains TOR usage detection using deep packet inspection to analyze the network traffic. Using this identification and blocking design, Onion router traffic can be blocked.

In [22] A real-time anomaly detection mechanism is explained. The method is suitable for both small as well as fast hardware implementation designed to be embedded in network devices. Each packet inside a network is characterized using a feature called hypercube hash functions, and then it can be used in hardware and software applications. For normal and abnormal feature region, this two-dimensional feature space is approximated into a binary bitmap. This solution is efficient for detection since it performs offline machine learning in real-time.

Building upon the various works to secure critical data, the proposed approach gives an added layer of security by implementing a customized script to detect the remote access traffic being placed on an open-source firewall to mitigate platform dependency and customize as per need. This proposed solution will aim to eliminate any kind of malicious activity inside the SSH tunnel so that an organization’s critical data will not be breached. In this solution, the python-based customized script is implemented so that it will see the SSH traffic payload inside the tunnel and if any traffic contains other than SSH protocol inside tunnel firewall will inspect the traffic which is integrated with this Script. The proposed solution contributes to the existing solutions in terms of agility, platform independence, and applicability.

3 Research Methodologies

In figure 1, client-server communication is happening while all the traffic is passing through the firewall, which keeps track of the traffic flow in its state table as would any state-full firewall.

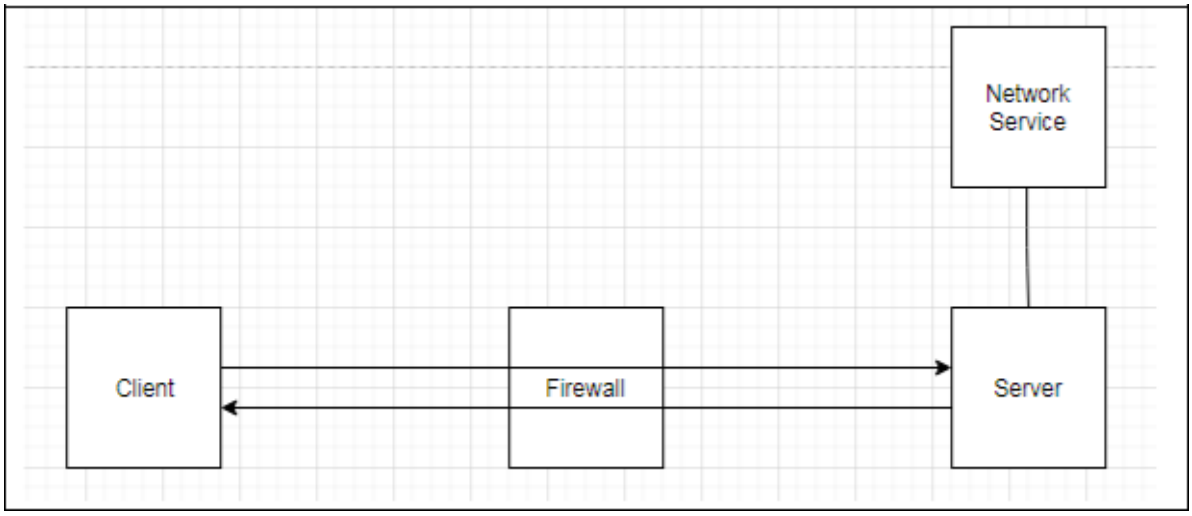


Figure 1: Simple Client-Server Communication

Basically, the client is accessing the hosted network services on the server. This remote access is taken with the help of SSH protocol due to its high level of security (encryption). This SSH protocol creates the SSH tunnel from client to server and all communication in between will occur securely. But, an attacker can make use of this encrypted tunnel and can use SSH tunneling or port-forwarding so that traffic other than SSH can easily be sent inside that tunnel meant to allow only SSH or port 22 accesses at firewall end and the firewall will perceive it to be normal SSH traffic. Such type of tunneling leads to major data loss and to prevent these data breaches, this approach provides a solution by providing an extra layer of security at perimeter level (Firewall) which is integrated using a customized script so that it will intercept the tunneling as shown in the below diagram.

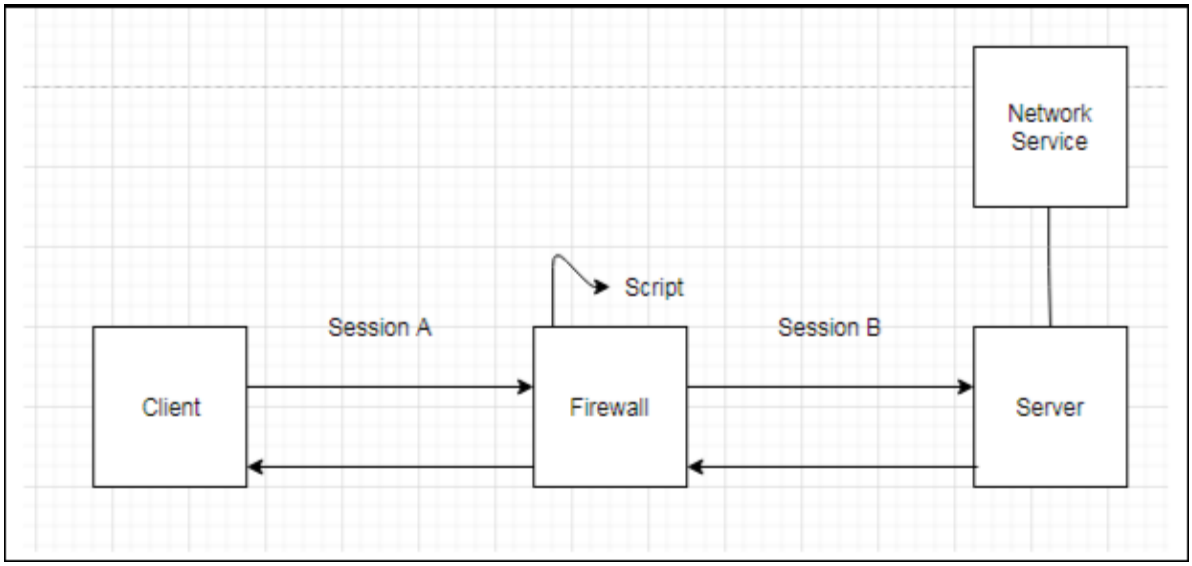


Figure 2: Implemented solution using Script

In figure2, the SSH tunnel is first inspected at the firewall end with the help of a script that is running parallel with an open-source firewall (PfSense). The client first establishes the session with a firewall and then at firewall it will check for any type of crafted packets passing inside the tunnel by checking the payload of network protocols and its size. If any type of suspicious behavior is identified at a firewall then this traffic is blocked so that any data loss will be prevented and if this tunnel is with pure SSH traffic then this will establish the connection between client and server. Below flow-chart also gives a clear picture of the methodology which is being used to detect malicious or crafted behavior during the client-server communication.

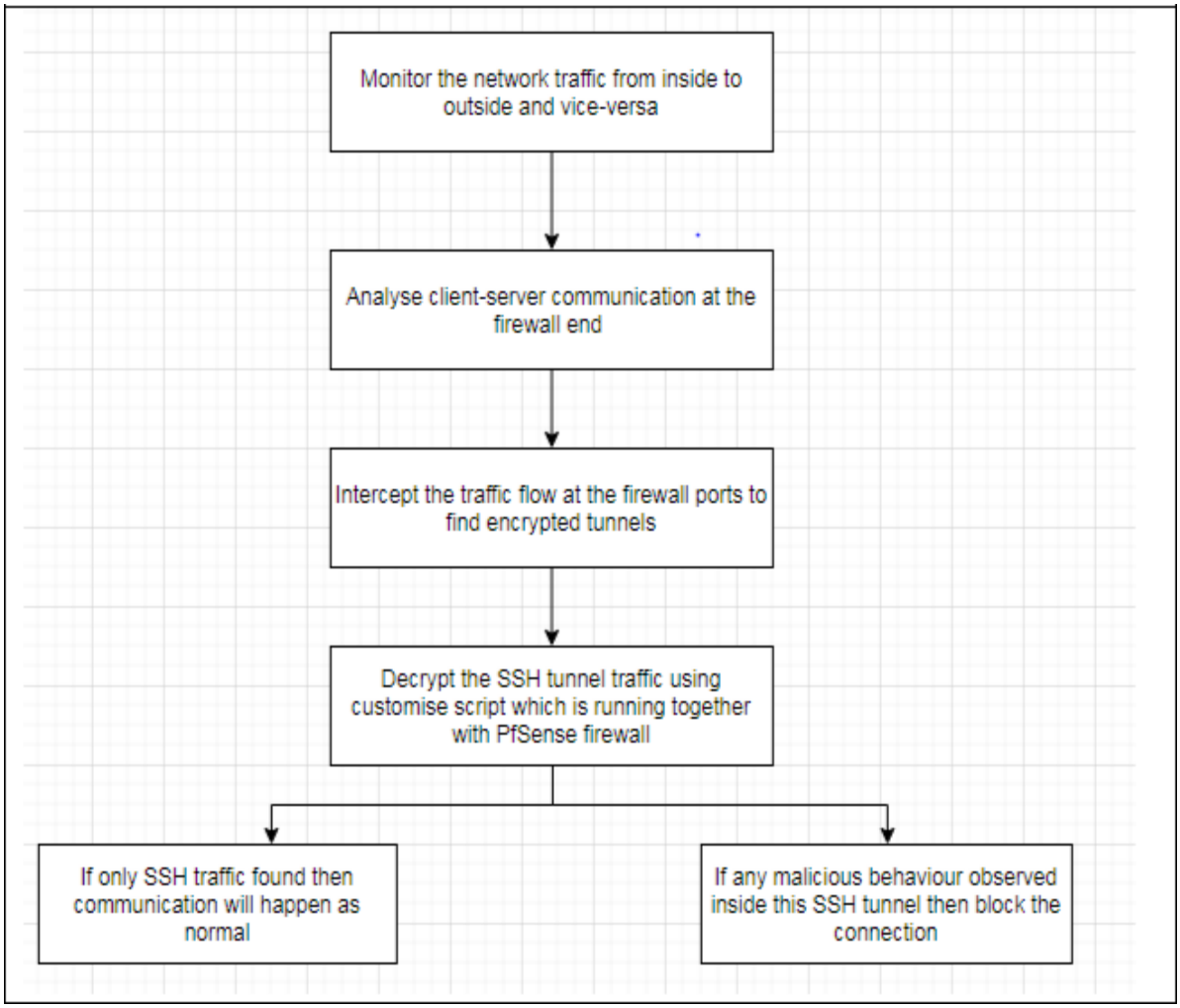


Figure 3: Workflow Diagram

Figure (3) is a flow diagram for detecting tunneled encrypted traffic in correspondence with realization. At first, the solution will monitor the overall network flow which can be useful while understanding the network flow. While at the second stage analysis of client-server communication is done at firewall end and after analyzing this traffic, interception, and decryption of traffic which is performed using a script which sits within the firewall at the perimeter level. Finally, if any malicious or crafted traffic is being found then this

will block the communication and if there is pure SSH protocol traffic is observed then communication between client and server will continue as it is. This customized script is basically making the decision on the basis of the payload for all the networking protocols and also considering its packet size. If the script finds anything other than SSH payload inside the tunnel then after decrypting the SSH tunnel, it will block that type of suspicious activity at the perimeter level. Inside python, there are built-in libraries, namely Wireshark for packet intercept and Pcapy, dpkt and Scapy modules which are used to detect the behavior of traffic and on the basis of the result, this script will decide whether there is any type of SSH tunneling or port-forwarding occurring in the network. This solution will provide an effective solution to help data loss prevention from inside the organization or an attacker who is trying to access the resources from outside the network.

4 Design Specifications

Figure 4 provides an overview of the architecture for the proposed solution. The network is divided into two parts which are namely inside and outside the network, where a firewall is placed in between two networks. As shown below, an SSH tunnel is established between the inside user to an outside server and vice-versa. Traffic which is from 10.1.56.85 to 10.1.56.45 is encrypted and port 22 (SSH protocol) is opened so that the communication between client and server is established. There are two scenarios 1.Pure SSH connection between client and server. 2. Another with some other services is using inside this SSH tunnel which is also referred to as SSH tunneling or port-forwarding. There are possibilities of data breaches from inside the network or from outside to access resources that are inside the network. Also, this outside to inside tunneling is referred to as reverse port-forwarding. Firstly, below lab is configured in a virtual machine and network traffic is intercepted using Wireshark and the same traffic is monitored at a firewall end. PfSense which is an open-source firewall means its source code is available for customization and also python based script which can be integrated at the firewall so that the traffic can be filtered at network boundary level.

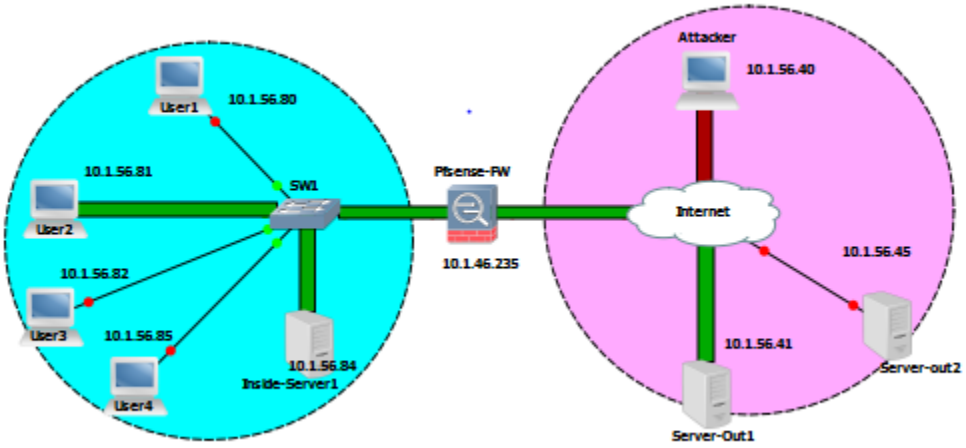


Figure 4: Design architecture of Implemented Network

The network is configured with a 10.1.56.0/24 IP range and in this scenario, a user is accessing the remote services which are served by a server using remote access via an SSH tunnel. Also, servers and attackers are in the outside internet world, and also they are accessing the internal resources from the outside means that the network is untrusted. There are several types of mechanisms or tools that can be used for taking remote access, in the Linux and Windows system, user can make use of OpenSSH and Putty for the tunneling feature. They can easily spoof the firewall as normal firewalls are not capable of capturing the content which is flowing from it but this solution proposes to mitigate this type of port-forwarding. Basically, this tunnel traffic is filtered and checks whether any type of crafted traffic is being transferred through this tunnel. At Eth1 and Eth2, which are the interfaces of firewall and traffic is segregated into inside and outside.

Figure 5 shows only traffic on port 22 is allowed at the firewall and the client is still accessing web service on 8080 by using SSH Tunneling or port-forwarding which is breaching the security of the organization. By using OpenSSH or putty one can easily establish the tunnel and send the non SSH traffic inside this encrypted SSH tunnel. Here firewall is filtering the traffic and found that only 22 port is allowed but due to its encapsulation packet, an attacker can transfer the HTTP, HTTPS, FTP, and etcetera traffic from this encrypted tunnel which is obviously causing major data loss and by doing this company's data integrity policies are breached.

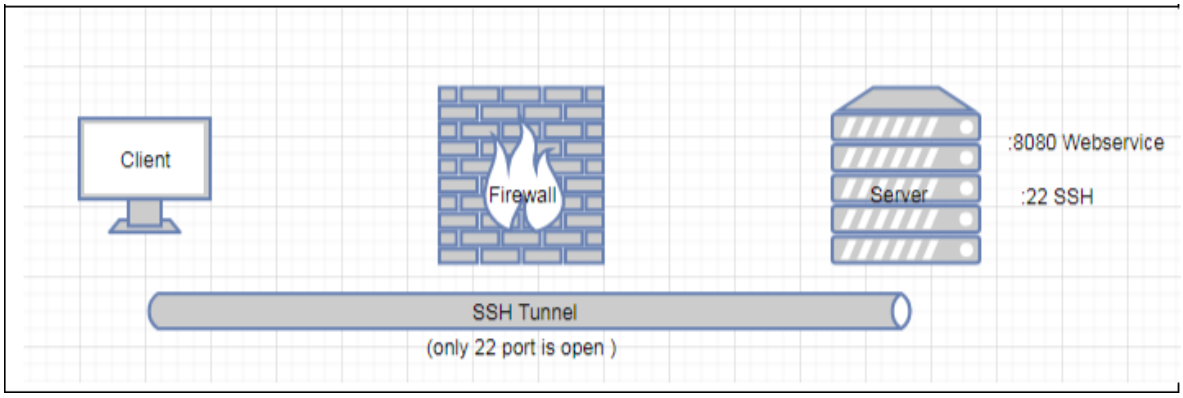


Figure 5: SSH Tunneling

This forwarding feature provides an encrypted connection from a client to a server inside the SSH tunnel. There are three types of tunneling or port-forwarding which are namely: local port-forwarding, remote port-forwarding and dynamic port-forwarding. This port-forwarding is nothing but an interaction between two applications that are TCP/IP applications, these applications are accessible through SSH connection. Even though the IT administrator blocks the non SSH traffic at firewall still the attacker can transfer the non SSH traffic inside the tunnel. In this architecture combination of firewall rules and script can decrypt and inspect the tunneling traffic and take necessary actions which port-provides an extra level of security which can prevent data loss.

5. Implementation/ Solution Deployment

SSH is a very important protocol that traces need to be monitor closely inside any organization.SSH is commonly used for purposes like remote control or file transfer. SSH tunnel can drill firewalls and all its NAT rules. Also, it can penetrate Intrusion detection and prevention systems and other security information and event management (SIEM) tools. SSH version 2 supports SOCKS5, which can provide users to set up a SOCKS5 proxy outside the organization and this can hide all the web activities.

SSH tunnels are divided into two types namely: Forward tunnel, means users a user from the inside can connect to an outside service by spoofing the edge monitoring or filtering devices. Another is a reverse tunnel that allows outside untrusted users to connect to inside servers and this can also be referred to as “autossh tunnel” which are explained in the following figures 6 and 7.

1. Forward SSH Tunnel

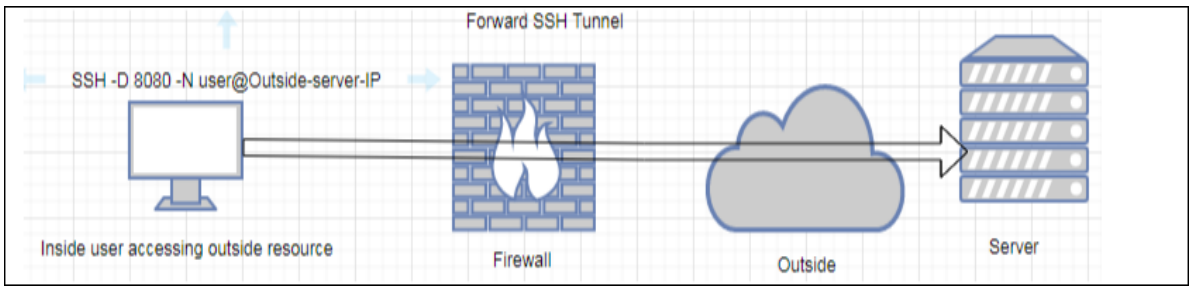


Figure 6: Forward SSH Tunnel

In diagram6, as internal users are connected to a server with SSH using a putty option called tunnel, a user can forward the non-SSH traffic using the command "SSH –D 8080 –N user@Outside-server-IP address" and access the other hosted service on an outside server using forwarded port 8080. In this command, SSH v2 –D allows full SOCKS5 features.

2. Reverse or autossh tunnel

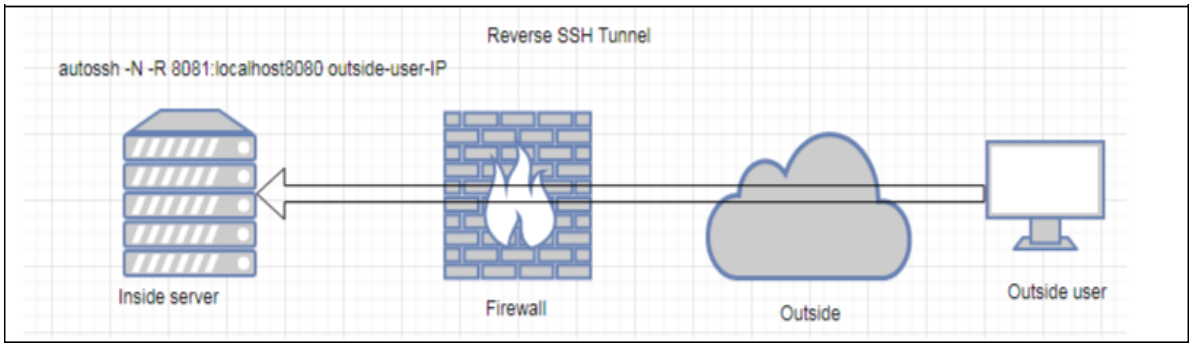


Figure 7: Reverse SSH Tunnel

In figure 7, an outside user can use the command "autossh –N –R 8081: localhost: 8080 outside-user-IP" and connect to any of the services which are listening on the inside server.

There are several ways by which SSH tunnels can spoof the firewall or network monitoring, below are some important ways of tunneling exploits:

1. Dynamic tunneling as a proxy: By using this dynamic feature of port forwarding, users can proxy web-based traffic through the encrypted SSH tunnel.

2. Single Flow traffic: If any organization is monitoring in and out traffic using various monitoring tools such as Syslog, SIEM and other network monitoring tools (NSM), these monitoring devices will see there is a single flow passing tunnel while hybrid traffic is passing inside this tunnel.

3. Anytime access into a network: Autossh can provide outsiders continual access to the core layer of the organizational network.

5.1 Detecting or Inspecting of SSH tunnel

It is of paramount importance to inspect the remote access traffic at the edge point of any organization. The proposed solution gives insights into the SSH traffic by detecting the non-Ssh traffic inside encrypted tunnels. The inspections of packets are carried out at header and footer level itself i.e. payload so that the actual message or the contents of the message are kept private and confidential. The aim is to detect only the nested traffic inside SSH tunnel and not the content of the packet which restores the data privacy of the user's message.

Below are the steps which are followed during the implementation for the detection of SSH tunneling traffic.

1. Customized python script, which uses various python modules or libraries.

Pyshark: This is a python packet parsing library which is using Wireshark dissectors. There are quite a few python packet parsing modules, but pyshark is different because it does not parse any network packets, instead it uses command-line utility. This allows parsing from a pcap (packet capture file) or lives traffic. This is done using installed Wireshark dissectors. By using the Pyshark module, required network device interfaces can be intercepted and then analyzed to capture network traffic for further decision-making.

Pcapy: This is a Python addendum module that can interface with a packet capture library (libpcap). Pcapy is a system-independent interface for network packet capture. This gives a lightweight portable framework for network monitoring and also for network debugging, security monitoring, and etcetera. Also, it is compatible with Linux/Unix and Windows operating systems and provides a simpler Object-Oriented API.

Scapy: This is Python's library which is generally used for packet manipulation in network traffic analysis. With the help of this, construction or Packet decoding can be done on a wide range of networking protocols, transfer them on the wire and capture them which can be read or saved using pcap files. This is designed in such a way that it can process fast packet prototyping using default values. This supports Python 2.7 and 3 across any operating system.

dpkt: This is a python library that is used to analyze the network traffic. It is fast, simple packet parsing/creation, also useful for TCP/IP protocol analysis. In this work, source IP, source port, destination IP, destination port and payload is extracted using Python code from .pcap file.

2. Integration of customize Python script with open-source firewall (PfSense).

PfSense is a Linux based firewall with a free and open-source operating system that can be installed on any hardware. It gives a user-friendly graphical user interface (GUI) and a command-line interface (CLI). It has many uses like one can configure PfSense as a router, proxy, DNS server, DHCP server and so on on a single hardware device. Source-code of this firewall is easily available for customization as per the requirement of the organization. This is flexible by design and therefore can be scaled.

3. Decision of port-forwarding or SSH tunneling

The above-integrated solution will check the traffic flow and then it will decide whether any type of malicious or crafted activity is happening inside a network. In this script, below are some of the main parameters that are used for the detection of SSH traffic flow. These are "client-address, client-port, source-address, source-port, and cipher suite" inside the traffic flow. To do the same, dpkt python library is checking the Ethernet frames and decides upon the size of that frame. These packet sizes are checked for all application-layer protocols. By checking all the data received from .pcap files collected by pyshark, it will detect if any nested tunnels are found inside a single SSH tunnel.

6 Evaluation/ Experimental Results

In this section, the experimental results are obtained by simulation. The simulation was performed using Wireshark. These simulations are run on a system with 8.00 gigs of RAM, 2.60 GHz Intel Core i5 processor and Windows 7 operating system. The SSH tunneling detection script is written in Python.

The figure8 shows the connection initiation process between the SSH client and server. Here TCP handshake occurs and once the connection is established, actual data frames start to flow. As per the flow graph, 10.1.56.45 is an SSH client and 10.1.56.85 is a server. After the connection, web-service is accessed on 10.1.56.219 via SSH-tunneling. This gives a clear view of the packets transferring through the SSH tunnel.

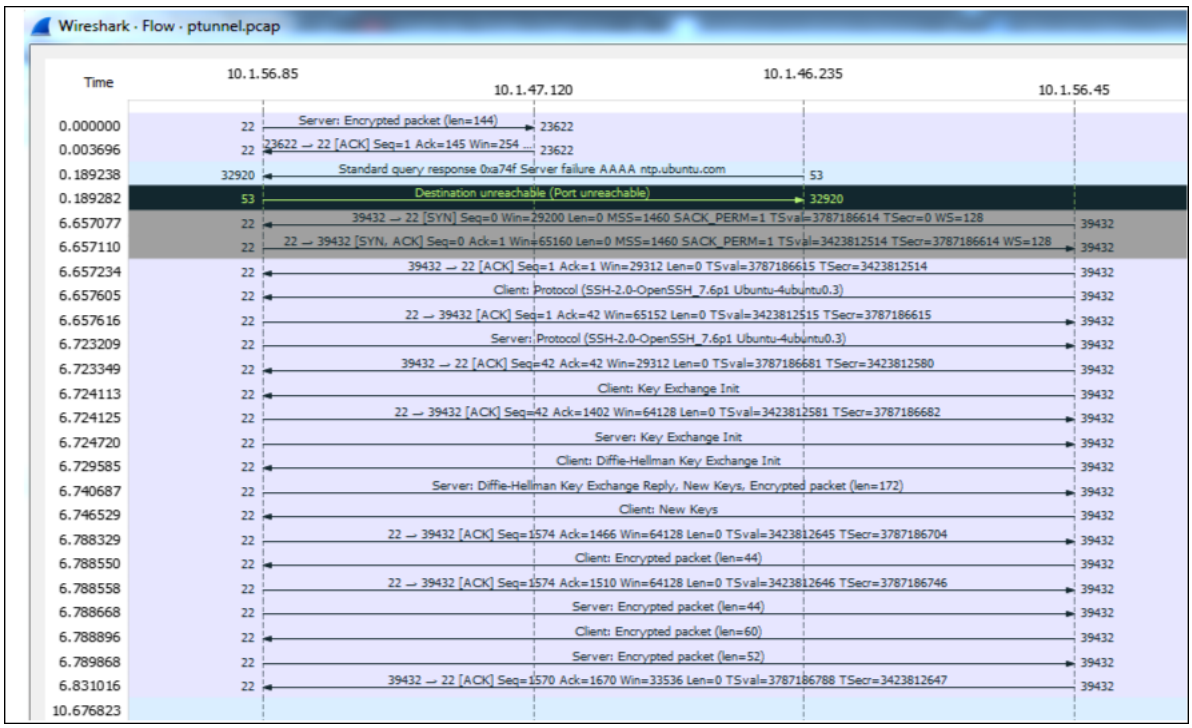


Figure 8: Client-Server handshake

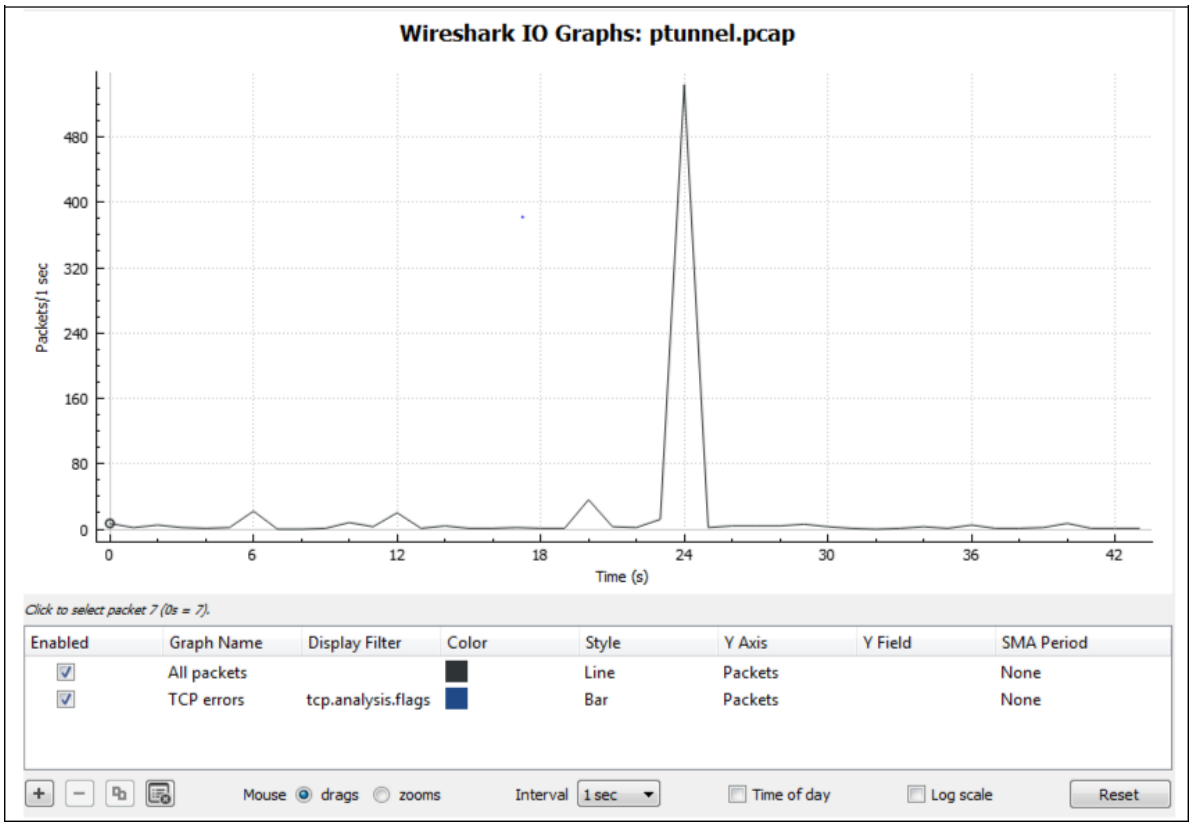


Figure 9: Tunnel traffic I/O graph

In graph 9, the tick interval is chosen to be 1 sec, meaning that the tunnel will be checked every second. These packets will be checked against the Python script if the tunnel contains any nested tunnel within it. As can be seen, the spike in the above graph indicates that hybrid traffic is found to be present in the tunnel. This hybrid traffic contains non-ssh traffic, i.e, HTTP traffic. This means that the script has been successful in identifying non-SSH traffic trying to be masked within an SSH tunnel.

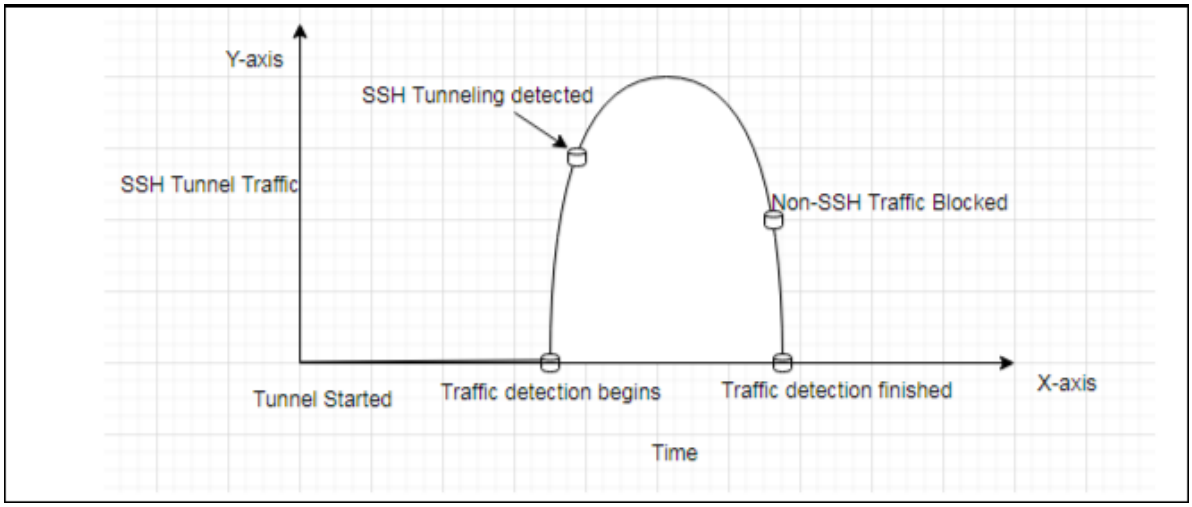
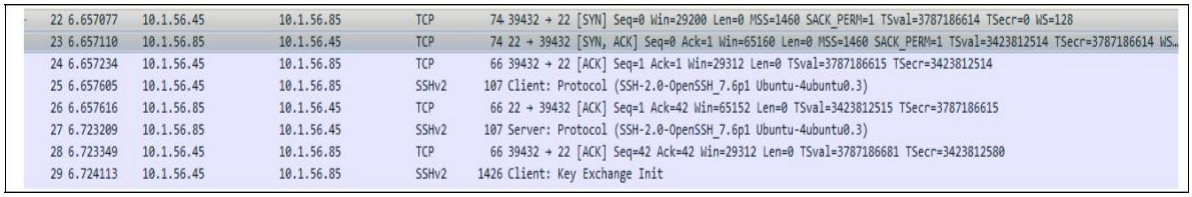


Figure 10: SSH Tunnel detection and Blocking

The graph 10 shows how the experimental results correlate with the actual process of detecting non-SSH traffic that is passing through an SSH tunnel. After the identification of such traffic, the traffic can be blocked. This would mitigate any risk of a data breach by an attacker trying to mask non-SSH traffic into SSH traffic. The blocking mechanism can be incorporated into the Python script as required and an alert may be sent to the network administrator.



22	6.657077	10.1.56.45	10.1.56.85	TCP	74 39432 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3787186614 TSecr=0 WS=128
23	6.657110	10.1.56.85	10.1.56.45	TCP	74 22 → 39432 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3423812514 TSecr=3787186614 WS…
24	6.657234	10.1.56.45	10.1.56.85	TCP	66 39432 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3787186615 TSecr=3423812514
25	6.657605	10.1.56.45	10.1.56.85	SSHv2	107 Client: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
26	6.657616	10.1.56.85	10.1.56.45	TCP	66 22 → 39432 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=3423812515 TSecr=3787186615
27	6.723209	10.1.56.85	10.1.56.45	SSHv2	107 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
28	6.723349	10.1.56.45	10.1.56.85	TCP	66 39432 → 22 [ACK] Seq=42 Ack=42 Win=29312 Len=0 TSval=3787186681 TSecr=3423812580
29	6.724113	10.1.56.45	10.1.56.85	SSHv2	1426 Client: Key Exchange Init

Figure 11: SSHv2 Traffic flow

The screenshot above shows client-server communication facilitated by SSH version 2. The python script makes use of the following parameter to identify the non-SSH traffic within the tunnel. Some of the most important ones are:

1. General flow statistics
2. Detection of cipher suite
3. Smallest possible packet size for cipher suite
4. Count of packets exchanged between the client and the server
5. Average lengths of client and server packets
6. Total ssh traffic data (in bytes) exchanges between the client and the server.

7. Average time between arrival of packets between the client and the server.

As seen from the experimental results obtained in the above discussion, it can be said the custom python script identifies the presence of non-SSH traffic in the SSH tunnel efficiently. This script can further be used to either block the traffic immediately if it has been initiated from the outside of the network administrator can be notified if some person from inside the network is trying to forward critical data outside the organization. The script sits at the firewall to the inside network, which acts as the filtering boundary for traffic to and fro the network. Thanks to the platform independency of the script, it can be incorporated on any type and device and customized as per requirement.

After the testing from research point of view, the evaluation is carried out by considering single client-server model for this research project, simultaneously the testing of integrated firewall can be carried out for multiple users and also for multiple SSH tunnels.

6.1 Evaluation of Research Question

Research Question: "How deep packet inspection technique can be used to detect and prevent malicious client-server communication traffic via Secure Shell (SSH) tunnel?"

After the evaluation of the results there are two possibilities
1. Packet with nested or crafted traffic inside SSH Tunnel is detected then it will detect the tunnel as a SSH tunnel detected.

2. Packet with genuine or authentic port 22 i.e. SSH traffic is detected as SSH tunnel not detected and further communication will establish.

6.2 Comparison between traditional firewall and firewall with integrated script

Sr. No	Traditional Firewall	Firewall with Integrated script
1	It will check only the port and IP based traffic.	It will check for port and IP based traffic as well as the payload i.e. header and footer of the packet.
2	It will inspect the traffic only at Layer4 in OSI model.	It will inspect the traffic up to application layer 7 in OSI model.
3	The nested traffic is undetectable and communication between client and server will not be detected.	The nested tunnel traffic is detected and client, server communication will gets detected.
4	There is no visibility of traffic in a network.	There is visibility of traffic in a network.

7 Conclusion and future work

The primary objective (deep packet inspection for intrusion detection) of the proposed solution has been achieved by implementing a python script to detect non-SSH traffic inside of an SSH tunnel. The solution is able to efficiently differentiate between normal SSH traffic and nested tunneling. The platform independence of the script means that it can be used across different types of operating systems. The script is also customizable according to the requirements of the organization. The solution can sit at the perimeter of the organizational network (the firewall) and block any masked attempt to access or breach critical data. Experimental results have shown that the implementation is successfully able to detect masked tunneling. This work is done on remote access (SSH protocol) to secure the tunnel, however, the same idea of scripts can be extended to other protocols such as DNS, HTTP, and HTTPs and so on. The script needs to continuously update for the features it uses to detect masked traffic because attackers keep evolving too. The script may be set up as a separate entity between any communicating networks with or without the presence of the firewall.

Acknowledge

I would like to extend my gratitude to my Supervisor Mr. Christos Grecos for his continuous guidance and motivation. Without your contribution, this project would not have consummated. I would also like to thank the School of Computing at the National College of Ireland for providing me with such an excellent opportunity to contribute to the body of knowledge. I would also like to thank my parents, colleagues, and friends for their support.

References

[1] I. Security, "Cost of a Data Breach Report 2019," Traverse City, Michigan 49686 USA, 2019.

[2] S. E. Security, "Symantec Enterprise Security Framework," in *Symantec Corporation*, Mountain View, CA 94043 USA, 2011.

[3] F. G. S. Maurizio Dusi, "A Preliminary Look at the Privacy of SSH Tunnels," in *IEEE*, St. Thomas, US Virgin Islands, USA, 3-7 Aug. 2008.

[4] M. Dusi, A. Este, F. Gringoli and L. Salgarelli, "Using GMM and SVM-Based Techniques for the Classification of SSH-Encrypted Traffic," in *IEEE*, Dresden, Germany, 14-18 June 2009.

[5] X. Tan, X. Su and Q. Qian, "The classification of SSH tunneled traffic using maximum likelihood classifier," in *IEEE*, Ningbo, China, 9-11 Sept. 2011.

[6] B. Qu, Z. Zhang, L. Guo and D. Meng, "On accuracy of early traffic classification," in *IEEE*, Xiamen, Fujian, China, 28-30 June 2012.

[7] M. Gebski, A. Penev and R. K. Wong, "Protocol Identification of Encrypted Network Traffic," in *IEEE*, Hong Kong, China, 18-22 Dec. 2006.

[8] M. Hirvonen and M. Sailio, "Two-phased method for identifying SSH encrypted application flows," in *IEEE*, Istanbul, Turkey, 4-8 July 2011.

[9] G. Mujtaba and D. J. Parish, "Detection of applications within encrypted tunnels using packet size distributions," in *IEEE*, London, UK, 9-12 Nov. 2009.

[10] S. Dharmapurikar, P. Krishnamurthy, T. Sproull and J. Lockwood, "Deep Packet Inspection using Parallel Bloom Filters," in *IEEE*, Stanford, CA, USA,, Jan.-Feb. 2004.

[11] P. Iyappan, K. S. Arvind, N. Geetha and S. Vanitha, "Pluggable Encryption Algorithm In Secure Shell(SSH) Protocol," in *IEEE*, Nagpur, India, 16-18 Dec. 2009.

[12] V. A. Foroushani and A. N. Zincir-Heywood, "Investigating application behavior in network traffic traces," in *IEEE*, Singapore, Singapore, 16-19 April 2013.

[13] Y. Imamverdiyev and L. Sukhostat, "Anomaly detection in network traffic using extreme learning machine," in *IEEE*, Baku, Azerbaijan, 12-14 Oct. 2016.

[14] B. Yang and D. Liu, "Research on Network Traffic Identification based on Machine Learning and Deep Packet Inspection," in *IEEE*, Chengdu, China, China, 15-17 March 2019.

[15] D. Smallwood and A. Vance, "Intrusion analysis with deep packet inspection: Increasing efficiency of packet based investigations," in *IEEE*, Hong Kong, China, 12-14 Dec. 2011.

[16] B. Yang and D. Liu, "Research on Network Traffic Identification based on Machine Learning and Deep Packet Inspection," in *IEEE*, Chengdu, China, China, 15-17 March 2019.

[17] S. Zamfir, T. Balan, F. Sandu and C. Costache, "Solutions for deep packet inspection in industrial communications," in *IEEE*, Bucharest, Romania, 9-10 June 2016.

[18] Z. M. Fadlullah, T. Taleb, N. Ansari, K. Hashimoto, Y. Miyake, Y. Nemoto and N. Kato, "Combating against attacks on encrypted protocols," in *IEEE*, Glasgow, UK, 24-28 June 2007.

[19] M. Al-hisnawi and M. Ahmadi, "Deep packet Inspection using Quotient Filter," in *IEEE*, Nov. 2016.

[20] N. K. Raja, K. Arulanandam and B. R. Rajeswari, "Two-Level Packet Inspection Using Sequential Differentiate Method," in *IEEE*, Cochin, Kerala, India, 9-11 Aug. 2012.

[21] F. A. Saputra, I. U. Nadhori and B. F. Barry, "Detecting and Blocking Onion Router Traffic Using Deep Packet Inspection," in *IEEE*, Denpasar, Indonesia, 29-30 Sept. 2016.

[22] N. Nwanze and D. Summerville, "Detection of anomalous network packets using lightweight stateless payload inspection," in *IEEE*, Montreal, Que, Canada, 14-17 Oct. 2008.