

# **Stealth Analysis of Network Topology using Spanning Tree Protocol**

Stephen Glackin

M.Sc. in Software  
and Network  
Security 2015





lyit

Institiúid  
Teicneolaíochta  
Leitir Ceanainn

Letterkenny  
Institute  
of Technology

Computing Department, Letterkenny Institute of Technology, Port Road, Letterkenny, Co.  
Donegal, Ireland.

# **Stealth Analysis of Network Topology using Spanning Tree Protocol**

Author: Stephen Glackin

Supervised by: John O' Raw

A thesis presented for the award of Master of Science  
in Software and Network Security.

Submitted to the Higher Quality and Qualifications Ireland (QQI)

Dearbhú Cáilíochta agus Cáilíochtaí Éireann

May 2015

## **Declaration**

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of Master of Science in Computing in Software and Network Security, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Signature of Candidate

Date

## **Acknowledgements**

I would like to thank my wife, kids, family members and extended family members for their support and encouragement, without them this project never would have been completed.

Also I would like to offer my greatest appreciation to my supervisor Mr. John O'Raw for all his help and support. His knowledge with regards to everything relating to computing is truly amazing and I am very grateful for the time he has given in assisting me to carry out this project.

## **Abstract**

Almost every network over the last 30 years is built on Spanning Tree Protocol (STP). This protocol makes topology information available to individual switches by exchanging Bridge Protocol Data Units (BPDUs) containing data fields which enable the Spanning Tree Algorithm (STA) to determine a hierarchy of switches on the network. A review of literature shows limited investigation into information leakage due to this protocol has been carried out since its first publication by the Digital Equipment Corporation (DEC) in 1985.

Scripts were developed using the Python Programming language accepting information from STP Packets with the aim of identifying the network topology of a Local Area Network (LAN) as well as information leakage from STP. Mitigation techniques for any information leakage discovered are discussed.

As a result of this project the viability of a security auditor using the developed scripts within a LAN in order to obtain a situation awareness of the network security perimeter of an organisation and assets within in this perimeter is also determined.

## Acronyms

<b>Acronym</b>	<b>Definition</b>
ANSI	American National Standards Institute
BPDU	Bridge Protocol Data Unit
BRCTL	Bridge Control
CWI	Centrum Wiskunde and Informatica
DEC	Digital Equipment Corporation
DHCP	Dynamic Host Configuration Protocol
DOS	Disk Operating System
DoS	Denial of Service
DSA	Digital Signature Algorithm
GUI	Graphical User Interface
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
IPS	Intrusion Protection System
LAN	Local Area Network
MAC	Media Access Control
MAN	Metropolitan Area Network
MIB	Management Information Base
MST	Multiple Spanning Tree Protocol
NAT	Network Address Translation
NIC	Network Interface Card
OS	Operating System
OSes	Operating Systems
OSI	Open Systems Interconnection
OUI	Organisational Unique Identifier
PC	Personal Computer

PVST+	Per-VLAN Spanning Tree
QoS	Quality of Service
RSTP	Rapid Spanning Tree Protocol
RPVST	Rapid -PVST
SCS	SSH Communications Security
SDN	Software Defined Networking
SNMP	Simple Network Management Protocol
SPB	Shortest Path Bridging
SSH	Secure Shell
STA	Spanning Tree Algorithm
STP	Spanning Tree Protocol
TCA	Topology Change Acknowledgement
TCN	Topology Change Notification
TCP	Transmission Configuration Protocol
TRILL	Transparent Interconnection of Lot of Links
UDP	User Datagram Protocol
VCSes	Version Control Systems
VLAN	Virtual Local Area Network
VSTP	VLAN Spanning Tree Protocol



# Table of Contents

<b>1. INTRODUCTION .....</b>	<b>1</b>
1.1 PROJECT SCOPE .....	3
<b>2. LITERATURE REVIEW .....</b>	<b>4</b>
2.1 SPANNING TREE PROTOCOL.....	4
2.1.1 Configuration BPDUs .....	5
2.1.2 Topology Change Notification (TCN) BPDUs.....	10
2.1.3 Recent Spanning Tree Developments.....	12
2.2 SPANNING TREE SECURITY .....	13
2.3 RAW SOCKETS .....	15
<b>3. TEST ENVIRONMENT.....</b>	<b>17</b>
3.1. ORACLE VM VIRTUALBOX.....	18
3.2. WIRESHARK.....	22
3.3. PYCHARM .....	23
3.4. SSH/OPENSASH/PUTTY .....	24
3.5. PUBLIC KEY AUTHENTICATION .....	25
3.6. LINUX SWITCH SET-UP .....	27
<b>4. SPANNING TREE PROTOCOL ANALYSIS .....</b>	<b>29</b>
4.1. SINGLE SWITCH SET-UP .....	30
4.2. TWO SWITCH SET-UP .....	32
4.3. THREE SWITCH SET-UP .....	35
4.4. FOUR SWITCH SET-UP .....	38
<b>5. SCRIPT DEVELOPMENT .....</b>	<b>40</b>
5.1. CODE DEVELOPMENT .....	40
5.2. APPLICATION TESTING .....	51
5.2.1. Test Network without TCNs and TCAs.....	51
5.2.2. Test network with TCNs and TCAs .....	53
<b>6. RECOMMENDATIONS /CONCLUSIONS .....</b>	<b>57</b>
6.1. INFORMATION LEAKAGE AND MITIGATION.....	57
6.2. DISCOVERY OF NETWORK TOPOLOGY .....	58
6.3. FUTURE WORK .....	59

# Appendices

Appendix A - configuration of network adapter using vboxmanage.exe.....	i
Appendix B - SSH set-up between administrative PC and network switch.....	iii
Appendix C - network switch configuration using BRCTL.....	viii
Appendix D - python script run on a stable network with no TCNs or TCAs.....	xvii
Appendix E - python script run on an unstable network with TCNs and TCAs.....	xxii

## Table of Figures

Figure 1 Configuration BPDU Parameters and Format(Institute of Electrical and Electronics Engineers <i>et al.</i> 2004, p. 62) .....	6
Figure 2 IEEE Default STP Costs(Menga 2004, para. 14) .....	7
Figure 3 Spanning Tree Algorithm(Menga 2004, para. 20) .....	8
Figure 4 Spanning Tree Port States (Menga 2004, para. 15) .....	9
Figure 5 Topology Change Notification BPDU Parameters and Format(Institute of Electrical and Electronics Engineers <i>et al.</i> 2004, p. 63) .....	10
Figure 6 Topology Change Notification working towards the root bridge (Molenaarin 2014, para. 37) .....	11
Figure 7 Topology Change Acknowledgement from Root Bridge(Molenaarin 2014, para. 42) .....	11
Figure 8 Initial Test Network Logical Diagram .....	17
Figure 9 Type 1 Bare-metal Hypervisor(Ribeiro 2009, para. 7) .....	18
Figure 10 Type 2 Hosted Hypervisor(Ribeiro 2009, para. 8) .....	19
Figure 11 Test Network Devices Set Up in Oracle VM VirtualBox.....	19
Figure 12 Eight Adapters shown in the *.vbox file for Switch1.....	21
Figure 13 Using VBoxmanage commands using the Windows Host's cmd .....	21
Figure 14 Initial login without public key authentication .....	25
Figure 15 Putty Set-Up.....	27
Figure 16 VLOOKUP used to lookup hexadecimal information received by packet analyser .....	29
Figure 17 Single Switch Set-Up .....	31
Figure 18 If you look at the topology change timer we see where the 34Secs, our timer here starts at 34.98Secs (max age + forward delay) .....	32
Figure 19 Two Switch Set-Up .....	33
Figure 20 Configuration BPDU (University of Virginia 2015, p. 14).....	34
Figure 21 Three Switch Set-Up.....	36
Figure 22 Four Switch Set-Up.....	39
Figure 23 Sample of Data obtained using the choice 1 in the application to Sniff Spanning Tree Packet .....	44
Figure 24 Topology Information obtained from Table1.....	52
Figure 25 Topology Information obtained from Table2.....	56

## Table of Code Listings

Code Listing 1 Raw Socket Creation (Moon, 2011) .....	40
Code Listing 2 Raw Socket used in Runner (Python Software Foundation, 2015) .....	41
Code Listing 3 Functions returning parsed Ethernet Header and Local Link information.....	42
Code Listing 4 Static toHex function used to parse from string to hex value (Kharechko, 2006) .....	42
Code Listing 5Function used to print Bpdu Fields to Screen .....	43
Code Listing 6 Printing STP Packet Data to Screen.....	43
Code Listing 7 Code to used to identify the Root Bridge .....	45
Code Listing 8 OUI Static Method .....	45
Code Listing 9 Identifying Bridges on the Network.....	46
Code Listing 10 Identification of a blocked/non-designated port.....	47
Code Listing 11 identifying other forwarding ports in the switch .....	48
Code Listing 12 Identification of network segment using Single TCN and TCA.....	49
Code Listing 13 Identification of Root Ports using multiple TCNs .....	50

## Table of Tables

Table 1 Application Test Results for a Stable Network .....	52
Table 2 Information obtained from unstable network .....	56

## 1. Introduction

Since its emergence in 1985 (Perlman, 1985) practically every network and network switch has utilised Spanning Tree Protocol (STP), for loop avoidance and enabling redundancy in a mesh network topology . It would be accurate to say that most enterprise networks in the world today use STP. When this protocol was initially developed network security was not as prevalent as it is today meaning that STP does not include the security features of a more recently developed protocol.

Every organisation has valuable IT assets such as computers, networks and data. Protecting those assets requires the organisation to conduct IT security audits in order to get a clear picture of network security risks and how to best deal with the threats they pose (IT Security, 2007).

In order to obtain an up-to-date situational awareness of the security perimeter and identify assets within that perimeter, security auditors have historically used open-source tools such as Nmap and Nessus (Webster, 2006). These tools use active monitoring techniques which inject test traffic onto a network and monitor its flow in order to identify network hosts (Sullivan, 2013). In their paper entitled, "Asset Tracking in Critical Power Communications Infrastructure using Passive Techniques" O'Raw et al. (2015) noted that active scans of this type could cause legacy equipment to crash, meaning that active scans are banned from certain critical infrastructure sites and alternative non-intrusive methods such as passive monitoring would be required in order to identify assets located within this type of network.

Instead of injecting artificial traffic into the network, passive monitoring entails monitoring traffic that is already on the network. Either specialised probes designed to capture network data or built-in capabilities on switches or other network devices can be used to capture network packets for analysis (Sullivan, 2013).

Previous studies have shown the use of various discovery protocols such as the Local Link Discovery Protocol (LLDP), Cisco Discovery Protocol (CDP) can be utilised along with the Simple Network Management Protocol (SNMP) in order to passively identify network

topology. SNMP is a Layer seven application OSI protocol which uses both management and agent software for network management in order to passively map network topology and carry out node characterisation, although It was noted that, “not all devices support SNMP” (Sans 2014, para. 30), which limits its effectiveness with regards to a networks which contain devices where SNMP is not supported. It could also be argued that the use of SNMP is not truly passive as its use requires the installation of agent and management software by a network administrator.

The author intends to establish if the long-standing STP which provides path redundancy while preventing undesirable loops in a network can be utilised by a security auditor as an alternative to SNMP for passively mapping the topology and security perimeter of the network via an administrative PC connected to a single switch interface.

"You can't protect assets simply by knowing what they are; you also have to understand how each individual asset is threatened"(IT Security 2007, para. 8). By passively monitoring the network the author also hopes to identify any vulnerability created by STP, while also providing mitigation techniques for a security professional.

## 1.1 Project Scope

The scope of the project is to investigate:

1. Information leakage by a passively monitored device using the standard IEEE 802.1D Spanning Tree Protocol(Institute of Electrical and Electronics Engineers *et al.*, 2004), while also illustrating vulnerabilities which may be utilised by a would be attacker.
2. Provision of a mitigation strategy for any vulnerability identified from information leakage.
3. The validity of using the standard IEEE 802.1D Spanning Tree Protocol(Institute of Electrical and Electronics Engineers *et al.*, 2004), to passively identify the network topology of a standard Local Area Network (LAN) by accessing a single switch interface located on the network ,while also determining if it can form the basis for a application which may be utilised by a security auditor for establishing a network's security perimeter.

As well as using a protocol analyser to examine the spanning tree packets, an application will be created using Raw Sockets in order to automatically abstract information at the Data Link Layer which can help in identifying network topology.

## **2. Literature Review**

Within the Literature Review STP is examined and described with emphasis on understanding the exchange of information between devices and examining current academic literature with regards to the current state of thinking in relation to STP security. It was noted that Denial of Service (DoS) attacks have been identified as the most prevalent vulnerability for STP with multiple methods for attacking the network being identified. Raw Sockets are used as a means by which an attacker can manipulate STP Packets in order to carry out a DoS attack, therefore a brief explanation with regards to Raw Sockets has also been given within this section.

### **2.1 Spanning Tree Protocol**

The Spanning Tree Protocol was originally developed by Radia Perlman for the Digital Equipment Corporation (DEC) in 1983 (Menga, 2004) and was first published in 1985 (Perlman, 1985) . The IEEE standard version of the protocol, 802.1D was published in 1990 and was superseded by RSTP 802.1W in the 2004 version (Iveson, 2013).

STP currently exists in the original Digital Equipment Corporation (DEC) implementation and the Institute of Electrical and Electronic Engineers (IEEE) standard which was developed from the latter and is almost exclusively used in networks today (Menga, 2004).

Prior to STP introduction any switch architecture which had a loop would end up with duplicate frames flooding between switches causing an outage; this is commonly referred to as a "broadcast storm". The Spanning Tree algorithm(STA) solved the problems associated with this physical loop by creating a logical blocking solution by stopping normal switch forwarding (Bahethi, 2014).

Once the root bridge has been selected, each non-root bridge determines the best path to reach the root bridge while blocking any other paths which introduce loops on the network.



In a converged STP, each port is either in a forwarding or blocking state. Ports which are considered the best path to the root bridge are placed into a forwarding state while all other ports are placed into a blocking state.

On their first initialisation, each switch generates a unique bridge ID used by spanning tree to uniquely identify the bridge. The bridge ID is a combination of the bridge MAC address plus a 2-byte bridge priority field, where the priority field can be altered by a network administrator to directly affect whether or not a bridge becomes the root bridge. The switch with the lowest bridge ID is the root bridge. When the bridge ID has been determined, this bridge begins to generate configuration Bridge Protocol Data Units (BPDUs) assuming that is the root bridge (Menga, 2004).

There are two types of BPDUs used in spanning tree, configuration BPDUs and Topology Change Notification (TCN) BPDUs. Configuration BPDUs are originated by the Root Bridge and flow outward along the active paths that radiate from the root bridge. TCN BPDUs flow upstream (towards the root bridge) to alert the root bridge that the active topology has changed (Rossi, 2000).

#### **2.1.1. Configuration BPDUs**

This type of BPDU is the main communication mechanism for STP and is used to determine the root bridge and whether or not a port should be forwarding or blocking state. The configuration BPDU has various fields which are used to indicate important parameters for the generation of the final STP by the Spanning Tree Algorithm (STA) (Menga, 2004).

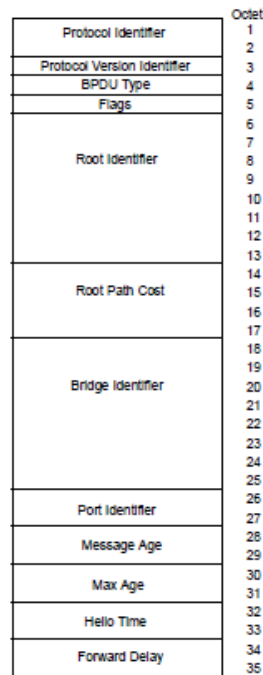


Figure 1 Configuration BPDU Parameters and Format(Institute of Electrical and Electronics Engineers *et al.* 2004, p. 62)

On receiving a configuration BPDU containing a lower root bridge ID, the bridge immediately considers this lower root bridge ID as the new root bridge and begins propagating the configuration BPDUs received from this bridge. Therefore in a network with multiple bridges, the bridge with the lowest bridge ID will eventually become the root bridge to all bridges.

Configuration BPDUs are generated by the root bridge only. Configuration BPDUs for non-root bridges are generated only when a configuration BPDU originated by the root bridge are received via the root port of the non-root bridge. The non-root bridge updates fields, such as root path cost, bridge ID and port ID in the received configuration BPDU and then propagates the updated configuration BPDU out all forwarding ports, except the root port upon which the BPDU was generated. This ensures that configuration BPDUs are propagated throughout the entire network.

After the root bridge has been selected each non-root bridge attempts to build a topology that forms the lowest-cost path to the root bridge. The lower this cost the more preferable the link. Depending on bandwidth each logical port has a default cost associated with it, as defined in the IEEE standard. This cost can be modified to influence root port selection.

Bandwidth (Mbps)	Cost
4	250
10	100
16	62
100	19
155	14
622	6
1000 (1 Gbps)	4
10000 (10 Gbps)	2

Figure 2 IEEE Default STP Costs(Menga 2004, para. 14)

Every port within STP transitions through several states upon port initialisation. Data from the user is only forwarded when a port is in the forwarding state. This approach is taken to prevent any loops from forming even for a short period, as a broadcast storm can bring down a network in seconds. There are five spanning tree port states, which are as follows:

### 1. Disabled State

A port is disabled when the port is down. This may be because the port has been administratively shut down or because of some issue with processing BPDUs. A port transitions from this state to the blocking state moving immediately to a listening state after it is initialised at the Layer 2 level.

### 2. Listening State

This is the phase where most of the important groundwork of generating a loop-free topology is carried out. Within this phase, spanning tree goes through the following processes:

#### i. Root Bridge Election

The bridge with the lowest bridge ID is selected as the root bridge.

#### ii. Root Port Selection

A single root port is selected from every non-root bridge providing the closest path to the root bridge.

#### iii. Designated bridge (port) selection for each segment

Each switch determines whether or not it represents the shortest path to the root bridge for each network segment attached to the non-root switch this excludes the network segment

attached to the root port. If this is the case, it configures itself as the designated bridge for the segment and configures the port as a designated port. Designated ports are placed into a forwarding state, while all other non-designated ports are placed into a blocking state. An exception to this configuration is if a non-designated port represents the root port on another switch. If this is the case, the root port on the other switch remains in a forwarding state, as well as remaining the designated port on the local switch.

Whether root bridge election or a root port selection, the same STA selection process is used for all decisions. Priority criteria are processed one by one, by comparing the configuration BPDUs received on a port with the configuration BPDUs that are sent out a port.

#### **Priority Criteria**

- 1 Select the lowest root bridge ID
- 2 Select the lowest root path cost
- 3 Select the lowest sender bridge ID
- 4 Select the lowest port priority

Figure 3 Spanning Tree Algorithm(Menga 2004, para. 20)

### *3. Learning State*

The bridge is accepting user data without forwarding it during the learning phase. The local MAC address table is populated on each bridge, so that once traffic is forwarded, the bridge does not need to flood a lot of traffic.

### *4. Forwarding State*

Depending on whether a port has been selected as either a root port or designated port, it is placed into the forwarding state. A port will remain in a forwarding state until a topology change occurs, where the port transitions to the listening phase and performs the appropriate selection processes.

### *5. Blocking State*

Where a port is found not to represent the shortest path to the root bridge it is placed in the blocking state. This is where the port is being blocked from sending or receiving any user data while still sending and receiving configuration BPDUs.

State	Description	User data being forwarded?
Disabled	The port is in a non-functional state, which might be due to a hardware failure or due to the port being administratively shut down.	No
Listening	The port is sending and receiving configuration BPDUs and is determining the root bridge and the role the port should take.	No
Learning	The switch is accepting user data on the port, but is not forwarding it, instead populating the bridge table with destination MAC address information. This ensures that the network is not suddenly flooded with unicast floods.	No
Forwarding	This state is transitioned to from the listening state. In this state, the port forwards all traffic. Only ports that represent the shortest path to the root bridge are placed into a forwarding state.	Yes
Blocking	The port is being blocked from sending or receiving any user data, but still sends and receives configuration BPDUs. A port is placed into the blocking state if it is determined to not represent the shortest path to the root bridge.	No

Figure 4 Spanning Tree Port States (Menga 2004, para. 15)

Timers determine how quickly or slowly a spanning-tree topology can react to a link or bridge failure converging to a new topology. There are three spanning-tree timers which are as follows:

1. Hello timer

The Interval between the generation of each configuration BPDU. The default is two seconds.

2. Max age timer

This timer controls the validity of a configuration BPDU after being received. The default is 20 seconds, meaning that if a configuration BPDU is not received within 20 seconds of the previous configuration BPDU, the previous configuration BPDU is no longer valid and a new root bridge must be selected.

3. Forward delay

This timer controls the amount of time spent by a bridge port in each of the listening and learning states before transitioning into a blocking to a forwarding state.

### 2.1.2. Topology Change Notification (TCN) BPDUs

Protocol Identifier	Octet 1
Protocol Version Identifier	2
BPDU Type	3
	4

Figure 5 Topology Change Notification BPDU Parameters and Format(Institute of Electrical and Electronics Engineers *et al.* 2004, p. 63)

A majority of BPDUs on a healthy network should be configuration BPDUs, although all bridged networks see at least a few of the second type of BPDUs, the Topology Change Notification (TCN) BPDUs. TCN BPDUs play a key role in handling changes in the active topology.

The TCN BPDUs are much simpler than the configuration BPDUs consisting of only three fields. TCN BPDUs are identical to the first three fields of a configuration BPDUs with the exception of a single bit in the type field, with the type field containing one of two hexadecimal values 0x00 (Binary: 0000 0000) indicating a configuration BPDUs or 0x80 (Binary: 1000 0000) indicating a TCN BPDUs.

Configuration BPDUs are only originated by the root bridge, but a TCN BPDUs will be generated by any switch in the network when either of two things happens:

1. A port has gone into forwarding state
2. A port has gone from forwarding or learning state into blocking state

"While the TCN BPDUs is important, it doesn't give the other switches a lot of detail. The TCN doesn't say exactly what happened, just that something happened"(Bryant 2015, para. 4).

When a bridge receives the topology TCN it will send a Topology Change Acknowledgement (TCA) on its designated port towards the downstream switch. It will create a TCN itself and send it on its root port as well.(Molenaarin, 2014)

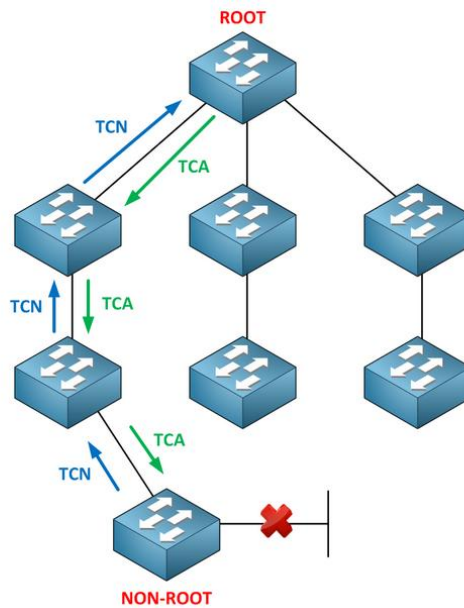


Figure 6 Topology Change Notification working towards the root bridge (Molenaarin 2014, para. 37)

When the root bridge receives the TCN, the root will also respond with an acknowledgement, taking the form of a configuration BPDU with the topology change bit/flag set. This indicates to all receiving bridges that the aging time for their MAC address tables should be changed from the default of 300 seconds/5 minutes to the forward delay spanning tree timer value (default 15 seconds).

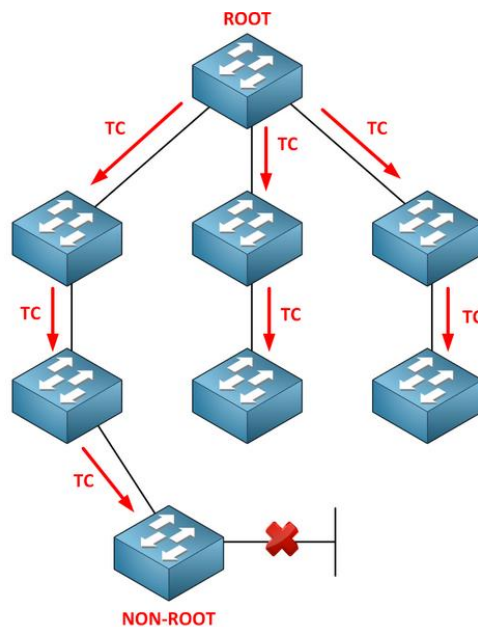


Figure 7 Topology Change Acknowledgement from Root Bridge(Molenaarin 2014, para. 42)

A Portfast enabled port changing STP state cannot result in the generation of a TCN BPDU. The most common usage of Portfast is when a single PC is connected directly to the bridge port, and since such a port going into forwarding state (or leaving it) doesn't impact STP operation, there's no need to alert the entire network about it. (Bryant, 2015)

### **2.1.3. Recent Spanning Tree Developments**

The IEEE has published a number of new specifications relating to STP, which include both Rapid Spanning Tree Protocol (RSTP) 802.1W and Multiple Spanning Tree (MST) 802.1S specifications.

The RSTP specification is the most significant development for spanning tree in recent times. RSTP has depreciated the 802.1D standard and redefines the states that bridge ports can be in, as well as how bridges detect failure and the associated convergence time. RSTP aims to reduce convergence times and also includes standards-based implementations of PortFast, UplinkFast, and BackboneFast.

The MST specification relates to how spanning tree interacts with topologies that include multiple Virtual Local Area Networks (VLANs). Modes of operation can be defined on Cisco Catalyst switches, which determine how the switch maintains STP for multiple VLANs. The following lists the common STP modes of operation:

#### **1. Common Spanning Tree (CST) 802.1Q**

This standard dictated that a single spanning-tree instance should be used for multiple VLANs. The reason for defining CST is to ensure interoperability with non-802.1Q bridges, as all STP communication is sent untagged on the native VLAN. By having only one spanning-tree instance each switch CPU needs to deal only with a single STP instance, although this has the drawback of being unable to use implement load sharing.

#### **2. Cisco Per-VLAN spanning tree (PVST+)**

This is a proprietary standard developed by Cisco, which allows multiple STP instances to operate in a Layer 2 network, while also allowing load sharing. This standard operates a unique STP instance per VLAN. Although allowing the load sharing, the implementation is



flawed as a single STP instance is required for each VLAN, even if VLANs share the same STP topology. This has a detrimental effect on CPUs in network environments which could support hundreds or thousands of VLANs.

### 3. Multiple Spanning Tree (MST)

MST is a combination of both 802.1Q and PVST+. It allows the user to map a configurable number of VLANs to a single STP instance, meaning that all VLANs that share the same STP topology can be supported by just one STP instance. Load sharing is achieved by having multiple STP instances, but the number of STP instances that must be maintained on each switch can be matched to the number of different logical topologies required for the network to implement load sharing (Menga, 2004).

## 2.2 Spanning Tree Security

Although STP is widely used in switching networks today, it was not until recently that its security performance has been studied by researchers. (Yeung *et al.*, 2006)

Network Layer 2 protocols such as STP have traditionally been considered trusted, in part because the local area networks (LANs) that they support are under the physical control of an organisation within a contained area. As a result Layer 2 infrastructure is not usually monitored unless there are connectivity issues. However, this assumption has been invalidated with the rise of the use of Layer 2 protocols over wide areas such as Metropolitan Area Networks (MANs), with attacks focused on the data link layer becoming more feasible and with the rise in insider threats within an organisation. Traditional Intrusion Detection Systems (IDSs) which usually operate at layer 3 or above on the TCP/IP stack have limited capabilities in dealing with threats which may occur at Layer 2 (Marro Mario, 2003).

In his paper entitled "Attacks at the Data Link Layer" Guillermo Marro Mario (2003) discusses characteristics of the STP/RSTP protocol which render it vulnerable to a Denial of Service (DoS) attacks by individuals with physical access to a network. DoS attacks exploit weaknesses in protocols and services by exhausting resources, causing service disruption or Quality of Service (QoS) degradation. Its main goal is to affect availability of the targeted service. The characteristics identified in this paper included:

1. Lack of authentication in BPDU messages
2. STP's slow convergence
3. Root role not fully monitored
4. More complex state machine in RSTP

Within this paper the following Flooding and Topology engagement attacks were listed which could be implemented on the network as a result of the previously identified vulnerabilities:

1. Flood of Configuration Message BPDUs with TC flag
2. Flood of Topology Change Notification BPDUs
3. Single-homed Root Role Claiming
4. Dual-homed Root Role Claiming
5. Internal Node Role Claiming
6. Tree Segmentation

In a paper entitled "Improving Network Infrastructure Security by Partitioning Networks Running Spanning Tree Protocol" Yeung et al. (2006) , discusses the current ROOT Guard and BPDU Guard techniques which have been developed by Cisco and there successful prevention of root role claiming attacks launched on a network. It was noted that these techniques don't stop all the attack vectors previously identified by Guillermo Marro Mario (2003).

Yeung et al. (2006) propose mitigation for all attack vectors associated with STP security by utilising boundary switches to partition a STP network into a hierarchy of switching domains, a Network Infrastructure (NI) which connects network infrastructure and a Non-Network Infrastructure (NNI) which connects to network hosts. The partitioning hides the detail of STP operation among domains and successfully avoids all STP attacks launched from the non-network infrastructure.

Rai et al. (2011) highlight the fact that Yeung et al.'s mitigation technique requires the implementation of specially designed switches, running a modified STP in a paper entitled, "

Exploit Detection Techniques for STP using Distributed IDS", while also proposing their own mitigation technique which introduces a cover based distributed Intrusion Detection Systems (IDSs) installed on network switches strategically placed on STP domains which enable the detection of all possible attacks on STP. It is stated that experimental results show that all possible attacks can be detected through this proposed scheme (Rai *et al.*, 2011).

In their paper entitled "Trust-Based Security for the Spanning Tree Protocol", Lai et al. (2014) noted a number of deficiencies in Rai et al.'s design ; firstly it is noted that this design could not identify root take-over attacks launched from the inside network, secondly it was noted that a new switch that connected to a peer switch and claimed to be the root role would be falsely considered an attacker and finally it was noted that the IDS deployment used also needed to be changed as the network topology changes.

Lai et al. (2014, p.1338), "propose a trust-based spanning tree protocol aiming at achieving a higher credibility of LAN switch with a simple and lightweight authentication mechanism." The authors concluded that their technique gave "comprehensive protection to the threats from both outside and inside the switches"(Lai *et al.* 2014, p.1343), with experiments showing that their improved protocol could effectively avoid the root take-over attack, flooding attacks, and other attacks.

## **2.3 Raw Sockets**

Data passes from one device to another by descending the TCP/IP stack on the originating machine and ascending the stack on the receiving machine. The complexity of this process is hidden from programmers in order to simplify the development of network capable software by sockets.

"A socket is a connection from one device to another. It can be viewed as a 'pipe' that is plugged into both devices. Data is put into the 'pipe' at one end and arrives at the other end without the programmers needing to consider how this is actually achieved" (Menzies 2002, p. 5).Socket end points are defined as exclusive combinations of port number and network

address, making it possible for a machine with a single network address to host several connections by via different port numbers.

Cooked sockets are where the data processing necessary to support the underlying network protocols is performed by the operating system. They greatly reduce the effort required to write network applications and are generally used by most application programmers.

"A raw socket is a socket that takes packets, bypasses the normal TCP/IP processing and sends them to the application that wants them" (Alder 2002, para.2). Where an application needs to programmatically control the details of the TCP and/or IP layers a programmer needs to be able both read and write the TCP and IP header information utilising raw sockets.

When using a raw socket the operating system does not perform the processing necessary to maintain the TCP and IP headers, these details are determined by the program that is making use of the raw socket. This introduces a higher level of complexity for the programmer and requires detailed knowledge of the protocols used, while also requiring much more code to be written. But it also provides a very high degree of flexibility as each packet can be crafted exactly as the developer desires (Menzies, 2002).

### 3. Test Environment

The Test Environment required for this project is strictly based on virtual devices composed of virtual machines running different Linux Operating Systems using the open -source Oracle VM Virtualbox Graphical User Interface Version 4.3.24r98716. The project uses both CentOS 6.6 Minimal Install (kernel 2.6.32-504.8.1.el6x86\_64) and Ubuntu 12.04 LTS Operating Systems (kernel 3.8.0-44-generic i686) with both being used within an internal network and connected to the internet using a single Network Address Translation (NAT) Interface for machine updates and downloading any applications and files which may be required. The host machine used for the project has a Microsoft Windows 7 Professional Operating System Version 6.1.7601 Service Pack1 Build 7601 with 8073 MB RAM installed.

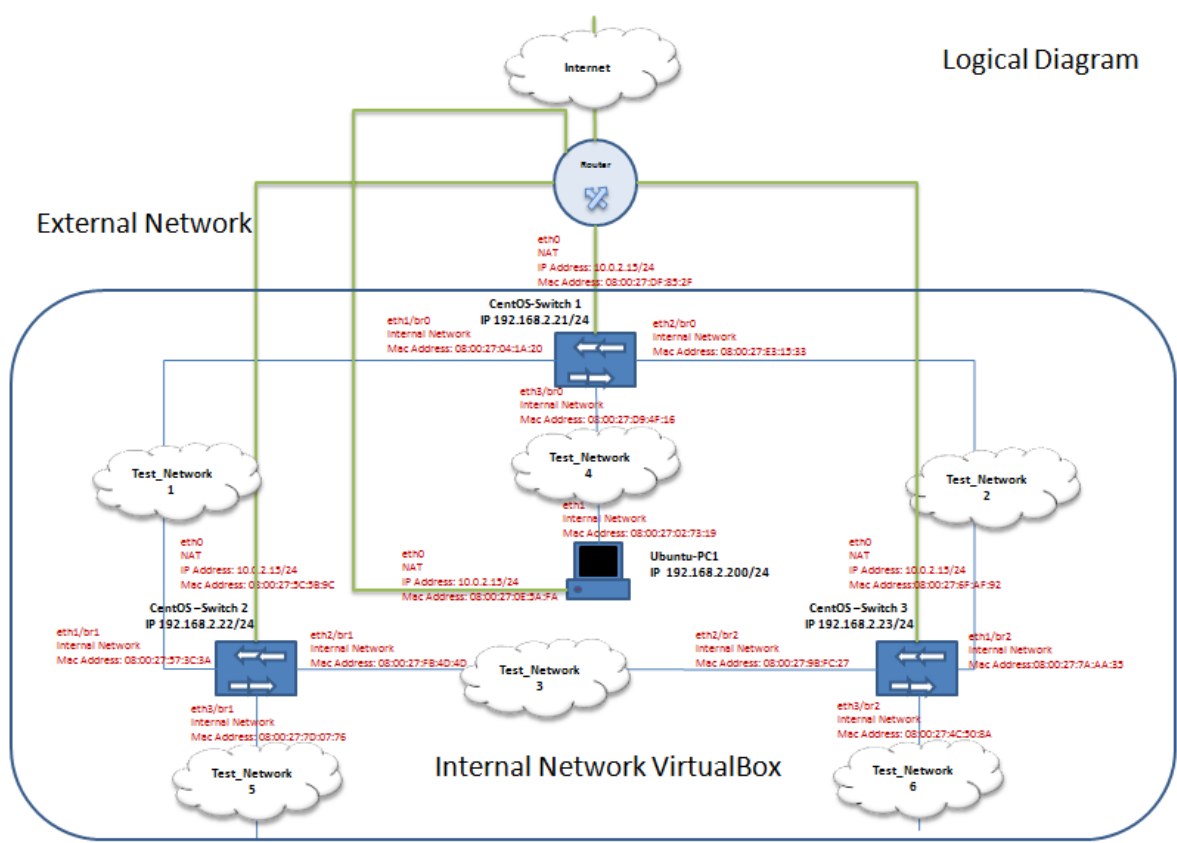


Figure 8 Initial Test Network Logical Diagram

### 3.1. Oracle VM VirtualBox

A virtualisation hypervisor comes in one of two forms: a bare-metal hypervisor or hosted hypervisor, also known as Type1 and Type 2.

Currently the most popular bare-metal Type 1 virtualisation hypervisors are:

1. VMware ESX and ESXi
2. Microsoft Hyper-V
3. Citrix Systems XenServer (Siebert 2011, para. 5)

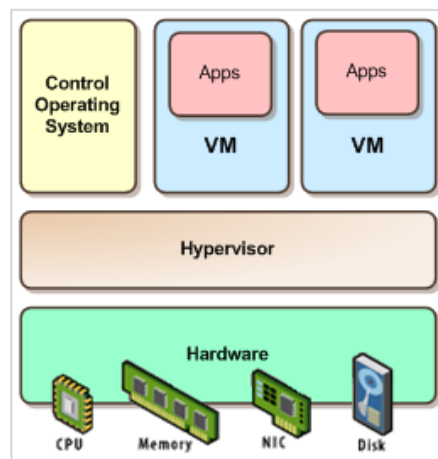


Figure 9 Type 1 Bare-metal Hypervisor(Ribeiro 2009, para. 7)

Currently the most popular hosted Type 2 virtualisation hypervisors are:

1. VMware Workstation, Server, Player and Fusion
2. Oracle VM VirtualBox
3. Microsoft Virtual PC
4. Parallels Desktop (Siebert 2011, para. 8)

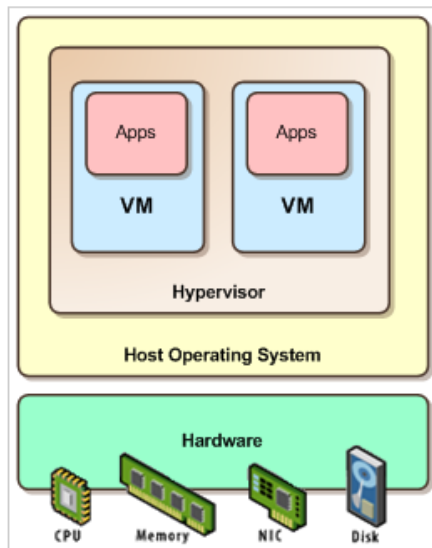


Figure 10 Type 2 Hosted Hypervisor(Ribeiro 2009, para. 8)

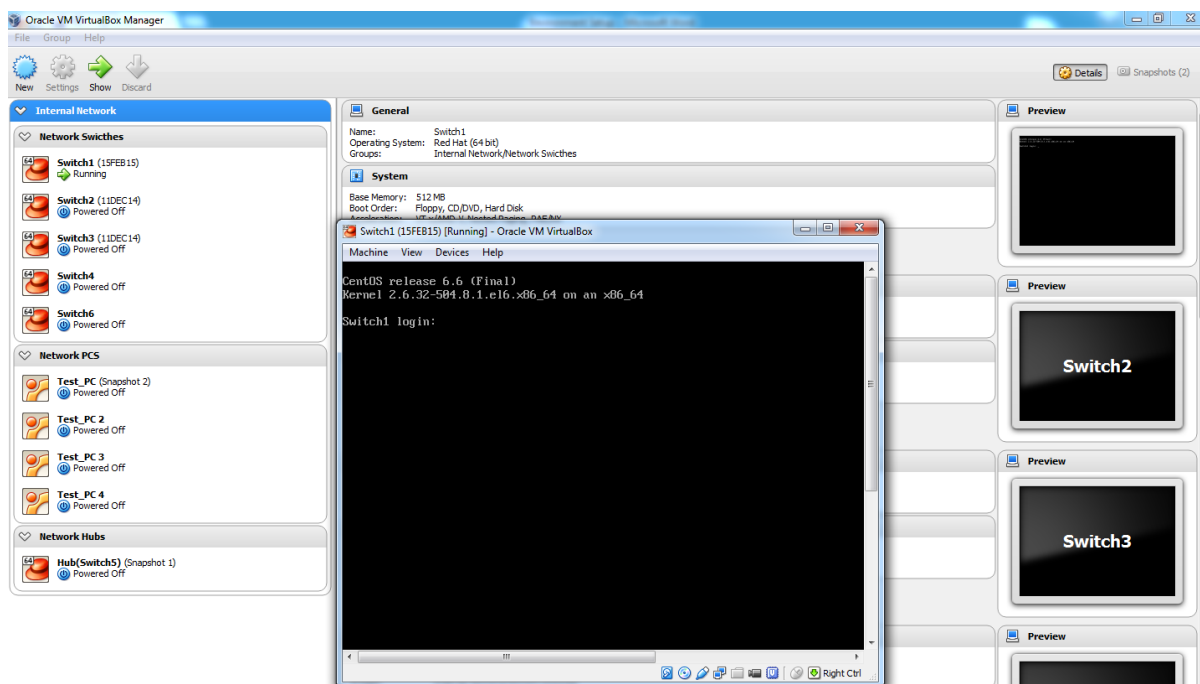


Figure 11 Test Network Devices Set Up in Oracle VM VirtualBox

For this project the open-source Oracle VM VirtualBox hypervisor is used to create the internal network for the development and testing of application software for the identification of a network topology using the STP.

VirtualBox was initially developed and released by Innotek GmbH in January 2007; Sun Microsystems acquired Innotek in February 2008, with the Oracle Corporation acquiring Sun

Microsystems in January 2010 and re-branding the product as "Oracle VM VirtualBox"(Tripathi, 2013).

The Oracle VM VirtualBox application installs on the host computer, extending its capabilities so that it can run multiple operating systems via virtual machines at the same time. Multiple virtual machines can be installed and run with the only practical limitations being the disk space and memory of the host machine.

Although Virtualbox supports up to eight network adapters. When a user looks into the \*.vbox file for the virtual machine they will see that eight network adapters have already been preconfigured but disabled. Inside the VirtualBox GUI the user is only able to configure up to four network adapters. In order for a user to configure the remaining network adapters the command line interface for VirtualBox , VBoxManage.exe is required (Neubert, 2013).

In order to set the path for VBoxManage.exe on the Windows host machine, the location of VBoxManage.exe file was found on the VirtualBox installation folder, located at C:\Program Files\Oracle\VirtualBox (Ang, 2012). Commands are available depending on the mode required for the network adapter(Neubert, 2013) (see Appendix A).



```

<Adapter slot="0" enabled="true" MACAddress="080027DF852F"
cable="true" speed="0" type="82540EM">

<Adapter slot="1" enabled="true" MACAddress="080027041A20"
cable="true" speed="0" type="82540EM">

<Adapter slot="2" enabled="true" MACAddress="080027E31533"
cable="true" speed="0" type="82540EM">

<Adapter slot="3" enabled="true" MACAddress="080027D94F16"
cable="true" speed="0" type="82540EM">

<Adapter slot="4" enabled="false" MACAddress="0800279B129B"
cable="true" speed="0" type="82540EM">

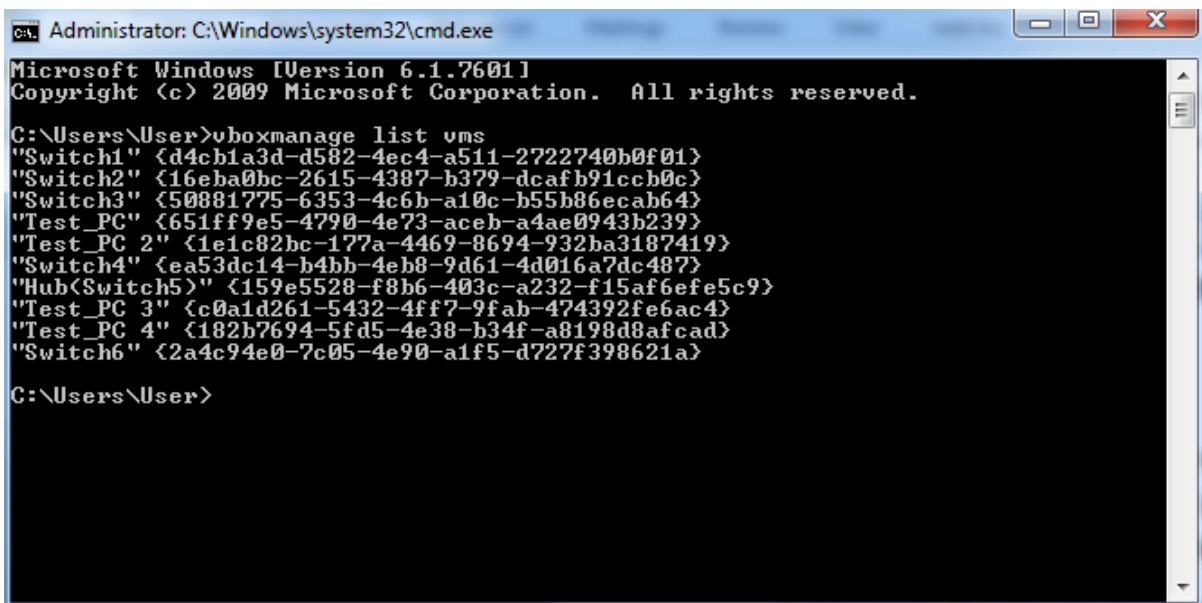
Adapter slot="5" enabled="false" MACAddress="08002781BF09"
cable="true" speed="0" type="82540EM">

<Adapter slot="6" enabled="false" MACAddress="080027D4E110"
cable="true" speed="0" type="82540EM">

<Adapter slot="7" enabled="false" MACAddress="0800274F598B"
cable="true" speed="0" type="82540EM">

```

Figure 12 Eight Adapters shown in the \*.vbox file for Switch1



```

Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\User>vboxmanage list vms
"Switch1" {d4cb1a3d-d582-4ec4-a511-2722740b0f01}
"Switch2" {16eba0bc-2615-4387-b379-dcafb91ccb0c}
"Switch3" {50881775-6353-4c6b-a10c-b55b86ecab64}
"Test_PC 2" {651ff9e5-4790-4e73-aceb-a4ae0943b239}
"Test_PC 2" {1e1c82bc-177a-4469-8694-932ba3187419}
"Switch4" {ea53dc14-b4bb-4eb8-9d61-4d016a7dc487}
"Hub(Switch5)" {159e5528-f8b6-403c-a232-f15af6efe5c9}
"Test_PC 3" {c0a1d261-5432-4ff7-9fab-474392fe6ac4}
"Test_PC 4" {182b7694-5fd5-4e38-b34f-a8198d8afcad}
"Switch6" {2a4c94e0-7c05-4e90-a1f5-d727f398621a}

C:\Users\User>

```

Figure 13 Using VBoxmanage commands using the Windows Host's cmd

### 3.2. Wireshark

Wireshark was developed as a tool for tracking down network problems and learning more about networking. It was originally named Ethereal and was initially released in July 1998, being re-named Wireshark in 2006.

Wireshark is an open source network packet analyser which attempts to capture network packets and display detailed packet data. Wireshark can be used for the following:

1. troubleshooting network problems
2. examining security problems
3. debugging protocol implementations
4. as a learning aid for network protocol internals(Lamping *et al.*, 2014)

It is important to note that while network analysis can be used to improve network performance and security it can also be used for malicious tasks. For example if password and username information were unencrypted it could be captured by a malicious user in order to compromise accounts. By learning network configuration information and listening to the traffic on the network an intruder can also exploit network vulnerabilities. Also malicious programs may include network analysis capabilities in order to sniff traffic on the network.

For these reasons companies should define specific policies regarding the use of a network analyser by stating, who can use it on the network and the "how, when and where" it may be used. These policies should be well known throughout the company. When using a consultant for performing network analysis services, a company should add a "network analysis" clause to any non-disclosure agreement they may have (Chappell, 2010).

For this project Wireshark was installed on each of the virtual Ubuntu PCs. STP packets generated by the virtual CentOS Switches on the virtual network were analysed using the Wireshark application. It is important to note that in order for Wireshark to capture and display packet information it must be run with root privileges.

### 3.3. Pycharm

Unlike Eclipse which is generally recognised as the industrial standard tool for Java Development, no equivalent tool could be identified with regards to Python. Peer reviews of Integrated Development Environments (IDEs) for Python, indicate that PyCharm compares very favourably with the other python IDEs currently available to developers. While IDEs such as PyScripter and Exedore are platform dependant or others such as Komodo and WingIDE are commercial applications, Pycharm is both platform independent and available as an open-source application. Also unlike PyDev which is an Eclipse plugin PyCharm is a standalone application which simplifies its initial installation and use for python code development.

The beta version of the PyCharm was released in July 2010 by the Czech company JetBrains(Taft, 2010) . PyCharm is available as both a Professional Proprietary Edition and an open source Community Edition which was released under the Apache License on 22 October 2013.(Jemerov, 2013)

For this project the Pycharm IDE Community Edition was installed on each of the virtual Ubuntu PCs. Pycharm was used to develop the python application for the project using raw sockets in order to access STP packets with the aim of establishing network topology. It is important to note that in order for Pycharm to create a raw socket and capture and display packet information, it must be run with root privileges.

Python was first created by Guido von Rossum in February 1991 while working for Centrum Wiskunde and Informatica (CWI) in Amsterdam, Netherlands. It is named after the Brit-com Monty Python's Flying Circus. "Many of Python's features originated from an interpreted language called ABC"(University of Michigan 1997, para. 1) which was also developed at CWI. It has since been developed by a large team of volunteers and is freely available from the Python Software Foundation(Lukaszewski, 2015).

"Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes "(University of Michigan 1997, para. 3). There are two recommended production-

ready versions of python at this point in time, because at the moment there are two branches of stable releases: 2.x and 3.x.

Python 2.7 has been used as the programming language for this project as there is currently more third party software available for Python 2 than for Python 3. It is important to note that Python 2 code will generally not run unchanged in Python 3.

### **3.4. SSH/OpenSSH/Putty**

Tatu Ylönen, a researcher at Helsinki University of Technology, Finland, designed the first version of the Secure Shell (SSH) protocol (SSH1) prompted by a password-sniffing attack on the university network. The protocol's goal was to replace the earlier rlogin, TELNET and rsh protocols, which did not provide strong authentication nor guarantee confidentiality. The first implementation of SSH was released as freeware in July 1995.

In December 1995, Ylönen founded SSH Communications Security (SCS) to market and develop SSH. The original version of the SSH software used various pieces of free software, such as GNU libgmp, but later versions released by SSH Communications Security evolved into increasingly proprietary software. The "SSH Secure Shell" (SSH2) was released by SCS as a commercial product in 1998 containing significant improvements in security, performance, and portability from the original protocol , although its license has been broadened to permit free use to the Linux, NetBSD, FreeBSD, and OpenBSD operating systems since its initial release.(Barrett, 2005)

OpenSSH provides the same functionality as SSH2 without conflicting with any intellectual property restrictions(Chapple, 2005). OpenSSH was created by the OpenBSD team as an alternative to the original SSH software developed by Tatu Ylönen. OpenSSH first appeared in OpenBSD 2.6 and the first portable release was made in October 1999.

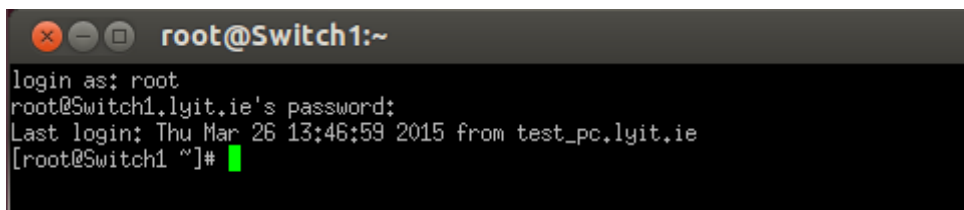
PuTTY was developed by Simon Tatham and was initially released in November 1998 (Tatham, 2011) , and it has been a usable SSH2 client since October 2000. While OpenSSH is probably the most used implementation of SSH in the world, PuTTY is likely the most used SSH client for the Microsoft Windows platform. Like OpenSSH, PuTTY is a very versatile tool for remote access to another computer.

Apart from being a SSH Client PuTTY also supports all the following protocols:

1. raw: The raw protocol is normally used for network debugging.
2. rlogin: This is an unencrypted UNIX remote login protocol that uses port 513 by default.
3. serial: The serial option is used to connect to a serial line. The most common purpose for this is to establish a serial connection between computers in lieu of an Ethernet or other network connection.
4. SSH: As already noted, SSH is an encrypted secure remote login protocol, which uses port 22 by default.
5. Telnet: Like rlogin, Telnet (telecommunication network) is an unencrypted remote login protocol. It typically uses port 23 and is available on many systems other than UNIX. Like rlogin, Telnet has waned in popularity due to privacy concerns.

In addition to these five protocols, PuTTY also supports features such as saved session configurations, session logging, locale (language) settings, and proxy sessions (Perrin, 2008).

### 3.5. Public Key Authentication



```
root@Switch1:~
login as: root
root@Switch1.lyit.ie's password:
Last login: Thu Mar 26 13:46:59 2015 from test_pc.lyit.ie
[root@Switch1 ~]#
```

Figure 14 Initial login without public key authentication

Unlike conventional password authentication where an individual is authenticated by providing the correct password, public key authentication is an alternative more secure and flexible means of identifying oneself to a switch.

For public key authentication key pair is generated, consisting of a public key (known to everyone) and a private key (kept secret). The private key is able to generate signatures. Signatures created using the private key cannot be forged by anybody who does not have the private key; but anybody who has the public key can verify that a particular signature is genuine.

Therefore when a key pair is generated on computer and the public key is copied to the switch, when the switch asks for the SSH client to authenticate itself, the SSH client can

generate a signature using its private key. The switch can verify that signature and allow the SSH client to log in.

A problem with this is that if a private key is stored unprotected on SSH client machine, then anybody who gains access to that machine will be able to generate signatures as if they were the SSH client. In order to eliminate this problem the private key is usually encrypted when it is stored on a local machine, using a passphrase of client's choice. In order to generate a signature, the SSH client must decrypt the key, by typing the passphrase.

This makes public-key authentication less convenient than password authentication as every time switch is accessed, instead of typing a short password a longer passphrase is required. The solution to this is use an authentication agent, a separate program which holds decrypted private keys and generates signatures on request. PuTTY's authentication agent is called Pageant.

When the user begins a Windows or Linux session, Pageant is started and private key is loaded into it by typing the passphrase once. For the rest of the session, PuTTY can be started any number of times and Pageant will automatically generate signatures. When the Windows or Linux session is closed, Pageant shuts down, without ever having stored the decrypted private key on disk.

Currently the most commonly used public-key algorithm available is RSA which was invented by Ron Rivest, Adi Shamir and Leonard Adleman. Another algorithm which may be used for public key encryption is the Digital Signature Algorithm (DSA) which is a Federal Information Processing Standard (Tatham, 2007).

For this project a SSH password-less automatic login from all Test\_PCs to Switches (see Appendix B). Where a single switch can easily be accessed using OpenSSH through the terminal, PuTTY proved particularly useful for accessing and saving multiple sessions to the various switches on the network.

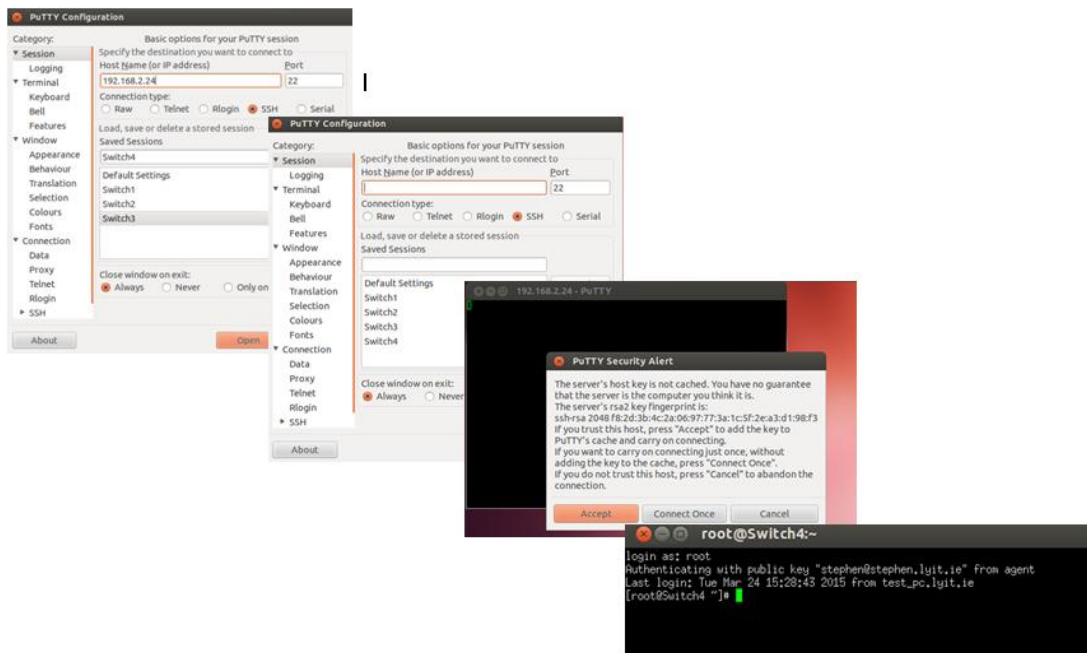


Figure 15 PuTTY Set-Up

### 3.6. Linux Switch Set-Up

A bridge is used to connect two Ethernet segments together in a protocol independent way, while a switch is a multi-port bridge connecting more than two Ethernet segments together. Unlike a router, bridges and switches forward packets based on Ethernet addresses, rather than IP addresses, meaning that all protocols can go transparently through a bridge. Within this paper the term “switch” or “bridge” may be interchangeable in their use when describing a single device used in order to connect Ethernet segments on the network.

Bridge Control (BRCTL) implements a subset of the ANSI/IEEE 802.1D standard and was initially introduced in Linux 2.2. It should be noted that the code for bridging has now been integrated into 2.4 and 2.6 kernel series.

Possible future enhancements for BRCTL include:

1. Document STP filtering
2. Netlink interface to control bridges (prototype in 2.6.18)
3. STP should be in user space
4. Support RSTP 802.1W and other 802.1D STP extensions(The Linux Foundation, 2009)

BRCTL is used to set up, maintain, and inspect the Ethernet bridge configuration in the Linux Kernel. To utilise the BRCTL function the user must be running as root or under sudo privileges.

The procedure used in order to setup a switch on the network using BRCTL is detailed in Appendix C. Before starting it was important to make sure all network interfaces are set up and working properly. The IP addresses for the switches need to be set after the switch has been configured, with no IP addresses being set for the individual interfaces. Dynamic Host Configuration Protocol (DHCP) start-up scripts should also be disabled for each interface.

Linux has a powerful tool which can mirror traffic known as the traffic control subsystem."Since it's a generic framework, its capabilities (including mirroring) are not limited to bridges; this means traffic can be mirrored for any interface(s) and sent to another interface(s), regardless of whether they are physical, virtual, part of a bridge or not" (Backreference.org 2014, para. 29 ). A mirrored port was also added to each switch in order for Test PCs to obtain STP packet information from all of the switch adapters which is also detailed as part of the switch setup in Appendix C.



## 4. Spanning Tree Protocol Analysis

For the purpose of STP analysis the internal network topology is gradually added to, initially containing a single switch and branching out into multiple switches on this network. This was done in order to gain a clear understanding of the protocol itself and in order to identify signatures which may identify network topology.

After the test network had been created STP packets were retrieved and analysed with the aid of Wireshark and Microsoft Excel. Within Microsoft Excel the VLOOKUP function (French, 2015) was initially used in order to make the retrieved packet information more readable by displaying the device name, port numbers and flags as plain text relative to the hexadecimal values obtained using the packet analyser.

Please note that as the first three bytes of a MAC Address (which is 08:00:27 in the case of the test network) are used as an Organisational Unique Identifier (OUI) for network interfaces and the final three bytes have been used in the results and diagrams to identify interfaces on the internal test network.

The screenshot shows a Microsoft Excel spreadsheet titled "STP Breakdown - Microsoft Excel". The formula bar contains the formula: `=VLOOKUP(CONCATENATE(C3,D3,E3,F3,G3,H3),Lookup!A:B,2,FAI`. The spreadsheet displays a table with columns for various STP packet fields. The table has 48 rows of data, with the first row (row 2) serving as the header. The columns are: No., Time(see), Destination Mac Address, Device lookup, Source Mac Address, Device lookup, Length, DSAP, SSAP, Control Field, Protocol ID, Version ID, Version, BPDU Type, BPDU lookup, Flags, Flag lookup, Root Bridge, Device lookup, and Root Path Cos. The data rows contain hexadecimal values for most fields, with some cells containing text like "Topology Change" or "Config & TCN". The bottom of the screenshot shows the Windows taskbar with several open applications and the system clock showing 12:27 on 29/03/2015.

Figure 16 VLOOKUP used to lookup hexadecimal information received by packet analyser

## 4.1. Single Switch Set-up

Based on the information obtained with regards STP a single switch should transition directly from its initial blocking state through both listening and learning states within 30 seconds depending if the default forward delay value is being used, all ports should then be placed in a forwarding state sending user data to and from the various nodes contained on the network. The root ID and bridge ID of each configuration BPDU should be equal and the BPDU Cost value should be zero as this is the root switch on the network. After all switch ports have transitioned into the forwarding state, configuration BPDUs propagated from the root switch should indicate a change of network topology by having their topology change bit/flag set.

On the preliminary set up of Switch1, a number of the switch ports were incorrectly connected to the same LAN segment. It was noted on start-up that within 0.0000224 seconds Switch1 continued forwarding packets from the interface 1 only (04:1a:20), which is the lowest interface MAC address on this device. The original 802.1D Spanning Tree Specification had anticipated the possibility of the root bridge having more than one port on the same LAN segment, and in that case, the port with the lowest port ID would become the designated port for that LAN segment, and put into forwarding mode sending traffic away from the root, while its other ports on that same LAN segment became non-designated ports put were immediately put into blocking state (Solie, 2002).

When the virtual network topology had been corrected and a SPAN/mirrored/monitor port had been created (Backreference.org, 2014) , packets were once again retrieved from a single switch running on the network. This was done for each switch separately to ensure that there could be a correlation between the results obtained.

## Logical Diagram-Internal Network VirtualBox

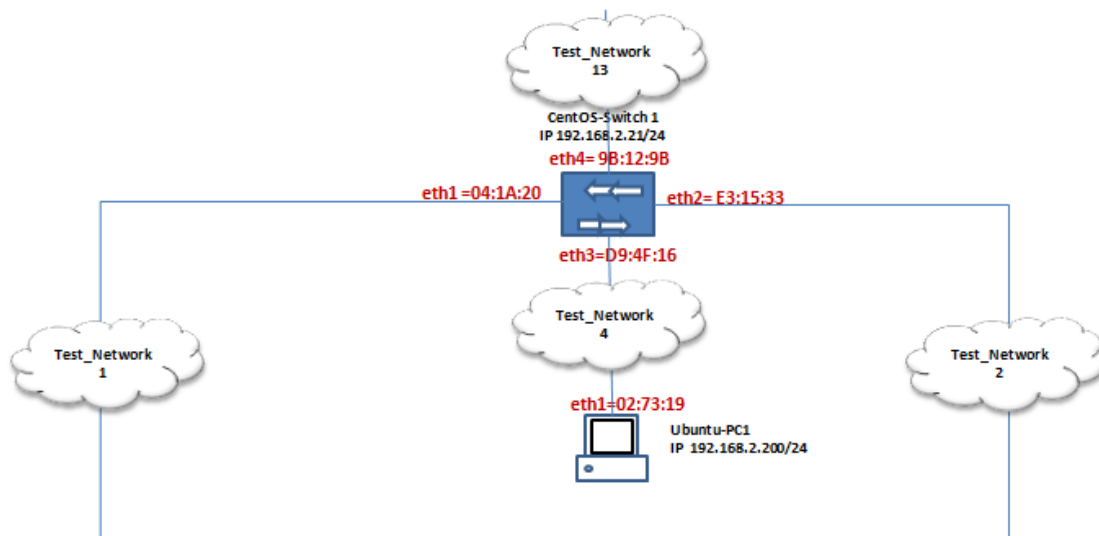


Figure 17 Single Switch Set-Up

It was noted that each switch immediately started to send configuration BPDUs through all ports on start-up although after a period of approx 30 seconds (listening and learning States = 2 x forward delay) ,with the switches ports entering the forwarding state the topology change bit/flag was set on all the configuration BPDUs from the root switch for the next 35 seconds ,a default combination of max age and forward delay, which originates from the root switch and normally signals a topology change situation and to all switches on the network allowing them to reduce their MAC table aging time to forward delay (Cisco.com, 2005).

Once the initial listening and learning states and TCA from the each switch had been completed, each switch continued to send configuration BPDUs through all designated ports.

```

[root@Switch1 ~]# brctl showstp br0
br0
bridge id          8000.080027041a20
designated root    8000.080027041a20
root port         0
max age           19.99
hello time        1.99
forward delay     14.99
ageing time       299.95
hello timer       1.30
topology change timer 0.00
hash elasticity   4
mc last member count 2
mc router        1
mc last member timer 0.99
mc querier timer 254.96
mc response interval 9.99
flags
path cost         0
bridge max age   19.99
bridge hello time 1.99
bridge forward delay 14.99
tcn timer        0.00
gc timer         4.30
hash max         512
mc init query count 2
mc snooping      1
mc membership timer 259.96
mc query interval 124.98
mc init query interval 31.24

```

```

[root@Switch1 ~]# brctl showstp br0
br0
bridge id          8000.080027041a20
designated root    8000.080027041a20
root port         0
max age           19.99
hello time        1.99
forward delay     14.99
ageing time       299.95
hello timer       1.49
topology change timer 34.17
hash elasticity   4
mc last member count 2
mc router        1
mc last member timer 0.99
mc querier timer 254.96
mc response interval 9.99
flags             TOPOLOGY_CHANGE TOPOLOGY_CHANGE_DETECTED
path cost         0
bridge max age   19.99
bridge hello time 1.99
bridge forward delay 14.99
tcn timer        0.00
gc timer         0.49
hash max         512
mc init query count 2
mc snooping      1
mc membership timer 259.96
mc query interval 124.98
mc init query interval 31.24

```

Figure 18 If you look at the topology change timer we see where the 34Secs, our timer here starts at 34.98Secs (max age + forward delay)

## 4.2. Two Switch Set-up

The next stage of the project involved connecting two switches, Switch1 and Switch2 on the network. Results were obtained by connecting PC1 to a mirrored port on Switch1, using PC2 as a man in the middle with two of this machines interfaces set up as a transparent bridge using BRCTL (Harris, 2014) and by connecting PC3 to the mirrored port on Switch2.

## Logical Diagram-Internal Network VirtualBox

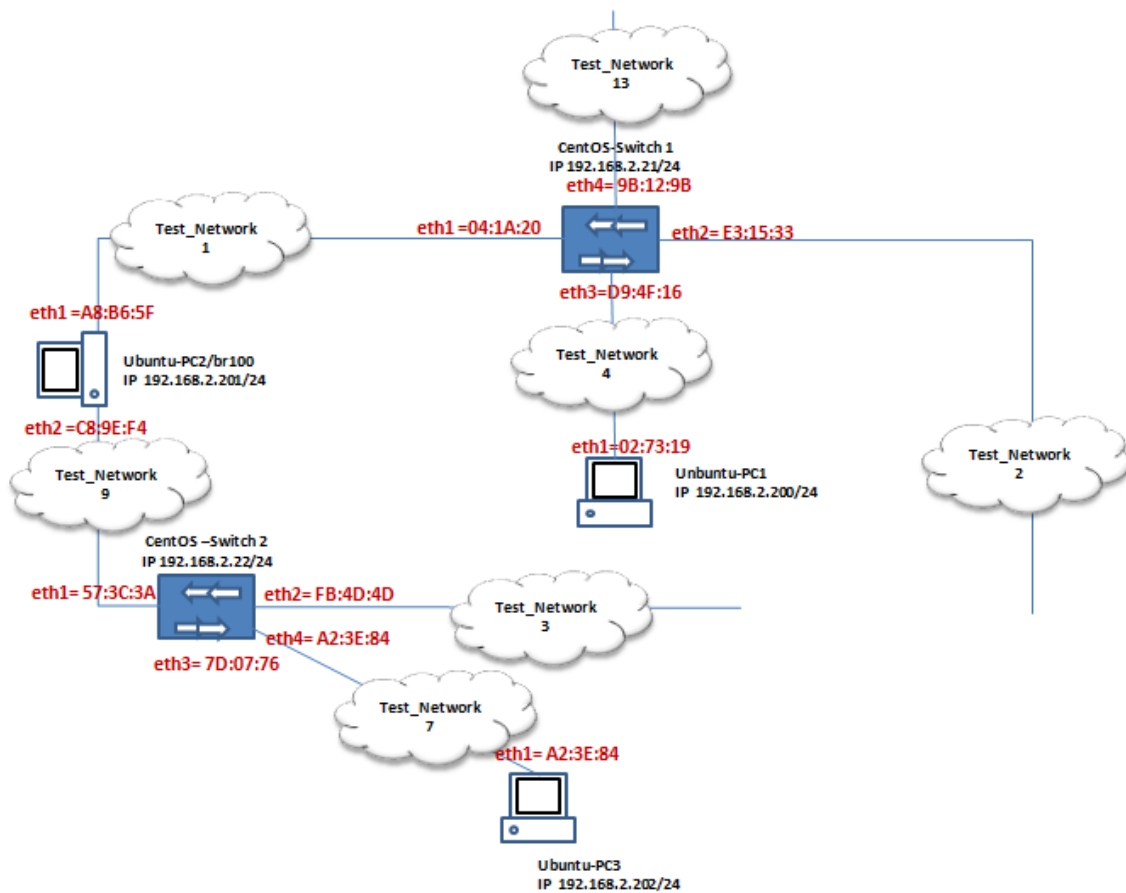


Figure 19 Two Switch Set-Up

The first test involved allowing Switch1 to bypass the initial start-up as described in 3.1 above. Once the switch started to consistently send configuration BPDUs from its designated ports, Switch2 was powered on. It should be noted that the priority value for all switches have been left at the default value of 32768, meaning that the root switch will be determined by the lowest MAC address of the switches used on the network (Switch1 in this instance). Based on previous studies Switch1 should be forwarding configuration BPDUs which are received by Switch2 via its root port. Configuration BPDUs then forwarded from Switch2 should have updated bridge ID, Cost and port ID fields. When Switch2 passes through both the listening and learning states a TCN will be forwarded via its root port to the root Switch (Switch1) indicating that this switch has entered a forwarding state and there is change in the network topology. On receiving the TCN Switch1 sets the topology change bit/flag in all configuration BPDUs it generates to indicate its attached network segments that there has been a change in network topology.

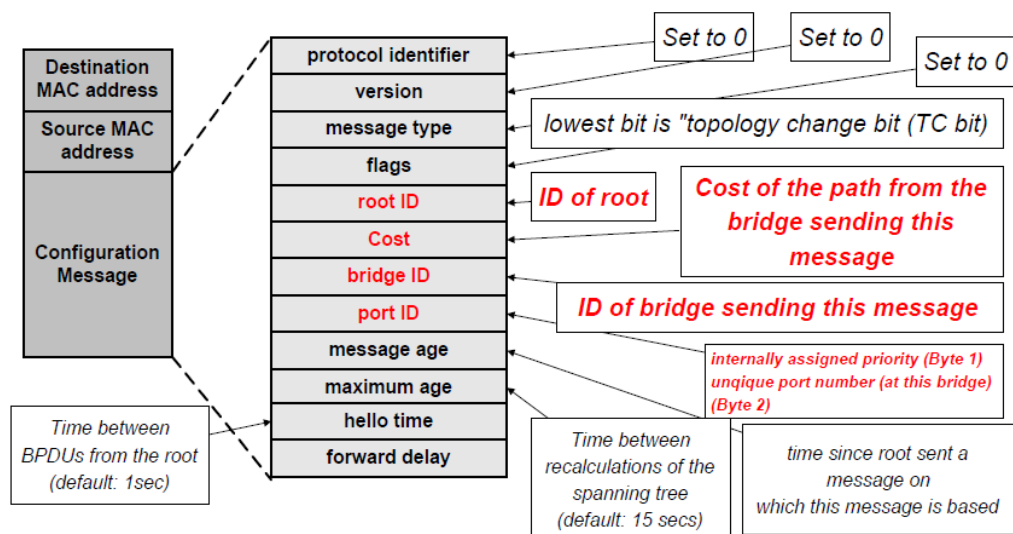


Figure 20 Configuration BPDUs (University of Virginia 2015, p. 14)

When looking at the information obtained from the three monitoring devices placed on the network, the following was noted:

1. It took 28.09895 seconds for Switch2 to send a TCN to Switch1 after packets started to be received using the packet analyser, this would indicate that Switch2 moved through both listening and learning states before forwarding BPDUs.
2. The root ID for all BPDUs has a value 32768/ 0 / 08:00:27:04:1a:20 indicating that Switch1 is the root Switch on this network, while also showing that the priority value is set to the default.
3. The root path cost is zero for a single interface at port id 0x8001 on Switch2 while all other interfaces have a cost of 4; this lower cost indicates that this port is the root port of Switch2. This port's bridge ID (32768 / 0 /08:00:27:04:1a:20) also confirms that Switch2 is receiving configuration BPDUs sent from Switch1 via this port, with all other interfaces forwarding updated configuration BPDUs to other network segments.
4. The source MAC address for the TCN was 57:3c:3a. It was noted from both mirrored ports on the Switches that the first configuration BPDUs with the topology change bit/flag set were broadcast via the designated port on which the TCN was received.
5. After receiving the TCN the root bridge broadcast configuration BPDUs with the topology change flag set for a duration of 33.00006 seconds.

The second test involved allowing Switch2 to bypass the initial start-up as described in 3.1 above. Once the switch started to consistently send configuration BPDUs from its designated ports, Switch1 was powered on. This test indicates what happens when a new root switch is placed on the network.

When looking at the information obtained from the three monitoring devices placed on the network, the following was noted:

1. When a new root switch was placed on the network there was no TCNs generated, this topology change is relayed to the network switches via the topology change bit/flag being set on the configuration BPDUs.
2. The root id and cost of all designated ports change when Switch2 receives a configuration BPDU via its root port from Switch1, with both root id and bridge id changing for the root port receiving configuration BPDUs sent from the root switch.
3. 28.27418 seconds after Switch2 receives its first configuration BPDU from Switch1, Switch1 commences to send configuration BPDUs with the topology change bit/flag set for duration of 33.999865 seconds.

**Note:** As the transparent bridge (PC2) used for the Two Switch Set-up did not provide any extra information with regards to the exchange of BPDUs it was decided that it would not be used for the remainder of testing.

### **4.3. Three Switch Set-up**

Stage three of the project involved connecting three switches Switch1, Switch2 and Switch3 on the network. Results were obtained by connecting PC1 to a mirrored port on Switch1, connecting PC3 to a mirrored port on Switch2 and finally connecting PC4 to a mirrored port on Switch3.

This set-up adds redundancy while introducing also loops to the network. Although each test varies slightly the final network topology will not differ after the network has converged using the STA. As the priority value has not been set, the hierarchy of switches on the network is determined using their MAC address. Switch1 is the root switch as it has the lowest root ID. As both Switch2 and Switch3 have equal root path cost fields, the designated

bridge for Test\_Network3 is established by comparing these switches bridge IDs. Switch3 has a lower bridge ID therefore it becomes the designated bridge for Test\_Network3, while the interface which connects Test\_Network3 to Switch2 transitions into a blocking state still receiving configuration BPDUs from Switch3's designated port.

Logical Diagram-Internal Network VirtualBox

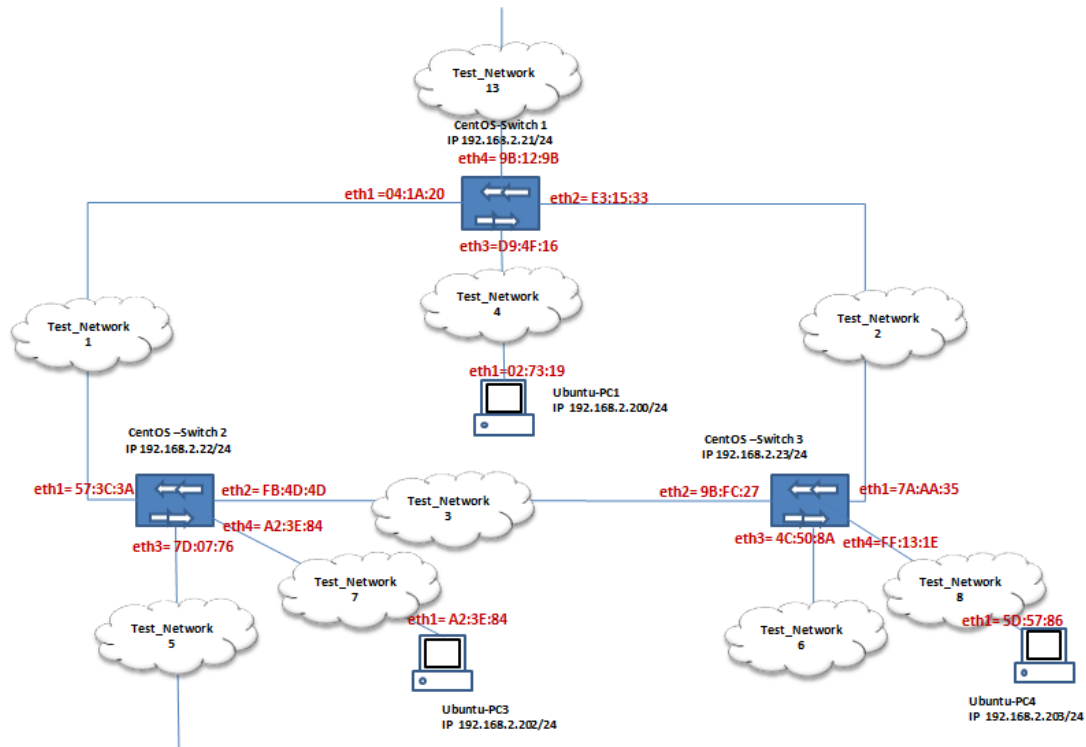


Figure 21 Three Switch Set-Up

This first test involved allowing Switch1 and Switch2 to bypass their initial start-up as described in 3.2 above. Once both switches started to consistently forward configuration BPDUs from their designated ports, Switch3 was powered on.

When looking at the information obtained from the three monitoring devices placed on the network, the following was noted:

1. Switch1 remains the root switch with the lowest bridge ID(04:1a:20) on the network
2. Two TCNs were received by Switch1 , the first TCN from 57:3C:3a and reply from 04:1a:20 indicated the port connected to Test\_Network3 had transitioned from a forwarding into a blocking state ,the second TCN from 7a:aa:35 and reply from e3:15:33 indicated that the designated ports on Switch3 had moved into a forwarding state from start-up (Cisco.com, 2005). As a result these topology changes two root ports



(57:3c:3a,7a:aa:35) and designated ports (04:1a:20,e3:15:33) have been identified on the network. This was also verified by tracing each TCN back to Switch1 and Switch3.

3. Prior to sending a TCN, Switch2 started to receive configuration BPDU's from Switch3's network interface (9b:fc:27) , while the fb:4d:4d interface on Switch2 stopped sending configuration BPDUs. From the above information the network segment between Switch2 (fb:4d:4d)and Switch3(9b:fc:27) has now been identified.

This second test involved allowing Switch1 and Switch3 to bypass their initial start-up phase. Once both switches started to consistently forward configuration BPDUs from their designated ports, Switch2 was powered on.

When looking at the information obtained from the three monitoring devices placed on the network, the following was noted:

1. One TCN was received by Switch1 from Switch2 on its transition to a forwarding state. This identifies the network segment between Switch1 designated port (04:1a:20) and Switch2 root port (57:3c:3a).
2. Switch2 is receiving two configuration BPDUs, one from Switch1 the root switch and the second from Switch3 (port relating to the non-designated/blocking port for Switch2) When two BPDUs are received on a switch because of redundant links in the network, the one with the higher root path cost is logically disabled and placed in a blocked state (Frazier, 2007) . Although this does not give the specific interfaces for each network segment it does give an indication of the network topology.

This third test involved allowing Switch2 and Switch3 to bypass their initial start-up phase. Once both switches started to consistently forward configuration BPDUs from their designated ports, Switch1 was powered on.

When looking at the information obtained from the three monitoring devices placed on the network, the following was noted:

1. The first packet sent out by Switch1 was a configuration packet without the topology change bit/flag set.

2. The topology bit/flag was set for a period of 59.99961 Seconds instead of the default 35 seconds
3. No TCN is was received by the root switch (Switch1)
4. A TCN was sent from Switch2 indicating that port 9b:fc:27 has changed from a forwarding port with a root path cost of 0 into a blocking port with a cost of 4. 57:3c:3a is now receiving configuration BPDUs from Switch1 with this port now becoming the root port 04:1a:20.
5. Switch3 does not generate any TCNs although there is a change in its ports root path cost from 0 to 4 apart from its root port
6. Switch3 received two configuration BPDUs from Switch1, the first was sent directly from Switch1 via e3:15:33 with a root path cost of 0 and the second was sent via Switch2 fb:4d:4d with a root path cost of 4. Switch3 continues to receive BPDUs from Switch1 but only receives this single configuration BPDU from Switch1 via Switch2. The ports on Switch3 do not change their state therefore no TCN is generated as its ports do not change any of their states to blocking.

#### **4.4. Four Switch Set-up**

Stage four of the project involved connecting four switches Switch1, Switch2, Switch3 and Switch4 on the network. This set-up introduces a peripheral switch which is connected to one of the core switches on the network. This test should provide a clear indication of how TCNs are propagated towards the root switch via the non-root switches root port, as well as showing TCAs sent from a non-root switch which is receiving a TCN via its designated port.

Results were obtained by connecting PC1 to a mirrored port on Switch1, connecting PC3 to a mirrored port on Switch2, connecting PC4 to a mirrored port on Switch3 and connecting PC5 to a mirrored port on Switch4.

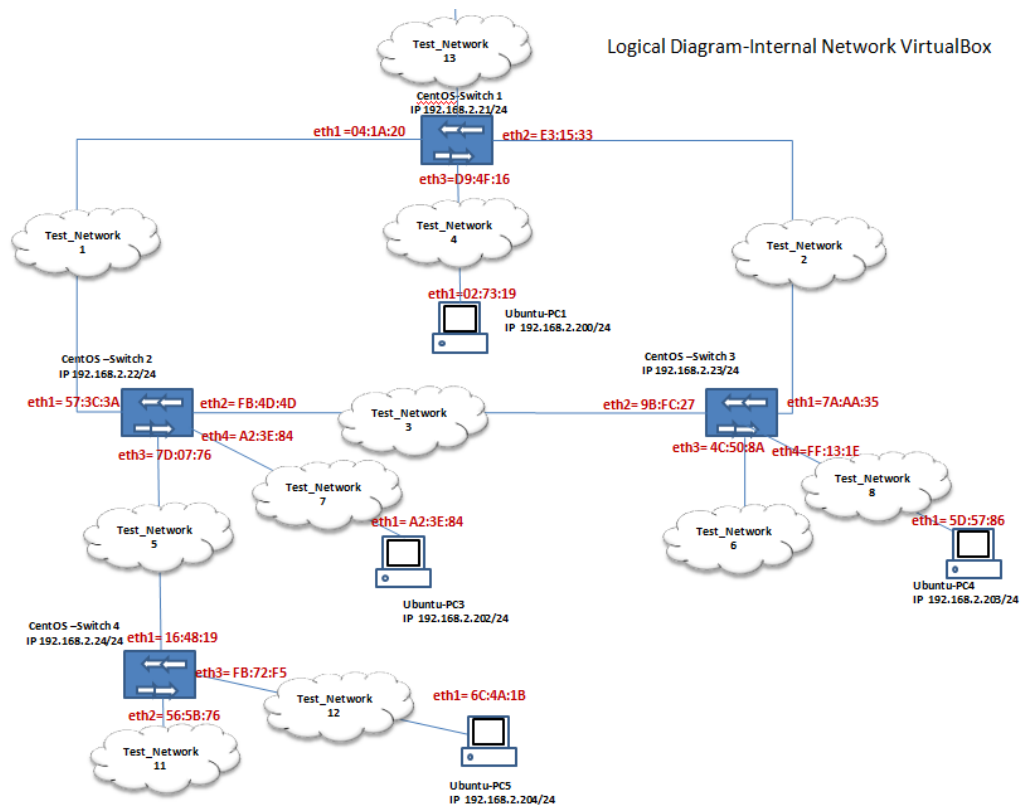


Figure 22 Four Switch Set-Up

This test involved allowing Switch1, Switch2 and Switch3 to bypass their initial start-up as described in 3.3 above. Once these switches started to consistently forward configuration BPDUs from their designated ports, Switch4 was powered on.

When looking at the information obtained from the three monitoring devices placed on the network, the following was noted:

1. One TCN was received from Switch2 by the root switch (i.e. Switch1).
2. Two TCNs were noted on Switch2 , one TCN was received by Switch2 from Switch4 and the second TCN was sent directly after receiving the first TCN from Switch2 to the root Switch
3. Switch3 did not receive or send any TCNs, although it was notified by a change in the network topology by the root switch
4. Switch4 receives configuration BPDUs from Switch3 via port 7d:07:76 with a cost of 4 and forwards configuration BPDUs with a cost of 8. The root sends BPDUs with the root path cost equal to 0, and this cost keeps increasing as the network diameter increases (Frazier, 2007).

## 5. Script Development

The Script Development section details the how script which uses the python programming language was developed in accordance with signatures obtained from the previous Spanning Tree Protocol Analysis section. The script was run on multiple PCs within the internal test network in order to determine if network topology could be fully identified. Although this is simple script to be used by a network administrator or security auditor, the code itself has been divided into object orientated classes making it more readable.

### 5.1. Code Development

```
# used to create raw socket

def raw_Socket(self):

    try:

        self.s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0003))

        # Bind socket to internal network interface

        interface = 'eth'

        num = input('please enter interface number')

        print ('You have selected interface no.' + str(num) + '\n')

        # concatenate number to interface string

        interface += str(num)

        self.s.bind((interface, 0))

        # Place the interface into promiscuous mode on the virtual machine as well as on
VirtualBox Hypervisor

        os.system('ifconfig ' + interface + ' promisc')

    except socket.error, msg:

        print ('Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + msg[1])
```

Code Listing 1 Raw Socket Creation (Moon, 2011)

When developing the script for identifying information from Spanning Tree Protocol Packets the Raw\_Socket Class contains a raw\_Socket function which will create a raw socket on a Linux based operating system. Within Code Listing 1, code has been added to allow the user to pick the network interface to which the raw socket is bound while also placing the selected interface into promiscuous mode, allowing the user to view all traffic on the network. If a socket is unable to be created an error message is displayed and the program shuts down.

```
# create Raw_Socket

    r.raw_Socket()

    #Use CTRL+C to exit loop

    print 'Use CTRL+C to exit...'

    time.sleep(2)

    try:

        while True:

            # retrieve packets from all available ports

            packet = r.s.recvfrom(65565)

            # packet string from tuple

            packet = packet[0]

            # parse ethernet header

            e = Ether(packet[0:17])

            # If an Spanning Tree Protocol or Topology Change Notification is received
determined by length

            if e.eth_protocol == 9728 or e.eth_protocol == 1792:

                # parse bpdu's to integer and string values

                b = Bpdu(packet[17:52])
```

Code Listing 2 Raw Socket used in Runner (Python Software Foundation, 2015)

Code Listing 2 relates to the STP\_Sniff\_Runner Class which is the main method, henceforth referred to as "Runner". This code shows that after the raw socket has been created packets are sniffed from all ports with each packet converted from a string to a read only list of characters, with the first seventeen characters relating to the Ethernet header of the packet. The ethertype value is parsed to an integer value by creating an Ether object using the Ether Class and if the integer values obtained relate to either a standard Spanning Tree Packet or a TCN ethertype the program continues to parse the remainder of the STP or TCN using the Bpdu Class.

```
# used to return ethernet header fields

def rtn_Header(self):

    return ' Time ' + str(datetime.datetime.now().time())+' Destination MAC : ' +
Conversion.toHex(self.dst_addr) + ' Source MAC : ' + Conversion.toHex(self.src_addr) + '
Protocol : ' + str(self.eth_protocol) + ' length : ' + str(self.length)

# used to return local link control

def rtn_llc(self):

    # print local link data

    return ' Local Link : ' + Conversion.toHex(self.local_link_control)
```

**Code Listing 3 Functions returning parsed Ethernet Header and Local Link information**

```
# static method used to convert string to hex

@staticmethod

def toHex(value):

    lst = []

    for ch in value:

        hv = hex(ord(ch)).replace('0x', '')

        if len(hv) == 1:

            hv = '0'+hv

        lst.append(hv)
```

**Code Listing 4 Static toHex function used to parse from string to hex value (Kharechko, 2006)**

Code Listing 3 shows two functions taken from the Ether Class which return information with regards to the Ethernet Header and Local Link Control of the packet to the Ether Object, 'e' in Code Listing 2 created in the Runner which can then be printed to screen. A timestamp have been added each packet to make it uniquely identifiable. The Conversion Class is used for the first time here in order to convert parsed strings into hexadecimal strings. This method is static meaning that it can be called directly from the class using the class name.

```
#print bpdu fields to screen

def print_Packet(self):

    print ' Protocol ID : ' + hex(self.protocol_ID) + ' Version ID : ' + hex(self.version_ID) + '
Bpdu Type : ' + hex(self.bpdu_Type) + ' Flag : ' + hex(self.flag) + ' Root Bridge Priority Value : '
+str(self.root_priority) + ' Root Bridge Mac : ' + Conversion.toHex(self.root_Id) + ' Root Path Cost
: ' + hex(self.root_Path_Cost)+'\n Bridge Priority Value :'+str(self.bridge_priority)+ ' Bridge
Identifier Mac: ' + Conversion.toHex(self.bridge_Id) + ' Port ID : ' + hex(self.port_ID) + ' Message
Age : ' + hex(self.message_Age) + ' Maximum Age : ' + hex( self.maximum_Age) + ' Hello Time : ' +
hex(self.hello_Time) + ' Forward Delay: ' + hex(self.forward_delay)
```

Code Listing 5 Function used to print Bpdu Fields to Screen

Code Listing 5 shows the print Packet function which print BPDU information of the BPDU Object, 'b' in Code Listing 2 created in the Runner to screen.

```
# print packet data

print '\n' + str(count) + e.rtn_Header() +'\n\n'+e.rtn_llc()+'\n'

# print all bpdu Fields

b.print_Packet()
```

Code Listing 6 Printing STP Packet Data to Screen

Code Listing 6 Shows the how Code Listing 3 and 4 are used within the Runner print STP packet information to Screen, which is choice 1"Sniff Spanning Tree Packets" in the Runner menu.

```
stephen@PC3: ~/Desktop/STP_Sniff
stephen@PC3:~$ cd Desktop
stephen@PC3:~/Desktop$ cd STP_Sniff
stephen@PC3:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      1
Choice 1 Selected: Sniff Spanning Tree Packets
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4

Use CTRL+C to exit...

1 Time 23:12:17.807058 Destination MAC : 0180c2000000 Source MAC : 080027a23e84 Protocol : 9728 length : 38
Local Link : 424203
Protocol ID : 0x0 Version ID : 0x0 Bpdu Type : 0x0 Flag : 0x1 Root Bridge Priority Value : -32768 Root Bridge Mac : 080027573c3a Root Path Cost : 0x0
Bridge Priority Value :-32768 Bridge Identifier Mac: 080027573c3a Port ID : 0x8004 Message Age : 0x0 Maximum Age : 0x1400 Hello Time : 0x200 Forward De
lay: 0xf00

2 Time 23:12:17.807372 Destination MAC : 0180c2000000 Source MAC : 0800277d0776 Protocol : 9728 length : 38
Local Link : 424203
Protocol ID : 0x0 Version ID : 0x0 Bpdu Type : 0x0 Flag : 0x1 Root Bridge Priority Value : -32768 Root Bridge Mac : 080027573c3a Root Path Cost : 0x0
Bridge Priority Value :-32768 Bridge Identifier Mac: 080027573c3a Port ID : 0x8003 Message Age : 0x0 Maximum Age : 0x1400 Hello Time : 0x200 Forward De
lay: 0xf00

3 Time 23:12:17.807504 Destination MAC : 0180c2000000 Source MAC : 080027fb4d4d Protocol : 9728 length : 38
Local Link : 424203
Protocol ID : 0x0 Version ID : 0x0 Bpdu Type : 0x0 Flag : 0x1 Root Bridge Priority Value : -32768 Root Bridge Mac : 080027573c3a Root Path Cost : 0x0
Bridge Priority Value :-32768 Bridge Identifier Mac: 080027573c3a Port ID : 0x8002 Message Age : 0x0 Maximum Age : 0x1400 Hello Time : 0x200 Forward De
lay: 0xf00

4 Time 23:12:17.807626 Destination MAC : 0180c2000000 Source MAC : 080027573c3a Protocol : 9728 length : 38
```

Figure 23 Sample of Data obtained using the choice 1 in the application to Sniff Spanning Tree Packet



Within the Runner, choice 2 of the menu enables the user to "Identify Network Switches and Topology" from the previous results identified in Section 4 Spanning Tree Analysis, using a number of if statements.

```
        # if current root is not the same as the previous bridge id and we don't have
a tcu packet 0x80
        if base[0] != b.root_Id and b.bpdu_Type != 128:
            # base value for comparison with next packet
            mac_Value = b.root_Id
            # if this Mac is not list of interfaces
            if mac_Value not in device:
                # append it to the list
                device.append(mac_Value)
                print 'Root Bridge ' + ' ' +
OUI.oui_Lookup(Conversion.toHex(mac_Value))+ ' ' + Conversion.toHex(mac_Value)
```

**Code Listing 7** Code to used to identify the Root Bridge

```
# static method used to look up organisation unique identifier
@staticmethod
def oui_Lookup(macValue):
    inter = EUI(macValue)
    oui=inter.oui
```

**Code Listing 8** OUI Static Method

Code Listing 7 is used to establish the root switch on the network while also identifying its designated ports. The OUI Class is used for the first time to identify a switch's Organisational Unique Identifier (OUI), the static method used is shown in Code Listing 8. The netaddr module(Moss, 2015) was imported into python in order to find the OUI for each noted MAC address which is then also printed to screen.

```

# if the current bridge is not the same as the previous bridge and we don't have a tcu packet

        if base[1] != b.bridge_Id and b.bpdu_Type != 128:

            # Create Mac item for comparison with other Macs added to the list of
devices

            mac_Value = b.bridge_Id

            # Create bridgeid item for comparison with other bridgeid's added to the
dictionary of costs and devices

            # this is used in to identify a non-designated/blocked port

            bridgeid_hex_form = str(Conversion.toHex(packet[34:42]))

            if mac_Value not in device:

                # append it to the list

                device.append(mac_Value)

                # increment the device count

                device_Count += 1

                print 'Bridge ID ' + str(device_Count) + ' ' +
OUI.oui_Lookup(Conversion.toHex(mac_Value))+ ' ' + Conversion.toHex(mac_Value)

                print 'Designated Port Interface ' +
str(Conversion.toHex(packet[9:12]))

```

**Code Listing 9 Identifying Bridges on the Network**

Code Listing 9 is used to establish the multiple switches on the network while also identifying their designated port. A local "bridgeid\_hex\_form" variable using the current bridge ID is created in order to identify if there is a blocked port on the switch being monitored, this is due to both priority and MAC being required in order to establish our hierarchy of bridges.

```

# if there is an equal root path cost and lower bridge_identifier we know that this device has a
blocked port

        if b.root_Path_Cost in dic:

            #find the bridge id value associated with the cost

            br = dic[b.root_Path_Cost]

            if br < bridgeid_hex_form:

print'*****'
*****

        print '\nATTENTION-' + bridgeid_hex_form + ' has a non-
designated/blocked port receiving packets from ' + br+'\n'

print'*****'
*****

        else:

print'*****'
*****

        print '\nATTENTION-' + br + ' has a non-designated/blocked port
receiving packets from ' + bridgeid_hex_form +'\n'

print'*****'
*****

```

**Code Listing 10 Identification of a blocked/non-designated port**

Code Listing 10 is used to identify a blocked/non-designated port searching a dictionary of root path costs and bridge identifier, if the root path cost is found in the dictionary and its associated bridge identifier is lower than the current bridge ID then the current bridge ID has a blocked port receiving packets from the stored bridge identifier and vice-versa. The root path cost and bridge ID dictionary is updated if a new bridge is detected and its MAC address has not already been stored is detected.

```

# find other forwarding ports associated with a bridge id

    if mac_Value == b.bridge_Id and b.bpdu_Type != 128:

        if b.bridge_Id in br_pid:

            # if this port id is not in the dictionary list of port id's associated with
the bridge id key value

                if b.port_ID not in br_pid[b.bridge_Id]:

                    br_pid[b.bridge_Id].append(b.port_ID)

                    print 'Designated Port Interface ' +
str(Conversion.toHex(packet[9:12]))

            # creates and updates dictionary of bridge id's and port id's

            br_pid.setdefault(b.bridge_Id, [b.port_ID])

```

**Code Listing 11 identifying other forwarding ports in the switch**

Code Listing 11 identifies multiple forwarding ports associated with a single bridge ID. If the previous bridge ID is different from the current bridge ID and if the current bridge ID has already been identified then port IDs are used in order to establish if this bridge has more than one designated port. If the ports IDs differ then the port ID is appended to a list of port IDs associated with the bridge ID and information with regards to the interface is printed to screen. This updated information is then stored in a dictionary of bridge ID and associated list of port IDs.

```

# the segment is identified by topology change acknowledgment (i.e the source mac address)
after a topology change notification

    if base[2] == 128 and b.flag == 129:

        src_addr = Conversion.toHex(e.src_addr)

        combined_segment_Values = src_addr + base[3]

        #stop duplication of segments

        if combined_segment_Values not in seg:

            print 'This is a network segment : Root Port ' + base[3] + ' to
Designated Port ' + Conversion.toHex(e.src_addr)

            # append this know segment to list of segments

            seg.append(combined_segment_Values)

```

**Code Listing 12 Identification of network segment using Single TCN and TCA**

Code Listing 12 identifies network segments including root and designated ports on the monitored machine by using both TCNs and TCAs. The code checks the list of variables stored from the previous packet to identify if a TCN was sent and checks if a TCA has been received in the current packet. If this is the case then the current source address is combined with the previous root or bridge ID and compared with a list of previously identified segments in order to stop duplication. If the identified segment is not contained within the list of previously identified segments then information with regards to the segment is printed to screen and the segment information is appended to the list of known network segments.

```
# if there is two tcn's in a row we know that the first tcn is the root port of a device connected to
the monitored device and the second tcn is the root port of the monitored device
```

```
    if base[2] == 128 and b.bpdu_Type == 128:

        src_addr = Conversion.toHex(e.src_addr)

        combined_segment_Values = src_addr + base[3]

        #stop duplication of segments

        if combined_segment_Values not in seg:

            print 'Root Port connected to this device from Interface ' + base[3]

            print 'Root Port from this device from Interface ' + src_addr

            # append this know segment to list of segments

            seg.append(combined_segment_Values)
```

**Code Listing 13 Identification of Root Ports using multiple TCNs**

Code Listing 13 identifies Root ports from a monitored device and switches connected to the monitored using multiple TCNs. The code checks a list of variables stored from the previous packet to check if a TCN sent or received and compares with the current packet to check if it is also a TCN. If this is the case then combines the source address of the current packet with source address of the previous packet and compares with the list of combined segment values. If this combined value is not stored within the list of combined segment values then ports to and from the monitored device are printed to screen and the combined value is appended to the list of known network segments/roots.

## 5.2. Application Testing

This section is divided into two sub sections in order to identify what information is obtained on a network which is stable without any TCNs being generated and with another network which is unstable and generating TCNs and TCAs.

### 5.2.1. *Test Network without TCNs and TCAs*

The first test used the four-switch network topology set-up detailed in Section 4 Spanning Tree Analysis to identify what information could be obtained using the developed test script on a Stable Network (see Appendix D). From this a table of information with regards to network topology information was obtained and a logical diagram drawn from the information given.

Within the stable network Switch1 is the root bridge as a result of having the lowest root ID on the network, propagating configuration BPDUs via its designated ports. Switch3 is the designated bridge for Test\_Network3 as it is equal in root path cost but has a lower bridge ID than Switch2. Switch3 is receiving configuration BPDUs via its root port and forwarding updated configuration BPDUs via its remaining designated ports. In addition to receiving configuration BPDUs via its root port Switch2 is also receiving configuration BPDUs from the designated bridge (Switch3) for Test\_Network3 via a blocked port while also forwarding updated configuration BPDUs to Switch4 via one of its remaining designated ports. In turn Switch4 updates both cost and bridge ID values in the configuration BPDUs received from Switch2 via its root port and forwards these updated BPDUs via its designated ports.

By using the developed test script network devices, designated ports and the blocked port of the converged network can be identified. It should be noted that no root ports are identified on the stable network due to the absence of TCNs on this type of network.

PC NO.	Identifier	Organisational Identifier	MAC Address	Designated Port	Root Port	Notes
1	Root ID	CADMUS	080027041a20	9b129b d94f16 e31533 041a20		This is the Root Switch
3	Root ID	CADMUS	080027041a20	041a20		This Switch is directly connected to the Root Bridge
	Bridge ID	CADMUS	080027573c3a	a23e84 7d0776		This is the Switch being monitored due to multiple forwarding ports
	Bridge ID	CADMUS	0800274c508a	9bfc27		This is a blocked
4	Root ID	CADMUS	080027041a20	e31533		This Switch is directly connected to the Root Bridge
	Bridge ID	CADMUS	0800274c508a	ff131e 4c508a 9bfc27		This is the Switch being monitored due to multiple forwarding ports
5	Root ID	CADMUS	080027041a20			This Switch is not directly attached to the Root Switch
	Bridge ID	CADMUS	080027573c3a	7d0776		This Switch is forwarding packets to the monitored switch
	Bridge ID	CADMUS	08002716481a	fb72f5 565b76		This is the Switch being monitored due to multiple forwarding ports

Table 1 Application Test Results for a Stable Network

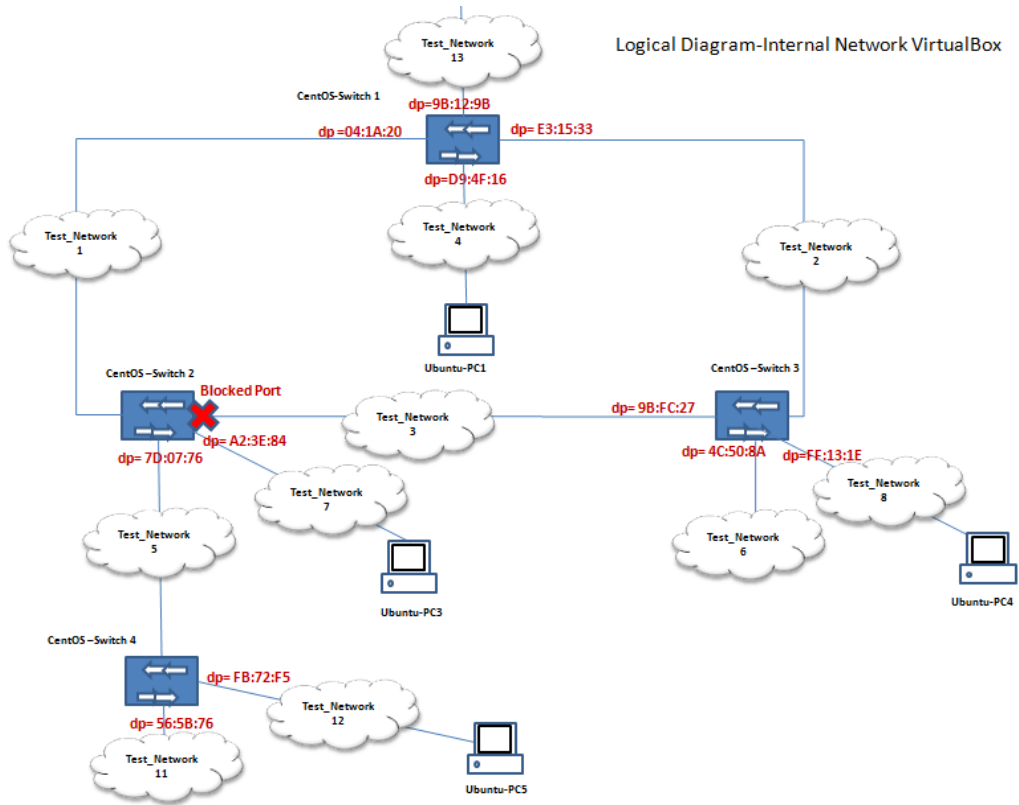


Figure 24 Topology Information obtained from Table1



### **5.2.2. Test network with TCNs and TCAs**

The second set of tests is also carried out on the four-switch set-up network topology described in Section 4 Spanning Tree Analysis but are carried out on an Unstable Network where TCNs and TCAs are being generated. There are a total of six tests shown where a port is going up or down, with the aim of generating TCNs which occur when a port/interface is transitioning into forwarding state or from forwarding or learning state into a blocking state (see Appendix E). From these tests a table of information with regards to network topology was obtained and a logical diagram drawn from the information given.

Test one indicates what information can be obtained by administratively placing the root port (7a:aa:35) of Switch3 into a disabled state. Switch1 remains the root bridge propagating configuration BPDUs throughout the network. Switch2's blocked port transitions from a blocked state into a forwarding state as it becomes the designated bridge for Test\_Network3. As Switch3 is no longer the designated bridge for Test\_Network3 and is no longer directly connected to the root bridge (Switch1), its designated port (9b:fc:27) to Test\_Network3 now becomes the root port for this switch. Configuration BPDU's are now received by Switch3 via Switch2, meaning that configuration BPDUs forwarded from Switch3 now have an updated root path cost field which reflects this change in network topology.

The script successfully identifies Test\_Network1 (04:1a:20 to 57:3c:3a) as a TCN is generated by Switch2 as a result of its blocked port transitioning from a blocking into a forwarding state. Switch2's previously blocked interface (fb:4d:4d) is identified, as it becomes the designated port for Test\_Network3. Finally, Switch3's new root port (9b:fc:27) is also identified as configuration BPDUs now being received via this port from Switch2.

Test two indicates what information can be obtained by administratively placing Switch3's previously disabled port (7a:aa:35) back into a forwarding state. This port once again becomes Switch3's root port, receiving configuration BPDUs directly from the root bridge (Switch1) resulting in a change in the root path cost field of the configuration BPDUs forwarded from Switch3. Switch2 is no longer the designated bridge for Test\_Network3 resulting in the designated port (fb:4d:4d) connected to this network segment being placed back into a blocking state and Switch3's interface(9b:fc:27) returning to the forwarding state as it once again becomes the designated port for this network segment.

The script successfully identifies Test\_Network2 (e3:15:33 to 7a:aa:35) as a TCN is generated by Switch3 as a result of its disabled port transitioning back into a forwarding state.

Test three indicates what information can be obtained by administratively placing Switch2's root port (57:3c:3a) into a disabled state. Switch1 remains the root bridge propagating configuration BPDUs throughout the network. Switch2's blocked port (fb:4d:4d) becomes its root port with configuration BPDUs forwarded by this switch having an updated root path cost to reflect the change in network topology. There is no change with regards to Switch3 which continues to forward configuration BPDUs via its designated ports.

The script successfully identifies Test\_Network3 (9b:fc:27 to fb:4d:4d) as a result of a TCN being generated when Switch2's port (fb:4d:4d) transitions from a blocked state into a forwarding state becoming the root port for this switch. The peripheral switch (Switch4) has a lower MAC address than Switch2 and as a result of bridge priority being left as the default value, it assumes it is the root bridge when Switch2's root port (57:3c:3a) is initially disabled prior to receiving configuration BPDUs from the actual root bridge (Switch1) via Switch3(9b:fc:27) and Switch2(7d:07:76). Switch4's root port and designated port (16:48:1a), as it is both receiving and sending information to the root bridge (Switch1) is identified. Switch4's root port(16:48:1a) was identified as a result of a TCN being generated when this switch transitions from a blocked state into a forwarding state after it initialises the STA when it assumes that it is the root bridge.

Test four indicates what information can be obtained by administratively placing Switch2's initial root port (57:3c:3a) back into a forwarding state. Switch1 remains the root bridge propagating configuration BPDUs throughout the network. There will be no change with regards to Switch3 which continues to forward configuration BPDUs via its designated ports. As Switch2's root port reverts back to the it interface (57:3c:3a) which is directly connected to the root bridge, interface fb:4d:4d once again enters a blocked state receiving configuration BPDUs from the root bridge via Switch3(9b:fc:27).

There is no duplicated information obtained as a result of Switch2's interface (57:3c:3a) transitioning from a disabled state back into a forwarding state and generating a TCN which would identify Test\_Network1 when using the test script.

Test five indicates what information can be obtained by administratively placing the designated port (9b:fc:27) of Switch3 into a disabled state. Switch1 remains the root bridge propagating configuration BPDUs throughout the network. Switch2's blocked port (fb:4d:4d) transitions into a forwarding state as now becomes the designated port for Test\_Network3.

There is no duplicated information obtained as a result of Switch2's blocked interface (fb:4d:4d) transitioning from a blocked into a forwarding state becoming the designated port for Test\_Network3 and generating a TCN which would identify Test\_Network1 when using the test script.

Test six indicates what information can be obtained by administratively placing the disabled port (9b:fc:27) of Switch3 into a forwarding state. Switch1 remains the root bridge propagating configuration BPDUs throughout the network. Switch2's designated port (fb:4d:4d) for Test\_Network3 transitions from a forwarding to a blocked state, as Switch3 once again becomes the designated bridge for Test\_Network3.

There is no duplicated information obtained as a result of Switch3's disabled port (9b:fc:27) transitioning from a disabled into a forwarding state which would generate a TCN indicating Test\_Network2 or as a result of Switch2's designated port (fb:4d:4d) transitioning from a forwarding into a blocking state which would generate a TCN indicating Test\_Network1 when using the script.

By carrying out these tests the entire topology of the test network was established, although it was noted that this was achieved by combining the information obtained by concurrently running the tests scripts on all four administrative PCs on the test network. As data relating to individual network segments was not duplicated by using the developed python script the process of establishing the test network topology was greatly simplified.

PC NO.	Identifier	Organisational Identifier	MAC Address	Designated Port	Root Port	Notes
1	Root ID	CADMUS	080027041a20	9b129b d94f16 e31533 041a20		Sw3eth1 down and up Test_Network1 and Test_Network2 identified
3	Root ID	CADMUS	080027041a20	041a20		
	Bridge ID	CADMUS	080027573c3a	a23e84 7d0776 fb4d4d	573c3a fd4d4d	Sw3eth1 down fd4d4d becomes dp sw2eth1down <ul style="list-style-type: none"> <li>fd4d4d becomes the rp</li> <li>network segment shown between SW2 and SW3 fd4d4d rp and 9bfc27 dp</li> </ul>
	Bridge ID	CADMUS	0800274c508a	9bfc27	9bfc27	Sw3eth1down 9bfc27 becomes rp
	Bridge ID	CADMUS	08002716481a	16481a	16481a	Sw2eth1downSwitch4 identified port acting as both dp and rp for this segment
4	Root ID	CADMUS	080027041a20	e31533		
	Bridge ID	CADMUS	0800274c508a	ff131e 4c508a 9bfc27	7aaa35	Sw2 eth1 down rp established as 7a:aa:35
5	Root ID	CADMUS	080027041a20	7d0776		Sw3eth1 down bridge identifies configuration BPDUs propagated via Switch2 dp
	Bridge ID	CADMUS	080027573c3a	7d0776		
	Bridge ID	CADMUS	08002716481a	fb72f5 565b76 16481a	16481a	Sw2eth1up 16481a established as dp

Table 2 Information obtained from unstable network

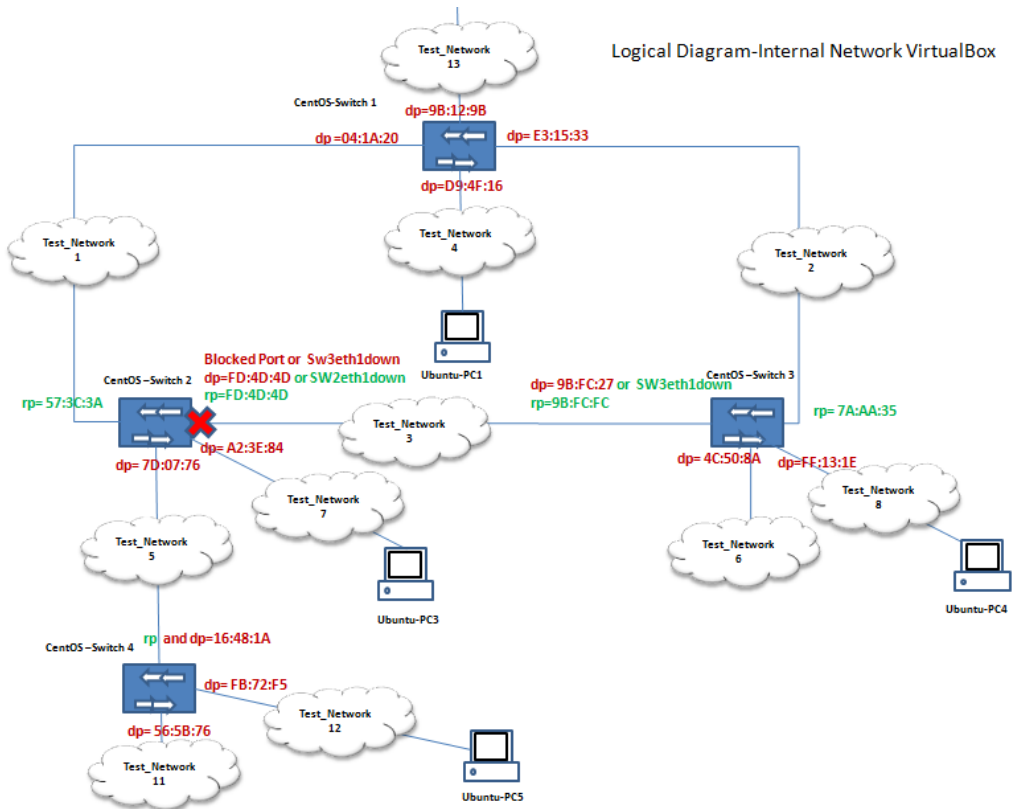


Figure 25 Topology Information obtained from Table2

## **6. Recommendations /Conclusions**

This section is broken into three sub sections with the first two sections relating to the project scope identified at the beginning of the dissertation and the final section detailing what the author believes is the next logical step as a result of the works carried out within this dissertation.

### **6.1. Information Leakage and Mitigation**

Developed 30 years ago and having a number of DoS attacks vectors identified as detailed in Section 2 Literature Review, STP does not appear to provide a large amount of data leakage which could prove advantageous to a would be attacker.

Having said this, depending on how many TCNs are generated and whether default priority values are contained within the root ID and bridge ID BPDU STP fields. One could assume what level of administration is being used on the targeted network. For example, if there is a large number of TCNs and the default priority values are used for STP then an attacker may view this as indicating a lack of administration present on the network; therefore indicating a possibility that devices contained within the network may have been installed without changes to their default administrative values. This combined with the "Organisation Unique Identifier" information contained within in the first three bytes of a layer two MAC address which provides information with regards to the device vendor which could provide the attacker with enough information allowing them to access and control a switch on the network. This would allow an attacker to change the priority value on the controlled switch making it the root bridge in STP and giving further MAC address information relating to network nodes contained within the controlled switches" MAC address table. This scenario could easily be avoided by having the correct administrative procedures in place.

With regards to the DoS attacks detailed in Section 2 Literature Review, alternative emerging secure methods for avoiding loops within the network such as Transparent Interconnection of Lots of Links (TRILL), Shortest Path Bridging (SPB) and Software Defined Networking (SDN) could be employed as an alternative to STP in order to mitigate for the

various flooding and topology engagement attacks identified. Otherwise a number of mitigation techniques are detailed within the Section 2 Literature Review which could be used in order to make STP more secure.

## **6.2. Discovery of Network Topology**

It was noted that the root switch provides the least amount of information with regards to network topology on a stable network as a result of having designated ports only. This means that no information with regards to switches connected to this device can be obtained as all configuration BPDUs originate here without information such as bridge ID or root path cost being updated. On an unstable network information with regards to the root ports of connected devices can be obtained via the source address of a TCN packet.

Non-root switches provide more information as a result of information they receive via their root port. This is where a configuration BPDU is received from a switch with a higher priority value and updates fields within its own configuration BPDUs, namely bridge ID and root path cost, which are forwarded from its designated ports to switches which have a lower priority on the network.

When a monitored switch provides information with regards to more than three bridge IDs, this switch is receiving configuration BPDUs via two of its ports. This indicates that the switch has a non-designated/blocked port, which can be identified by finding equal root path costs and then calculating the lower bridge priority by finding the lowest bridge ID.

If combined with TCAs, TCNs can provide information with regards to the segments used on a LAN. This is achieved by using the TCN source address indicating the root port by which the TCN was sent, with the TCA source address indicating the designated port of the switch receiving the TCN. If two TCNs are received in succession then the last TCN is to be used in this way, with the first TCN relating to the root port of a switch of lower priority connected to the monitored switch.

When TCAs are sent from the root switch using the topology change bit/flag no real topology information is given as this is sent via configuration BPDUs indicating to all receiving switches that the default aging time for their MAC tables should be changed from the default of 300 seconds to its forward delay with a default value of 15 seconds.

In order to accurately identify the network topology STP information from multiple switches is required; this was shown by the amalgamation of the data obtained in the tables detailing information with regards to stable and unstable networks in Section 5 Script Development. This is also verified by a prior study showing how the physical topology of a network can be established using the Spanning Tree Protocol by obtaining STP information of all switches' ports in Management Information Base (MIB) using SNMP (HePeng PanHeng *et al.*, 2010)

### **6.3. Future Work**

Further study and development of the script is required in order to make it fully object orientated. Secure coding guidelines with regards to python would also need to be investigated and the code re-factored in accordance to these guidelines.

Sudo privileges are required when running the test scripts which provides some security with regards to its use on the administrative PC, although as the file is left unencrypted on the administrators desktop this would allow for its abstraction and use on a PC controlled by a would be attacker. In order to provide extra security the administrator could use Cryptkeeper (Pot, 2011) or eCryptfs (eCryptfs.org, 2015) to encrypt the python folder located on the desktop.

As all testing was carried out in a virtual network, further tests would be required within an enterprise environment in order to identify if it functions correctly and to ascertain if it has any value as part of a security audit of the network.

As a result of BRCTL currently complementing ANSI/IEEE802.1D and with plans for future enhancements including RSTP 802.1W, further research with regards to the establishment of network topology using the enhanced BRCTL completing IEEE802.1W should be carried out when it is released.

As a result of the works carried out the author believes that the tools and techniques used could form the basis for a diagnostic tool for STP, which could be used to identify faulty or disabled interfaces on the network.





## References

- Alder, 2002 Alder, R. (2002). *Raw Sockets*. [Online]. Available at: [http://www.linuxchix.org/content/courses/security/raw\\_sockets](http://www.linuxchix.org/content/courses/security/raw_sockets) [Accessed: 20 March 2015].
- Ang,2012 Ang, D. (2012). *VBoxManage Manage VirtualBox From Command Line - The Life of an Automation Engineer*. [Online]. Available at: <http://www.anggrianto.com/blog/vboxmanage-manage-virtualbox-from-command-line/> [Accessed: 26 March 2015].
- Backreference.org., 2014 Backreference.org. (2014). *Port mirroring with Linux bridges* « 1. [Online]. Available at: <http://backreference.org/2014/06/17/port-mirroring-with-linux-bridges/> [Accessed: 15 February 2015].
- Bahethi,2014 Bahethi, P. (2014). What is 802.1D Spanning Tree Protocol? *Network Design Implementation Consultation - Shilpa Systems Inc. USA*. [Online]. Available at: <http://www.shilpasys.com/articles/what-is-802-1d-spanning-tree-protocol/> [Accessed: 11 January 2015].
- Barrett,2005 Barrett, D. J. (2005). *SSH, the secure shell: the definitive guide*. 2nd ed. Sebastopol, CA: O'Reilly.
- Bryant,2015 Bryant, C. (2015). *CCNP SWITCH Tutorial: TCN BPDUs*. [Online]. Available at: <http://www.thebryantadvantage.com/CCNPBCMSNExamTCNBPDU.htm> [Accessed: 21 March 2015].
- Cecil,2014 Cecil, A. (2014). *A Summary of Network Traffic Monitoring and Analysis Techniques*. [Online]. Available at: [http://www.cs.wustl.edu/~jain/cse567-06/ftp/net\\_monitoring/index.html](http://www.cs.wustl.edu/~jain/cse567-06/ftp/net_monitoring/index.html) [Accessed: 30 December 2014].
- Chappell,2010 Chappell, L. (2010). *Wireshark network analysis: the official Wireshark*

*certified network analyst study guide*. San Jose, , CA: Protocol Analysis Institute, Chappell University.

- Cisco.com,2005 Cisco.com. (2005). *Understanding Spanning-Tree Protocol Topology Changes - Cisco*. [Online]. Available at: <http://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/12013-17.html> [Accessed: 27 March 2015].
- eCryptfs.org,2015 eCryptfs.org. (2015). *eCryptfs*. [Online]. Available at: <http://ecryptfs.org/> [Accessed: 28 May 2015].
- Frazier,2007 Frazier, J. (2007). *Hacking lan switches*. 1st ed. Indianapolis, IN: Cisco Press.
- French,2015 French, T. (2015). *How to use VLOOKUP in Excel*. [Online]. Available at: <http://spreadsheets.about.com/od/excelfunctions/ss/vlookup.htm> [Accessed: 29 March 2015].
- Gordon, 2007 Gordon, P. (2007). *Data Leakage - Threats and Mitigation*. SANS Institute InfoSec Reading Room.
- Harris,2014 Harris, G. (2014). *CaptureSetup/Ethernet - The Wireshark Wiki*. [Online]. Available at: <https://wiki.wireshark.org/CaptureSetup/Ethernet> [Accessed: 29 March 2015].
- HePeng PanHeng et al.,2010 HePeng PanHeng, LiXiangdong and ZhengQiusheng. (2010). *Physical topology discovery based on spanning tree protocol*. In: 2010, IEEE, p.V14–V308 – V14–V311. [Online]. Available at: doi:10.1109/ICCCAS.2010.5622296 [Accessed: 28 April 2015].
- IEEE et al, 2004 Institute of Electrical and Electronics Engineers, IEEE Computer Society, LAN/MAN Standards Committee, American National Standards Institute and IEEE-SA Standards Board. (2004). *IEEE standard for local*

*and metropolitan area networks media access control (MAC) bridges.*  
New York, N.Y.: Institute of Electrical and Electronics Engineers.  
[Online]. Available at:  
<http://ieeexplore.ieee.org/servlet/opac?punumber=9155> [Accessed:  
18 March 2015].

- IT Security,2007                      IT Security. (2007). *10 Steps to Creating Your Own IT Security Audit - IT Security.* [Online]. Available at: <http://www.itsecurity.com/features/it-security-audit-010407/> [Accessed: 28 April 2015].
- Iveson,2013                              Iveson, S. (2013). *Speaker for the Dead - Spanning Tree Protocol - Packet Pushers Podcast.* [Online]. Available at:  
<http://packetpushers.net/speaker-for-the-dead-spanning-tree-protocol/> [Accessed: 18 March 2015].
- Jemerov,2013                            Jemerov, D. (2013). PyCharm 3.0 Community Edition source code now available. *JetBrains PyCharm Blog.* [Online]. Available at:  
<http://blog.jetbrains.com/pycharm/2013/10/pycharm-3-0-community-edition-source-code-now-available/> [Accessed: 23 March 2015].
- Kaufman,2012                           Kaufman, L. (2012). *Create Shortcuts on the Desktop to Run Programs as Root in Ubuntu 11.10.* [Online]. Available at:  
<http://www.howtogeek.com/112700/create-shortcuts-on-the-desktop-to-run-programs-as-root-in-ubuntu-11.10/> [Accessed: 23 March 2015].
- Khareckko,2006                        Kharechko, M. (2006). *Convert string to hex « Python recipes « ActiveState Code.* [Online]. Available at:  
<http://code.activestate.com/recipes/496969-convert-string-to-hex/>  
[Accessed: 11 April 2015].
- Kumar, 2014                              Kumar, J. (2014). *How to Create and Manage VM Groups in VirtualBox - Easily Manage Virtual Machines.* [Online]. Available at:  
<http://www.sysprobs.com/how-to-create-manage-vm-groups-in->

virtualbox-virtual-machines [Accessed: 29 March 2015].

- Lai et al.,2014                      Lai, Y., Pan, Q., Liu, Z., Chen, Y. and Zhou, Z. (2014). *Trust-Based Security for the Spanning Tree Protocol*. In: May 2014, IEEE, p.1338–1343. [Online]. Available at: doi:10.1109/IPDPSW.2014.150 [Accessed: 28 April 2015].
- Lamping et al, 2014                Lamping, U., Sharpe, R. and Warnicke, E. (2014). *Wireshark User's Guide Revision 3.2*. [Online]. Available at: [https://www.wireshark.org/docs/wsug\\_html\\_chunked/index.html](https://www.wireshark.org/docs/wsug_html_chunked/index.html) [Accessed: 23 March 2015].
- Lukaszewski                        Lukaszewski, A. (2015). *A Brief History of Python - About Python*. [Online]. Available at: [http://python.about.com/od/gettingstarted/ss/whatispython\\_2.htm](http://python.about.com/od/gettingstarted/ss/whatispython_2.htm) [Accessed: 24 March 2015].
- Menga, 2004                        Menga, J. (2004). Chapter 4. Spanning Tree. In: *CCNP Practical Studies: Switching*. [Online]. Available at: [https://www.informit.com/library/content.aspx?b=CCNP\\_Studies\\_Switching&seqNum=29](https://www.informit.com/library/content.aspx?b=CCNP_Studies_Switching&seqNum=29) [Accessed: 24 March 2015].
- Marro Mario,2003                 Marro Mario, G. (2003). *Attacks at the Data Link Layer*. Master of Science in Computer Science, University of California at Davis.
- Menzies, 2002                     Menzies, T. (2002). *The Raw and the Uncooked: The Windows XP Raw Sockets Saga Final Words (Hopefully)*. SANS Institute, Bethesda, MD.
- Molenaarin,2014                 Molenaarin, R. (2014). *Spanning Tree Topology Change Notification (TCN) - Networklessons.com*. *Networklessons.com*. [Online]. Available at: <http://networklessons.com/switching/spanning-tree-topology-change-notification-tcn/> [Accessed: 21 March 2015].
- Moon,2011                         Moon, S. (2011). *Code a network packet sniffer in python for Linux*.

- BinaryTides*. [Online]. Available at:  
<http://www.binarytides.com/python-packet-sniffer-code-linux/>  
[Accessed: 29 December 2014].
- Moss,2015 Moss, D. (2015). *Installing netaddr — netaddr 0.7.14 documentation*. [Online]. Available at:  
<https://pythonhosted.org/netaddr/installation.html> [Accessed: 11 April 2015].
- Neubert,2013 Neubert, A. (2013). *More than 4 Network Cards in Virtualbox [EanderAlx's Page]*. [Online]. Available at:  
[https://www.eanderalx.org/virtualization/8\\_network\\_card\\_vbox](https://www.eanderalx.org/virtualization/8_network_card_vbox)  
[Accessed: 27 February 2015].
- O' Raw et al,2015 O'Raw ,J,Laverty,D, Morrow,J.(2015) Asset Tracking in Critical Power Communications Infrastructure using Passive Techniques. 2015 IEEE 15<sup>th</sup> International Conference on Environment and Electrical Engineering.
- Perlman,1985 Perlman, R. (1985). An algorithm for distributed computation of a spanningtree in an extended LAN. *ACM SIGCOMM Computer Communication Review*, 15 (4), p.44–53. [Online]. Available at:  
[doi:10.1145/318951.319004](https://doi.org/10.1145/318951.319004) [Accessed: 19 March 2015].
- Perrin ,2008 Perrin, C. (2008). *Use PuTTY as an SSH client on Windows - TechRepublic*. [Online]. Available at:  
<http://www.techrepublic.com/blog/it-security/use-putty-as-an-ssh-client-on-windows/> [Accessed: 24 March 2015].
- Pot,2011 Pot, J. (2011). Encrypt & Protect Your Computer Files With CryptKeeper [Linux]. *MakeUseOf*. [Online]. Available at:  
<http://www.makeuseof.com/tag/encrypt-protect-computer-files-cryptkeeper-linux/> [Accessed: 28 May 2015].

- Python Software Foundation,2015 Python Software Foundation. (2015). 7.3. *struct* — Interpret strings as packed binary data — Python 2.7.10rc0 documentation. [Online]. Available at: <https://docs.python.org/2/library/struct.html#format-characters> [Accessed: 10 April 2015].
- Rai et al.,2011 Rai, A., Barbhuiya, F. A., Sur, A., Biswas, S., Chakraborty, S. and Nandi, S. (2011). *Exploit detection techniques for STP using distributed IDS*. In: December 2011, IEEE, p.939–944. [Online]. Available at: doi:10.1109/WICT.2011.6141374 [Accessed: 28 April 2015].
- Ribeiro,2009 Ribeiro, M. (2009). *Virtualization Basics | Thoughts on Information Technology*. [Online]. Available at: <https://itechthoughts.wordpress.com/2009/11/10/virtualization-basics/> [Accessed: 22 March 2015].
- Rouse,2005 Rouse, M. (2005). *What is security audit? - Definition from WhatIs.com*. [Online]. Available at: <http://searchcio.techtarget.com/definition/security-audit> [Accessed: 28 April 2015].
- Rossi,2000 Rossi, L. R. (2000). *Cisco catalyst LAN switching*, McGraw-Hill Cisco technical expert series. New York: McGraw-Hill.
- Saive,2012 Saive, R. (2012). *SSH Passwordless Login Using SSH Keygen in 5 Easy Steps*. [Online]. Available at: <http://www.tecmint.com/ssh-passwordless-login-using-ssh-keygen-in-5-easy-steps/> [Accessed: 11 February 2015].
- Sans, 2014 Sans.org. (2014). *Intrusion Detection FAQ: How can passive techniques be used to audit and discover network vulnerability?*. Available: [http://www.sans.org/security-resources/idfaq/passive\\_vuln.php](http://www.sans.org/security-resources/idfaq/passive_vuln.php). Last [accessed 13th December 2014].
- Siebert,2011 Siebert, E. (2011). *Understanding hosted and bare-metal virtualization*

- hypervisor types*. [Online]. Available at:  
<http://searchservervirtualization.techtarget.com/tip/Understanding-hosted-and-bare-metal-virtualization-hypervisor-types> [Accessed: 22 March 2015].
- Solie,2002 Solie, K. (2002). *CCIE practical studies*, Cisco Press practical studies series. Indianapolis, IN: Cisco Press.
- Sullivan,2013 Sullivan, D. (2013). Active vs Passive Network Monitoring - What's on Your Network? The Need for Passive Monitoring. *Tom's IT Pro*. [Online]. Available at:  
[http://www.tomsitpro.com/articles/network\\_monitoring-netflow-it\\_security-networking-snmp,2-561-2.html](http://www.tomsitpro.com/articles/network_monitoring-netflow-it_security-networking-snmp,2-561-2.html) [Accessed: 30 December 2014].
- Taft, 2010 Taft, D. (2010). *JetBrains Strikes Python Developers with PyCharm 1.0 IDE*. [Online]. Available at: <http://www.eweek.com/c/a/Application-Development/JetBrains-Strikes-Python-Developers-with-PyCharm-10-IDE-304127> [Accessed: 23 March 2015].
- Tatham,2007 Tatham, S. (2007). *Using public keys for SSH authentication*. [Online]. Available at:  
<http://the.earth.li/~sgtatham/putty/0.60/html/doc/Chapter8.html> [Accessed: 25 March 2015].
- Tatham,2011 Tatham, S. (2011). *Simon Tatham: About Me*. [Online]. Available at:  
<http://www.chiark.greenend.org.uk/~sgtatham/me.html> [Accessed: 24 March 2015].
- The Linux Foundation,2009 The Linux Foundation. (2009). *bridge | The Linux Foundation*. [Online]. Available at:  
<http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge> [Accessed: 22 April 2015].

- Tomicki,2012 Tomicki, L. (2012). *Attacking the Spanning-Tree Protocol*. [Online]. Available at: <http://tomicki.net/attacking.stp.php> [Accessed: 29 December 2014].
- Tripathi,2013 Tripathi, A. (2013). *What is Oracle VirtualBox?*. [Online]. Available at: <http://peoplesofttutorial.com/what-is-oracle-virtualbox/> [Accessed: 23 March 2015].
- University of Michigan,1997 University of Michigan. (1997). *The Python Programming Language*. [Online]. Available at: <http://groups.engin.umd.umich.edu/CIS/course.des/cis400/python/python.html> [Accessed: 24 March 2015].
- University of Virginia,2015 University of Virginia. (2015). *Chapter 5 Lan Switching Pdf download ~ dch360.com*. [Online]. Available at: <http://dch360.com/file/3c2b943> [Accessed: 7 April 2015].
- Webster,2006 Seth Webster, R. L. (2006). *Experience Using Active and Passive Mapping for Network Situational Awareness*. In: 2006, IEEE, p.19–26. [Online]. Available at: doi:10.1109/NCA.2006.23 [Accessed: 7 April 2015].
- Yeung et al.,2006 Yeung, K. H., Yan, F. and Leung, C. (2006). *Improving Network Infrastructure Security by Partitioning Networks Running Spanning Tree Protocol*. In: 2006, IEEE, p.19–19. [Online]. Available at: doi:10.1109/ICISP.2006.13 [Accessed: 28 April 2015].

## Bibliography

ActiveState Software Inc. (2015). *Komodo IDE -- One Cross-Platform IDE, All Your Languages*. [Online]. Available at: <http://komodoide.com/> [Accessed: 23 April 2015].

Anon. (2014). *002 - CentOS 6.5 Installation on VirtualBox*. [Online].



Available at: [https://www.youtube.com/watch?v=7pQIIQ-mY90&feature=youtube\\_gdata\\_player](https://www.youtube.com/watch?v=7pQIIQ-mY90&feature=youtube_gdata_player) [Accessed: 29 November 2014].

Artemjev, O. and Mjasnyankin, V. (2003). *Fun with the Spanning Tree Protocol*. [Online]. Available at: <http://phrack.org/issues/61/12.html> [Accessed: 7 April 2015].

Deibel, S. (2015). *IntegratedDevelopmentEnvironments - Python Wiki*. [Online]. Available at: <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments> [Accessed: 20 May 2015].

Etherton, E. (2013). *Building a Virtual Lab with VirtualBox for Penetration Testing and Hacking Tests - YouTube*. [Online]. Available at: <https://www.youtube.com/watch?v=AiWRmMzwwJM> [Accessed: 25 March 2015].

Fruit, J. (2014). *Comparison of Python IDEs for Development | Python Central*. [Online]. Available at: <http://www.pythoncentral.io/comparison-of-python-ides-development/> [Accessed: 20 May 2015].

Gordon, S. (2013). *Creating a Virtual Network of Linux Guests using VirtualBox | Steven Gordon*. [Online]. Available at: <https://sandilands.info/sgordon/creating-a-virtual-network-of-linux-guests-using-virtualbox> [Accessed: 25 March 2015].

Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A. and Pras, A. (2014). Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys & Tutorials*, 16 (4), p.2037–2064. [Online]. Available at: doi:10.1109/COMST.2014.2321898 [Accessed: 1 April 2015].

Pydev.org. (2015). *PyDev*. [Online]. Available at: <http://pydev.org/> [Accessed: 23 April 2015].

Kroger, P. (2014). Choosing the Best Python IDE. *Pedro Kroger*. [Online]. Available at: <http://pedrokroger.net/choosing-best-python-ide/> [Accessed: 20 May 2015].

Lutz, M. (1996). *Programming Python*, A Nutshell handbook. 1st ed. Bonn ; Sebastopol, CA: O'Reilly.

Marian Zburlea. (2014). *Install putty and generate ssh key to auto log in to Ubuntu server 14*. [Online]. Available at: [https://www.youtube.com/watch?v=SUswyZzqrUM&feature=youtu\\_gdata\\_player](https://www.youtube.com/watch?v=SUswyZzqrUM&feature=youtu_gdata_player) [Accessed: 11 February 2015].

Siebert, E. (2011). *Top 10 hypervisors: Choosing the best hypervisor technology*. [Online]. Available at: <http://searchservervirtualization.techtarget.com/tip/Top-10-hypervisors-Choosing-the-best-hypervisor-technology> [Accessed: 22 March 2015].

SourceForge. (2015). *BGINFO4X - BGINFO for X and for Windows! / Wiki / Documentation*. [Online]. Available at: <http://sourceforge.net/p/bginfo4x/wiki/Documentation/> [Accessed: 17 April 2015].

Tapia Gutierrez, S. (2013). *The Best Python IDEs You Can Use for Development | Python Central*. [Online]. Available at: <http://www.pythoncentral.io/the-best-python-ides-you-can-use-for-development/> [Accessed: 20 May 2015].

Watters, A. (1996). *Internet programming with Python*. New York: M&T Books.

## **Appendix A - configuration of network adapter using vboxmanage.exe**

The fifth network Adapter for Switch1 can be enabled/configured for an Internal Network using the following vboxmanage commands:

**1. To enable the fifth network adapter on Switch1 for an internal network named test\_network1**

```
C:\Users\User>vboxmanage modifyvm Switch1 --nic5 intnet
```

```
C:\Users\User>vboxmanage modifyvm Switch1 --intnet5 "test_network1"
```

**2. To enable VMs Promiscuous mode**

```
C:\Users\User>vboxmanage modifyvm Switch1 --nicpromisc5 allow-all
```

**3. Use other hardware type (Intel Pro/1000 MT Server)**

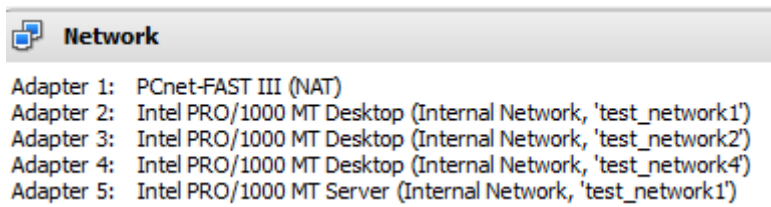
```
C:\Users\User>vboxmanage modifyvm Switch1 --nictype5 82545EM
```

**4. Disconnect cable**

```
C:\Users\User>vboxmanage modifyvm Switch1 --cableconnected5 off
```

**5. Connect cable**

```
C:\Users\User>vboxmanage modifyvm Switch1 --cableconnected5 on
```

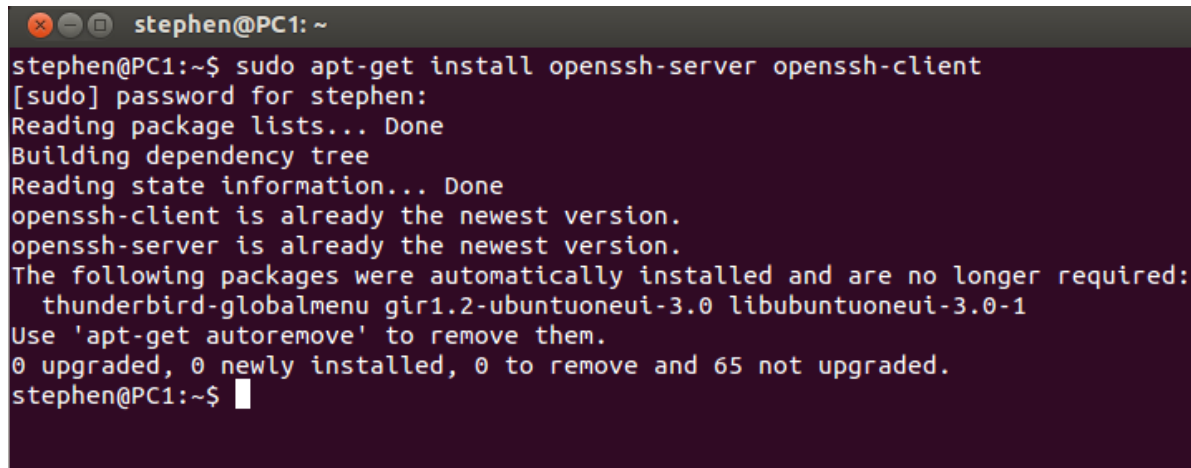


Note "Adapter 5", a fifth network adapter has now been added to Switch

## **Appendix B - SSH set-up between administrative PC and network switch**

This example illustrates the setup of a SSH password-less automatic login from PC1 (192.168.2.200) as user root to Switch4 (192.168.2.24) with user root.

**Step1:** Install OpenSSH on the PC1

A terminal window titled 'stephen@PC1: ~' showing the command 'sudo apt-get install openssh-server openssh-client' and its output. The output indicates that the packages are already installed and lists some automatically installed packages that are no longer required.

```
stephen@PC1:~$ sudo apt-get install openssh-server openssh-client
[sudo] password for stephen:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version.
openssh-server is already the newest version.
The following packages were automatically installed and are no longer required:
  thunderbird-globalmenu gir1.2-ubuntuoneui-3.0 libubuntuoneui-3.0-1
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 65 not upgraded.
stephen@PC1:~$
```

**Step 2:** Create Authentication SSH-Keygen Keys on PC1 (192.168.2.200)

First login into PC1 (192.168.2.200) with user root and generate a pair of public keys.

```
stephen@PC1: ~
stephen@PC1:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/stephen/.ssh/id_rsa):
/home/stephen/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/stephen/.ssh/id_rsa.
Your public key has been saved in /home/stephen/.ssh/id_rsa.pub.
The key fingerprint is:
25:2f:12:b2:db:69:a3:bd:f7:c2:1d:52:db:de:bb:07 stephen@PC1
The key's randomart image is:
+--[ RSA 2048]-----+
|
|   . . . .
|  o . +.
| . . S..o
| o o..o . E
| . =. o o . .
| + .+ . . . .
| . oo o.  o+
+-----+
stephen@PC1:~$
```

**Step 3:** Create .ssh Directory on Switch4 (192.168.2.24)

Use SSH from PC1 (192.168.2.200) to connect to Switch4 (192.168.2.24) using root as user and create .ssh directory under it.

```
[root@Switch4 ~]# ls -a
. 002.1d .bash_history .bash_profile .cshrc install.log.syslog .tcshrc
.. anaconda-ks.cfg .bash_logout .bashrc install.log .pki .viminfo
[root@Switch4 ~]#
```

```
stephen@PC1:~$ ssh root@192.168.2.24 mkdir -p .ssh
The authenticity of host '192.168.2.24 (192.168.2.24)' can't be established.
RSA key fingerprint is f8:2d:3b:4c:2a:06:97:77:3a:1c:5f:2e:a3:d1:98:f3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.24' (RSA) to the list of known hosts.
root@192.168.2.24's password:
```

```
[root@Switch4 ~]# ls -a
. 002.1d .bash_history .bash_profile .cshrc install.log.syslog .ssh .viminfo
.. anaconda-ks.cfg .bash_logout .bashrc install.log .pki .tcshrc
[root@Switch4 ~]#
```

#### Step 4: Upload Generated Public Keys to Switch4 (192.168.2.24)

Use SSH from PC1 (192.168.2.200) and upload new generated public key (id\_rsa.pub) on Switch4 (192.168.2.24) under root's .ssh directory as a file name authorized\_keys.

```
stephen@PC1:~$ cat .ssh/id_rsa.pub | ssh root@192.168.2.24 'cat >> .ssh/authorized_keys'
root@192.168.2.24's password:
```

#### Step 5: Set Permissions on Switch4 (192.168.2.24)

Due to different SSH versions on servers, we need to set permissions on .ssh directory and authorized\_keys file.



```
stephen@PC1:~$ ssh root@192.168.2.24 "chmod 700 .ssh; chmod 640 .ssh/authorized_keys"
```

**Step 6:** Login from PC1 (192.168.2.200) to Switch4 (192.168.2.24) without Password

```
stephen@PC1:~$ ssh root@192.168.2.24  
Last login: Tue Mar 24 15:07:33 2015  
[root@Switch4 ~]#
```

From now onwards you can log into Switch4 as root user from server PC1 as root user without a password.(Saive, 2012)

## **Appendix C - network switch configuration using BRCTL**

This procedure was used in order to setup Switch1 on the CentOS 6.6 minimal Install using BRCTL

**1. Install all needed packages:**

```
[root @Switch1 ~]#yum install bridge-utils
```

```
[root @Switch1 ~]#yum install tunctl
```

**2. Disable NetworkManager and enable network at boot time:**

```
[root @Switch1 ~]#chkconfig NetworkManager off
```

```
[root @Switch1 ~]#chkconfig --levels 35 network on
```

```
[root @Switch1 ~]#/etc/init.d/NetworkManager stop
```

```
[root @Switch1 ~]#/etc/init.d/network restart
```

**3. Create br0 configuration**

```
[root @Switch1 ~]#vi /etc/sysconfig/network-scripts/ifcfg-br0
```

```
DEVICE=br0

TYPE=Bridge

BOOTPROTO=static

ONBOOT=yes

STP =on

IPADDR=192.168.2.21

NETMASK=255.255.255.255

NETWORK=192.168.2.0

GATEWAY=0.0.0.0
```

#### 4. Create eth1,eth2,eth3,eth4 configuration

```
[root @Switch1 ~]#vi /etc/sysconfig/network-scripts /ifcfg-eth1
```

```
DEVICE=eth1

HWADDR=08:00:27:04:1A:20

TYPE=ethernet

BOOTPROTO=static

ONBOOT=yes

BRIDGE=br0

IPV6INIT=no
```

```
[root @Switch1 ~]#vi /etc/sysconfig/network-scripts /ifcfg-eth2
```

```
DEVICE=eth2

HWADDR=08:00:27:E3:15:33

TYPE=ethernet

BOOTPROTO=static

ONBOOT=yes

BRIDGE=br0

IPV6INIT=no
```

```
[root @Switch1 ~]#vi /etc/sysconfig/network-scripts /ifcfg-eth3
```

```
DEVICE=eth3  
  
HWADDR=08:00:27:D9:4F:16  
  
TYPE=ethernet  
  
BOOTPROTO=static  
  
ONBOOT=yes  
  
BRIDGE=br0  
  
IPV6INIT=no
```

```
[root @Switch1 ~]#vi /etc/sysconfig/network-scripts /ifcfg-eth4
```

```
DEVICE=eth4  
  
HWADDR=08:00:27:9B:123:9B  
  
TYPE=ethernet  
  
BOOTPROTO=static  
  
ONBOOT=yes  
  
BRIDGE=br0  
  
IPV6INIT=no
```

## 5. Restart network or reboot machine

```
[root@Switch1 ~]#/etc/init.d/network restart
```

```
[root@Switch1 ~]#reboot
```

## 6. Enable Spanning Tree Protocol

In order to enable and ensure that Spanning Tree Protocol (STP) is on the following BRCTL commands are used:

```
[root@Switch1 ~]# brctl stp br0 on
```

```
[root@Switch1 ~]#brctl show br0
```

```
[root@Switch1 ~]# brctl stp br0 on
[root@Switch1 ~]# brctl show br0
bridge name      bridge id        STP enabled      interfaces
br0              8000.080027041a20  yes              eth1
                                                         eth2
                                                         eth3
                                                         eth4

[root@Switch1 ~]#
```

You can see the STP parameters by entering the following:

```
[root@Switch1 ~]#brctl showstp br0
```

```

[root@Switch1 ~]# brctl showstp br0
br0
bridge id          8000.000027041a20
designated root    8000.000027041a20
root port         0
max age           19.99
hello time        1.99
forward delay     0.00
ageing time       299.95
hello timer       1.72
topology change timer 0.00
hash elasticity   4
mc last member count 2
mc router        1
mc last member timer 0.99
mc querier timer 254.96
mc response interval 9.99
path cost         0
bridge max age    19.99
bridge hello time 1.99
bridge forward delay 0.00
tcn timer         0.00
gc timer          0.72
hash max          512
mc init query count 2
mc snooping       1
mc membership timer 259.96
mc query interval 124.98
mc init query interval 31.24
flags

eth1 (0)
port id           0000
designated root    8000.000027041a20
designated bridge  8000.000027041a20
designated port    8001
designated cost    0
mc router        1
state             forwarding
path cost         4
message age timer 0.00
forward delay timer 0.00
hold timer        0.71
flags

eth2 (0)
port id           0000
designated root    8000.000027041a20
designated bridge  8000.000027041a20
designated port    8002
designated cost    0
mc router        1
state             forwarding
path cost         4
message age timer 0.00
forward delay timer 0.00
hold timer        0.71
flags

eth3 (0)
port id           0000
designated root    8000.000027041a20
designated bridge  8000.000027041a20
designated port    8003
designated cost    0
mc router        1
state             forwarding
path cost         4
message age timer 0.00
forward delay timer 0.00
hold timer        0.71
flags
[root@Switch1 ~]#

```



There are a number of parameters related to the STP that can be configured. It is important to note that the code auto detects the speed of the link and other parameters, so these usually don't need to be changed. For the purposes of this project the bridge priority of each switch has been left as the default value meaning the lowest MAC address determines the root bridge on this network.

## **7. Port mirroring**

The mirrored port is enabled by running script within the `"/etc/rc.local"` folder of the CentOS Linux OS.

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
ifconfig eth1 promisc
ifconfig eth2 promisc
ifconfig eth3 promisc

sif=eth1
tif=eth2
dif=eth3

#ingress eth1 to eth3
tc qdisc add dev "$sif" ingress
tc filter add dev "$sif" parent ffff: \
    protocol all \
    u32 match u8 0 0 \
    action mirred egress mirror dev "$dif"

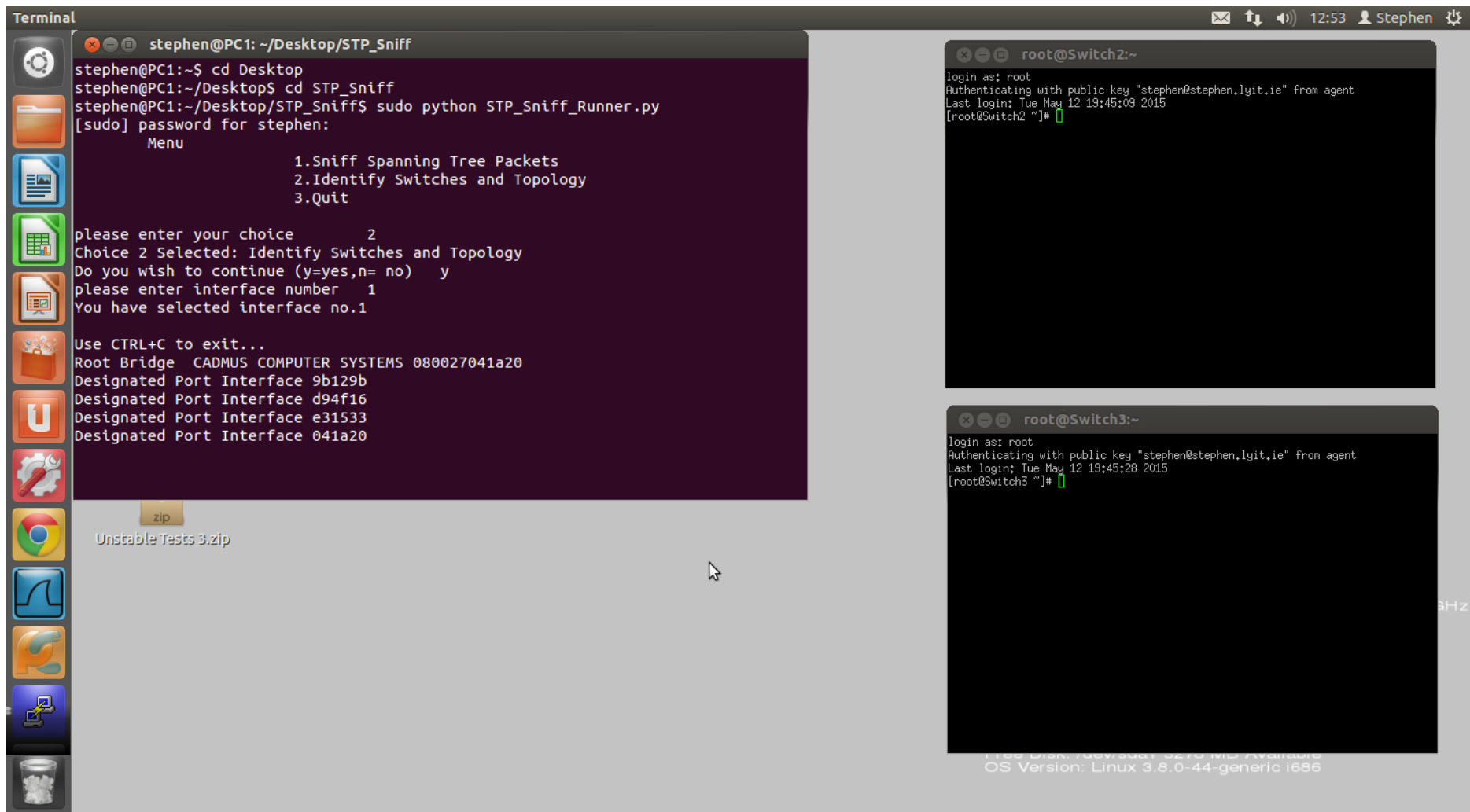
#egress eth1 to eth3
tc qdisc add dev "$sif" handle 1: root prio
tc filter add dev "$sif" parent 1: \
    protocol all \
    u32 match u8 0 0 \
    action mirred egress mirror dev "$dif"

#ingress eth2 to eth3
tc qdisc add dev "$tif" ingress
tc filter add dev "$tif" parent ffff: \
    protocol all \
    u32 match u8 0 0 \
    action mirred egress mirror dev "$dif"

#egress eth2 to eth3
tc qdisc add dev "$tif" handle 2: root prio
tc filter add dev "$tif" parent 2: \
    protocol all \
    u32 match u8 0 0 \
    action mirred egress mirror dev "$dif"

~
~
~
"/etc/rc.local" [readonly] 44L, 1018C
```

## **Appendix D - python script run on a stable network with no TCNs or TCAs**



Script run on PC1/Switch1 - Stable Network

```
stephen@PC3: ~/Desktop/STP_Sniff
stephen@PC3:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 041a20
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface a23e84
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface 9bfc27
*****
ATTENTION- 8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
```

Script run on PC3/Switch2 - Stable Network

```
stephen@PC4: ~/Desktop/STP_Sniff
stephen@PC4:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface e31533
Bridge ID 1 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface ff131e
Designated Port Interface 4c508a
Designated Port Interface 9bfc27
█
```

Script run on PC4/Switch3 -Stable Network

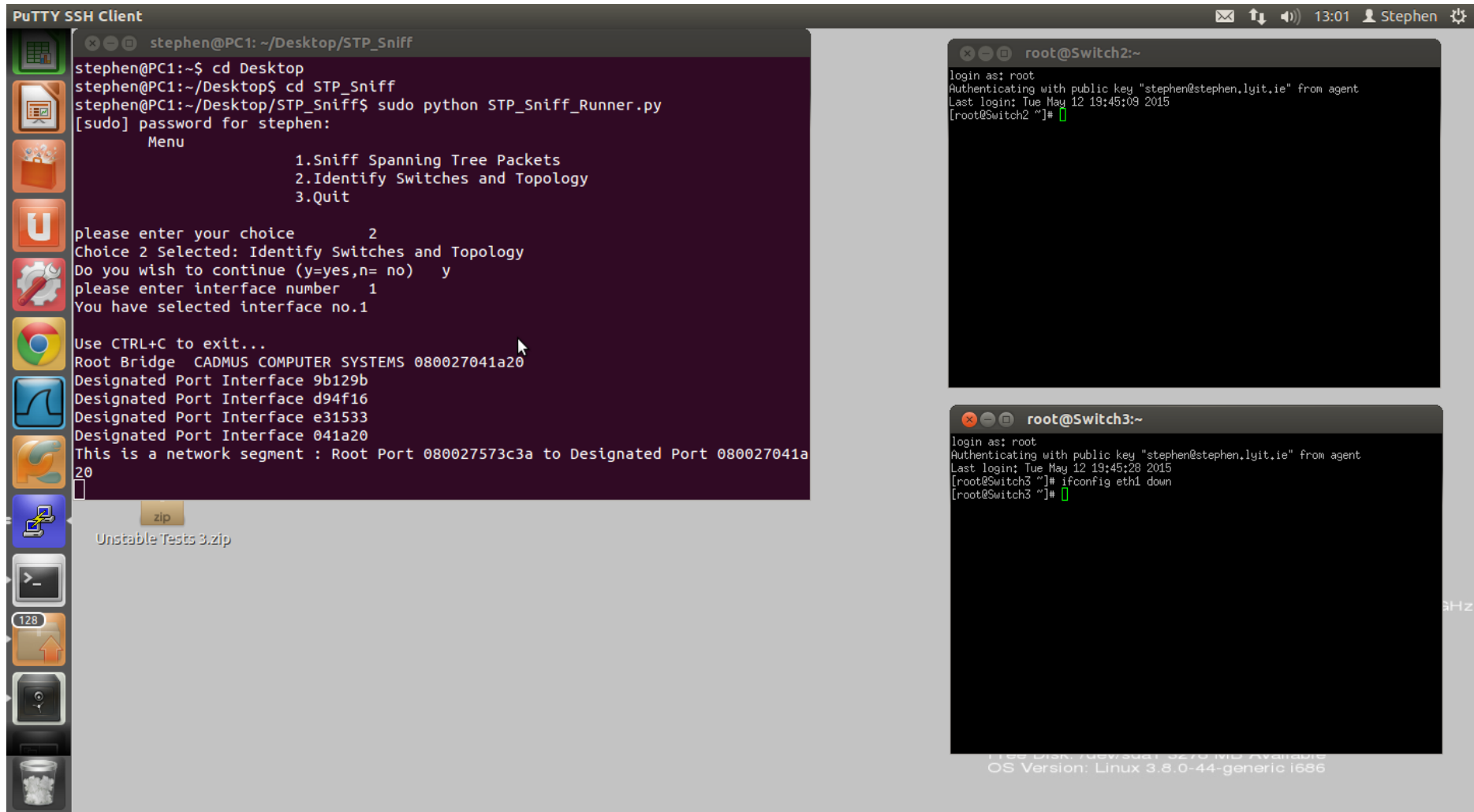
```
stephen@PC5: ~/Desktop/STP_Sniff
stephen@PC5:~$ cd Desktop
stephen@PC5:~/Desktop$ cd STP_Sniff
stephen@PC5:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  6
You have selected interface no.6
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 7d0776
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface fb72f5
Designated Port Interface 565b76
█
```

Script run on PC5/Switch4 - Stable Network

## **Appendix E - python script run on an unstable network with TCNs and TCAs**



**Test 1 Switch 3 eth1 down** (In this test adapter 7a:aa:35 is taken down on Switch 3.)



(Test1) Script PC1/Switch1 - Unstable Network

```
stephen@PC3: ~/Desktop/STP_Sniff
stephen@PC3:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice          2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number    4
You have selected interface no.4

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 041a20
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface a23e84
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface 9bfc27
*****
ATTENTION- 8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
Designated Port Interface fb4d4d
Root Port connected to this device from 0800279bfc27
Root Port from this device 080027573c3a
This is a network segment : Root Port 080027573c3a to Designated Port 080027041a20
□
```

(Test1) Script PC3/Switch2 - Unstable Network

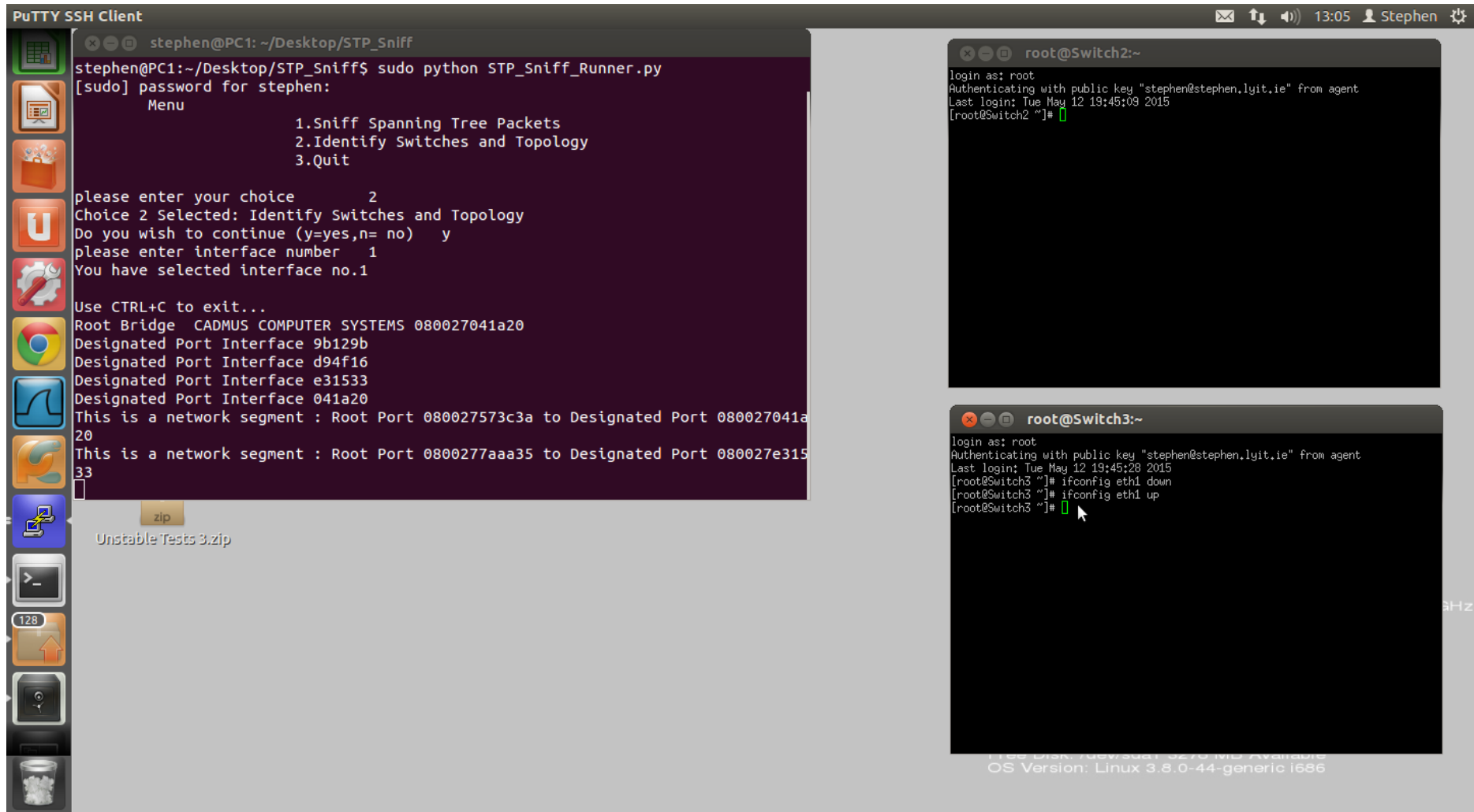
```
stephen@PC4: ~/Desktop/STP_Sniff
stephen@PC4:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice          2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number    4
You have selected interface no.4
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface e31533
Bridge ID 1 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface ff131e
Designated Port Interface 4c508a
Designated Port Interface 9bfc27
Bridge ID 2 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface fb4d4d
*****
ATTENTION-8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
```

(Test1) Script PC4/Switch3 - Unstable Network

```
stephen@PC5: ~/Desktop/STP_Sniff
stephen@PC5:~$ cd Desktop
stephen@PC5:~/Desktop$ cd STP_Sniff
stephen@PC5:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  6
You have selected interface no.6
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 7d0776
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface fb72f5
Designated Port Interface 565b76
```

(Test1) Script PC5/Switch4 -Unstable Network

Test 2 Switch 3 eth1 up (In this test adapter 7a:aa:35 is taken up on Switch 3).



(Test2) Script PC1/Switch1 - Unstable Network

```
stephen@PC3: ~/Desktop/STP_Sniff
stephen@PC3:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice          2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 041a20
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface a23e84
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface 9bfc27
*****
ATTENTION- 8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
Designated Port Interface fb4d4d
Root Port connected to this device from 0800279bfc27
Root Port from this device 080027573c3a
This is a network segment : Root Port 080027573c3a to Designated Port 080027041a20
```

(Test2) Script PC3/Switch2 - Unstable Network

```
stephen@PC4: ~/Desktop/STP_Sniff
stephen@PC4:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice          2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number     4
You have selected interface no.4
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface e31533
Bridge ID 1 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface ff131e
Designated Port Interface 4c508a
Designated Port Interface 9bfc27
Bridge ID 2 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface fb4d4d
*****
ATTENTION-8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
This is a network segment : Root Port 0800277aaa35 to Designated Port 080027e31533
```

(Test2) Script PC4/Switch3 - Unstable Network

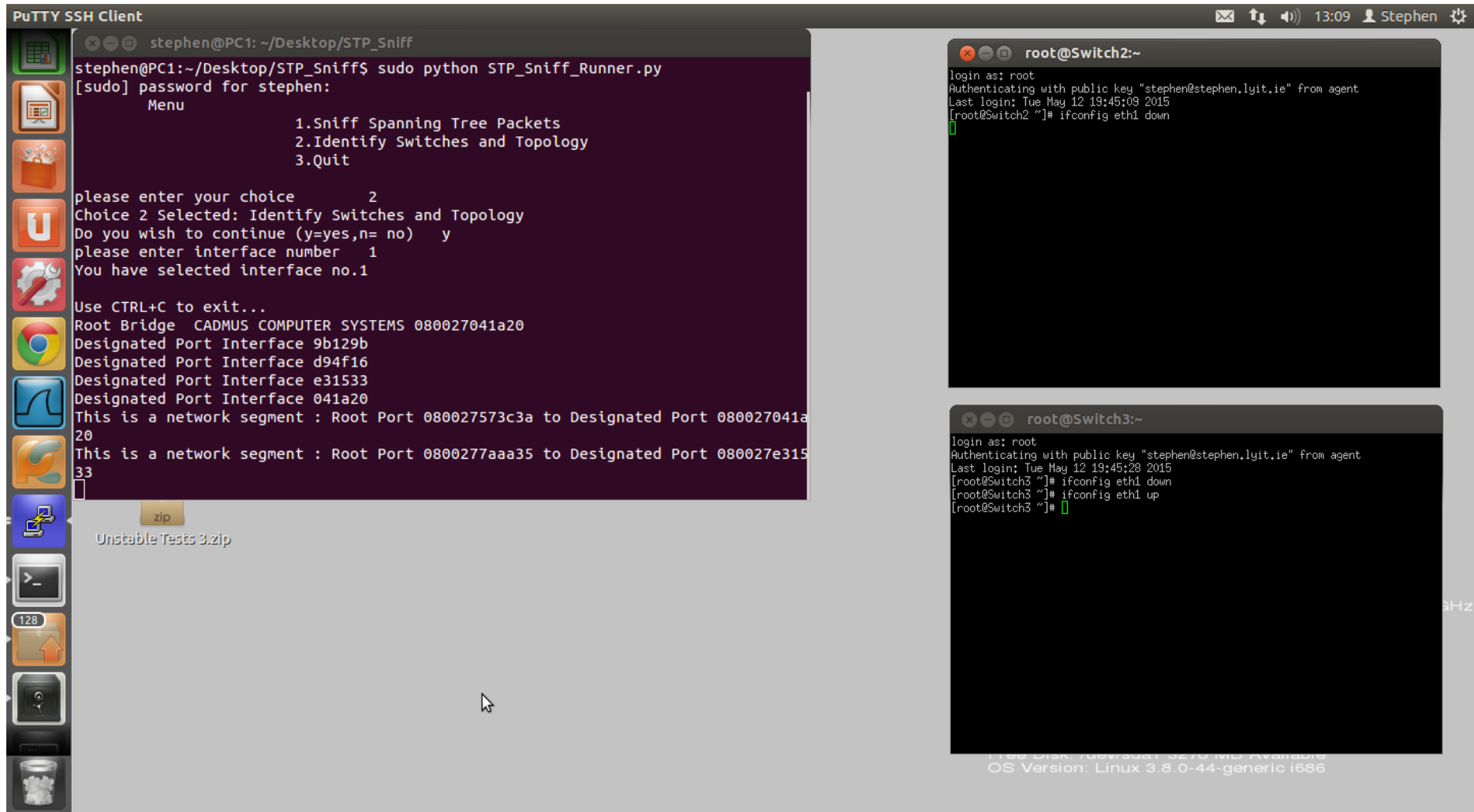
```
stephen@PC5: ~/Desktop/STP_Sniff
stephen@PC5:~$ cd Desktop
stephen@PC5:~/Desktop$ cd STP_Sniff
stephen@PC5:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  6
You have selected interface no.6

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 7d0776
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface fb72f5
Designated Port Interface 565b76
```

(Test2) Script PC5/Switch4 - Unstable Network



**Test 3 Switch 2 eth1 down** (In this test adapter 57:3c:3a is taken down on Switch 2).



(Test3) Script PC1/Switch1 -Unstable Network

xxxi

```
stephen@PC3: ~/Desktop/STP_Sniff
stephen@PC3:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit

please enter your choice          2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number    4
You have selected interface no.4

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 041a20
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface a23e84
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface 9bfc27
*****
ATTENTION- 8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
Designated Port Interface fb4d4d
Root Port connected to this device from 0800279bfc27
Root Port from this device 080027573c3a
This is a network segment : Root Port 080027573c3a to Designated Port 080027041a20
Root Bridge  CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface 16481a
Root Port connected to this device from 08002716481a
Root Port from this device 080027fb4d4d
This is a network segment : Root Port 080027fb4d4d to Designated Port 0800279bfc27
```

(Test3) Script PC3/Switch2 -Unstable Network

```
stephen@PC4: ~/Desktop/STP_Sniff
stephen@PC4:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice          2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number    4
You have selected interface no.4

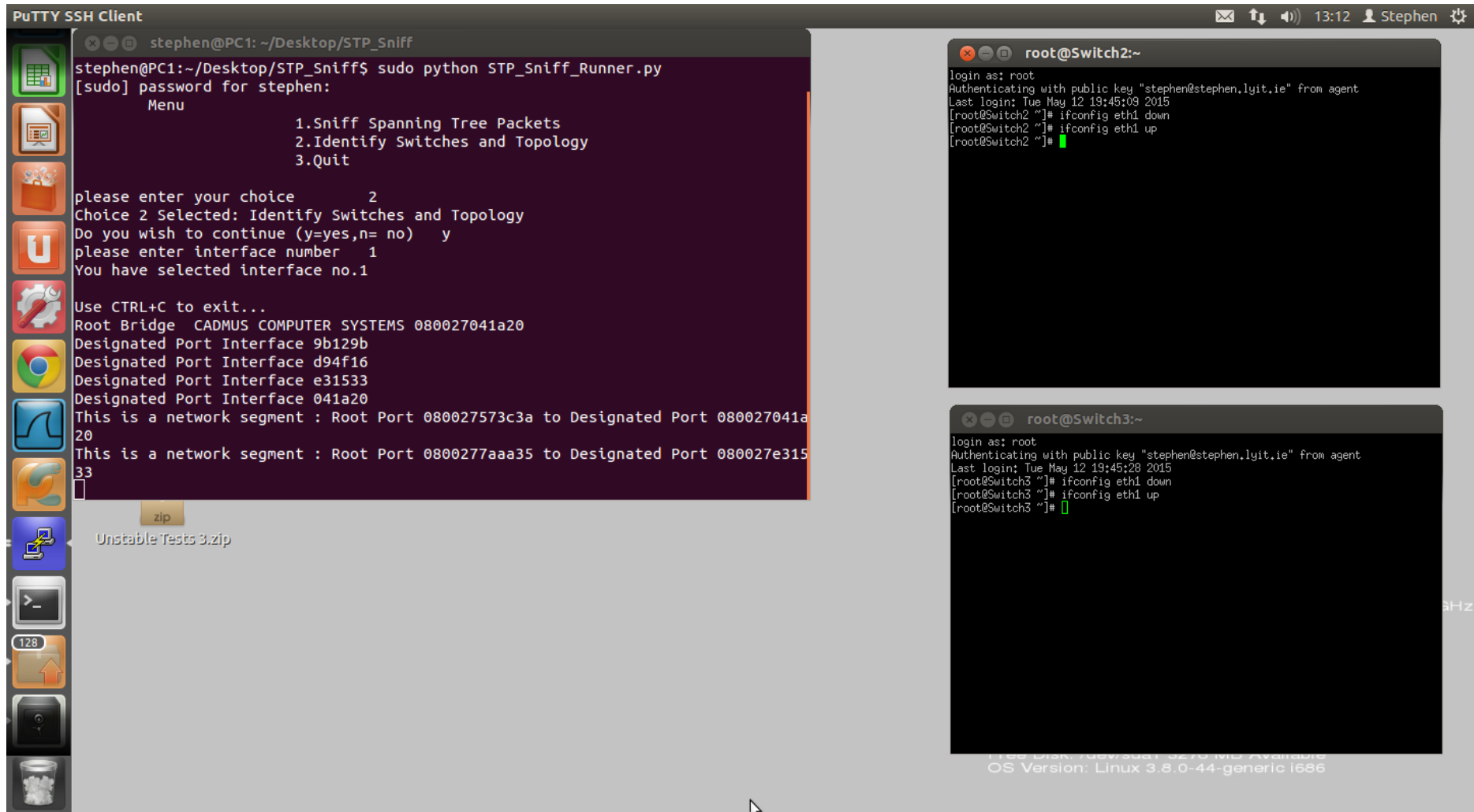
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface e31533
Bridge ID 1 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface ff131e
Designated Port Interface 4c508a
Designated Port Interface 9bfc27
Bridge ID 2 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface fb4d4d
*****
ATTENTION-8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
This is a network segment : Root Port 0800277aaa35 to Designated Port 080027e31533
Root Port connected to this device from 080027fb4d4d
Root Port from this device 0800277aaa35
█
```

(Test3) Script PC4/Switch3 - Unstable Network

```
stephen@PC5: ~/Desktop/STP_Sniff
stephen@PC5:~$ cd Desktop
stephen@PC5:~/Desktop$ cd STP_Sniff
stephen@PC5:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  6
You have selected interface no.6
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 7d0776
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface fb72f5
Designated Port Interface 565b76
Designated Port Interface 16481a
█
```

(Test3) Script PC5/Switch4 - Unstable Network

#### Test 4 Switch 2 eth1 up (In this test adapter 57:3c:3a is taken up on Switch 2)



(Test4) Script PC1/Switch1 - Unstable Network

XXXV

```
stephen@PC3: ~/Desktop/STP_Sniff
stephen@PC3:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit

please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 041a20
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface a23e84
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface 9bfc27
*****
ATTENTION- 800080027573c3a has a non-designated/blocked port receiving packets from 8000800274c508a
*****
Designated Port Interface fb4d4d
Root Port connected to this device from 0800279bfc27
Root Port from this device 080027573c3a
This is a network segment : Root Port 080027573c3a to Designated Port 080027041a20
Root Bridge  CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface 16481a
Root Port connected to this device from 08002716481a
Root Port from this device 080027fb4d4d
This is a network segment : Root Port 080027fb4d4d to Designated Port 0800279bfc27

```

(Test4) Script PC3/Switch2 - Unstable Network

```
stephen@PC4: ~/Desktop/STP_Sniff
stephen@PC4:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice          2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number    4
You have selected interface no.4
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface e31533
Bridge ID 1 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface ff131e
Designated Port Interface 4c508a
Designated Port Interface 9bfc27
Bridge ID 2 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface fb4d4d
*****
ATTENTION-8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
This is a network segment : Root Port 0800277aaa35 to Designated Port 080027e31533
Root Port connected to this device from 080027fb4d4d
Root Port from this device 0800277aaa35
█
```

(Test4) Script PC4/Switch3 -Unstable Network

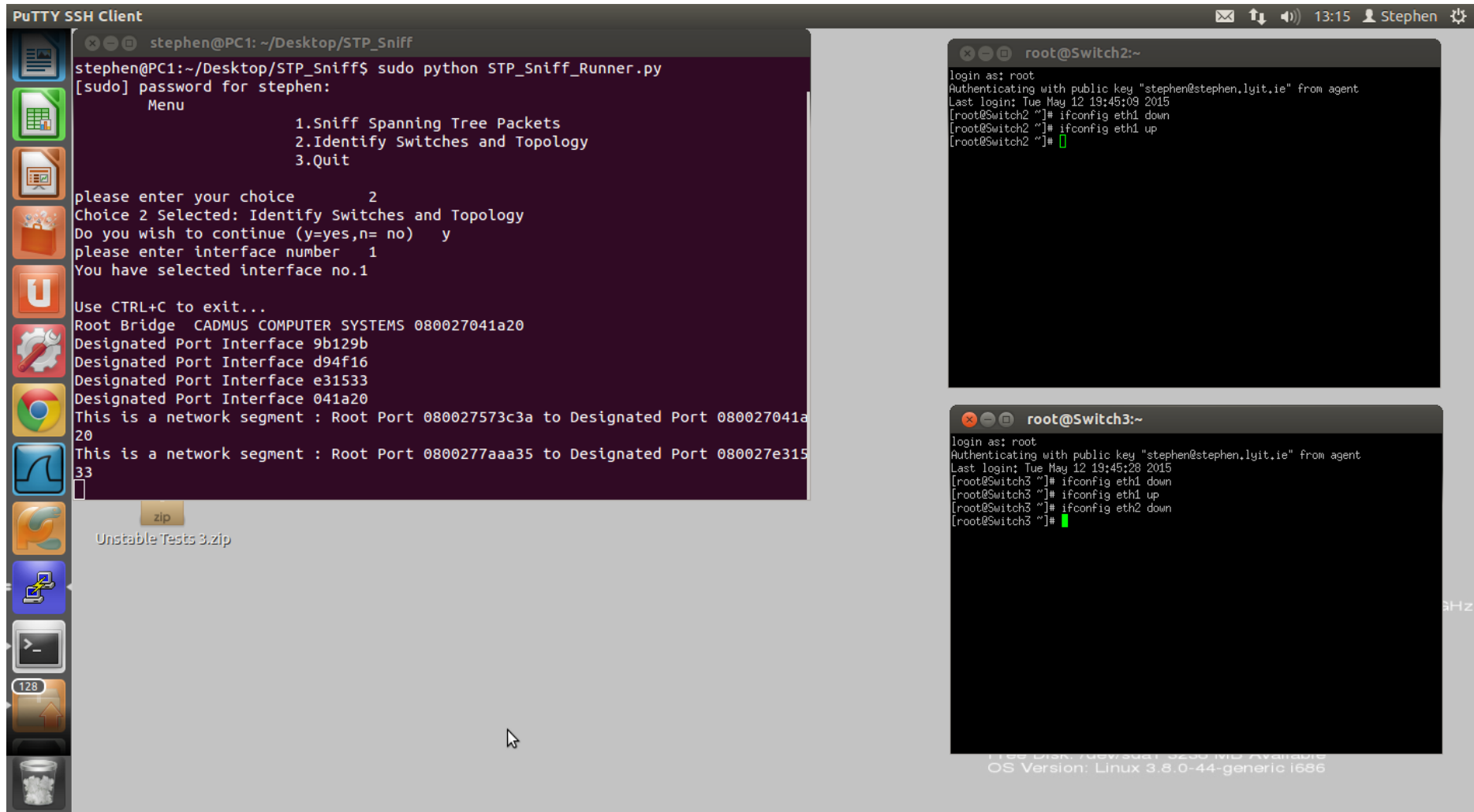
```
stephen@PC5: ~/Desktop/STP_Sniff
stephen@PC5:~$ cd Desktop
stephen@PC5:~/Desktop$ cd STP_Sniff
stephen@PC5:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  6
You have selected interface no.6

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 7d0776
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface fb72f5
Designated Port Interface 565b76
Designated Port Interface 16481a
█
```

(Test4) Script PC5/Switch4 - Unstable Network



### Test 5 Switch 3 eth2 down (In this test adapter 9b:fc:27 is taken down on Switch 3)



(Test5) Script PC1/Switch1 -Unstable Network

```
stephen@PC3: ~/Desktop/STP_Sniff
stephen@PC3:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 041a20
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface a23e84
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface 9bfc27
*****
ATTENTION- 800080027573c3a has a non-designated/blocked port receiving packets from 8000800274c508a
*****
Designated Port Interface fb4d4d
Root Port connected to this device from 0800279bfc27
Root Port from this device 080027573c3a
This is a network segment : Root Port 080027573c3a to Designated Port 080027041a20
Root Bridge  CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface 16481a
Root Port connected to this device from 08002716481a
Root Port from this device 080027fb4d4d
This is a network segment : Root Port 080027fb4d4d to Designated Port 0800279bfc27

```

(Test5) Script PC3/Switch2 - Unstable Network

```
stephen@PC4: ~/Desktop/STP_Sniff
stephen@PC4:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice          2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface e31533
Bridge ID 1 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface ff131e
Designated Port Interface 4c508a
Designated Port Interface 9bfc27
Bridge ID 2 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface fb4d4d
*****
ATTENTION-8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
This is a network segment : Root Port 0800277aaa35 to Designated Port 080027e31533
Root Port connected to this device from 080027fb4d4d
Root Port from this device 0800277aaa35

```

(Test5) Script PC4/Switch3 - Unstable Network

```
stephen@PC5: ~/Desktop/STP_Sniff
stephen@PC5:~$ cd Desktop
stephen@PC5:~/Desktop$ cd STP_Sniff
stephen@PC5:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  6
You have selected interface no.6
Use CTRL+C to exit...
Root Bridge CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 7d0776
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface fb72f5
Designated Port Interface 565b76
Designated Port Interface 16481a
█
```

(Test5) Script PC5/Switch4 - Unstable Network

## Test 6 Switch 3 eth2 up (In this test adapter 9b:fc:27 is taken up on Switch 3 )

The screenshot displays a PuTTY SSH Client window with three terminal panes. The left pane shows the execution of a Python script on PC1, which identifies network segments and lists designated ports. The top-right pane shows the configuration of interface eth1 on Switch2, and the bottom-right pane shows the configuration of interfaces eth1 and eth2 on Switch3.

```
stephen@PC1: ~/Desktop/STP_Sniff
stephen@PC1:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  1
You have selected interface no.1
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 9b129b
Designated Port Interface d94f16
Designated Port Interface e31533
Designated Port Interface 041a20
This is a network segment : Root Port 080027573c3a to Designated Port 080027041a20
This is a network segment : Root Port 0800277aaa35 to Designated Port 080027e31533
33

root@Switch2:~
login as: root
Authenticating with public key "stephen@stephen.lyit.ie" from agent
Last login: Tue May 12 19:45:09 2015
[root@Switch2 ~]# ifconfig eth1 down
[root@Switch2 ~]# ifconfig eth1 up
[root@Switch2 ~]#

root@Switch3:~
login as: root
Authenticating with public key "stephen@stephen.lyit.ie" from agent
Last login: Tue May 12 19:45:28 2015
[root@Switch3 ~]# ifconfig eth1 down
[root@Switch3 ~]# ifconfig eth1 up
[root@Switch3 ~]# ifconfig eth2 down
[root@Switch3 ~]# ifconfig eth2 up
[root@Switch3 ~]#

OS Version: Linux 3.8.0-44-generic i686
```

(Test6) Script PC1/Switch1 - Unstable Network

```
stephen@PC3: ~/Desktop/STP_Sniff
stephen@PC3:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit

please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 041a20
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface a23e84
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface 9bfc27
*****
ATTENTION- 800080027573c3a has a non-designated/blocked port receiving packets from 8000800274c508a
*****
Designated Port Interface fb4d4d
Root Port connected to this device from 0800279bfc27
Root Port from this device 080027573c3a
This is a network segment : Root Port 080027573c3a to Designated Port 080027041a20
Root Bridge  CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface 16481a
Root Port connected to this device from 08002716481a
Root Port from this device 080027fb4d4d
This is a network segment : Root Port 080027fb4d4d to Designated Port 0800279bfc27

```

(Test6) Script PC3/Switch2 - Unstable Network

```
stephen@PC4: ~/Desktop/STP_Sniff
stephen@PC4:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit
please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  4
You have selected interface no.4
Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface e31533
Bridge ID 1  CADMUS COMPUTER SYSTEMS 0800274c508a
Designated Port Interface ff131e
Designated Port Interface 4c508a
Designated Port Interface 9bfc27
Bridge ID 2  CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface fb4d4d
*****
ATTENTION-8000080027573c3a has a non-designated/blocked port receiving packets from 80000800274c508a
*****
This is a network segment : Root Port 0800277aaa35 to Designated Port 080027e31533
Root Port connected to this device from 080027fb4d4d
Root Port from this device 0800277aaa35
█
```

(Test6) Script PC4/Switch3 - Unstable Network

```
stephen@PC5: ~/Desktop/STP_Sniff
stephen@PC5:~$ cd Desktop
stephen@PC5:~/Desktop$ cd STP_Sniff
stephen@PC5:~/Desktop/STP_Sniff$ sudo python STP_Sniff_Runner.py
[sudo] password for stephen:
Menu
    1.Sniff Spanning Tree Packets
    2.Identify Switches and Topology
    3.Quit

please enter your choice      2
Choice 2 Selected: Identify Switches and Topology
Do you wish to continue (y=yes,n= no)  y
please enter interface number  6
You have selected interface no.6

Use CTRL+C to exit...
Root Bridge  CADMUS COMPUTER SYSTEMS 080027041a20
Designated Port Interface 7d0776
Bridge ID 1 CADMUS COMPUTER SYSTEMS 080027573c3a
Designated Port Interface 7d0776
Bridge ID 2 CADMUS COMPUTER SYSTEMS 08002716481a
Designated Port Interface fb72f5
Designated Port Interface 565b76
Designated Port Interface 16481a

```

(Test6) Script PC5/Switch4 - Unstable Network