# TNE30019 – Unix for Telecommunications

*Learning Summary Report*

*Michael Reeb*

## Introduction

This report summarises what I have learnt in Unix for Telecommunications. It includes a summary of the completed tasks within the Unit Portfolio and a reflection on my learning.

## Assessment Summary

Please complete the tables below to summarise completion of individual tasks in this Portfolio

| | Pass | Credit | Distinction | High Distinction | | |
|---|---|---|---|---|---|---|
| | | | | HD1 | HD2 | HD3 |
| Lab Report 1 | x | | | | | |
| Lab Report 2 | | x | | | | |
| Lab Report 3 | | | x | | | |
| Jails Report | | | x | | | |
| Research Report | | | | | x | |
| Project | | | | | | x |
| Presentation | | | x | | | |

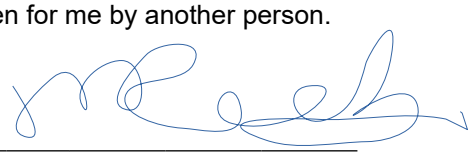
*Graded Portfolio Tasks*

| | Included (please tick) |
|---|---|
| Pass Tasks (14 tasks) | |
| Credit Tasks (3 tasks) | |
| Distinction Tasks (5 tasks) | |

*Non-Graded Portfolio Tasks*

## Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature: 

# Reflection

## The most important things I learnt:

This unit taught me the fundamentals of Unix from command line. It has removed apprehension and uncertainty that I have faced when using *nix previously.

Entering this unit I had never used Python or Bash before, so learning these two scripting languages was very useful and has increased the amount of programming tools in my tool-chest.

## The things that helped me most were:

Google, various textbooks, and Jonathon.

## I found the following topics particularly challenging:

CUPS; printers are evil.

## I found the following topics particularly interesting:

I really enjoyed learning about the underlying operating system structure. I learnt a lot about how computer operating systems work this semester.

I also really enjoyed the Nmap and TCPDump modules. It was interesting doing some basic port-scanning and network analysis. I might even do some further reading into network security when I find the time.

It was also interesting learning about all the different Unix systems. I am amazed about how easy it is to get applications to run (once you know what to do of course), and how compatible everything is.

That said, I also found that it can be very tedious trying to navigate poorly documented obfuscated manuals and the like, especially when it comes to niche libraries and modules ( I am looking at you Apple).

I also found the research papers (Jails and my own topic) to be very interesting. I really enjoy learning, so reading papers is fun.

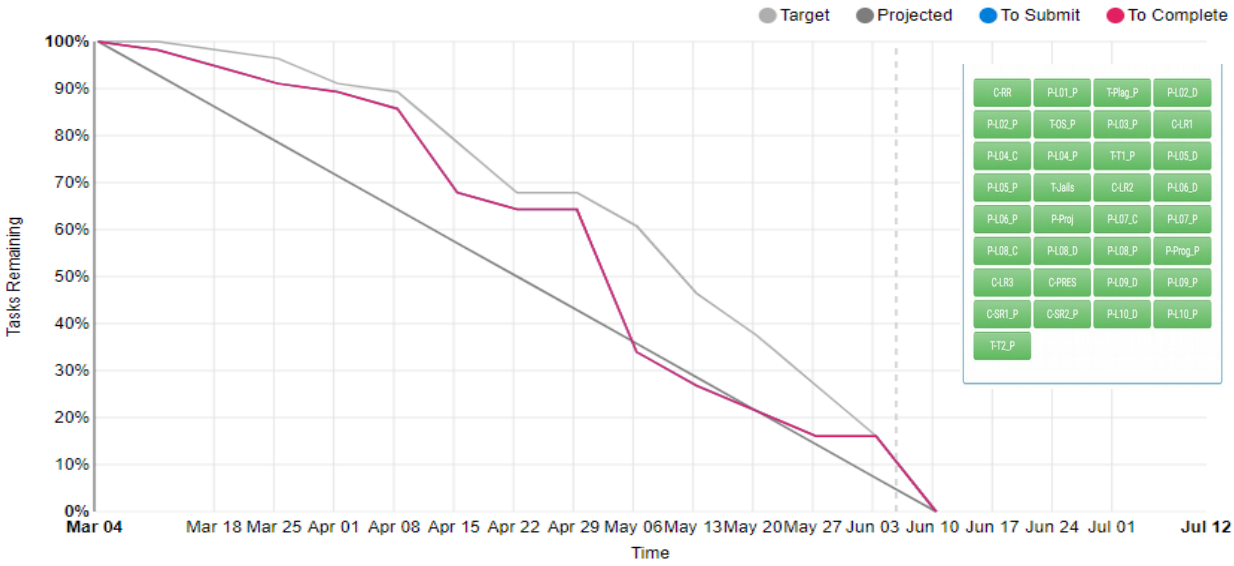
## I feel I learnt these topics, concepts, and/or tools really well:

- I learnt to use Python to beginner level, and learnt about some of its limitations.
- I think that I learnt the course content well, but I am well aware that the course was just a brief introduction to different Unix applications and systems. I am not going to claim expertise in any of these topics.

## I still need to work on the following areas:

Everything really. We only had introductions to all of the topics, and any more understanding of these would be useful.

## My progress in this unit was …:




## This unit will help me in the future:

This unit has made me comfortable enough and confident to now to start using command line *nix in my own projects. I have a few raspberry Pis that I intend on using to their limits.

I also want to build a home server system, and find an openwrt compatible router so I can play around with it and see if I can optimise my home network.

I really am happy to have had a course on command-line Unix; exposure to this will benefit me greatly.

## If I did this unit again I would do the following things differently:

I don't think I would do much differently. I should have put a bit more effort into the Research Report to get the 3rd tier HD, but I wouldn't know what to improve without pinpoint feedback. I am also livid that I got the question wrong on Test 1 about the FreeBSD '&&' vs '| |', as I got them the wrong way around in the test ( I even wrote them both down on the paper).

I should have done better on Test 2. Next time I would have to study more.

I really gave this unit as much effort as I was willing to commit. As the saying goes: "I did my best, I have no regrets".

Swinburne University of Technology

Unix for Telecommunications

Michael Reeb

# Portfolio Submission

*Submitted By:*
Michael Reeb
4936094

*Tutor:*
Jonathan Kua

June 5, 2019

# Contents

## 2 Overall Task Status

| Task | Status | Times Assessed |
|---|---|---|
| Communications - Research Report | Complete | 1 |
| Practical - Lab 1: Introduction to Unix | Complete | 1 |
| Theory - Plagiarism Test (P) | Complete | 1 |
| Practical - Lab 2: LaTeX (D) | Complete | 1 |
| Practical - Lab 2: LaTeX (P) | Complete | 1 |
| Theory - Operating Systems (P) | Complete | 1 |
| Practical - Lab 3: Bind (P) | Complete | 1 |
| Communications - Lab Report 1 | Complete | 1 |
| Practical - Lab 4: Basic Apache (C) | Complete | 1 |
| Practical - Lab 4: Basic Apache (P) | Complete | 1 |
| Theory - Test 1 (P) | Complete | 1 |
| Practical - Lab 5: Advanced Apache (D) | Complete | 1 |
| Practical - Lab 5: Advanced Apache (P) | Complete | 1 |
| Theory - Jails | Complete | 1 |
| Communications - Lab Report 2 | Complete | 1 |
| Practical - Lab 6: PCAP Programming (D) | Complete | 1 |
| Practical - Lab 6: PCAP Programming (P) | Complete | 1 |
| Practical - Project | Complete | 1 |
| Practical - Lab 7: NMap (C) | Complete | 1 |
| Practical - Lab 7: NMap (P) | Complete | 1 |
| Practical - Lab 8: TCPDump (C) | Complete | 1 |
| Practical - Lab 8: TCPDump (D) | Complete | 1 |
| Practical - Lab 8: TCPDump (P) | Complete | 1 |
| Practical - Programming Tutorial | Complete | 2 |
| Communications - Lab Report 3 | Complete | 1 |
| Communications - Project Presentation | Complete | 1 |
| Practical - Lab 9: Samba (D) | Complete | 1 |
| Practical - Lab 9: Samba (P) | Complete | 1 |
| Communications - Speaker Reflection 1 | Complete | 1 |
| Communications - Speaker Reflection 2 | Complete | 1 |
| Practical - Lab 10: CUPS (D) | Complete | 1 |
| Practical - Lab 10: CUPS (P) | Complete | 1 |
| Theory - Test 2 (P) | Complete | 1 |

# 3 Learning Outcomes

## 3.1 ULO1

Use navigation processes in an installed Unix System

| Task | Rating | Status | Times Assessed |
|---|---|---|---|
| Practical - Lab 1: Introduction to Unix | ♦♦♦◊◊ | Complete | 1 |
| Theory - Plagiarism Test (P) | ♦♦◊◊◊ | Complete | 1 |
| Theory - Test 1 (P) | ♦♦◊◊◊ | Complete | 1 |
| Practical - Project | ♦♦♦♦◊ | Complete | 1 |
| Theory - Test 2 (P) | ♦♦◊◊◊ | Complete | 1 |

## 3.2 ULO2

Appreciate the operation of a Unix-based Operating System

| Task | Rating | Status | Times Assessed |
|---|---|---|---|
| Theory - Operating Systems (P) | ♦♦♦♦◊ | Complete | 1 |
| Practical - Lab 3: Bind (P) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 4: Basic Apache (C) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 4: Basic Apache (P) | ♦♦♦◊◊ | Complete | 1 |
| Theory - Test 1 (P) | ♦♦◊◊◊ | Complete | 1 |
| Practical - Project | ♦♦♦♦◊ | Complete | 1 |
| Theory - Test 2 (P) | ♦♦◊◊◊ | Complete | 1 |

## 3.3 ULO3

Conduct the administration of a Unix server or workstation

| Task | Rating | Status | Times Assessed |
|---|---|---|---|
| Theory - Test 1 (P) | ♦♦◊◊◊ | Complete | 1 |
| Practical - Project | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 6: PCAP Programming (D) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 6: PCAP Programming (P) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 7: NMap (C) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 7: NMap (P) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 8: TCPDump (C) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 8: TCPDump (D) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 8: TCPDump (P) | ♦♦♦◊◊ | Complete | 1 |
| Theory - Test 2 (P) | ♦♦◊◊◊ | Complete | 1 |

## 3.4 ULO4

Configure common network services, devices and security

| Task | Rating | Status | Times Assessed |
|---|---|---|---|
| Theory - Test 1 (P) | ♦♦◊◊◊ | Complete | 1 |
| Practical - Lab 5: Advanced Apache (D) | ♦♦♦♦♦ | Complete | 1 |
| Practical - Lab 5: Advanced Apache (P) | ♦♦♦♦♦ | Complete | 1 |
| Practical - Project | ♦♦♦♦◊ | Complete | 1 |
| Practical - Programming Tutorial | ♦♦♦♦◊ | Complete | 2 |
| Communications - Project Presentation | ♦♦♦♦◊ | Complete | 1 |
| Practical - Lab 9: Samba (D) | ♦♦♦♦◊ | Complete | 1 |
| Practical - Lab 9: Samba (P) | ♦♦♦♦◊ | Complete | 1 |
| Theory - Test 2 (P) | ♦♦◊◊◊ | Complete | 1 |

### 3.5 ULO5

Demonstrate the use of administration tools on Unix systems

| Task | Rating | Status | Times Assessed |
|---|---|---|---|
| Theory - Test 1 (P) | ♦♦◊◊◊ | Complete | 1 |
| Communications - Lab Report 2 | ♦♦♦♦◊ | Complete | 1 |
| Theory - Jails | ♦♦♦♦♦ | Complete | 1 |
| Practical - Project | ♦♦♦♦◊ | Complete | 1 |
| Theory - Test 2 (P) | ♦♦◊◊◊ | Complete | 1 |

### 3.6 ULO6

Design and construct unfamiliar services

| Task | Rating | Status | Times Assessed |
|---|---|---|---|
| Practical - Lab 10: CUPS (P) | ♦♦♦♦♦ | Complete | 1 |
| Practical - Lab 10: CUPS (D) | ♦♦♦♦♦ | Complete | 1 |
| Practical - Project | ♦♦♦♦◊ | Complete | 1 |
| Theory - Test 2 (P) | ♦♦◊◊◊ | Complete | 1 |
| Theory - Test 1 (P) | ♦♦◊◊◊ | Complete | 1 |

### 3.7 ULO7

Generate documentation for laboratory work and a research assignment

| Task | Rating | Status | Times Assessed |
|---|---|---|---|
| Communications - Research Report | ♦♦♦♦♦ | Complete | 1 |
| Practical - Lab 2: LaTeX (D) | ♦♦♦◊◊ | Complete | 1 |
| Practical - Lab 2: LaTeX (P) | ♦♦♦◊◊ | Complete | 1 |
| Theory - Operating Systems (P) | ♦♦♦♦◊ | Complete | 1 |
| Communications - Lab Report 1 | ♦♦♦♦◊ | Complete | 1 |
| Communications - Speaker Reflection 1 | ♦♦♦♦◊ | Complete | 1 |
| Theory - Test 1 (P) | ♦♦◊◊◊ | Complete | 1 |
| Communications - Lab Report 2 | ♦♦♦♦◊ | Complete | 1 |
| Practical - Project | ♦♦♦♦◊ | Complete | 1 |
| Communications - Lab Report 3 | ♦♦♦♦◊ | Complete | 1 |
| Communications - Project Presentation | ♦♦♦◊◊ | Complete | 1 |
| Communications - Speaker Reflection 2 | ♦♦♦♦◊ | Complete | 1 |
| Theory - Test 2 (P) | ♦♦◊◊◊ | Complete | 1 |

# 4 Practical - Lab 2: LaTeX (D)

Lab task - LaTeX (Distinction)

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 2: LaTeX (D)

*Submitted By:*
Michael Reeb
4936094
2019/03/20 11:41

*Tutor:*
Jonathan Kua

March 20, 2019

# Sample Title

Sum Gai*
Ann Otherguy*
sumgai@corporation.com
aotherguy@coperation.com
FSET Swinburne
Hawthorn, VIC

## ABSTRACT

This is the abstract. admire its abstractness. Nam id fermentum dui. Suspendisse sagittis tortor a nulla mollis, in pulvinar ex pretium. Sed interdum orci quis metus euismod, et sagittis enim maximus. Vestibulum gravida massa ut felis suscipit congue. Quisque mattis elit a risus ultrices commodo venenatis eget dui. Etiam sagittis eleifend elementum.

Nam interdum magna at lectus dignissim, ac dignissim lorem rhoncus. Maecenas eu arcu ac neque placerat aliquam. Nunc pulvinar massa et mattis lacinia.

## KEYWORDS

component, formatting, style, styling, insert

**ACM Reference Format:**
Sum Gai and Ann Otherguy. 2018. Sample Title. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

## 1 SECTION

This is an example of a section.

### 1.1 Subsection

This is an example of a subsection

*1.1.1 Subsubsection.* this is an example of a subsubsection. **Now we are in bold font**!

## 2 LISTS

### 2.1 Itemised List

(1) This is the first item
(2) This is the second item
(3) This is the third item
(4) The list goes on
  (a) this list is nested in another list
  (b) WOW
    (i) and now we get deeper
    (ii) This text is underlined!
(5) *This line is in italics*

### 2.2 Bullet List

- This list uses bullets
- BANG
  - Now we step in again
  - pow
    - THIS IS IN SMALL CAPS
    - hello

## 3 IMAGE/FIGURE INCLUDING REFERENCING

$$x = \sqrt{\pi + e} \int_0^\infty \frac{\sinh \theta}{\theta^2} d\theta \tag{1}$$

**Figure 1: This is an equation**

**Figure 2: This is an picture. Seems familiar eh?**

This Figure above (Figure 1) can be referenced by using the `\ref{label}` command. As can the picture which can be seen in Figure 2.

*Both authors contributed equally to this research.


*Woodstock '18, June 03–05, 2018, Woodstock, NY*

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

## 4 TABLE

**Table 1: This is a table.**

| Column 1 | Column 2 | Column 3 |
|---|---|---|
| Hello<br>but under | I am a pretty<br>this line | nice table<br>is a hard line |
| now | lets merge | some columns |
| Merged column takes up 2 spots | | purple |
| this one takes up all three yeahhhh | | |

We can also reference tables similarly to what we did in Section 3, by labelling and referencing. eg Table 1.

## REFERENCING

Acknowledging works similar to referencing, but we use the `\cite{bibID}` command instead [**?** ]. These can be used anywhere in a document. Note the LaTeXjargon regarding cite and referencing means different things. referencing references a label, while cite cites the bibliography

## ACKNOWLEDGMENTS

To Unix, for being useful. To Jonathan for randomly choosing the same picture as me.

## A RESEARCH METHODS

### A.1 Part One

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi malesuada, quam in pulvinar varius, metus nunc fermentum urna, id sollicitudin purus odio sit amet enim. Aliquam ullamcorper eu ipsum vel mollis. Curabitur quis dictum nisl. Phasellus vel semper risus, et lacinia dolor. Integer ultricies commodo sem nec semper.

### A.2 Part Two

Etiam commodo feugiat nisl pulvinar pellentesque. Etiam auctor sodales ligula, non varius nibh pulvinar semper. Suspendisse nec lectus non ipsum convallis congue hendrerit vitae sapien. Donec at laoreet eros. Vivamus non purus placerat, scelerisque diam eu, cursus ante. Etiam aliquam tortor auctor efficitur mattis.

## B ONLINE RESOURCES

Nam id fermentum dui. Suspendisse sagittis tortor a nulla mollis, in pulvinar ex pretium. Sed interdum orci quis metus euismod, et sagittis enim maximus. Vestibulum gravida massa ut felis suscipit congue. Quisque mattis elit a risus ultrices commodo venenatis eget dui. Etiam sagittis eleifend elementum.

Nam interdum magna at lectus dignissim, ac dignissim lorem rhoncus. Maecenas eu arcu ac neque placerat aliquam. Nunc pulvinar massa et mattis lacinia.

## REFERENCES

[1] D. Doom, M. Evil, and I. Dominate, "Using LaTex for world domination," Soc. World Enslavement For Evil Masterminds. Bermuda, vol. A247, pp. 529–551, April 2009.

# 5 Practical - Lab 2: LaTeX (P)

Lab task - LaTeX

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 2: LaTeX (P)

*Submitted By:*
Michael Reeb
4936094
2019/03/15 14:38

*Tutor:*
Jonathan Kua

March 15, 2019

# Sample Title

1st Author 1
*Author 1 Organisation*
*name of organisation*
City, Country
email address

Author 2
*Author 2 Organisation*
*name of organisation*
City, Country
email address

***Abstract*—This is the abstract. admire its boldness. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.**

***Index Terms*—component, formatting, style, styling, insert**

## I. Section

This is an example of a section.

### A. Subsection

This is an example of a subsection

*1) Subsubsection:* this is an example of a subsubsection. **Now we are in bold font**!

## II. Lists

### A. Itemised List

1) This is the first item
2) This is the second item
3) This is the third item
4) The list goes on
    a) this list is nested in another list
    b) WOW
        i) and now we get deeper
        ii) This text is underlined!
5) *This line is in italics*

### B. Bullet List

- This list uses bullets
- BANG
    - Now we step in again
    - pow
        * THIS IS IN SMALL CAPS
        * hello

## III. Image/Figure including referencing

$$x = \sqrt{\pi + e} \int_0^{\infty} \frac{\sinh \theta}{\theta^2} d\theta \tag{1}$$

Fig. 1. This is an equation

Fig. 2. This is an picture. Seems familiar eh?

This Figure above (Figure 1) can be referenced by using the `\ref{label}` command. As can the picture which can be seen in Figure 2.

## IV. Table

TABLE I
THIS IS A TABLE.

| Column 1 | Column 2 | Column 3 |
|---|---|---|
| Hello<br>but under | I am a pretty<br>this line | nice table<br>is a hard line |
| now | lets merge | some columns |
| Merged column takes up 2 spots | | purple |
| this one takes up all three yeahhhh | | |

We can also reference tables similarly to what we did in Section III, by labelling and referencing. eg Table I.

## REFERENCING

Acknowledging works similar to referencing, but we use the `\cite{bibID}` command instead [1]. These can be used anywhere in a document. Note the LaTeXjargon regarding cite and referencing means different things. referecning refernces a label, while cite cites the bibliography

## REFERENCES

[1] D. Doom, M. Evil, and I. Dominate, "Using LaTex for world domination," Soc. World Enslavement For Evil Masterminds. Bermuda, vol. A247, pp. 529–551, April 2009.

# 6 Practical - Lab 3: Bind (P)

Lab task - Bind

| Outcome | Weight |
| --- | --- |
| ULO2 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 3: Bind (P)

*Submitted By:*
Michael Reeb
4936094
2019/03/20 21:10

*Tutor:*
Jonathan Kua

March 20, 2019

```
Marking P-Lab-03-Bind-P on RULE host rule61
TNE30019/TNE80014 - Portfolio Task - P-Lab03 - BIND - Pass
====================================
********************************************************************************
 Portfolio Task: P-Lab-03-Bind, Pass Task

 Configure your rule host (rule61) with a forwarding DNS server

 Tasks:
  Query for: labbind.unix        ; Correct answer: 136.186.230.61
  Query for: labnmap.unix        ; Correct answer: 136.186.230.21
  Query for: caia.swin.edu.au    ; Correct answer: 136.186.229.16
  Query for: 136.186.230.61      ; Correct answer: ns1.unix
********************************************************************************

Date: 20190320_2108
Last 5 Logons:
student    pts/0    vpn247-80.cc.swin.edu. Wed Mar 20 20:34   still logged in
student    pts/0    vpn247-80.cc.swin.edu. Wed Mar 20 20:10 - 20:34  (00:23)
root       pts/0    vpn247-80.cc.swin.edu. Wed Mar 20 14:42 - 17:51  (03:09)
student    pts/1    vpn247-80.cc.swin.edu. Wed Mar 20 12:03 - 15:13  (03:09)
student    pts/0    vpn247-80.cc.swin.edu. Wed Mar 20 11:08 - 14:17  (03:09)

RULE host Socket information
bind     named      24347 20 tcp4   136.186.230.61:53     *:*
bind     named      24347 21 tcp4   136.186.230.61:953    *:*
bind     named      24347 512 udp4  136.186.230.61:53     *:*
bind     named      24347 513 udp4  136.186.230.61:53     *:*
bind     named      24347 514 udp4  136.186.230.61:53     *:*
bind     named      24347 515 udp4  136.186.230.61:53     *:*
bind     named      24347 516 udp4  136.186.230.61:53     *:*
bind     named      24347 517 udp4  136.186.230.61:53     *:*
bind     named      24347 518 udp4  136.186.230.61:53     *:*
bind     named      24347 519 udp4  136.186.230.61:53     *:*
bind     named      24347 520 udp4  136.186.230.61:53     *:*
bind     named      24347 521 udp4  136.186.230.61:53     *:*
bind     named      24347 522 udp4  136.186.230.61:53     *:*
bind     named      24347 523 udp4  136.186.230.61:53     *:*
bind     named      24347 524 udp4  136.186.230.61:53     *:*
bind     named      24347 525 udp4  136.186.230.61:53     *:*
bind     named      24347 526 udp4  136.186.230.61:53     *:*
bind     named      24347 527 udp4  136.186.230.61:53     *:*
bind     named      24347 528 udp4  136.186.230.61:53     *:*
bind     named      24347 529 udp4  136.186.230.61:53     *:*
bind     named      24347 530 udp4  136.186.230.61:53     *:*
bind     named      24347 531 udp4  136.186.230.61:53     *:*
bind     named      24347 532 udp4  136.186.230.61:53     *:*
bind     named      24347 533 udp4  136.186.230.61:53     *:*
bind     named      24347 534 udp4  136.186.230.61:53     *:*
tempuser sshd       18410 3  tcp4   136.186.230.61:22     136.186.247.80:57007
tempuser sshd       18410 7  tcp4   136.186.230.61:6010   *:*
root     sshd       18406 3  tcp4   136.186.230.61:22     136.186.247.80:57007
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*

Performing DNS lookups on 136.186.230.61
========================================

Querying for (labbind.unix) using DNS server (136.186.230.61) - Correct answer (136.186.230.61)
Using domain server:
```

```
Name: 136.186.230.61
Address: 136.186.230.61#53
Aliases:

labbind.unix has address 136.186.230.61
Test Result: PASSED


Querying for (labnmap.unix) using DNS server (136.186.230.61) - Correct answer (136.186.230.21)
Using domain server:
Name: 136.186.230.61
Address: 136.186.230.61#53
Aliases:

labnmap.unix has address 136.186.230.21
Test Result: PASSED


Querying for (caia.swin.edu.au) using DNS server (136.186.230.61) - Correct answer (136.186.229.16)
Using domain server:
Name: 136.186.230.61
Address: 136.186.230.61#53
Aliases:

caia.swin.edu.au has address 136.186.229.16
Test Result: PASSED


Querying for (136.186.230.61) using DNS server (136.186.230.61) - Correct answer (ns1.unix.)
Using domain server:
Name: 136.186.230.61
Address: 136.186.230.61#53
Aliases:

61.230.186.136.in-addr.arpa domain name pointer ns1.unix.
Test Result: PASSED


********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 7 Communications - Lab Report 1

Required Lab Report 1

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦♦◊ |

| Date | Author | Comment |
|---|---|---|
| 2019/04/10 09:18 | Jonathan Kua | Great work! Well-written results and discussion sections. The content has achieved Distinction level. One thing you might try for future reports is to develop your own diagrams in EPS format, if applicable. |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Communications - Lab Report 1

*Submitted By:*
Michael Reeb
4936094
2019/04/09 22:12

*Tutor:*
Jonathan Kua

April 9, 2019

# TNE30019 Unix for Telecommunications Lab 4 - Basic-Apache

Michael Reeb (ID: 4936094)
Swinburne University of Technology
Melbourne, Australia
4936094@student.swin.edu.au

***Abstract*—Apache is a web server application that can be used to build complex web servers. The main configuration file is `httpd.conf` which is used to change basic settings and be used to call other configuration files, such as the `/htaccess` file called in the later sections to create a password authenicated section of the site.**

## I. Introduction to Apache

Apache, or Apache HTTP Server is an open-source web server application. It supports many different compiled modules which can be used to extend Apaches functionality. Apache is widely used throughout the world, and accounts for 67% of all web-servers. Apache is used to serve, or handle the HTTP requests from connecting client devices.

## II. Aim

In this lab, a basic Apache web-server was implemented, and a private page section was implemented demonstrating more of Apaches functionality. This sever was to serve a basic HTML homepage, and the private segment of the site was to be password protected.

## III. Equipment

RULE host 61 was used as the server. SSH access was though PuTTy running on an external PC connected to Swinburne via VPN.

## IV. Methodology

As per lab handout.

## V. Results


Fig. 1. Capture of the main index page from host RULE62

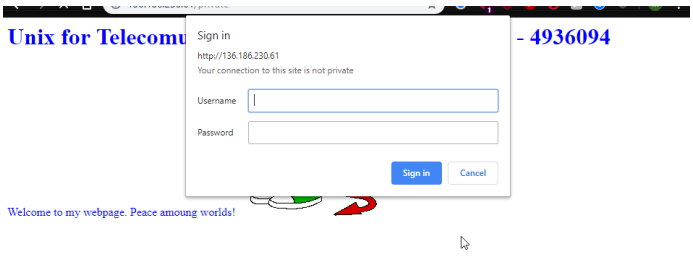



Fig. 2. Authentication prompt before being able to access the private section of the site


Fig. 3. Capture of the private section. Authentication is needed before being able to see this page

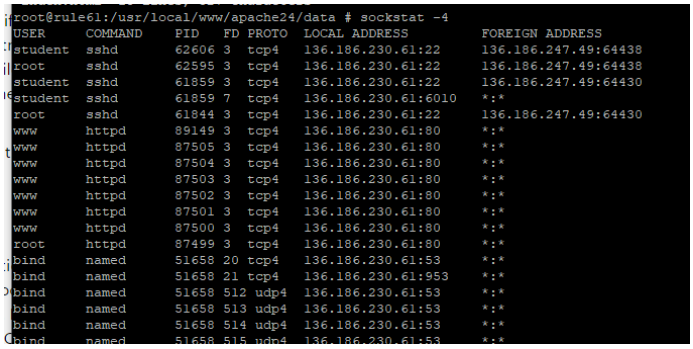



Fig. 4. Output of the command SOCKSTAT -4 while Apache is running. Note that Apache service is identified as HTTPD, and it is running a www service from port 80

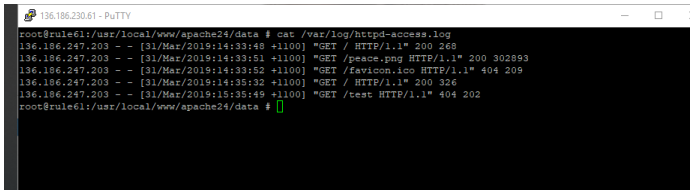



Fig. 5. Output of the access log of the Apache server.

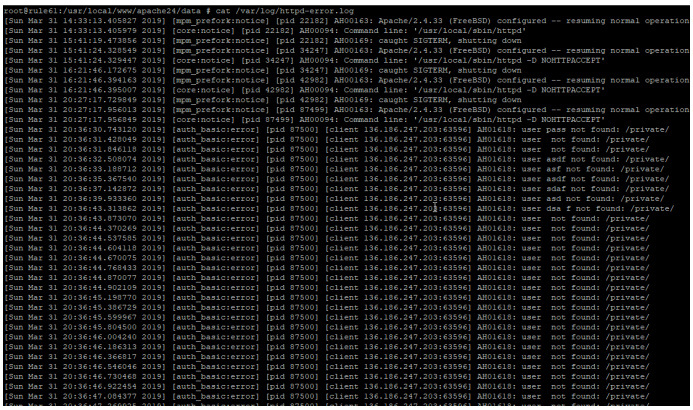



Fig. 6. Output of the access log of the Apache server.

## VI. Discussion

### A. Apache Introduction

The /etc/services is the service file that contains a list of all the common services that are run on Unix, the port-number they common use, the protocol the particular service uses, as well as any applicable aliases that the service has. For example, when `sockstat-4` command is ran, this `services` file is queried so the aliases of the services can be seen. Figure 4 is an example of this [3].

### B. Apache Configuration File

The Apache configuration `httpd.conf` file determines the basic configuration of the Apache service. This file is main configuration file and can be used to change basic settings, as well as call other modules and other configuration files. Changes to this configuration file require Apache (or the httpd service) to restarted to take effect [4].
In this file, we can see that the port that Apache will listen on when running is configured to port 80, which is the well-known port associated with HTTP services. Also it can be seen that the directory that documents will be served from is: `/usr/local/www/apache24/data`. The error log and access log can be found at: `/var/log/httpd-error.log` and `/var/log/httpd-access.log` respectively.

### C. Installing a Simple Web Page

A basic page with the name `index.html` was place in the server directory `/usr/local/www/apache24/data`. The page can be seen in Figure 1.

### D. Starting Apache

With Apache running, the `sockstat-4` command will show the open IPv4 sockets and which process they relate to. The Apache services have the name `httpd`, and the local address is the RULE IP connected to socket 80 as outlined previously. These processes have no associated "foreign" addresses, which means that they are listen for incoming connections.
The Access Log file as see in Figure 5 contains the HTTP requests, the IP address of the requesting client, and at what time, and the status code of what the server sends back to the client. In this log for example, we see that client with IP address of `136.186.247.203` requested ((`GET/HTTP/1.1`)) that the server should send (`"GET"` part) the domain root file (the `/` by itself means the domain root which refers to in this case `index.html`) using the protocol HTTP 1.1 (`HTTP/1.1`). The server responds with the status code (200 in this case) and the size of the returned object (268 bytes). Requesting a page that does not exist would result in a 404 error as the servers response [5].

### E. Researching Simple Authentication

`.htaccess` is the configuration file used by Apache to change configurations of different directories so web-server behaviour can be changed. The .htaccess file operates on a directory level, and when enabled in the main configuration file (httpd.conf), Apache will override the main configuration with the configuration in .htaccess. This is done on a per-directory basis, so a server with multiple directories could have many .htaccess files change behaviour per directory [1].

### F. Create the password file

It is important not to store the password file in a publicly served directory so it is not illegally accessed or modified in anyway. Storing the password file inside the server private directory does not grantee safety either. By placing the file outside of the Apache serving directory, there is no chance that it may accidentally be accessed by clients, and the Apache software will still be access to the password file. The directory chosen for this section was: `/usr/local/etc/apache24/.htpasswd`, which is not inside the Apache serving directory.

### G. Create the .htaccess file

the .htaccess file was created in the Private directory with the folloing commands:

- AuthUserFile /usr/local/etc/apache24/.htpasswd
- AuthName "No Unauthorised access."
- AuthType Basic
- Require valid-user

The `AuthUserFile` command directs Apache to the username/password database to use.
The `AuthName` command is the prompt that is to the client that will appear on the login box.
`AuthType` tell Apache what kind of Authentication to use.Here just the basic authentication is used.
`Require` tell Apache which authenticated users will be granted access [2].

### H. Testing

As can be seen in Figure 2, upon trying to access the private section of the page access was not given until a valid user id and password was given. Upon doing so, the private index was served and we where able to see Figure 3. Invalid credentials were also tested, and the page was redirected to the 401 Unauthorised error page. This can also been seen in the `.htaccess-error` log file in Figure 6. It can be seen that many different attempts of login without valid credentials where attempted.

## VII. Conclusion

Apache is a popular web serving platform that is used for building web servers. It is highly configurable, and can be easily configured for individual-to-individual basis. In this lab, we explored basic configuration, and simple directory security. The `httpd.conf` file is the main configuration file, which can be overwritten on a per-directory basis by `.htaccess` configurations. Password access to private directories is also achievable, and it is important that passwords are stored outside of the Apache serving directories so the password database is not illegally accessed or modified.

## References


[1] What is .htaccess? Retrieved from http://www.htaccess-guide.com/
[2] PasswordBasicAuth. (n.d.). Retrieved from https://wiki.apache.org/httpd/PasswordBasicAuth
[3] Haas, J. (2018, December 18). What Is /etc/services in Linux/Unix? Retrieved from https://www.lifewire.com/what-is-etc-services-2196940
[4] Configuration Files. (n.d.). Retrieved from https://httpd.apache.org/docs/2.4/configuring.html
[5] Understanding the Apache Access Log - KeyCDN Support. (n.d.). Retrieved from https://www.keycdn.com/support/apache-access-log

# 8 Practical - Lab 4: Basic Apache (C)

Lab task - Basic Apache (Credit)

| Outcome | Weight |
|---|---|
| ULO2 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 4: Basic Apache (C)

*Submitted By:*
Michael Reeb
4936094
2019/03/31 20:49

*Tutor:*
Jonathan Kua

March 31, 2019

```
Marking P-Lab-04-Basic-Apache-C on RULE host rule61
TNE30019/TNE80014 - Laboratory - Basic Apache
==============================================
*******************************************************************************
 Portfolio Task: P-Lab-04-Basic-Apache, Credit Task

 Configure your rule host (rule61) with Apache to serve a single web site

 Tasks:
  Apache running on rule61
  Browsing to http://rule61/private will fail with error code 401
  Browsing to http://rule61/private with username(ruleperson)
    password(ruleperson) will return a valid web page
*******************************************************************************

Date: 20190331_2048
Last 5 Logons:
student    pts/1    vpn247-203.cc.swin.edu Sun Mar 31 19:57   still logged in
student    pts/0    vpn247-203.cc.swin.edu Sun Mar 31 18:59   still logged in
student    pts/3    vpn247-203.cc.swin.edu Sun Mar 31 14:27 - 18:36  (04:09)
student    pts/1    vpn247-136.cc.swin.edu Sun Mar 31 11:54 - 14:45  (02:51)
student    pts/8    atc330-19.ict.swin.edu Mon Mar 25 16:10 - 16:14  (00:04)

RULE host Socket information
www      httpd      89149 3  tcp4   136.186.230.61:80     *:*
www      httpd      87505 3  tcp4   136.186.230.61:80     *:*
www      httpd      87504 3  tcp4   136.186.230.61:80     *:*
www      httpd      87503 3  tcp4   136.186.230.61:80     *:*
www      httpd      87502 3  tcp4   136.186.230.61:80     *:*
www      httpd      87501 3  tcp4   136.186.230.61:80     *:*
www      httpd      87500 3  tcp4   136.186.230.61:80     *:*
root     httpd      87499 3  tcp4   136.186.230.61:80     *:*
tempuser sshd       83724 3  tcp4   136.186.230.61:22     136.186.247.203:63507
root     sshd       83721 3  tcp4   136.186.230.61:22     136.186.247.203:63507
tempuser sshd       82089 3  tcp4   136.186.230.61:22     136.186.247.203:63502
tempuser sshd       82089 7  tcp4   136.186.230.61:6011   *:*
root     sshd       82086 3  tcp4   136.186.230.61:22     136.186.247.203:63502
tempuser sshd       71710 3  tcp4   136.186.230.61:22     136.186.247.203:63419
tempuser sshd       71710 7  tcp4   136.186.230.61:6010   *:*
root     sshd       71706 3  tcp4   136.186.230.61:22     136.186.247.203:63419
bind     named      51658 20 tcp4   136.186.230.61:53     *:*
bind     named      51658 21 tcp4   136.186.230.61:953    *:*
bind     named      51658 512 udp4  136.186.230.61:53     *:*
bind     named      51658 513 udp4  136.186.230.61:53     *:*
bind     named      51658 514 udp4  136.186.230.61:53     *:*
bind     named      51658 515 udp4  136.186.230.61:53     *:*
bind     named      51658 516 udp4  136.186.230.61:53     *:*
bind     named      51658 517 udp4  136.186.230.61:53     *:*
bind     named      51658 518 udp4  136.186.230.61:53     *:*
bind     named      51658 519 udp4  136.186.230.61:53     *:*
bind     named      51658 520 udp4  136.186.230.61:53     *:*
bind     named      51658 521 udp4  136.186.230.61:53     *:*
bind     named      51658 522 udp4  136.186.230.61:53     *:*
bind     named      51658 523 udp4  136.186.230.61:53     *:*
bind     named      51658 524 udp4  136.186.230.61:53     *:*
bind     named      51658 525 udp4  136.186.230.61:53     *:*
bind     named      51658 526 udp4  136.186.230.61:53     *:*
bind     named      51658 527 udp4  136.186.230.61:53     *:*
bind     named      51658 528 udp4  136.186.230.61:53     *:*
bind     named      51658 529 udp4  136.186.230.61:53     *:*
```

```
bind     named      51658 530 udp4  136.186.230.61:53     *:*
bind     named      51658 531 udp4  136.186.230.61:53     *:*
bind     named      51658 532 udp4  136.186.230.61:53     *:*
bind     named      51658 533 udp4  136.186.230.61:53     *:*
bind     named      51658 534 udp4  136.186.230.61:53     *:*
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*
?        ?          ?     ?  tcp4   136.186.230.61:51564  136.186.230.61:80
?        ?          ?     ?  tcp4   136.186.230.61:14865  136.186.230.61:80
?        ?          ?     ?  tcp4   136.186.230.61:80     136.186.247.203:63645

HTTP QUERY http://rule61/private
====================================
--2019-03-31 20:48:23--  http://rule61/private
Resolving rule61 (rule61)... 136.186.230.61
Connecting to rule61 (rule61)|136.186.230.61|:80... connected.
HTTP request sent, awaiting response... 401 Unauthorized

Username/Password Authentication Failed.


Test Result: PASSED


HTTP QUERY http://rule61/private
 - Authentication parameters (--user=ruleperson --password=ruleperson)
====================================
--2019-03-31 20:48:23--  http://rule61/private
Resolving rule61 (rule61)... 136.186.230.61
Connecting to rule61 (rule61)|136.186.230.61|:80... connected.
HTTP request sent, awaiting response... 401 Unauthorized
Authentication selected: Basic realm="You wish to entery the private part. Please Enter Password"
Reusing existing connection to rule61:80.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://rule61/private/ [following]
--2019-03-31 20:48:23--  http://rule61/private/
Reusing existing connection to rule61:80.
HTTP request sent, awaiting response... 200 OK
Length: 259 [text/html]
Saving to: ’/tmp/wget_save.boy7QkPe’

     0K                                                       100% 29.6M=0s

2019-03-31 20:48:23 (29.6 MB/s) - ’/tmp/wget_save.boy7QkPe’ saved [259/259]


====================================
Downloaded Page contents
====================================
<html>

<head>
<title> private part </title>
</head>

<body bgcolor="white" text="blue">

<h1> Oh wow, hello there. </h1>
SHhhhhhh this is my private part
Michael Reeb  - 4936094
```

```
<img src="private.gif" alt="Saitama eye walk">
</body>

</html>====================================

Test Result: PASSED


********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 9 Practical - Lab 4: Basic Apache (P)

Lab task - Basic Apache

| Outcome | Weight |
|---|---|
| ULO2 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 4: Basic Apache (P)

*Submitted By:*
Michael Reeb
4936094
2019/03/31 15:44

*Tutor:*
Jonathan Kua

March 31, 2019

```
Marking P-Lab-04-Basic-Apache-P on RULE host rule61
TNE30019/TNE80014 - Laboratory - Basic Apache
==========================================
********************************************************************************
 Portfolio Task: P-Lab-04-Basic-Apache, Pass Task

 Configure your rule host (rule61) with Apache to serve a single web site

 Tasks:
  Apache running on rule61
  Browsing to http://rule61 will return a valid web page
********************************************************************************

Date: 20190331_1542
Last 5 Logons:
student    pts/3    vpn247-203.cc.swin.edu Sun Mar 31 14:27   still logged in
student    pts/1    vpn247-136.cc.swin.edu Sun Mar 31 11:54 - 14:45  (02:51)
student    pts/8    atc330-19.ict.swin.edu Mon Mar 25 16:10 - 16:14  (00:04)
student    pts/16   atc330-19.ict.swin.edu Mon Mar 25 14:48 - 15:28  (00:39)
student    pts/0    vpn247-80.cc.swin.edu. Wed Mar 20 23:47 - 23:50  (00:02)

RULE host Socket information
www      httpd      34253 3  tcp4   136.186.230.61:80     *:*
www      httpd      34252 3  tcp4   136.186.230.61:80     *:*
www      httpd      34251 3  tcp4   136.186.230.61:80     *:*
www      httpd      34250 3  tcp4   136.186.230.61:80     *:*
www      httpd      34249 3  tcp4   136.186.230.61:80     *:*
www      httpd      34248 3  tcp4   136.186.230.61:80     *:*
root     httpd      34247 3  tcp4   136.186.230.61:80     *:*
tempuser sshd       20964 3  tcp4   136.186.230.61:22     136.186.247.203:63191
root     sshd       20961 3  tcp4   136.186.230.61:22     136.186.247.203:63191
tempuser sshd       20954 3  tcp4   136.186.230.61:22     136.186.247.203:63189
tempuser sshd       20954 7  tcp4   136.186.230.61:6011   *:*
root     sshd       20951 3  tcp4   136.186.230.61:22     136.186.247.203:63189
bind     named      51658 20 tcp4   136.186.230.61:53     *:*
bind     named      51658 21 tcp4   136.186.230.61:953    *:*
bind     named      51658 512 udp4  136.186.230.61:53     *:*
bind     named      51658 513 udp4  136.186.230.61:53     *:*
bind     named      51658 514 udp4  136.186.230.61:53     *:*
bind     named      51658 515 udp4  136.186.230.61:53     *:*
bind     named      51658 516 udp4  136.186.230.61:53     *:*
bind     named      51658 517 udp4  136.186.230.61:53     *:*
bind     named      51658 518 udp4  136.186.230.61:53     *:*
bind     named      51658 519 udp4  136.186.230.61:53     *:*
bind     named      51658 520 udp4  136.186.230.61:53     *:*
bind     named      51658 521 udp4  136.186.230.61:53     *:*
bind     named      51658 522 udp4  136.186.230.61:53     *:*
bind     named      51658 523 udp4  136.186.230.61:53     *:*
bind     named      51658 524 udp4  136.186.230.61:53     *:*
bind     named      51658 525 udp4  136.186.230.61:53     *:*
bind     named      51658 526 udp4  136.186.230.61:53     *:*
bind     named      51658 527 udp4  136.186.230.61:53     *:*
bind     named      51658 528 udp4  136.186.230.61:53     *:*
bind     named      51658 529 udp4  136.186.230.61:53     *:*
bind     named      51658 530 udp4  136.186.230.61:53     *:*
bind     named      51658 531 udp4  136.186.230.61:53     *:*
bind     named      51658 532 udp4  136.186.230.61:53     *:*
bind     named      51658 533 udp4  136.186.230.61:53     *:*
bind     named      51658 534 udp4  136.186.230.61:53     *:*
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
```

```
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*
?        ?          ?     ?  tcp4   136.186.230.61:80     136.186.247.203:63226
?        ?          ?     ?  tcp4   136.186.230.61:80     136.186.247.203:63230
?        ?          ?     ?  tcp4   136.186.230.61:57343  136.186.230.61:80

HTTP QUERY http://rule61
====================================
--2019-03-31 15:42:14--  http://rule61/
Resolving rule61 (rule61)... 136.186.230.61
Connecting to rule61 (rule61)|136.186.230.61|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 326 [text/html]
Saving to: '/tmp/wget_save.NxJcb4NA'

     0K                                                       100% 34.7M=0s

2019-03-31 15:42:14 (34.7 MB/s) - '/tmp/wget_save.NxJcb4NA' saved [326/326]


====================================
Downloaded Page contents
====================================
<html>

<head>
<title> favorites / bookmark title goes here </title>
</head>

<body bgcolor="white" text="blue">

<h1>Unix for Telecomunications - Lab 4 - Michael Reeb  - 4936094</h1>
Welcome to my webpage. Peace amoung worlds!

<img src="peace.png" alt="Rick and Morty peace amoung worlds">
</body>

</html>
====================================

Test Result: PASSED


********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

## 10 Theory - Test 1 (P)

In class test run during tutorial in week 5 of semester

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO6 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO2 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO5 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO1 | ♦♦◊◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Theory - Test 1 (P)

*Submitted By:*
Michael Reeb
4936094
2019/04/12 14:43

*Tutor:*
Jonathan Kua

April 12, 2019

# Unix for Telecommunications

Portfolio Task – T-Test-1
**Pass Level Task**

Factors per 1.1

| | |
|---|---|
| **Student ID** | 4936094 |
| **Student Name** | Michael Roob |
| **Test Date** | Week 5 |
| **Time Allowed** | 30 minutes |

## Assessment – Staff Use Only

In order to pass the test you **MUST** score a minimum of **80%**

| Question: | 1 | 2 | 3 | 4 | 5 | Total |
|---|---|---|---|---|---|---|
| Points: | 4 | 6 | 5 | 6 | 7 | 28 |
| Score: | 4 | 6 | 4 | 6 | 5.5 | 25.5 |

91%
PASS

# 11 Communications - Research Report

Research report required on Assignment topic

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦♦♦ |

| Date | Author | Comment |
|---|---|---|
| 2019/04/30 15:11 | Jonathan Kua | Excellent work! Great job! |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Communications - Research Report

*Submitted By:*
Michael Reeb
4936094
2019/04/17 10:54

*Tutor:*
Jonathan Kua

April 17, 2019

# A Brief Overview of LTE Video Broadcast Technologies

Michael Reeb - 4936094
*TNE30019 Unix for Telecommunications*
*Communications Report*
4936094@student.swin.edu.au

***Abstract*—Currently the standard technology used is in cellar mobile networks is LTE, and its successor LTE-A, a Fourth Generation technology that offers bandwidth up-to 1Gbps in the down link and mulitcast/broadcast capabilities via eMBMS. Modern mobile devices are being released with chipsets that are compatible with this broadcast technology, and coupled with the benefits of the H.264 encoding which allows for the implementation of DASH (and the related HLS) media transport systems, the future for efficient broadcasting high quality, live video over cellular networks with great user experience, is a certainty.**

## I. Introduction

With the widespread adoption of portable, powerful smart devices, the demand for high-quality, flexible streaming services has also increased. Currently video content counts for almost 58% of global internet traffic [1], and this consumer video demand has placed a large strain upon cellar telecommunication services who are currently implementing the latest cellular technology to improve network efficiency, improve customer Quality of Service (QoS), and maximise bandwidth capabilities.

The streaming of events, particularly live sport and entertainment, has become a profitable and marketable service for many mobile telecommunication providers who are now moving away from traditional services (such as hardwired phone and voice), and focusing on media streaming services and platforms. This paper overviews the different technology that is needed for broadcasting live-video via a cellular network, and explores cellular technology and future enhancements, video codecs, and media distribution systems.

## II. On-demand Streaming Operation

There are two main models for serving streaming media: the push method where the transmission of data is requested by the server, and the pull method where the request for transmission is performed by the client devices. Each method has its advantages and disadvantages with respect to performance, and application.

### A. Push Streaming

A push service is one where the request of the transmission of payload data is performed by the server. Once the client and server have established a connection, the server pushes the new data to the client until the client stops or interrupts the session [8]. This is useful for streaming services, as it allows the server to perform multi-cast (ie, sending the same payload data to multiple services) which can lower the bandwidth requirements of the server thus reducing redundant transmission. An example a push based service is Internet Radio. A client subscribes to the stream, and data is pushed as it is created by the server. There are no "rewind" type capabilities that are performed by the server( however they may be some local caching on the client device that allows short rewind capabilities) as the client only receives what the server sends.

In these systems, the client device application can be considered passive; there is very little complexity in the client application as it only needs to be able to connect to the server, receive, decode, and buffer the data as before using it. The server holds the complexity and sends commands to the user device regarding QoS and other session changes [8]. Example protocols are SIP (Session Initiation Protocol, used in telephony applications including video conferencing) and RTP (Real Time Protocol, which is used often in live-streaming video and audio, as it also supports multi-cast) [8].

### B. Pull Streaming

A pull service is a service where the request for transmission of data is performed on the client side. The content is pre-encoded and retrieved piecemeal by requests from the client. Multiple users each have a unique connection to the server and request the data that they would like to be served. This is useful for streaming services , including live streaming services, as it allows each client to watch the same media as when it is convenient for client, have rewind/playback capabilities (for example, YouTube live has limited rewind capabilities for live streams), and dynamic quality control which can be adjusted on the fly . With pull-type services, the complexity lies in the client application; the server simply serves the requests of the client devices. While less efficient than Push based services this is not a large issue if large bandwidth is available to user devices.

Modern pull based streaming services use HTTP for as the transport protocol, which itself uses TCP to guarantee packet delivery. Unlike the Push streaming examples, there is no built in broadcast or multi-cast support for HTTP/TCP as each user device acts streams the media independently from each other.

This can particularly inefficient if devices are streaming the same data as the server will send redundant copies of the media to each device independently.
Example services of Pull type services are Youtube,(including Youtbue Live), Netflix, and Spotify [8].

### C. Streaming Implementations

Streaming protocols are used to transport the encoded video data (explored further in Section II-F) from the sever to the client mobile devices. These implementations wrap the encoded video, perform QoS, and live-video quality switching. Historically, RTP and other UDP-based services were used as the transport protocol overhead was small, and thus efficient as bandwidth was often limited and inconsistent. However with modern network technology offering larger bandwidth and faster throughput, this low-overhead has primary become a less of concern with reliability of delivery becoming more of a focus. This focus shift has lead to HTTP over TCP becoming the most common media streaming transport protocol used in current devices [9].
While there are many different HTTP streaming implementations, including Adobe HTTP Dynamic Streaming, and Microsoft's Smooth Streaming, this paper will primary focus on the two most popular: Apples HTTP Live Streaming (HLS), and the MPEG standardised Dynamic Adaptive Streaming over HTTP (MPEG-DASH, or DASH).

### D. HLS

HLS is an adaptive bitrate streaming protocol that was developed by Apple and natively supported in QuickTime, Safari, iOS and other Apple related devices and technologies. The popularity of HLS has increased with the popularity of Apple Products and Devices,

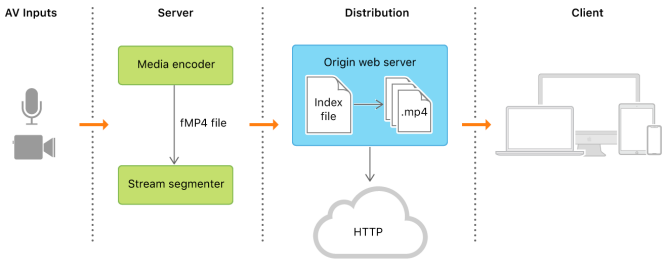



Fig. 1. Flow chart of Apple's HLS [11]

The HLS Media Preparation Server is responsible for taking input streams of audio and video input and digitally encoding them using an encoding scheme (as explored in Section II-F). The encoded content is then placed into an MPEG Transport Stream container, where it is delivered to the Stream Segmenter, where the stream is segmented into chunks approximately 5-10 seconds long of MPEG-Transport Stream video files (`.ts` files). It is here that the video also can be encoded at different qualities for adaptive stream capabilities. The Stream Segmenter also produces the index files that describe the available chunked `.ts` files at the differing quality levels. This chunking and indexing process can also be performed in real-time, for live-streaming content. [9]
The `.ts` files and index files are then pushed to a HTTP web sever where they are published. The web server is only responsible for delivering the requested chunks [9].
Finally, the client device establishes a connection with the server, where it requests the index and `.ts` files to download. The client is responsible for selection which quality tier to choose from and adapting the quality during the stream as bandwidth and throughput fluctuates [9].

### E. MPEG-DASH

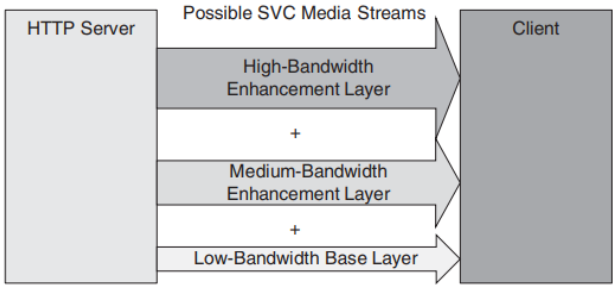



Fig. 2. Example of a 3-Layer SVC. To achieve highest possible quality, the base layer and the two additional enhancement layer streams are needed [9].

DASH is term that describes any HTTP streaming service that allows the client devices to select various quality tiers offered by the HTTP server. DASH is not a protocol, or system itself, but a set of formats that can be used with HTTP to deliver streamable media. Thus, a total analysis of DASH is outside the scope of this paper will not be performed, but a brief overview of MPEG-DASH will be analysed; a specific instance of DASH defined in the ISO/IEC FCD 23009-1 standard [9].

Similarly to HLS, MPEG-DASH utilises HTTP over TCP for transport, and supports various encoding schemes. MPEG-DASH also segments the input audio/video stream into individual chunks of differing quality, and clients connected to the MPEG-DASH web server decide which video quality tier to stream. However, differences lie in the operation of this quality selection.
With MPEG-DASH the adjustment of the quality of the stream is performed by the addition of Scalable Video Codec (SVC) layers. This means that a client will always receive the base SVC layer, and increasing the quality is achieved by the addition of higher enhancement layers. An example of this can be seen in Figure 2.

### F. Encoder and Decoder Algorithms, and Protocols

Another important aspect of efficient transmission is the encoder/encoder (codec) algorithm and structure. As wireless broadcast technology is inherently lossy, its is highly likely that during streaming there is a likelihood that packets

will be lost. Now, particularly if this occurs during an a LTE-Broadcast (which is inherently a push based service) there is no backwards channel for the mobile devices to request re-transmission of the lost packet. Thus, the codec used has to find a balance between data compression, and ability to withstand packet losses in transmission. [7].
Historically, many different codecs have existed,such as MPEG-2, and MPEG-4. These have been supplanted by modern codecs that are doubly as effective (ie for a given bandwidth are able to provide twice the video quality [9]). The commonly used codec for live-streaming is H.264. H.264 is a industry standard video compression system that commonly is used for video streaming services, being both used in HLS and DASH delivery systems outlined in Section II-D and Section II-E.

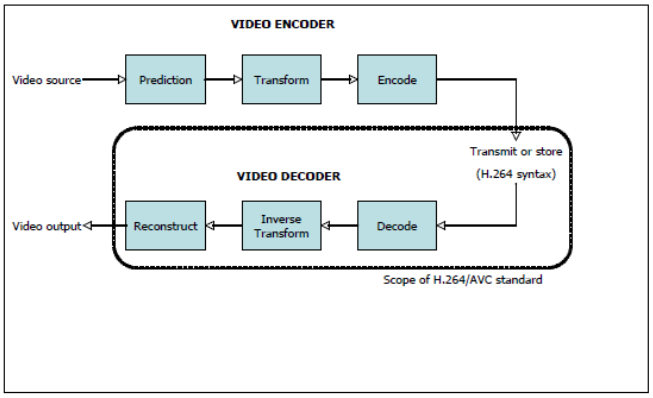



Fig. 3. H264 encoding and decoding process [13].

The encoding and decoding process may vary between implementations of H.264, but all will follow a similar process to that in Figure 3. First, the encoder processes breaks down the frame of the video in what is known as a Macroblock (a $16 \times 16$ pixel box) which is compared to previous coded data of the current frame (known as Intra prediction), or previously coded frames (known as Inter prediction). The encoder subtracts the prediction from the current macro-blocks to obtain the residual. A block of these residuals is then transformed into a set of weighted co-coefficients which is then quantised to some value, called the Quantisation Parameter (QP) [13]. This QP can be changed depending on the desired Quality/Compression ratio (ie higher compression means lower quality, and high quality means low compression ) [9].
The output of the encoding processes is then used to create a bit stream. This bit stream is the information that is sent to servers and received by client devices. The decoding process is the reverse of this, and a video image is reconstructed.
A H.264 stream colonists of different frames, categorised by the amount of prediction used by the encoder when encoding said frame. An I-frame contains no prediction at all, and are sent at certain intervals to ensure that the codec reconstruction process has something to synchronise to. If I-frames are lost, any frames that reference this I-frame cannot be reconstructed. A P-frame is a frame that has prediction from a previous I-frame, or from another P-frame. And a B-frame contains predictions from both previous and subsequent frames [9].
Reference frames will be significantly smaller than non-referenced frames, and as such I-frames will cause peak-traffic while P-frames and B-frames will only cause minimal amounts of traffic [9].

The compression performance of H.264 is significantly greater than that of its predecessors. H.264 can deliver significantly better performance than that of MPEG-2 and MPEG-4, however at the price of greater computational power. As computational power has increased in user devices, H.264 has seen adaptation throughout video technology from media streaming to media storage.

## III. Long Term Evolution

### A. LTE Operation Overview

Long Term Evolution (LTE) is a communications standard for wireless designed for cellular networks to simplify the network architecture than of the 3G (and previous) protocols by being based on an all-IP packet switched network [3, pg.470]. This allowed LTE to have significantly higher bit-rates, whilst still maintaining simultaneous access for multiple users.
In LTE the core network and air interface was separated into two distinct entities: eNodeB and LTE EUTRAN respectively. This modularisation allows for cost-effective improvement and advancements to be implemented in either of these systems without having the whole network having to be reconfigured. This simple network structure results in simpler operation, and thus greater performance [3, pg.473].

### B. LTE Air Interface

LTE uses Orthogonal Frequency Division Multiple Access (OFDMA) in the down-link channels [3, pg.473], and Frequency Division Multiple Access (FDMA) in the up-link. These modulation technologies were chosen for high bit-rates, lower latency, and lower power consumption (particularly in the uplink as this transmission would be powered by the mobile device which often use battery power). With 20Mhz wide channels, the performance for LTE was 100Mbps in the down-link, and 50Mbps in the uplink.
The LTE network architecture was greatly simplified from the 3G technologies, so all QoS traffic was now packet-switched (including voice and multimedia), unlike the older 3G technologies whose infrastructure was comprised of both packet and circuit-switched services.
The implementation of all LTE transmissions being apart of a QoS system allows for prioritisation of different transmission packets, which greatly improves throughput for high resource real time applications, such as live-video and live-audio.
LTE also supports multiple antenna transmission (MIMO)

which improves spectral efficiency, bandwidth, and lowers interference.

### C. LTE Advanced: True 4G

LTE Advanced was an upgrade to LTE that met the standards to be a true 4G technology. These advancements allows 1 Gbps peak rates in the down link, and 500Mbps peak rates in the uplink [3, pg.502]. This increase in data rates was achieved through many different advancements in various technological aspects of the system, namely: increasing the bandwidths of the channels to a maximum of 100Mhz via carrier aggregation, MIMO enhancements, introduction of smaller mobile-cell subdivisions and relays, and enhanced interference modelling [3].

## IV. LTE Operation and Real Time Streaming

Video traffic comprises a significantly large portion of current mobile traffic which requires larger capacities on the network back-haul, but also requires reliable connectivity over the radio interface. LTE and LTE-Advanced both have QoS built into them allowing the priority of certain services over others. This is important for Real Time Streaming over LTE, as packet delay and losses over the radio link can cause user experience to decline when streaming media.

LTE also has issues with bandwidth consumption with multiple concurrent streaming clients. Each mobile device has its own assigned channel (a piece of bandwidth allocated to this device for data transmission) and due to the intensive data needed and QoS requirements for video streaming, the LTE transmitter and backhaul network can quickly reach maximum capacity. This will cause any new customers trying to connect to the network to either fail, or force current customers to share some of their allocated bandwidth which can effect the video streaming quality, and reliability.

If multiple clients are streaming the exact same content, (say a live sport game) then a lot of these transmissions are going to be replicas of each other and thus redundant. Instead of sending multiple transmissions over separate channels of the same data, and given broadcast nature of cellular networks, a more efficient way of transmitting would be to have all the mobile devices subscribe to the same channel to receive the media data.This technique is know as cellular-broadcasting, was introduced by the LTE group in its Evolved Multimedia Broadcast/Multicast Services (eMBMS) specification, also commonly known as LTE-Broadcast.

## V. Evolved Multimedia Broadcast Multicast Service

eMBMS is the LTE implementation of the of Multimedia Broadcast/Multicast Service (MBMS) which brings the advantages of LTE to MBMS allowing HD broadcast service by taking advantage of the higher bandwidth channels that LTE offers.

eMBMS is a “point-to-multi-point” service where data is transmitted from a single-source transmitter to multiple receivers [4] allowing for multiple devices to receive the same broadcast transmission within the same cell. eMBMS reduces cellular network bandwidth by eliminating the redundancy caused by having multiple devices streaming the same data over many uni-cast links. Thus for live-streaming services (such as live-event broadcasting or television over LTE) this can radically free the cellar network bandwidth, and reduces the load on the cellular back-haul

Up to 60 percent of an LTE frequency carrier can be used for MBMS transmissions, and using 20MHz LTE eMBMS allows for approximately 5 HD streams to be broadcast, parallel with normal unicast services [4].

Currently eMBMS needs unique chip set on the device so backwards compatibility is not possible with older devices. This is currently a limiting factor of adoption for using eMBMS services as software decoding is CPU intensive and given most mobile devices are battery powered, a software fix is not viable [4]. However,the latest mobile devices do support eMBMS, so as consumers upgrade their devices the rollout of eMBMS services will be able to increase.

### A. Efficiency with eMBMS

Efficiency and resource savings for eMBMS is not as simple as a linear-relationship (ie two devices on a broadcast channel does not save half the amount of resources). For a unicast transmission the LTE network is able to adapt the signal modulation and coding rate to suit the current reception conditions, which is not possible to do with broadcast transmission. The broadcast transmitter must use a robust modulation and encoding schemes that are able to reach devices located on the edge of the cellular coverage area. Devices on the edge of the cell will suffer from larger signal losses due to distance, but will also receive interference from neighbouring cell transmissions. Thus, for adequate broadcast transmission a larger amount of resources are required compared to unicast transmission; which equates to many devices uni-casting the same stream before it is more efficient for the network to switch over to multi-cast [4].

### B. Single Frequency Network

However, if multiple neighbouring cells transmit the same data fully synchronised so that the neighbouring radio signals are interfering constructively, the received signal (particularly at the cell edges) will increase.

This transmission technique is called “signal frequency network” and can greatly increase the total cellular network efficiency. This is particularly effective for popular live events, such as live sport finals (ie AFL grand final) where many user devices located in different neighbouring cells are casting the same media.

The neighbouring cell radio transmitters purposefully select the same frequency blocks to transmit the media on, such that the signal is indistinguishable from each other. As this frequency blocks are relativity static, the rest of the blocks are able to be used for regular, non-SFN transmissions (including other LTE broadcasts, and regular unicast traffic). For LTE base-stations separated by 500m, the efficiency of MBMS with

SFN transmission was evaluated to be 3bytes/Hz as compared with 0.25 bytes/Hz for MBMS over UMTS (a 3G technology) [4].

### C. Stream Authentication with eMBMS

A large issue with eMBMS is ensuring that only paying customers are able to access the eMBMS service. Due to the broadcast nature cellular networks, the eMBMS signal must be adequate encrypted so unauthorised devices cannot listen in the live stream. Secondly, as the encryption keys need to be distributed among a large number of UE devices and need to be changed at regular intervals to prevent unauthorised distribution of them to free loading devices who wish to access the media illegally.

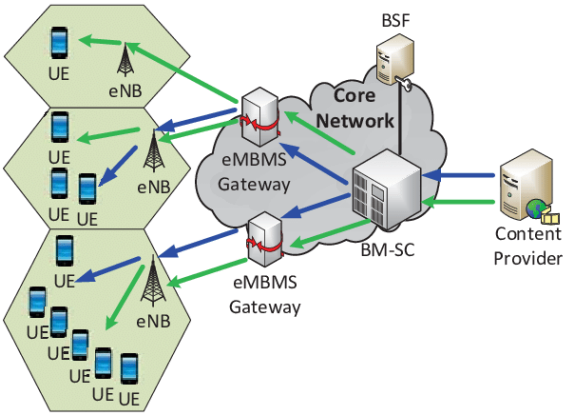



Fig. 4. A simplified example of eMBMS architecture [6]

The Key Management Mechanism, which is used by the multicast network provide forward and backward security, consists of the following parts: a Bootstrapping Server Function (BSF), Broadcast Multi-cast Service Centre (BM-SC), a content provider, an eMBMS gateway, and mobile user devices (UEs). A simplified diagram of this system can be seen in Figure 4.

The BSF is a part of the the Generic Bootstrapping Architecture that is responsible for establishing shared secrets between the mobile devices and the BM-SC. The BM-SC is the entry-point for the content providers and forwards the broadcasts packets to the eMBMS Gateway, where the packets are then further distributed to the eNodeB. The eNodeB then broadcasts the packets to the mobile user equipment (UE in the diagram).

To protect the eMBMS data, four security keys are used which are: the MBMS Reqeust Key (MRK), the MBMS Service Key (MSK), the MBMS Traffic Key (MTK), and the MBMS User Key (MUK) [5].

THe MRK is used for authentication a mobile device to a BM-SC when performing the key requests. The MUK is used for securing the distribution the MSK, and the MSK is used for securing eMBMS sessions and the distribution of the MTK. The MTK is used for encrypting and decrypting of the eMBMS packets [5]. Thus, the MTK, MUK, and MSK are used for protecting data, while the MRK is used for authenticating mobile devices. During the eMBMS service, the MSK and MTK are updated (or rekeyed) to protect the stream [5].

A simplified User Service Join Procedure is as follows, where $UE_{\{1..n\}}$ denote the mobile devices that have already joined [5]:

1) The new UE user device, $\text{UE}_{n+1}$, sends a service join request to the BM-SC. BM-SC then will ask that the joining device $\text{UE}_{n+1}$ to begin the bootstrap authentication procedure with the BSF [5].
2) Device $\text{UE}_{n+1}$ performs this bootstrap procedure with the BSF to obtain a $\text{MRK}_{n+1}$. This MRK key is then used by the mobile device to derive the $\text{MUK}_{n+1}$ key.
3) After deriving $\text{MRK}_{n+1}$ and $\text{MUK}_{n+1}$, $\text{UE}_{n+1}$ performs authentication with the BM-SC using $\text{MRK}_{n+1}$
4) The BM-SC generates a new MSK, $\text{MSK}_{new}$ which unicasts this to all existing mobile devices $\text{UE}_k$ already on the service, encrypted with the $\text{UE}_k$s $\text{MUK}_k$ key.
5) Finally, the BM-SC generates a new MTK, which is encrypted with $\text{MSK}_{new}$ and broadcast out to all user equipment devices that have joined the MBMS service.

This solution outlined above for rekeying every-time a device joins or leaves the service group works well for a small number of devices, but degrades quickly if there is a large number of devices in the usergroup (live sporting events may attract thousands of user devices for example). Performance quickly degrades as the devices continuously joining and dropping at seemly random intervals will cause frequent rekeying. This frequent rekeying will cause a large amount of re-authentication traffic, given such a large user-group of user devices can cause swap the Core Network causing it to under-perform.

Also, if the mobile devices were not able to receive the new keys from the broadcast for some reason (say, from momentary small signal fade, or interference) than this device will have to reauthenticate to the user group from the beginning. Frequent rekeying means that the probability that a device will miss the re-keying is increased, and thus can cause an endless loop of rekeying transmissions [5].

Alternatively having a longer fixed time-frame before rekeying will cause the keys to be renewed less often. While this will prevent the Core Network from being overwhelmed with reauthentication traffic and prevent rekeying loops, the likelihood of the multicast being compromised and having freeloading devices access the stream for free is higher, which results in lost revenue for the content provider. This balance between the stream security and performance of the networks is known as the security-performance trade-off, and is an important metric used by the service providers to adjust the network security protocols to find an acceptable threshold between the two criteria.

A solution the security-performance is the Dynamic Rekeying Algorithm (DRA). This algorithm is performed by the network operators to determine when rekeying in necessary. DRA is

used designed to update security keys based on a dynamic time slot, rather than when a mobile user device joins or leaves, or after a set period of time. This algorithm allows the network operators to define a performance metric between acceptable revenue loss of the content provider, and network performance of the cellular network , and find the optimal interval time that performs acceptably on both of these metrics.
An example algorithm for determining the interval time is given as:

$$\phi = \sum_{k=N(t_i-1)+1}^{N(t_i)} \tag{1}$$

where: $\phi$ is the accumulated free enjoy time interval,
$N(t_i)$ is the number of user devices leaving the multi-cast group between $t = 0$and$t = t_i$,
and $s_k$ is the time when one user leaves the user-group. Thus, the time interval between $[s_k, t_i]$ the user is still able to freely accesses the multi-cast content because it still hold the valid encryption keys.
When the accumulated free-enjoy time exceeds a defined limit, then the keys are updated and rekeyed. Operators are thus able to easily adjust this limit as circumstances change to ensure the optimal network performance and revenue lost metric is achieved.

## VI. Mobile-to-Mobile Cooperation

Another interesting technology that could improve QoS of live streaming is Mobile-to-Mobile cooperation. As video-streaming requires particularly high-data rates to maintain a high QoS, a mobile device connected via LTE has large energy drain on the device battery as the LTE radio system is not as optimised for energy efficient than that of a low power transmission protocols such as Bluetooth [12].

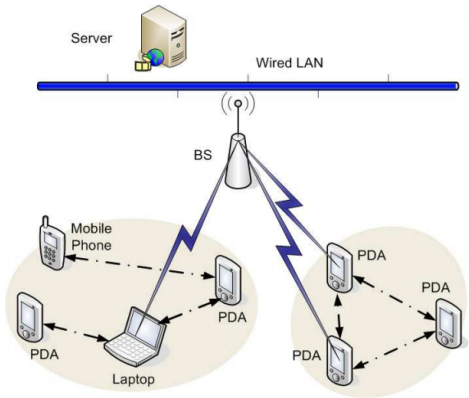



Fig. 5. An example of a clustered network. Only once device connects via the long range link, and the payload is distributed to the other devices via a short-range protocol [14]

As can be seen in Figure 5, the principle of Mobile-to-mobile cooperation is as follows: Mobile devices wanting to stream the data from the Long Range LTE link base station that are also geographically near each other create an ad-hoc wireless network cluster via a short range energy efficient protocol, such as Bluetooth. Then in each cluster a single device becomes the cluster head, and receives the data packets from Long Range link, and then further broadcasts them to the rest of the UT via the Short Range. Only a single device at a time needs to connect to the LR at anytime, which can greatly reduce LR link bandwidth requirements by reducing the amount of Unicast connections. The average power consumption across all devices in cluster is significantly lower as only one device is using the high power network protocols where the other devices use the low-power devices to receive the relayed data. [12] [14].

## VII. Conclusion

The future for broadcasting live video over cellular networks is currently being implemented worldwide. The rollout of eMBMS enabled user devices, and user demands for streaming live-video will see subsequent services provided by cellular network providers who aim to efficiently optimise network bandwidth and connectivity. Coupled with the improvement in codec as mobile device computational power has increased; it is near certainty that customers will be able to live-stream high quality, high definition video media on their mobile devices.

## References


[1] Sandvine releases 2018 Global Internet Phenomena Report. (n.d.). Retrieved from https://www.sandvine.com/press-releases/sandvine-releases-2018-global-internet-phenomena-report

[2] Bing, B. (2013). *Chapter 4* Broadband wireless multimedia networks. Hoboken, NJ: John Wiley & Sons.

[3] Beard, C., & Stallings, W. (2016). *Chapter 13 - 15* Wireless Communication Networks and Systems. Pearson.

[4] Lecompte, D., & Gabin, F. (2012). Evolved multimedia broadcast/-multicast service (eMBMS) in LTE-advanced: Overview and Rel-11 enhancements. IEEE Communications Magazine, 50(11), 68-74. doi:10.1109/mcom.2012.6353684

[5] Y. Ren, J. Chen, J. Chin & Y. Tseng, (2016) "Design and Analysis of the Key Management Mechanism in Evolved Multimedia Broadcast/-Multicast Service," in IEEE Transactions on Wireless Communications, 15(12), 8463-8476. doi: 10.1109/TWC.2016.2615605

[6] (2019). Retrieved from: https://www.researchgate.net/figure/A-simplified-example-of-eMBMS-architecture_fig2_309081966

[7] Kumar, U., & Oyman, O. (2013). QoE evaluation for video streaming over eMBMS. 2013 International Conference on Computing, Networking and Communications (ICNC). doi:10.1109/iccnc.2013.6504146

[8] Begen, A., Akgul, T., & Baugher, M. (2011). Watching Video over the Web: Part 1: Streaming Protocols. IEEE Internet Computing, 15(2), 54-63. doi:10.1109/mic.2010.155

[9] Barz, H. W., & Bassett, G. A. (2016). *Chapter 3, Chapter 9* Multimedia networks: Protocols, design, and applications. West Sussex, U.K.: Wiley.

[10] HTTP Live Streaming. (n.d.). Retrieved from https://developer.apple.com/documentation/http_live_streaming

[11] Understanding the HTTP Live Streaming Architecture. (n.d.). Retrieved from https://developer.apple.com/documentation/http_live_streaming/understanding_the_http_live_streaming_architecture

[12] Ma, D., Peng, J., Li, H., Liu, W., Huang, Z., & Zhang, X. (2015). Energy efficient video streaming over wireless networks with mobile-to-mobile cooperation. 2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM). doi:10.1109/pacrim.2015.7334849

[13] An Overview of H.264 Advanced Video Coding. (n.d.). Retrieved from https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/

[14] Al-Kanj, L., & Dawy, Z. (2010). Optimized energy efficient content distribution over wireless networks with mobile-to-mobile cooperation. 2010 17th International Conference on Telecommunications. doi:10.1109/ictel.2010.5478815

# 12 Practical - Lab 5: Advanced Apache (D)

Lab task - Advanced Apache (Distinction)

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦♦♦♦ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 5: Advanced Apache (D)

*Submitted By:*
Michael Reeb
4936094
2019/04/08 15:28

*Tutor:*
Jonathan Kua

April 8, 2019

```
Marking P-Lab-05-Advanced-Apache-D on RULE host rule61
TNE30019/TNE80014 - Laboratory - Advanced Apache
================================================
********************************************************************************
 Portfolio Task: P-Lab-04-Advanced-Apache, Pass Task

 Configure your rule host (rule61) with Apache to serve a virtual hosted web site

 Tasks:
  Set rule61 as DNS Server
  Query for: aristocrats.unix    ; Correct answer: 136.186.230.61
  Apache running on rule61
  Browsing to http://aristocrats.unix  will return a correct web page
********************************************************************************

Date: 20190408_1527
Last 5 Logons:
student    pts/18   atc330-21.ict.swin.edu Mon Apr  8 14:56   still logged in
student    pts/17   atc330-21.ict.swin.edu Mon Apr  8 14:56   still logged in
student    pts/1    vpn247-242.cc.swin.edu Sun Apr  7 09:49 - 12:59  (03:09)
student    pts/0    vpn247-242.cc.swin.edu Sun Apr  7 08:52 - 13:01  (04:09)
student    pts/1    vpn247-228.cc.swin.edu Sat Apr  6 23:17 - 02:27  (03:09)

RULE host Socket information
www      httpd      52418 3  tcp4   136.186.230.61:80     *:*
www      httpd      52417 3  tcp4   136.186.230.61:80     *:*
www      httpd      52416 3  tcp4   136.186.230.61:80     *:*
www      httpd      52415 3  tcp4   136.186.230.61:80     *:*
www      httpd      52414 3  tcp4   136.186.230.61:80     *:*
root     httpd      52413 3  tcp4   136.186.230.61:80     *:*
tempuser sshd       43937 3  tcp4   136.186.230.61:22     136.186.15.217:55890
root     sshd       43930 3  tcp4   136.186.230.61:22     136.186.15.217:55890
tempuser sshd       43898 3  tcp4   136.186.230.61:22     136.186.15.217:55876
root     sshd       43891 3  tcp4   136.186.230.61:22     136.186.15.217:55876
bind     named      41933 20 tcp4   136.186.230.61:53     *:*
bind     named      41933 21 tcp4   136.186.230.61:953    *:*
bind     named      41933 512 udp4  136.186.230.61:53     *:*
bind     named      41933 513 udp4  136.186.230.61:53     *:*
bind     named      41933 514 udp4  136.186.230.61:53     *:*
bind     named      41933 515 udp4  136.186.230.61:53     *:*
bind     named      41933 516 udp4  136.186.230.61:53     *:*
bind     named      41933 517 udp4  136.186.230.61:53     *:*
bind     named      41933 518 udp4  136.186.230.61:53     *:*
bind     named      41933 519 udp4  136.186.230.61:53     *:*
bind     named      41933 520 udp4  136.186.230.61:53     *:*
bind     named      41933 521 udp4  136.186.230.61:53     *:*
bind     named      41933 522 udp4  136.186.230.61:53     *:*
bind     named      41933 523 udp4  136.186.230.61:53     *:*
bind     named      41933 524 udp4  136.186.230.61:53     *:*
bind     named      41933 525 udp4  136.186.230.61:53     *:*
bind     named      41933 526 udp4  136.186.230.61:53     *:*
bind     named      41933 527 udp4  136.186.230.61:53     *:*
bind     named      41933 528 udp4  136.186.230.61:53     *:*
bind     named      41933 529 udp4  136.186.230.61:53     *:*
bind     named      41933 530 udp4  136.186.230.61:53     *:*
bind     named      41933 531 udp4  136.186.230.61:53     *:*
bind     named      41933 532 udp4  136.186.230.61:53     *:*
bind     named      41933 533 udp4  136.186.230.61:53     *:*
bind     named      41933 534 udp4  136.186.230.61:53     *:*
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
```

```
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*
?        ?          ?     ?  tcp4   136.186.230.61:46601  136.186.230.61:80

CHECKING DNS CONFIGURATION
==========================
Querying for (aristocrats.unix) using DNS server (136.186.230.61) - Correct answer (136.186.230.61)
Using domain server:
Name: 136.186.230.61
Address: 136.186.230.61#53
Aliases:

aristocrats.unix has address 136.186.230.61
Test Result: PASSED


ADDING DNS SERVER 136.186.230.61 TO SYSTEM RESOLV.CONF
==============================================

HTTP QUERY http://aristocrats.unix
====================================
--2019-04-08 15:27:43--  http://aristocrats.unix/
Resolving aristocrats.unix (aristocrats.unix)... 136.186.230.61
Connecting to aristocrats.unix (aristocrats.unix)|136.186.230.61|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: '/tmp/wget_save.9kNVIFCG'

     0K                                                       30.8K=0.02s

2019-04-08 15:27:43 (30.8 KB/s) - '/tmp/wget_save.9kNVIFCG' saved [494]


====================================
Downloaded Page contents
====================================
<html><head><title>Aristocrats Family Theater Resturant</title></head><body>
<center><h1>Hello!</h1></center><p>
<center><h2>The Aristocrats Family Theater Restaurant site is under construction...</h2></center></p
>
<center><h3>Check back soon for updates.</h3></center></p>
<center><h3>Remember: It is only <i>-394079263</i> seconds until our first location opens!</h3/><cen
ter>
<center><h3>Right now it is Mon Apr 8 15:27:43 EST 2019 and we open on Thu Oct 12 13:00:00 EST 2006
</body></html>
====================================

Checking Downloaded page contents

Test Result: PASS(download) + PASS(contents)


REMOVING DNS SERVER 136.186.230.61 FROM SYSTEM RESOLV.CONF
================================================

********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 13 Practical - Lab 5: Advanced Apache (P)

Lab task - Advanced Apache

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦♦♦♦ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 5: Advanced Apache (P)

*Submitted By:*
Michael Reeb
4936094
2019/04/07 09:51

*Tutor:*
Jonathan Kua

April 7, 2019

```
Marking P-Lab-05-Advanced-Apache-P on RULE host rule61
TNE30019/TNE80014 - Laboratory - Advanced Apache
================================================
********************************************************************************
 Portfolio Task: P-Lab-04-Advanced-Apache, Pass Task

 Configure your rule host (rule61) with Apache to serve a virtual hosted web site

 Tasks:
  Set rule61 as DNS Server
  Query for: armitagechemicals.unix    ; Correct answer: 136.186.230.61
  Apache running on rule61
  Browsing to http://armitagechemicals.unix  will return a correct web page
********************************************************************************

Date: 20190407_0951
Last 5 Logons:
student    pts/1    vpn247-242.cc.swin.edu Sun Apr  7 09:49   still logged in
student    pts/0    vpn247-242.cc.swin.edu Sun Apr  7 08:52   still logged in
student    pts/1    vpn247-228.cc.swin.edu Sat Apr  6 23:17 - 02:27  (03:09)
student    pts/2    vpn247-228.cc.swin.edu Sat Apr  6 22:02 - 23:15  (01:13)
student    pts/7    vpn247-49.cc.swin.edu. Mon Apr  1 14:25 - 17:35  (03:09)

RULE host Socket information
www      httpd      47022 3  tcp4   136.186.230.61:80     *:*
www      httpd      47021 3  tcp4   136.186.230.61:80     *:*
www      httpd      47020 3  tcp4   136.186.230.61:80     *:*
www      httpd      47019 3  tcp4   136.186.230.61:80     *:*
www      httpd      47018 3  tcp4   136.186.230.61:80     *:*
root     httpd      47017 3  tcp4   136.186.230.61:80     *:*
tempuser sshd       46525 3  tcp4   136.186.230.61:22     136.186.247.242:59761
tempuser sshd       46525 7  tcp4   136.186.230.61:6011   *:*
root     sshd       46522 3  tcp4   136.186.230.61:22     136.186.247.242:59761
bind     named      41933 20 tcp4   136.186.230.61:53     *:*
bind     named      41933 21 tcp4   136.186.230.61:953    *:*
bind     named      41933 512 udp4  136.186.230.61:53     *:*
bind     named      41933 513 udp4  136.186.230.61:53     *:*
bind     named      41933 514 udp4  136.186.230.61:53     *:*
bind     named      41933 515 udp4  136.186.230.61:53     *:*
bind     named      41933 516 udp4  136.186.230.61:53     *:*
bind     named      41933 517 udp4  136.186.230.61:53     *:*
bind     named      41933 518 udp4  136.186.230.61:53     *:*
bind     named      41933 519 udp4  136.186.230.61:53     *:*
bind     named      41933 520 udp4  136.186.230.61:53     *:*
bind     named      41933 521 udp4  136.186.230.61:53     *:*
bind     named      41933 522 udp4  136.186.230.61:53     *:*
bind     named      41933 523 udp4  136.186.230.61:53     *:*
bind     named      41933 524 udp4  136.186.230.61:53     *:*
bind     named      41933 525 udp4  136.186.230.61:53     *:*
bind     named      41933 526 udp4  136.186.230.61:53     *:*
bind     named      41933 527 udp4  136.186.230.61:53     *:*
bind     named      41933 528 udp4  136.186.230.61:53     *:*
bind     named      41933 529 udp4  136.186.230.61:53     *:*
bind     named      41933 530 udp4  136.186.230.61:53     *:*
bind     named      41933 531 udp4  136.186.230.61:53     *:*
bind     named      41933 532 udp4  136.186.230.61:53     *:*
bind     named      41933 533 udp4  136.186.230.61:53     *:*
bind     named      41933 534 udp4  136.186.230.61:53     *:*
tempuser sshd       36568 3  tcp4   136.186.230.61:22     136.186.247.242:56913
tempuser sshd       36568 7  tcp4   136.186.230.61:6010   *:*
```

```
root     sshd       36565 3  tcp4   136.186.230.61:22     136.186.247.242:56913
tempuser sshd       34472 3  tcp4   136.186.230.61:22     136.186.247.242:56009
root     sshd       34469 3  tcp4   136.186.230.61:22     136.186.247.242:56009
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*
?        ?          ?     ?  tcp4   136.186.230.61:48605  136.186.230.61:80

CHECKING DNS CONFIGURATION
==========================
Querying for (armitagechemicals.unix) using DNS server (136.186.230.61) - Correct answer (136.186.23
0.61)
Using domain server:
Name: 136.186.230.61
Address: 136.186.230.61#53
Aliases:

armitagechemicals.unix has address 136.186.230.61
Test Result: PASSED


ADDING DNS SERVER 136.186.230.61 TO SYSTEM RESOLV.CONF
==========================================

HTTP QUERY http://armitagechemicals.unix
====================================
--2019-04-07 09:51:07--  http://armitagechemicals.unix/
Resolving armitagechemicals.unix (armitagechemicals.unix)... 136.186.230.61
Connecting to armitagechemicals.unix (armitagechemicals.unix)|136.186.230.61|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 303 [text/html]
Saving to: '/tmp/wget_save.19SndrMV'

     0K                                                       100% 34.1M=0s

2019-04-07 09:51:07 (34.1 MB/s) - '/tmp/wget_save.19SndrMV' saved [303/303]


====================================
Downloaded Page contents
====================================
<html>
<head><title>Welcome to Armitage Chemicals</title></head>
<body>
<center>
<h1>Welcome to Armitage Chemicals</h1>
<img src=armitagechemicals.jpg>
<p>Armitage Chemicals Headquarters<p>
To contact us, please deliver all letters/emails to the guard at the front gate.<br>

</center>

</body>
</html>
====================================

Checking Downloaded page contents

Test Result: PASS(download) + PASS(contents)
```

```
REMOVING DNS SERVER 136.186.230.61 FROM SYSTEM RESOLV.CONF
==========================================

********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 14 Communications - Lab Report 2

New Description

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦♦◊ |

| Outcome | Weight |
|---|---|
| ULO5 | ♦♦♦♦◊ |

| Date | Author | Comment |
|---|---|---|
| 2019/04/30 15:23 | Jonathan Kua | Excellent work! Good to see you have included your program codes in the report, not the easiest thing with LaTeX! |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Communications - Lab Report 2

*Submitted By:*
Michael Reeb
4936094
2019/04/27 18:21

*Tutor:*
Jonathan Kua

April 27, 2019

# TNE30019 Unix for Telecommunications Lab 6 - PCAP Programming

Michael Reeb (ID: 4936094)
Swinburne University of Technology
Melbourne, Australia
4936094@student.swin.edu.au

***Abstract*—PCAP is a traffic analysis API that can be used to create programs that can capture network packets and perform analysis on them. dkpt is a Python wrapper of the Unix libpcap implementation of PCAP, and can easily be used to sort and manage the captured traffic packets, and output useful human-readable information.**

## I. Introduction to PCAP

PCAP (**P**acket **Cap**ture) is an application programming interface (API) that is used for capturing network traffic. The PCAP API is implemented in Unix-like systems under the name libpcap, and on Windows under the name Npcap. These PCAP libraries are able to save captured packets to file, read the saved packets from file; and this data can also be analysed and manipulated. These API implementations are written in C [1], and the information in the pcap file is store in their hexadecimal values.
In this lab, we used the Python module dkpt, which is a wrapper that sits on top of the C library [**?**].

## II. Aim

Dkpt was used for analysis of given PCAP files which contained previously captured TCP and UDP traffic.Relevant IP packet information was pulled from the traffic dump, and printed to the console.
In Part 2 the packets where sorted into flows and information was regarding the flow behaviour was analysed. A flow is simply a unique combination of the following parameters: the Source IP and Source Socket, the Destination IP and Destination Socket, and the protocol type. All this information was then printed to screen in a user-readable manner.

## III. Equipment

RULE host 61 was used as the server. SSH access was though PuTTY running on an external PC connected to Swinburne via VPN. Python2.7 was used as a scripting language to perform the required tasks with the PCAP libraries. The dpkt python library was used for parsing the PCAP packets.

## IV. Methodology

As per lab handout.

## V. Discussion

### A. Part 1 : mtcdump

The purpose of mtcdump was to analyse a PCAP file, and for each file in the dump, print the packet information (Timestamp, Source IP and socket, Destination IP and socket, Protocol, Packet-length and TTL) to the console.
The !# was the first line in the file and pointed to the Python2.7 binaries so when executed this script would be able to locate the Python interpreter.
In the main method of the program, the PCAP file had to be opened using the argument string given when the script was executed. This was achieved using the `open()` Python command. This file was then passed to the `dkpt.pcap.Reader()`, which is parses the file extracting the buffered hexadecimal data. This `Reader()` also implements an iterator, which will be used to cycle through to get each the next packet of hexadecimal data.
As the script iterated through the dump, each element in the iterator was extracted as a Ethernet Frame, using `eth=dpkt.ethernet.Ethernet(buf)`. Then, the IP packet was extracted from the frame using `ip=eth.data`.
With the IP packet isolated, it was now possible to extracted the information needed for printing to screen. The source IP was extracted with `ip.source` member, and was converted into the usual IPv4 format (A.B.C.D) using the socket module command: `src_ip=socket.inet_ntoa(ip.src)`. The same method was used to extract the destination port, however using `ip.dst` member to access the destination IPv4 address.

The source and destination ports were extracted using: `src_port=ip.data.sport` and `dst_port=ip.data.dport` respectively. The protocol type was extracted using the ip.p member which is an integer. This was compared to the `dpkt.ip.IP_PROTO_UDP` which is just a enumerator of all the IP protocols. Since all our traffic was either UDP or TCP, a simple if/else statement was used to determine the correct IP protocol, and then assign a correctly formatted string to a local variable to use for printing to screen.
The ttl and packet length fields were accessed using `ip.ttl` and `ip.len`. Finally, at the end of the loop a running packet count would increment per packet, and the total number of bytes would accumulate by adding the `ip.len` to a running total. All this information would be printed to screen.

### B. *Part 2 : tcpdump*

The aim of part 2 was to extract the flow information from the PCAP dump, and determine how many packets per flow there were, and the total number of bytes each flow had .
It was decided to contain all the relevant flow information within a class, called `tcpFlow`. This class would manage the flow identification, the timestamp of the beginning of each flow, the total number packets, and the total number of bytes. A constructor was would initialised these class members, importantly noting that the packet counter was initialised as 1, not 0, ensuring no off-by-one errors
In the main loop of the program, similarly to Part 1, the PCAP file was opened, read, and iterated through using the exact same method. However, instead of printing the packets individually, each packets flow was identified and if determined to be unique (ie a new flow), it would be added to a Python dictionary collection. The dictionary was chosen because it maps a key with a value.
In this case, for each packet, a key was generated with a method (`getFlowKeyNiceAndConcise(ip)`) which took an ip packet as an argument and would output a TCP flow identifier (source ip address, source port, destination ip address, destination port, and ip protocol concatenated together) as a string. This key was then checked to see if it already existed in the dictionary.
If this key was unique, then an instance of the tcpFlow() was constructed, and the key/instance pair was added in the dictionary. If the flow was seen before, then in that tcpFlow the packet count was incremented and the amount of bytes of the packet was added to the running total (using the class functions `incrementPacket()`, and `incrementBytes(bytes)` which takes the `ip.len` as an argument).
This system would find each unique flow and add them to the dict structure, however as Dictionary structure would not have the data in order needed to be marked by the RULE marking system. To solve this, a quick workaround was achieved by taking each unique flow in the dictionary, mapping the tcpFlow instances into another dict with the timestamp (of the first flow packet) as a key, and then sorting them by this by the timestamp. This issue was to do with the choice of the dictionary collection, as their functionality was confused with the C++ std::map structure functionality.
A better system would be to have used either a python-tuple or another class as the key, which would of allowed for sorting the first dictionary; or instead of using a dictionary, an array or other list could have been used, mitigating the limitations of the dictionary structure all together.

### C. *Making the scripts executable from the directory*

To make the scripts executable from the directory, the `chmod-xscriptName` command was used in root privileged. This would allow the script to be executable.

## VI. Conclusion

PCAP is a useful API for analysing network traffic data. In this lab a Python implementation was used, and a simple PCAP traffic analyser was created. In the second part, this was expanded to be able to analyse traffic flows.

## References


[1] Pcap. (2019, April 19). Retrieved from https://en.wikipedia.org/wiki/Pcap
[2] API Reference¶. (n.d.). Retrieved from https://dpkt.readthedocs.io/en/latest/api/index.html

## VII. Results

Listing 1. tcpsummary.py

```
#!/usr/local/bin/python2.7
#version 1 by Michael Reeb
################################################################################
# Sample Python program with printing functionaility for P-Lab-06-PCAP-Programming-P
################################################################################

##########
# You will probably want all these libraries for your actual program
##########
import datetime
import time
import sys
import dpkt
import socket

##########
# print_flow_summary_header()
#
# Prints the first lines of output as required by the marking script
##########
def print_flow_summary_header():
    print '\n---------- PER-FLOW INFO ----------\n'

##########
# print_flow_summary_footer()
#
# Prints the last line of output as required by the marking script
##########
def print_flow_summary_footer():
    print'-----------------------------------------------------'

##########
# print_flow_summary_single(u_int isrc_ip, u_short src_port, u_int idst_ip, u_short dst_port,
#                           char *protocol, struct timeval *start_time, u_int flow_packets, u_int flow_
#
# isrc_ip      - 32-bit IP address
# src_port     - Source port number
# idst_ip      - 32-bit IP address
# dst_port     - Destination port number
# protocol     - String containing protocol (should be 'tcp' or 'udp')
# start_time   - Timestamp of first packet in the nominated flow
# flow_packets - Integer count of total packets in this flow
# flow_bytes   - Integer sum of all packet sizes in this flow
#
# Print the lines of data as required by the marking script for ALL packets in ONE flow from the PCAP f
# The function will ensure formatting and convert IP addresses to strings for display
##########
def print_flow_summary_single(src_ip, src_port, dst_ip, dst_port, protocol, \
                  start_time, flow_packets, flow_bytes):

    print 'Flow: %s:%s-%s:%s' % (src_ip, src_port, dst_ip, dst_port)
    print 'Protocol: %s' % (protocol)
    print 'Start time: %s' % (start_time)
    print 'Total packets: %u\nTotal bytes: %u\n' % (flow_packets, flow_bytes)

##########
# run_example
#
```

```
59 # Main function of sample program to demonstrate use of provided functions
60 #
61 # 1) Parse command line parameter. Fail if PCAP filename not provided
62 # 2) Call print_flow_summary_header() to display initial output
63 # 3) Generate fake data for flow 1 to print as an example
64 # 4) Call print_flow_summary_single() to display fake data. You will need to this once for each flow in
65 # 5) Generate more fake data for flow 2 to print as an example
66 # 6) Call print_flow_summary_single() to display flow 2 fake data
67 # 7) Call print_flow_summary_footer() to display final output
68 ##########
69
70 def getFlowKeyNiceAndConcise(ip):
71     src_ip = socket.inet_ntoa(ip.src)
72     src_port = ip.data.sport
73     dst_ip = socket.inet_ntoa(ip.dst)
74     dst_port = ip.data.dport
75     if ip.p == dpkt.ip.IP_PROTO_UDP:
76         protocol = "udp"
77     else:
78         protocol = "tcp"
79
80
81
82     temp ='%s:%s -> %s:%s (%s)' % \
83       (src_ip, src_port, dst_ip, dst_port, protocol)
84     return temp
85
86 class tcpFlow:
87 #-----------------------------------------------------------------------
88     def __init__(self,ip_packet,time_stamp):
89         self.source_ip = socket.inet_ntoa(ip_packet.src)
90         self.src_port = ip_packet.data.sport
91         self.dst_ip = socket.inet_ntoa(ip_packet.dst)
92         self.dst_port = ip_packet.data.dport
93         self.time_stamp = time_stamp
94         self.total_packets = 1
95         self.total_bytes =ip_packet.len
96         if ip_packet.p == dpkt.ip.IP_PROTO_UDP:
97             self.protocol = "udp"
98         else:
99             self.protocol = "tcp"
100
101
102     # if ip_packet.p == dpkt.ip.IP_PROTO_UDP:
103  #       self.protocol = "udp"
104     # else:
105  #       self.protocol = "tcp"
106
107 #-----------------------------------------------------------------------
108     def getTimeStamp(self):
109         return self.time_stamp
110
111
112     def incrementPacket(self):
113         self.total_packets +=1
114 #-----------------------------------------------------------------------
115     def incrementBytes(self,bytes):
116         self.total_bytes += bytes
117         #print the summary of the flow
118     def printInstanceFlowSummary(self):
119         print_flow_summary_single(self.source_ip,self.src_port,self.dst_ip,self.dst_port,self.protocol,
120
121
```

```
122 ##############################################################################################
123
124 def run_example():
125
126     global total_packet_count, total_bytes_count
127
128     try:
129     sys.argv[1]
130     dmp_file = sys.argv[1]
131     fp_dmp_file = open(dmp_file)
132     except Exception as e:
133     print 'Error: please supply pcap filename!\n'
134     return
135
136     print_flow_summary_header()
137
138     # need to move this into a class file, then create a list of these new classes for each flow
139     pcap = dpkt.pcap.Reader(fp_dmp_file)
140
141     #declare tcp flows dict to manage flows. dicts use key:flow arrangement
142     tcp_flows = dict()
143
144     for ts, buf in pcap:
145         eth = dpkt.ethernet.Ethernet(buf)
146         ip = eth.data
147         #create key varible here to make below look nicer
148         key = getFlowKeyNiceAndConcise(ip)
149        # print'---------------------------- current flow key%s' %(key)
150         #check to see if flow already exists. if it does, increment the counters, if not construct new
151         if key in tcp_flows.keys():
152             tcp_flows[key].incrementPacket()
153             tcp_flows[key].incrementBytes(ip.len)
154         else: #add new instance of a TCP flows to list, have to pass constructor time and ip packet
155             timestamp = str(datetime.datetime.utcfromtimestamp(ts))
156             tcp_flows[key] = tcpFlow(ip,timestamp)
157             #tcp_flows.update(key:tcpFlow(ip,str(datetime.datetime.utcfromtimestamp(ts))))
158     #-------------------------------------------------------------------------------------------
159     #create a second list because python dict sucks and is nothing like cpp map
160     #so we have to find a stupid work around to sort the damn thing.
161     #so we create a new dictionary that associates the timestamp to the class.
162     sorted_flows = dict()
163     for keys in tcp_flows:
164         tempKey = tcp_flows[keys].getTimeStamp()
165         sorted_flows[tempKey]=tcp_flows[keys]
166
167        #now we sort that list by timestamp and print the summary to the screen so we have things in the
168     for k in sorted(sorted_flows):
169         sorted_flows[k].printInstanceFlowSummary()
170
171
172
173
174     # start_time = time.time()
175     # src_ip = '1.2.3.4'
176     # src_port = '99'
177     # dst_ip = '5.6.7.8'
178     # dst_port = '199'
179     # protocol = 'tcp'
180
181     # print_flow_summary_single(src_ip, src_port, dst_ip, dst_port, protocol, start_time, 100, 150000)
182
183     # start_time = time.time()
184     # src_ip = '10.0.0.1'
```

```
185     # src_port = '123'
186     # dst_ip = '10.1.1.200'
187     # dst_port = '456'
188     # protocol = 'udp'
189
190     # print_flow_summary_single(src_ip, src_port, dst_ip, dst_port, protocol, start_time, 8, 12000)
191
192     print_flow_summary_footer()
193
194     fp_dmp_file.close()
195
196 if __name__ == '__main__':
197     run_example()
```

Listing 2. mtcpdump.py

```
1  #!/usr/local/bin/python2.7
2  #version 1 by Michael Reeb
3  ################################################################################
4  # Sample Python program with printing functionaility for P-Lab-06-PCAP-Programming-P
5  ################################################################################
6
7  ##########
8  # You will probably want all these libraries for your actual program
9  ##########
10 import datetime
11 import time
12 import sys
13 import dpkt
14 import socket
15
16 ##########
17 # print_packet_info(ts, src_ip, src_port, dst_ip, dst_port, protocol, pkt_len, ttl)
18 #
19 # ts       - Packet timestamp
20 # isrc_ip  - 32-bit IP address
21 # src_port - Source port number
22 # idst_ip  - 32-bit IP address
23 # dst_port - Destination port number
24 # protocol - String containing protocol (should be 'tcp' or 'udp')
25 # pkt_len  - Packet length as integer
26 # ip_ttl   - IP Packet TTL as byte value
27 #
28 # Print the line of data as required by the marking script for ONE packet in the PCAP file
29 # The function will ensure formatting and convert IP addresses to strings for display
30 ##########
31 def print_packet_info(ts, src_ip, src_port, dst_ip, dst_port, protocol, pkt_len, ttl):
32
33     # get UTC timestamp, source/destination IP/port
34     utc_timestamp = str(datetime.datetime.utcfromtimestamp(ts))
35
36     # create log line
37     log_line = '[%s] - %s:%s -> %s:%s (%s, len=%d, ttl=%d)' % \
38       (utc_timestamp, src_ip, src_port, dst_ip, dst_port, protocol, pkt_len, ttl)
39
40     print log_line
41
42 ##########
43 # print_summary(total_packet_count, total_bytes_count, average_pkt_size)
44 #
45 # total_pkts - Integer count of total packets
46 # total_bytes - Integer count of sum of packet sizes
47 # avg_pkt     - Integer representation of average packet size
```

```
48  #
49  # Print the three parameters in the format as required by the marking script
50  ##########
51  def print_summary(total_packet_count, total_bytes_count, average_pkt_size):
52
53      print '\n---------- SUMMARY ----------\n'
54      print 'Total packets: %i' % (total_packet_count)
55      print 'Total bytes (bytes): %i' % (total_bytes_count)
56      print 'Average packet size (bytes): %i' % (average_pkt_size)
57      print '\n-----------------------------'
58
59  ##########
60  # run_example
61  #
62  # Main function of sample program to demonstrate use of provided functions
63  #
64  # 1) Parse command line parameter. Fail if PCAP filename not provided and file cannot be opened
65  # 2) Generate fake data to print as an example
66  # 3) Call print_packet_info() to display fake data. You will need to do this once for each packet in th
67  # 4) Call print_summary() to display a fake summary. You will need to do this with your accumulated dat
68  ##########
69  def run_example():
70
71      global total_packet_count,  total_bytes_count
72      pkt_count = 0
73      byte_count = 0
74
75      try:
76      sys.argv[1]
77      dmp_file = sys.argv[1]
78      fp_dmp_file = open(dmp_file)
79      except Exception as e:
80      print 'Error: please supply pcap filename!\n'
81      return
82
83  #For each packet in the dmp_file
84
85      pcap = dpkt.pcap.Reader(fp_dmp_file)
86
87      for ts, buf in pcap:
88          eth = dpkt.ethernet.Ethernet(buf)
89          ip = eth.data
90          src_ip = socket.inet_ntoa(ip.src)
91          src_port = ip.data.sport
92          dst_ip = socket.inet_ntoa(ip.dst)
93          dst_port = ip.data.dport
94      if ip.p == dpkt.ip.IP_PROTO_UDP:
95          protocol = "udp"
96      else:
97          protocol = "tcp"
98      ttl = ip.ttl
99      len = ip.len
100         print_packet_info(ts, src_ip, src_port, dst_ip, dst_port, protocol, len, ttl)
101         pkt_count += 1
102         byte_count += ip.len
103
104
105
106
107
108
109 #For each packet in the dmp_file
110
```

```
111
112
113
114         print_summary(pkt_count, byte_count, byte_count/ pkt_count)
115
116         fp_dmp_file.close()
117
118 if __name__ == '__main__':
119         run_example()
```

# 15 Practical - Lab 6: PCAP Programming (D)

Lab task - PCAP Programming (Distinction)

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 6: PCAP Programming (D)

*Submitted By:*
Michael Reeb
4936094
2019/04/16 14:58

*Tutor:*
Jonathan Kua

April 16, 2019

```
Marking P-Lab-06-PCAP-Programming-D on RULE host rule61
TNE30019/TNE80014 - Laboratory - PCAP Programming
================================================
********************************************************************************
 Portfolio Task: P-Lab-06-PCAP-Programming, Distinction Task

 Develop a tcpsummary program to output a summary of all flows within a tcpdump
 PCAP file

 Tasks:
  Create an executable as /home/student/lab06/tcpsummary
  Executable takes one parameter - name of PCAP file
  Output standard summary block of text for each unique flow in PCAP file
********************************************************************************

Date: 20190416_1455
Last 5 Logons:
student    pts/8    vpn247-130.cc.swin.edu Tue Apr 16 12:05   still logged in
student    pts/4    vpn247-202.cc.swin.edu Mon Apr 15 21:59 - 01:09  (03:09)
student    pts/4    vpn247-202.cc.swin.edu Mon Apr 15 21:25 - 21:59  (00:34)
student    pts/5    ses236-63.cc.swin.edu. Mon Apr 15 17:18 - 17:30  (00:12)
student    pts/5    ses236-63.cc.swin.edu. Mon Apr 15 17:15 - 17:16  (00:00)

CHECKING EXECUTABLE
====================
Checking executable (/usr/home/student/pcap_lab/tcpsummary), permissions(rwx??????)...

-rwxr-xr-x 1 0 1001 6942 Apr 16 14:52 /usr/home/student/pcap_lab/tcpsummary

RUNNING TESTS
=============
============================================================
Running test: /usr/home/student/pcap_lab/tcpsummary test1.pcap

--- Program output ---------------------------------------

--------------- PER-FLOW INFO -----------------

Flow: 172.16.11.2:61016-172.16.10.2:80
Protocol: tcp
Start time: 2018-08-16 02:48:10.238506
Total packets: 6
Total bytes: 466

Flow: 172.16.10.2:80-172.16.11.2:61016
Protocol: tcp
Start time: 2018-08-16 02:48:10.238523
Total packets: 5
Total bytes: 986

Flow: 172.16.11.3:37269-172.16.10.2:12345
Protocol: udp
Start time: 2018-08-16 02:48:13.537505
Total packets: 5
Total bytes: 7490

Flow: 172.16.10.2:12345-172.16.11.3:37269
Protocol: udp
Start time: 2018-08-16 02:48:13.591438
Total packets: 1
```

```
Total bytes: 1498

-----------------------------------------------
------------------------------------------------------------

Test Result: PASSED

============================================================
Running test: /usr/home/student/pcap_lab/tcpsummary test2.pcap

--- Program output ---------------------------------------

--------------- PER-FLOW INFO -----------------

Flow: 172.16.11.72:33711-172.16.10.62:5001
Protocol: tcp
Start time: 2018-08-01 03:08:34.575832
Total packets: 156
Total bytes: 234000

Flow: 172.16.10.62:5001-172.16.11.72:33711
Protocol: tcp
Start time: 2018-08-01 03:08:34.576840
Total packets: 78
Total bytes: 4056

-----------------------------------------------
------------------------------------------------------------

Test Result: PASSED

============================================================
Running test: /usr/home/student/pcap_lab/tcpsummary test3.pcap

Program output: Hidden

Test Result: PASSED

********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 16 Practical - Lab 6: PCAP Programming (P)

Lab task - PCAP Programming

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 6: PCAP Programming (P)

*Submitted By:*
Michael Reeb
4936094
2019/04/15 22:33

*Tutor:*
Jonathan Kua

April 15, 2019

```
Marking P-Lab-06-PCAP-Programming-P on RULE host rule61
TNE30019/TNE80014 - Laboratory - PCAP Programming
================================================
********************************************************************************
 Portfolio Task: P-Lab-06-PCAP-Programming, Pass Task

 Develop a mini-tcpdump program to output a summary line for each packet in a
 tcpdump PCAP file

 Tasks:
  Create an executable as /home/student/lab06/mtcpdump
  Executable takes one parameter - name of PCAP file
  Output standard summary line for each packet in PCAP file
  Output summary of total bytes and packets captured, and average packet size
********************************************************************************

Date: 20190415_2227
Last 5 Logons:
student    pts/4    vpn247-202.cc.swin.edu Mon Apr 15 21:59   still logged in
student    pts/4    vpn247-202.cc.swin.edu Mon Apr 15 21:25 - 21:59  (00:34)
student    pts/5    ses236-63.cc.swin.edu. Mon Apr 15 17:18 - 17:30  (00:12)
student    pts/5    ses236-63.cc.swin.edu. Mon Apr 15 17:15 - 17:16  (00:00)
student    pts/15   ses236-63.cc.swin.edu. Mon Apr 15 17:02 - 17:14  (00:11)

CHECKING EXECUTABLE
===================
Checking executable (/usr/home/student/pcap_lab/mtcpdump), permissions(rwx??????)...

-rwxr-xr-x 1 0 1001 3722 Apr 15 22:03 /usr/home/student/pcap_lab/mtcpdump

RUNNING TESTS
=============
============================================================
Running test: /usr/home/student/pcap_lab/mtcpdump test1.pcap

--- Program output ---------------------------------------
[2018-08-16 02:48:10.238506] - 172.16.11.2:61016 -> 172.16.10.2:80 (tcp, len=60, ttl=63)
[2018-08-16 02:48:10.238523] - 172.16.10.2:80 -> 172.16.11.2:61016 (tcp, len=60, ttl=64)
[2018-08-16 02:48:10.288710] - 172.16.11.2:61016 -> 172.16.10.2:80 (tcp, len=52, ttl=63)
[2018-08-16 02:48:10.288738] - 172.16.11.2:61016 -> 172.16.10.2:80 (tcp, len=198, ttl=63)
[2018-08-16 02:48:10.288746] - 172.16.10.2:80 -> 172.16.11.2:61016 (tcp, len=52, ttl=64)
[2018-08-16 02:48:10.288860] - 172.16.10.2:80 -> 172.16.11.2:61016 (tcp, len=770, ttl=64)
[2018-08-16 02:48:10.288865] - 172.16.10.2:80 -> 172.16.11.2:61016 (tcp, len=52, ttl=64)
[2018-08-16 02:48:10.339317] - 172.16.11.2:61016 -> 172.16.10.2:80 (tcp, len=52, ttl=63)
[2018-08-16 02:48:10.339331] - 172.16.11.2:61016 -> 172.16.10.2:80 (tcp, len=52, ttl=63)
[2018-08-16 02:48:10.339563] - 172.16.11.2:61016 -> 172.16.10.2:80 (tcp, len=52, ttl=63)
[2018-08-16 02:48:10.339586] - 172.16.10.2:80 -> 172.16.11.2:61016 (tcp, len=52, ttl=64)
[2018-08-16 02:48:13.537505] - 172.16.11.3:37269 -> 172.16.10.2:12345 (udp, len=1498, ttl=63)
[2018-08-16 02:48:13.549001] - 172.16.11.3:37269 -> 172.16.10.2:12345 (udp, len=1498, ttl=63)
[2018-08-16 02:48:13.560213] - 172.16.11.3:37269 -> 172.16.10.2:12345 (udp, len=1498, ttl=63)
[2018-08-16 02:48:13.571413] - 172.16.11.3:37269 -> 172.16.10.2:12345 (udp, len=1498, ttl=63)
[2018-08-16 02:48:13.582614] - 172.16.11.3:37269 -> 172.16.10.2:12345 (udp, len=1498, ttl=63)
[2018-08-16 02:48:13.591438] - 172.16.10.2:12345 -> 172.16.11.3:37269 (udp, len=1498, ttl=64)

------------------- SUMMARY -------------------

Total packets: 17
Total bytes (bytes): 10440
Average packet size (bytes): 614
```

```
-------------------------------------------
------------------------------------------------------------

Test Result: PASSED

============================================================
Running test: /usr/home/student/pcap_lab/mtcpdump test2.pcap

--- Program output ---------------------------------------
[2018-08-01 03:08:34.575832] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.576818] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.576840] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.577830] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.578830] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.578863] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.579829] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.580817] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.580849] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.581830] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.582831] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.582863] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.583830] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.584818] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.584853] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.585855] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.586855] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.586927] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.587854] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.588842] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.588911] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.589854] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.590895] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.590953] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.591899] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.592885] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.592942] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.593898] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.594898] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.594956] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.595898] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.596893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.596948] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.597872] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.598855] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.598882] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.599855] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.600883] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.600928] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.601898] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.602897] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.602942] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.603899] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.604884] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.604930] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.605896] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.606898] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.606943] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.607900] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.608883] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.608940] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
```

```
[2018-08-01 03:08:34.609896] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.610897] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.610954] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.611872] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.612860] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.612904] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.613872] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.614873] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.614916] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.615872] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.616859] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.616902] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.617868] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.618871] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.618913] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.619867] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.620884] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.620941] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.621897] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.622897] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.622954] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.623897] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.624884] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.624940] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.625898] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.626876] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.626921] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.627855] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.628842] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.628913] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.629891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.630893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.630934] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.631892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.632880] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.632921] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.633892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.634892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.634933] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.635892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.636880] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.636920] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.637892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.638867] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.638898] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.639893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.640854] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.640885] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.641893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.642869] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.642900] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.643894] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.644856] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.644886] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.645892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.646867] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.646897] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.647892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.648839] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.648888] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
```

```
[2018-08-01 03:08:34.649892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.650871] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.650907] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.651893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.652880] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.652914] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.653893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.654893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.654928] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.655892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.656879] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.656912] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.657892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.658893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.658926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.659892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.660878] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.660911] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.661891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.662893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.662926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.663892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.664879] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.664913] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.665892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.666892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.666926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.667892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.668879] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.668912] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.669892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.670892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.670925] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.671892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.672878] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.672912] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.673891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.674892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.674925] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.675891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.676878] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.676911] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.677894] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.678892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.678924] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.679893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.680878] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.680911] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.681893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.682893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.682926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.683891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.684880] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.684913] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.685891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.686893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.686926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.687893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.688879] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.688913] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
```

```
[2018-08-01 03:08:34.689893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.690892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.690926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.691893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.692880] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.692913] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.693900] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.694891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.694924] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.695893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.696877] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.696910] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.697893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.698891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.698924] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.699890] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.700881] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.700914] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.701892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.702894] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.702927] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.703892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.704878] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.704911] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.705893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.706893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.706926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.707893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.708880] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.708913] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.709890] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.710893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.710926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.711891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.712881] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.712914] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.713894] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.714893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.714926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.715891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.716878] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.716912] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.717893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.718892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.718925] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.719891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.720879] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.720913] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.721893] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.722891] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.722926] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.723868] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.724878] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.724911] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.725892] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.726868] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.726887] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
[2018-08-01 03:08:34.727867] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.728857] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.728873] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)
```

```
[2018-08-01 03:08:34.729868] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.730868] - 172.16.11.72:33711 -> 172.16.10.62:5001 (tcp, len=1500, ttl=63)
[2018-08-01 03:08:34.730884] - 172.16.10.62:5001 -> 172.16.11.72:33711 (tcp, len=52, ttl=64)

------------------- SUMMARY -------------------

Total packets: 234
Total bytes (bytes): 238056
Average packet size (bytes): 1017

-----------------------------------------------
-------------------------------------------------------------

Test Result: PASSED

=============================================================
Running test: /usr/home/student/pcap_lab/mtcpdump test3.pcap

Program output: Hidden

Test Result: PASSED

*******************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 17 Practical - Lab 7: NMap (C)

Lab task - Nmap (Credit)

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

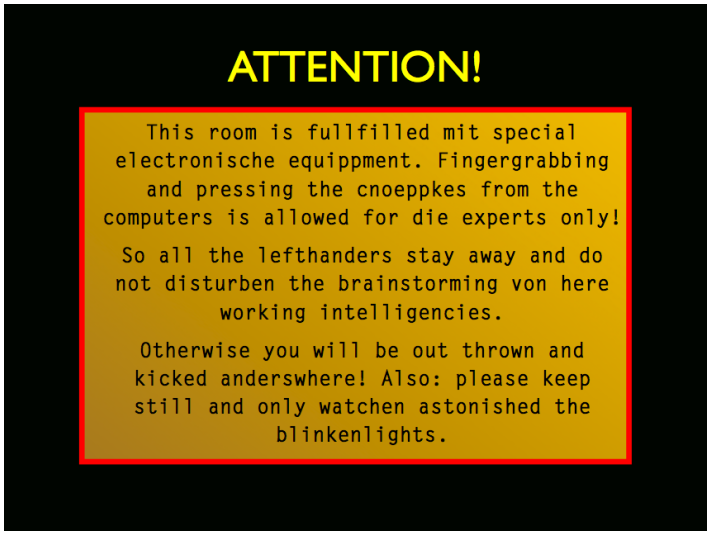
Doubtfire Submission

# Practical - Lab 7: NMap (C)

*Submitted By:*
Michael Reeb
4936094
2019/04/12 16:02

*Tutor:*
Jonathan Kua

April 12, 2019




# Secret Message: No spitzensparken for the new RULE system

# 18 Practical - Lab 7: NMap (P)

Lab task - Nmap

| Outcome | Weight |
| --- | --- |
| ULO3 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 7: NMap (P)

*Submitted By:*
Michael Reeb
4936094
2019/04/12 15:59

*Tutor:*
Jonathan Kua

April 12, 2019

```
student@rule61:~ % nmap -v -r -p- -sV  rule21.caia.swin.edu.au
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-12 15:46 EST
NSE: Loaded 43 scripts for scanning.
Initiating Ping Scan at 15:46
Scanning rule21.caia.swin.edu.au (136.186.230.21) [2 ports]
Completed Ping Scan at 15:46, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 15:46
Completed Parallel DNS resolution of 1 host. at 15:46, 0.00s elapsed
Initiating Connect Scan at 15:46
Scanning rule21.caia.swin.edu.au (136.186.230.21) [65535 ports]
Discovered open port 7/tcp on 136.186.230.21
Discovered open port 13/tcp on 136.186.230.21
Discovered open port 21/tcp on 136.186.230.21
Discovered open port 22/tcp on 136.186.230.21
Discovered open port 23/tcp on 136.186.230.21
Discovered open port 79/tcp on 136.186.230.21
Discovered open port 80/tcp on 136.186.230.21
Discovered open port 110/tcp on 136.186.230.21
Discovered open port 143/tcp on 136.186.230.21
Increasing send delay for 136.186.230.21 from 0 to 5 due to
max_successful_tryno increase to 4
Connect Scan Timing: About 9.14% done; ETC: 15:52 (0:05:08 remaining)
Connect Scan Timing: About 18.29% done; ETC: 15:52 (0:04:33 remaining)
Connect Scan Timing: About 27.43% done; ETC: 15:52 (0:04:01 remaining)
Connect Scan Timing: About 36.58% done; ETC: 15:52 (0:03:30 remaining)
Connect Scan Timing: About 45.72% done; ETC: 15:52 (0:02:59 remaining)
Connect Scan Timing: About 54.87% done; ETC: 15:52 (0:02:29 remaining)
Connect Scan Timing: About 64.01% done; ETC: 15:52 (0:01:59 remaining)
Connect Scan Timing: About 73.16% done; ETC: 15:52 (0:01:28 remaining)
Connect Scan Timing: About 82.31% done; ETC: 15:52 (0:00:58 remaining)
Discovered open port 53959/tcp on 136.186.230.21
Completed Connect Scan at 15:52, 328.98s elapsed (65535 total ports)
Initiating Service scan at 15:52
Scanning 10 services on rule21.caia.swin.edu.au (136.186.230.21)
Completed Service scan at 15:52, 6.01s elapsed (10 services on 1 host)
NSE: Script scanning 136.186.230.21.
Initiating NSE at 15:52
Completed NSE at 15:52, 0.07s elapsed
Initiating NSE at 15:52
Completed NSE at 15:52, 0.00s elapsed
Nmap scan report for rule21.caia.swin.edu.au (136.186.230.21)
Host is up (0.000073s latency).
rDNS record for 136.186.230.21: passTest.unix
Not shown: 65525 closed ports
PORT      STATE SERVICE    VERSION
7/tcp     open  echo
13/tcp    open  daytime
21/tcp    open  ftp        WU-FTPD or MIT Kerberos ftpd 6.00LS
22/tcp    open  ssh        OpenSSH 7.2 (FreeBSD 20160310; protocol 2.0)
23/tcp    open  telnet     BSD-derived telnetd
79/tcp    open  finger     FreeBSD fingerd
```

```
80/tcp    open  http       thttpd 2.27 19Oct2015
110/tcp   open  tcpwrapped
143/tcp   open  tcpwrapped
53959/tcp open  http       thttpd 2.27
Service Info: Host: rule21; OSs: Unix, FreeBSD; CPE: cpe:/o:freebsd:freebsd

Read data files from: /usr/local/share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 336.26 seconds
```

# 19 Practical - Programming Tutorial

Successful completion of programming task in tutorial

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦♦♦◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Programming Tutorial

*Submitted By:*
Michael Reeb
4936094
2019/05/09 13:12

*Tutor:*
Jonathan Kua

May 9, 2019

```c
// visit https://www.binarytides.com/receive-full-data-with-recv-socket-function-in-c/


#include <stdio.h>
#include <stdlib.h>

#define __USE_GNU
#include <string.h>

#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#include <errno.h>
#include<unistd.h>
#define WWW_PORT    80

//-------------------------------------------------------------------------
// Print an error message to stderr and terminate the program immediately
void printerror(char *pcErrStr) {
    fprintf(stderr, "ERROR: %s\n\n", pcErrStr);
    exit(1);
}

//-------------------------------------------------------------------------
// Input parameters are the name of the server to query and a pointer to a
// sockaddr_in structure to fill.  The function performs a lookup of the
// specified Internet name and stores the address along with the web port
// number (WWW_PORT) in the provided sockaddr_in structure
void FillAddress(char *pcURL, struct sockaddr_in *psAddress) {
    struct hostent      *psHostDetails;

    if ((psHostDetails = gethostbyname(pcURL)) == NULL) printerror("Can't determine address of website");

    memcpy(&(psAddress->sin_addr), psHostDetails->h_addr_list[0], 4);
    psAddress->sin_family = AF_INET;
    psAddress->sin_port = htons(WWW_PORT);

    printf("Server details:\n");
    printf("  Server:\t%s\n", pcURL);
    printf("  IP Address:\t%s\n", inet_ntoa(psAddress->sin_addr));
}

//-------------------------------------------------------------------------
int main(int iArgC, char **ppcArgV) {
    struct sockaddr_in  sAddress;
    char                *pcWWWServer, *pcPage, *pcPos;

    if (iArgC != 2) printerror("Command Line requires one parameter = website");

```

```
52      // If the provided URL begins with "http://" then set pcWWWServer to point
53      // to the next character otherwise set it to point to the entire provided URL
54      if (pcWWWServer = strstr(ppcArgV[1], "http://")) pcWWWServer+= 7; else
        ↪  pcWWWServer = ppcArgV[1];
55
56      // If there is a "/" in the URL (after the http:// has been stripped) then
57      // replace it with a 0 so the string terminates as just the name.  Create
58      // a new string pointer (pcPage) to be a copy of the remaining string OR
59      // just "/" if no page details have been provided
60      if (pcPos = strstr(pcWWWServer, "/"))
61      {
62          pcPage = strdup(pcPos);
63          pcPos[0] = 0;
64      } else
65          pcPage = strdup("/");
66
67      // Call FillAddress() to fill sAddress with the IP address details (and port
68      // number 80) of the requested URL
69      FillAddress(pcWWWServer, &sAddress);
70
71      // Put rest of code here_____________________________________________________
        ↪  ________________________
72      //create a socket. note socket returns an integer based upon its success
73      //
74      // int socket(int domain, int type, int protocol);
75      int mySock=socket(AF_INET,SOCK_STREAM,0);
76      if( mySock<0) {
77          printf("Error creating socket\n");
78          exit(1);
79      }
80      //try and connect socket to server
81      //
82      //  int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
83       if(connect(mySock,(struct sockaddr*)&sAddress, sizeof(sAddress) ) != 0){
84          printf("Error in connecting socket\n");
85             exit(1);
86       }
87  // create a plain-text GET request for HTTP. should included other HTTP versions
88       //
89       // GET <pgName> HTTP/1.1\r\nHost: <hostName>\r\nConnection: close\r\n\r\n"
90  char httpRequest[1024];
91  sprintf(httpRequest, "GET %s HTTP/1.1\r\nHost: %s\r\nConnection: close\r\n\r\n",
    ↪  pcPage, pcWWWServer);
92
93  //http://man7.org/linux/man-pages/man2/send.2.html
94  //send the httpMessage through the socket to the server
95  //ssize_t send(int socket, const void *buffer, size_t length, int flags);
96  //returns the amount of bytes sent.
97  //returns -1 if there is an error
98  ssize_t hasSent = send(mySock,httpRequest,sizeof(httpRequest),0);
99  //check if it has sent without issues. exit on error
100 if(hasSent < 0){
101     printf("Error in socket sending data\n");
```

```
102     exit(1);
103     }
104
105
106 //http://man7.org/linux/man-pages/man2/recv.2.html
107 //listen to socket for response.
108 //recv function is used to do this.
109 //
110 // ssize_t recv(int sockfd, void *buf, size_t len, int flags);
111 // until a message is ready to receive, recv returns -1
112 //also need a buffer
113 char recvBuffer[16000];
114 //open a file with fopen() for writing the scrape to
115 FILE* outputTxtFile;
116 if ((outputTxtFile = fopen("scrape.txt","w")) == NULL){
117     printf("Error cannot open file\n");
118     exit(1);
119 }
120
121 while(recv(mySock,recvBuffer, sizeof(recvBuffer),0) > 0 ){
122   //printf("%s\n",recvBuffer);
123  //print buffer to file. add null chars on end and flush
124  // fprintf(outputTxtFile,recvBuffer);
125   fputs(recvBuffer,outputTxtFile);
126   fflush(outputTxtFile);
127 }
128 //fprintf(outputTxtFile,recvBuffer);
129 //printf("%s\n",recvBuffer);
130
131
132 //http://man7.org/linux/man-pages/man3/fopen.3.html
133 // /*fopen(const char *pathname, const char *mode);
134 //mode "w" Truncate file to zero length or create text file for writing.
135 //         The stream is positioned at the beginning of the file.
136 //outputTxtFile
137
138
139 fclose(outputTxtFile);
140 printf("%s\n",recvBuffer);
141 printf("-------------------Scrape Completed---------------------- \n");
142 close(mySock);
143     return 0;
144 }
```

# 20 Theory - Jails

Brief report summarising jail operations

| Outcome | Weight |
|---|---|
| ULO5 | ♦♦♦♦♦ |

| Date | Author | Comment |
|---|---|---|
| 2019/04/30 15:04 | Jonathan Kua | Excellent work! |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Theory - Jails

*Submitted By:*
Michael Reeb
4936094
2019/04/27 21:59

*Tutor:*
Jonathan Kua

April 27, 2019

# A Brief Overview of Operating System Virtualisation

Michael Reeb - 4936094
*TNE30019 Unix for Telecommunications*
*Communications Report*
4936094@student.swin.edu.au

***Abstract*—Hardware virtualisation allows a computer to logically divide its resources to operate like separate machines. These separate machines can run differing operating systems each with its own kernel, in the case of a virtual machine, or can be a compartmentalised space that shares its kernel with the other instances but processes are limited to that instances workspace, such is the case with FreeBSD Jails.**

## I. Introduction

Hardware virtualisation is the process of logically dividing its resources to allow multiple Virtual Machines (VM) to run simultaneously. These VMs perform like a regular physical machine, and ideally to the Operating System (OS) and/or applications running on them, VMs are indistinguishable from a physical machine.

## II. Virtualisation

The simplest form of Virtualisation is emulation: the host computer simulates all the underlying physical hardware (memory, CPU, CPU registers, etc) in the emulator. The Guest Operating system; that is the system that is to be emulated, then runs inside this emulator and behaves as it would if it was running on physical hardware, without any support from the host Operating System.
While hardware emulation works well for modern computers emulating old (and less powerful) hardware; it is not feasible for a computer to emulate an equivalent system.
A Virtual Machine Monitor (VMM), or Hypervisor allows the Guest OS to run directly on the CPU, and when the Guest OS attempts to execute privileged instructions (which it does not have permission to perform as it is operating in user mode) the Hypervisor traps them, and then steps in and emulates the instruction on the virtual machine. Thus, a hypervisor simply monitors the guest OS and emulates the Guest OS privileged instruction calls, such as memory allocation, I/O operation, context switching etc.

A Virtual Machine Monitor, or Hypervisor creates the illusion that there are multiple machines running on the same physical hardware. It is essential that a Hypervisor be able to score well in following three dimensions to ensure that the Virtual Machine behaves identically to a real physical machine:

1) **Saftey**: the Hypervisor should be in full control of the visualised resources.
2) **Fidelity**: the behaviour of a program executing in a VM should be identical to that of one executing on a physical machine.
3) **Efficiency** As much as possible of the program code should run without the Hypervisor intervening.

Thus, for a machine to achieve virtualisation, the sensitive instructions must be a subset of the privileged instructions.

There is no clear distinction between the categories of Hypervisors, and certain Hypervisor implementations can fall into multiple categories. However, in this paper we will draw two-distinct categories for hypervisors: type 1, where the virtual machine runs directly on a Hypervisor incorporated with the kernel, and Type-2 hypervisors where the virtual machine runs as an independent application on the host OS like any other application.

### A. Type 1 Hypervisor

A Type-1 Hypervisor is similar to an operating system as it is the only process that is running in the most privileged mode and has direct access to the systems hardware. This is colloquially known as a ”bare-metal ” hypervisor.

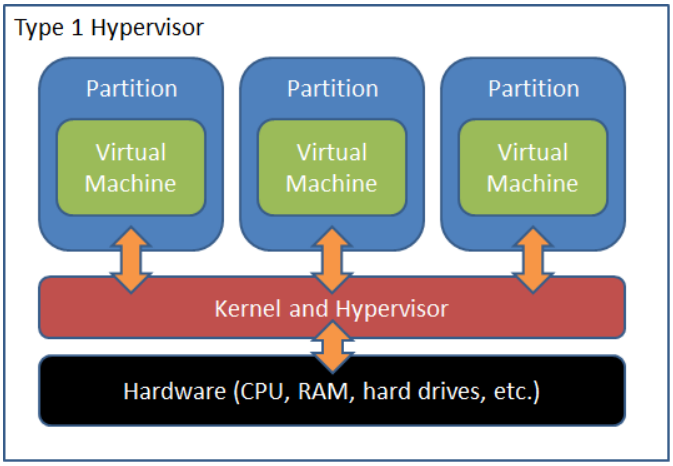



Fig. 1. A Type-1 hypervisor access flow. The Virtual Machine accesses the Hypervisor which then translates these access calls to system hardware [1]

Type 1 Hypervisors directly manage the hardware and memory to to create ”virtual machines” that guest operating systems

can execute within. An example of the Type-1 Hypervisor flow can be seen in Figure 1.

### B. Type 2 Hypervisor

Unlike a Type-1 Hypervisor which runs in the most privileged mode and has access to system hardware, Type-2 Hypervisor is simply an application that runs on a ”host” operating system which creates a virtual machine that the guest operating systems can execute in.

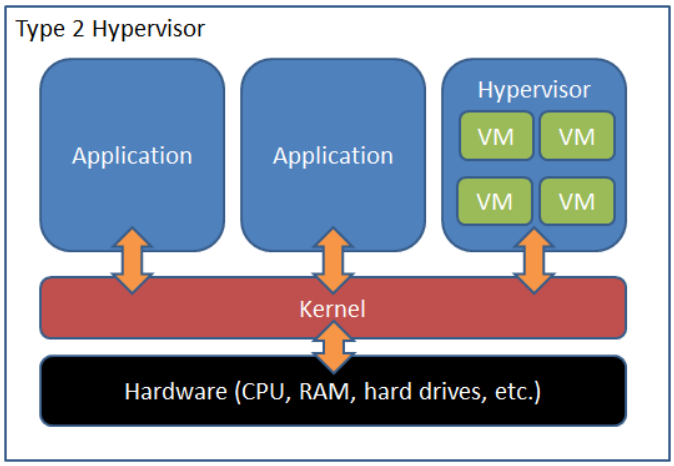



Fig. 2. A Type-1 hypervisor access flow. The Virtual Machine accesses the Hypervisor which then translates these access calls to system hardware [1]

Management of memory and hardware is performed by the Host OS like any other application that is run.

### C. Virtual Machine Uses

Virtual machines are useful as they are both able to ensure complete isolation from each other, as well the ability to run different operating systems that can be configured to optimise the running application. An example use of a Type-1 Hypervisor running multiple virtual machines is having a single piece hardware running multiple VMs that each host a different server. That way, if one server is compromised, the hackers bound to only that virtual machine. Secondly, if a server crashes the operating system, only that virtual machine will be affected and the other VMs operate normally.
An example use of a Type-2 system is a software developer who is developing a cross-platform application. Instead of having to have multiple different physical computers, the developer could run the other operating system inside the virtual machine and perform all the code compilation and testing inside the VM.

## III. Jails

### A. chroot Jail

`chroot` is a Unix operation that changes the apparent working directory for a given process and its children. The processes cannot see or access files outside the designated directory tree, and the artificial root directory is called a ”chroot jail”.

A chroot jail is not a good idea for security purposes, such as isolating server systems from each-other, as is possible for privileged root users to escape the jail via performing a second ”chroot”, or mounting a file system.
Instead, chroot is used in applications such as testing and development, where isolation of the processes simplifies the operation for the user. An example of this is having to compile software where multiple conflicting libraries can cause linkage errors. Instead, each project can be ran in its own chroot jail, with the nessisary libraries present and thus isolating the conflicting libraries from each other.

### B. FreeBSD Jail

FreeBSD jail operates in a similar way to a Type 2 Hypervisor, where an underlying Host OS instance of FreeBSD is used for communication to the hardware.
Each jail is partitioned into its own mini-system where applications can operate. Unlike a Virtual machine, a FreeBSD jail shares the FreeBSD OS and kernel with all other jails. Therefore, there is no Virtual Machine that is being ran, so the overhead of the FreeBSD is vastly minimised.

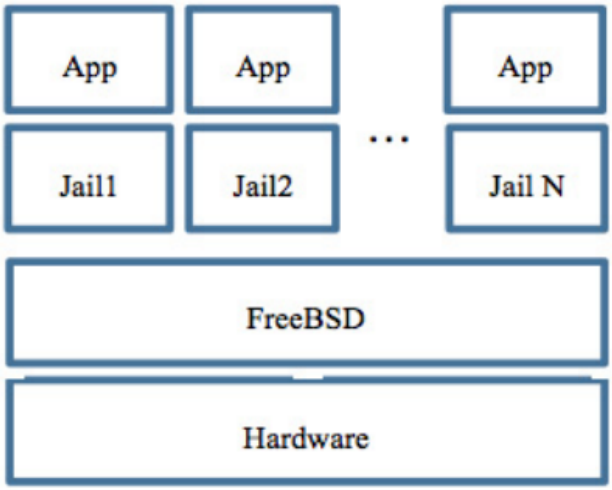



Fig. 3. An example of the structure of a FreeBSD jail. Note that each jail shares the same OS and kernel [2]

Each jail is a sand-boxed virtual environment and has its own files, processes, IP-addresses, and user/superuser accounts. Processes that are ran on one jail are not allowed to interact with those of another jail so that each jail is effectively sealed from one another. Each jail can run different software (even the same software of differing versions and configurations) as decided upon by the jail owner.
Limitiation are introduced in Free-BSD jail enviroment as each jail is bound to the kernel of the base system. It is simply not possible for a jail needs to run a different kernel or different kernel configuration, which may be needed for optimal application performance. Secondly, this host kernel is a single-point of failure: if some bug or exploit is used to crash

the base kernel, then all services in all jails will also cease to function.
Thus Free-BSD jails can be used to compartmentalised sensitive and vulnerable services from each other. Given that a central-server is used to support multiple services; in this example a FTP service and a Web server are running from the same machine. By placing the separate services in differing jails, if the Web server is compromised and the attacker is able to browse through the directory tree, they will never be able to see the FTP service information.

### *C. Jail-Like tools offered by other OS*

There are many alternative applications on other operating systems that can offer process isolation. For windows, software such as Sandboxie can perform application isolation. Linux Containers allow multiple running isolated Linux systems on a single Linux kernel, and MacOS has applications such as Docker.

## REFERENCES


[1] https://www.altaro.com/hyper-v/hyper-v-terminology-host-operating-system-or-parent-partition/
[2] Performance of Jails versus Virtualization for Cloud Computing Solutions
[3] http://www.cs.yale.edu/homes/aspnes/pinewiki/Virtualization.html
[4] https://corensic.wordpress.com/2011/12/12/virtual-machines-the-traps/

# 21 Practical - Lab 8: TCPDump (C)

Lab task - TCPDump (Credit)

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 8: TCPDump (C)

*Submitted By:*
Michael Reeb
4936094
2019/04/17 22:51

*Tutor:*
Jonathan Kua

April 17, 2019

# TNE30019-Unix For Telecommunications
# Lab 8 - TCPDump

Michael Reeb - 4936094

April 17, 2019

## 1 Results

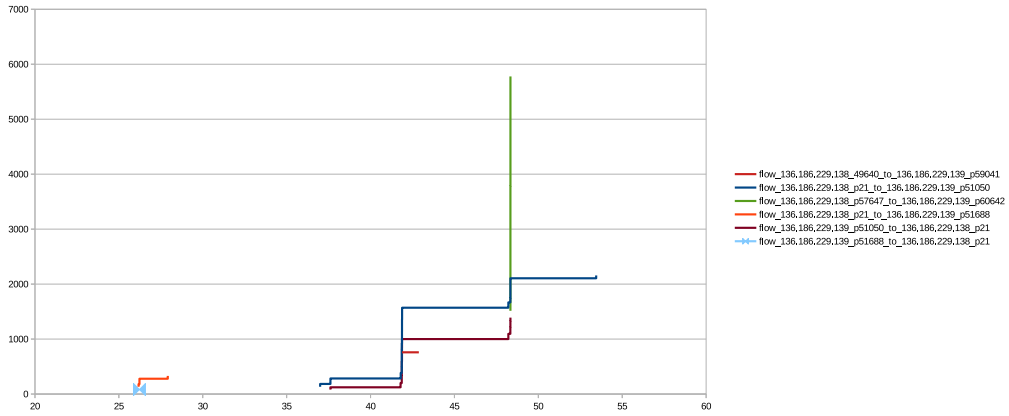



Figure 1: File Transfer Protocol (FTP) flows.

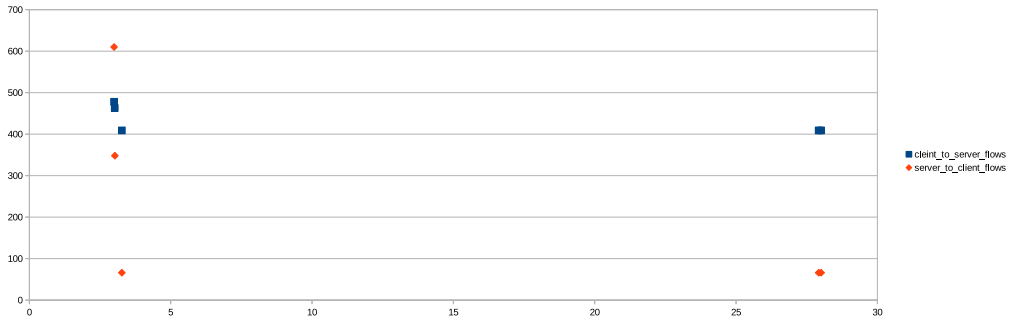



Figure 2: HTTP Flows. Note that each point is a separate flow. Flows from the Client-to-Server are blue, and flows from Server-to-Client are in orange.

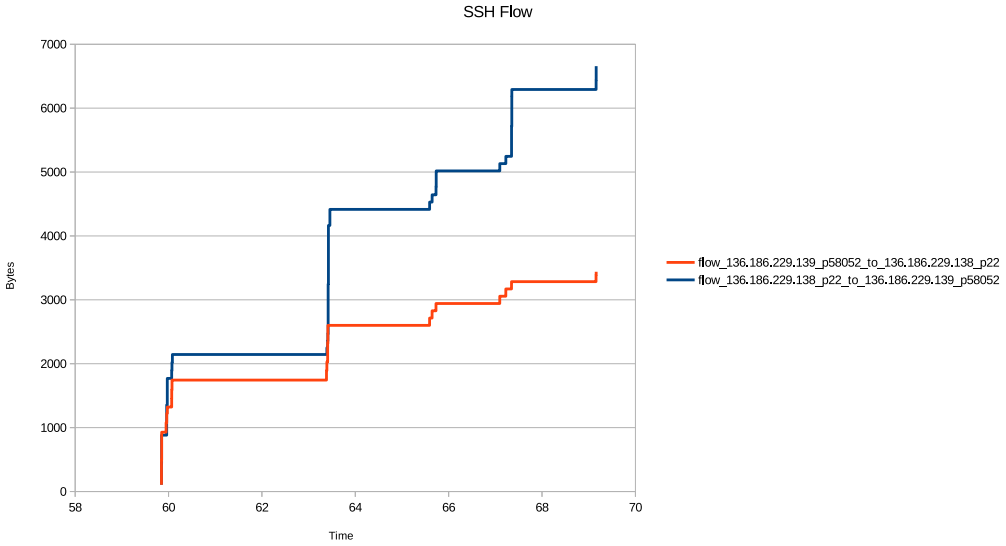



Figure 3: Secure Shell (SSH) flows.

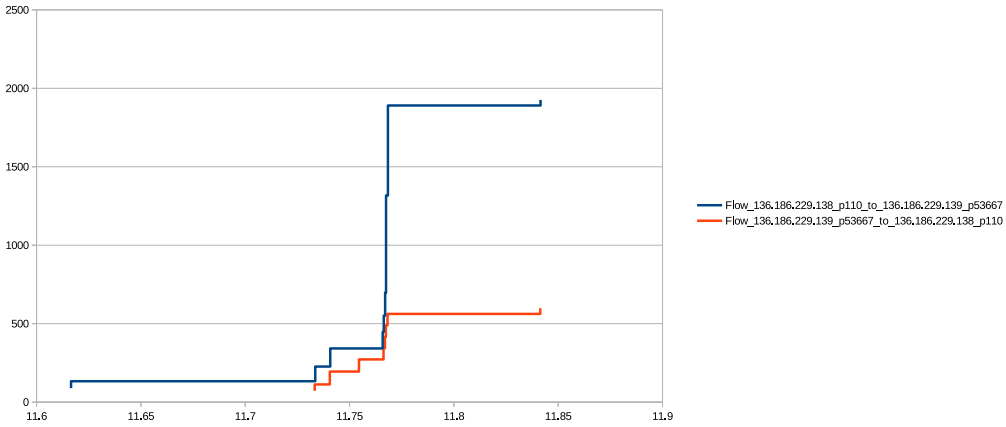



Figure 4: Post Office Protocol (POP) flows.

# 22 Practical - Lab 8: TCPDump (D)

Lab task - TCPDump (Distinction)

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 8: TCPDump (D)

*Submitted By:*
Michael Reeb
4936094
2019/04/17 23:58

*Tutor:*
Jonathan Kua

April 18, 2019

# TNE30019-Unix For Telecommunications
# Lab 8 - TCPDump

Michael Reeb - 4936094

April 17, 2019

## Contents



---

## 1 Results

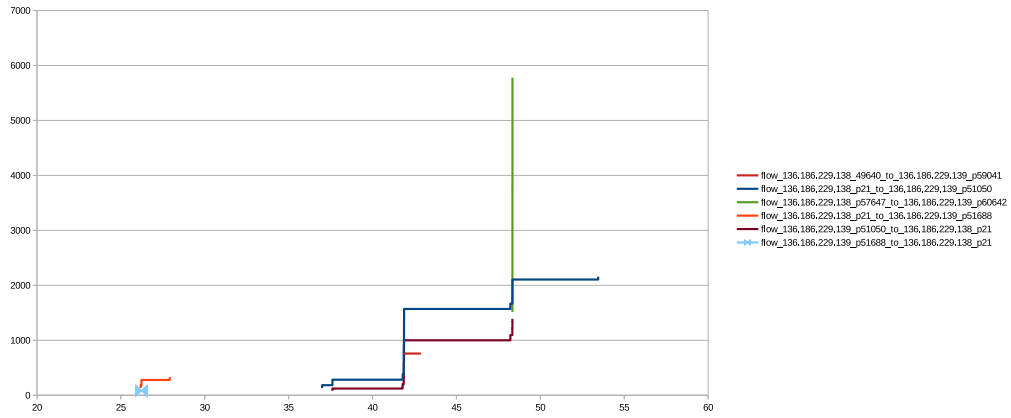



Figure 1: File Transfer Protocol (FTP) flows.

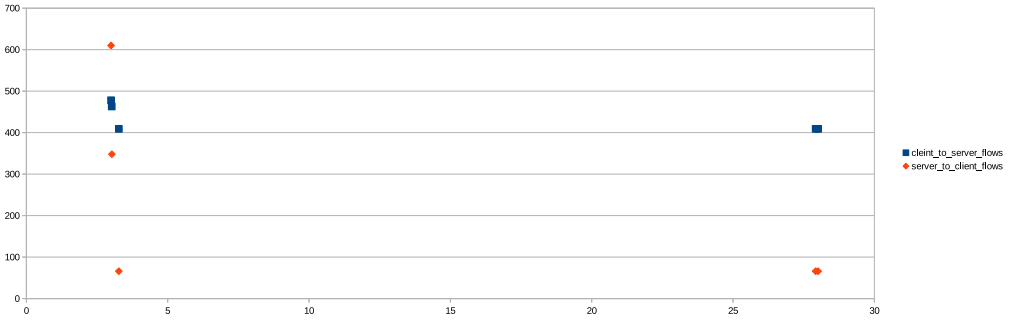



Figure 2: HTTP Flows. Note that each point is a separate flow. Flows from the Client-to-Server are blue, and flows from Server-to-Client are in orange.

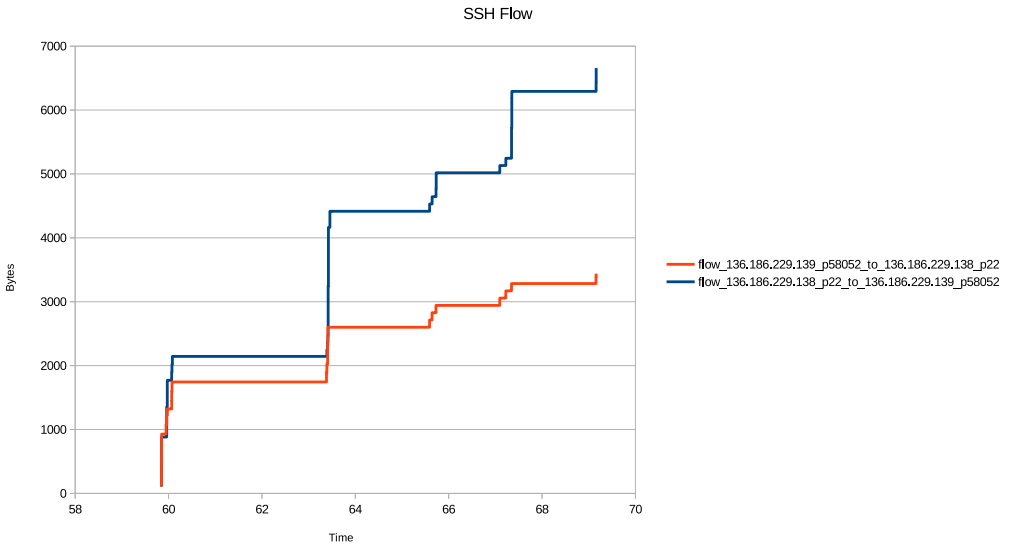



Figure 3: Secure Shell (SSH) flows.

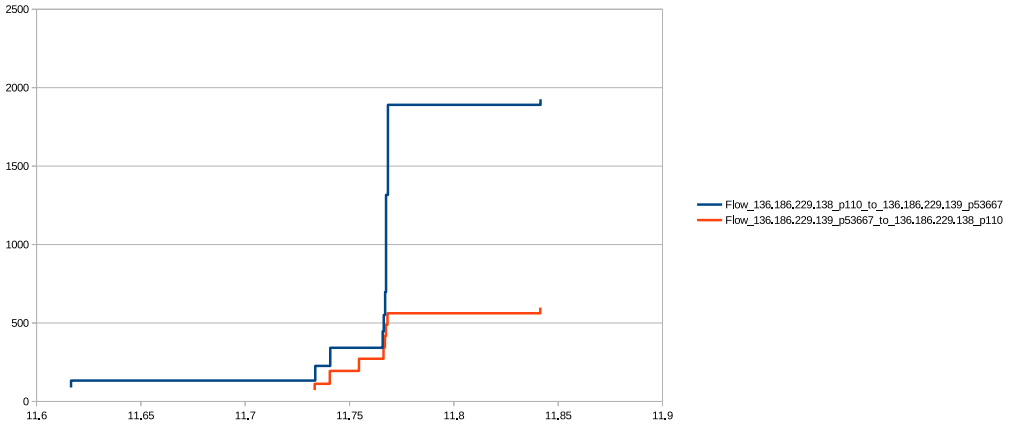



Figure 4: Post Office Protocol (POP) flows.

# 2 Discussion

In this lab, the Server was located on IP `136.186.229.138`, while the client was located on IP `136.186.229.139`

## 2.1 FTP

In Figure 1, it can seen that there are six distinct flows, 2 of which are from the client device to the server, and four are from the server to the client. The first "burst" of communication was the failed login attempt using `anonymous` as username, and the servers response to the failed login, and when the client abruptly disconnected. In the second burst, the client successfully logged into the FTP server. Here we see that the server has two flows to the client, which is the way FTP operates. FTP sends commands over one connection, and the data over the second connection.
This transmission is bursty, as we can see large peaks of data transfer, followed by periods of inactivity.

## 2.2 HTTP

As seen in Figure 2 there are many numerous flows of traffic. This is because modern web browsers operate using parallel HTTP connects so web pages are able to be downloaded faster. This can be seen around the first burst of activity, where multiple connections are used to download the web-page. The flows that occur later on have to do with the browser making requests attempting to download the web-page's `favicon.ico`; the little icon that appears next a web-pages name in the tab-bar on modern browsers.
Again, this traffic is bursty as there are periods of high traffic, and then periods of no traffic.

## 2.3 SSH

In Figure 3 we can see that SSH has two distinct flows: one from the client to server, the other from server to client. we can see that the flows are steady, until there is a large amount

of data transferred from the server to the client, before remaining steady again.This may indicate that a file had been downloaded.
While not as severe as the previous protocols, these SSH flows are still bursty, particularly in the flow from the server to the client.

### 2.4 POP

In Figure 4 we can see that POP protocol has two distinct flows: one from the client to the server, and the other from the server to the client. The initial traffic is due to the login and verification process ( we can see the password is *mypassword*, and this is followed by a large burst in the Server→Client caused by the Client asking to retrieve an email which was then subsequently served.
Again, this traffic is bursty, as can been seen by the large data transmission caused by the email retrieval.

## 3 Conclusion

Bursty traffic can cause negative user experience in networks with limited bandwidth, such as the Last Mile between ISPs and home networks. There is no simple solution to bursty traffic without increasing bandwidth, which is costly endeavour for ISP.
Due to the nature of operation TCP congestion control which attempts to probe the network for maximum throughput , traffic bursts will be an inevitable part of internet usage. Burst Traffic can be minimised by the use of buffers, which attempt to absorb the burst traffic. However, this can introduce frustrating RTT times into a network as buffered need to be cleared, which can particularly effect Real Time services.

# 23 Practical - Lab 8: TCPDump (P)

Lab task - TCPDump

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

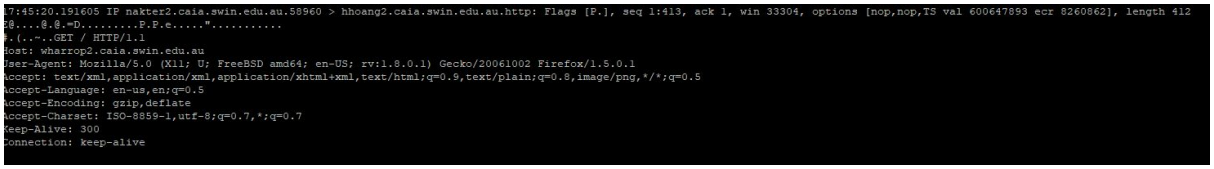
# Practical - Lab 8: TCPDump (P)

*Submitted By:*
Michael Reeb
4936094
2019/04/12 23:36

*Tutor:*
Jonathan Kua

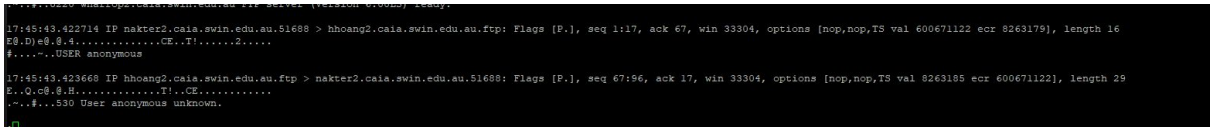
April 12, 2019

1) Over what timeframe was the trace captured? From 17:45:17.195008 until 17:46:26 (approx 1 min)
2) First was http request, then pop3 (email) request, then FTP transfer, then SSH
3) wharrop2.caia.swin.edu.au virtual host thttpd located on the Hhoang2.caia.swin.edu.au server (Hhoang2.caia.swin.edu.au). This site is the Armitage Chemicals website from Lab 5.



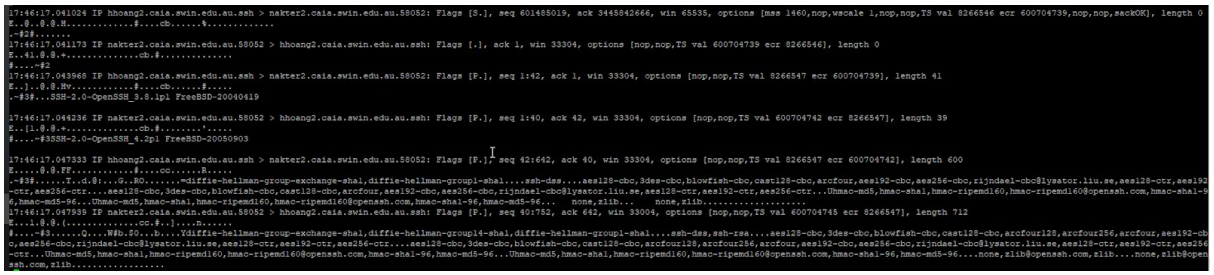


4)He typed the wrong username, user anonymous does not exist




5)Laurence retrieve a copy (RETR) of the blkmarble-desktopImage.jpg image


6) It is not possible to see the contents of the SSH session as they are encrypted. However we were able to see SSH perform the public-key exchange part of the Diffie-Hellman key exchange.

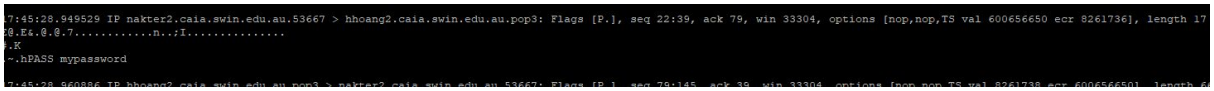



7)the contents of the email was:

```
17:45:28.963395 IP hhoang2.caia.swin.edu.au.pop3 > nakter2.caia.swin.edu.au.53667: Flags
E..'.[@.@.D.........n........;t...........
.~.k#.K.+OK sending message ending with a '.' on a line by itself
From wharrop@wharrop2.caia.swin.edu.au Tue Oct  3 17:08:30 2006
Return-Path: <wharrop@wharrop2.caia.swin.edu.au>
Received: from wharrop2.caia.swin.edu.au (localhost.caia.swin.edu.au [127.0.0.1])
        by wharrop2.caia.swin.edu.au (8.13.3/8.13.3) with ESMTP id k9378UJ7005779
        for <lslowman@wharrop2.caia.swin.edu.au>; Tue, 3 Oct 2006 17:08:30 +1000 (EST)
        (envelope-from wharrop@wharrop2.caia.swin.edu.au)
Received: (from wharrop@localhost)
        by wharrop2.caia.swin.edu.au (8.13.3/8.13.3/Submit) id k9378Ums005778
        for lslowman; Tue, 3 Oct 2006 17:08:30 +1000 (EST)
        (envelope-from wharrop)
Date: Tue, 3 Oct 2006 17:08:30 +1000 (EST)
From: Warren Harrop <wharrop@wharrop2.caia.swin.edu.au>
Message-Id: <200610030708.k9378Ums005778@wharrop2.caia.swin.edu.au>
To: lslowman@wharrop2.caia.swin.edu.au
Subject: Labs?

Lawrence Slowman, you have yet to hand in any labs. Is there any reason for this? Are
you still enrolled?

Warren Harrop

.
```

It was sent by wharrop@wharrop2.caia.swin.edu.au (Warren Harrop) a 17:08:30

8) Laurences' password is: mypassword



# 24 Communications - Speaker Reflection 1

Reflection of presentation of first Industry Invited Speaker

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦♦◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Communications - Speaker Reflection 1

*Submitted By:*
Michael Reeb
4936094
2019/05/20 15:39

*Tutor:*
Jonathan Kua

May 20, 2019

# Unix for Telecommunications

Portfolio Task – C-Speaker-Reflection-Form

Name: Michael Keeb Date: 20/5/2019

1) Reflect on one thing you learnt today that you were completely surprised by

Job turn around. industry has no loyalty? or do employees have no loyalty?

2) How do you think todays presentation changed the way you look at your career following graduation?

Going to focus on big data, AI/machine learning as a ~~career~~ career path wrt software + comp sci

P

# 25 Communications - Lab Report 3

New Description

| Outcome | Weight |
| --- | --- |
| ULO7 | ♦♦♦♦◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Communications - Lab Report 3

*Submitted By:*
Michael Reeb
4936094
2019/05/02 22:58

*Tutor:*
Jonathan Kua

May 2, 2019

# TNE30019 Unix for Telecommunications
# Lab 9 - Samba

Michael Reeb (ID: 4936094)
Swinburne University of Technology
Melbourne, Australia
4936094@student.swin.edu.au

***Abstract*—The software package Samba allows for the creation of network directories, or shares. This allows for cross-platform file sharing (ie from Unix to Windows), and the creation of public share drives.**

## I. Introduction to PCAP

Samba is a software package that implements the SMB protocol, and can be used to enable file and print sharing across multiple Operating Systems (OS) who implement different file structures. A Samba ”share” is simply a directory on Unix that is offered by Samba via the SMB protocol to a connecting client to use a regular directory.

## II. Aim

In this lab, Samba was used to create networked drives that could be remotely accessed by a Windows OS from Windows File Explorer using the Network Drive wizard. These shares were private, so only the account owner would be able to access them. Any files transferred or modified on this drive would transfer to the Unix local host drive of the account.
In the second part of the lab, a public shared drive was implemented so all accounts were able create, and modify files located within this share.

## III. Equipment

RULE host 61 was used as the server. SSH access was though PuTTY running on an external PC connected to Swinburne via VPN. Samba version 4.8.2 was used to implement the service.

## IV. Methodology

As per lab handout.

## V. Discussion

### A. Basic Samba Configuration

Initially, Samba was configured in the `rc.conf` script so that it would run on startup, and so that the `service` commands would operate. This was performed by appending `samba_server_enable="YES"` to the `rc.conf` file located in `/etc` directory. The `[global]` section of the configuration file determines the server-wide settings, as well as the options that will apply to all shares. It was under this section that the work-group was assigned (using the `workgroup=MSHOME` option), server string was set (using `serverstring=rule61.caia.swin.edu.au`), and to ensure that authentication is used with encrypted passwords, both the options `security=user` and `encryptpasswords=yes` were activated. To ensure that only clients from the Swinburne were able to connect, the option `hostsallow136.186.*.*` was implemented, and `*` is the wild card character meaning that any IP in the `136.186.0.0` domain is allowed to connect.
Finally, the `ntlmauth=ntlmv1-permitted` was enable under the global settings to ensure that NTMLv1 and above is used for all clients.

### B. Create Users

Create new accounts on the Rule host is accomplished using the FreeBSD `adduser` command ran from the shell. Two accounts were created, one called **samba** with the password *samba*, the second username was **autocollector** with the password being *autocollector*.
Next, a Samba password had to be created for each of the above accounts. This was done using the `smbpasswd-aUSERNAME` shell command, where USERNAME the account name (samba and autocollector) that was created above.
To ensure these accounts were correctly created, the

shell command `pdbedit-L` was used. This command generates a list of the users that are located in the Samba database.

### C. Create Shares

In the `smb4.conf` file, the section denoted `[homes]` is used to configure services for connecting clients to their home directories. These are created at run-time by the Samba server, by using the Samba variable substitutions **%u** which replaces the variable with the current Samba session username.
Thus, the home shares were created with the following configuration option: `path=/home/%u` which sets the Share path depending on the user connecting (ie for the account autocollector, the Samba server will change this option to: `path=/home/autocollector` ).
The `readonly=no` allows the creation and modification of files on the share. Finally, the `browseable=no` option hides the
path[homes] share from being able to be accessed, but will allow the user to autoconnect to their directory as explained above. If this option is not set, then any user will be able to see all the shares of the all accounts in the `homes` share, albeit a user will still only be able to access their own share.
Funcionality was then tested by logging in to each account on Windows, creating a file, and then checking the accounts home directory on Rule 61 to ensure that the file is there too.

### D. Creation of the public share

A public share can be created in the `smb4.conf` so all users can access. This was created in the `[public]` section. As given in the lab specification the location shared folder was set with the `path=/home/samba/pubStuff`. Again the `readonly=no` option was used to files can be created and modified. The `browsable=yes` option is used to ensure that the share is seen in the list of availible shares by the client.
The `guestok=yes` ensures that no password is needed for connecting to this service.
The `createmask=777` determines the permission files created in this share. This value is bit-wise ”And” with created files permissions. The `forcecreatemode=777` also is used to dermine the permissions and is bite-wise ORed onto the file. `777` means that any user of any group can read, write, and exectute these files created in this share.
Finally, to have this share work correctly, either the share directory permission in Unix must be changed (by using the command `chmod777_-R/home/samba/pubStuff` which will also apply this to any directories), or by using the `forceuser=samba` option is the `[public]` share which will change any account accessing this share to use the share as if they were the user *samba*

## VI. Conclusion

Samba is a software package that can be used to create shared network drives across multiple OS platforms. In this lab it was demonstrated that Samba can be used to generate private shares at run time on a per account basis, and that a public share directory can be setup that allows any user to create, edit and modify files.

## References

[1] Configuration Manual for smb.conf. (n.d.). Retrieved from https://www.samba.org/samba/docs/4.7/man-html/smb.conf.5.html

## VII. Results

Listing 1. smb4.conf

```
[global]
workgroup = MSHOME
server string = rule61.caia.swin.edu.au
security = user
encrypt passwords = yes
hosts allow 136.186.*.*
ntlm auth = ntlmv1-permitted

[home]
comment = %u  Home Directory
path = /home/%u
read only = no
#valid users = %u
#---------------------add this line to prevent the "homes" from appearing
browseable = no

[public]
comment = Public Shared Directory
path = /home/samba/pubStuff
read only = no
guest ok = yes
browseable = yes
create mask = 777
#
#either need to force user as below, or change permissions of public folder to
#read/write/executable access (chmod 777)
;force user = samba
force create mode = 777
```

# 26 Practical - Lab 9: Samba (D)

Lab task - Samba (Distinction)

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦♦♦◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 9: Samba (D)

*Submitted By:*
Michael Reeb
4936094
2019/04/29 22:37

*Tutor:*
Jonathan Kua

April 29, 2019

```
Marking P-Lab-09-Samba-D on RULE host rule61
TNE30019/TNE80014 - Laboratory - Samba
=====================================
********************************************************************************
 Portfolio Task: P-Lab-09-Samba, Distinction Task

 Configure your rule host (rule61) with Samba running a public share

 Tasks:
  Two Samba users: samba(password=samba) and autocollector(password=autocollector)
  Samba share available to both users: //rule61/public
  Share maps to /home/samba/pubStuff
  Read/write access to both users
  Both users can edit/delete each others files
********************************************************************************

Date: 20190429_2236
Last 5 Logons:
student    pts/2    vpn247-162.cc.swin.edu Mon Apr 29 21:52   still logged in
student    pts/7    atc330-20.ict.swin.edu Mon Apr 29 15:32 - 16:24  (00:52)
student    pts/5    atc330-20.ict.swin.edu Mon Apr 29 14:31 - 16:23  (01:51)
student    pts/7    vpn247-53.cc.swin.edu. Sat Apr 27 15:29 - 18:39  (03:09)
student    pts/7    vpn247-53.cc.swin.edu. Sat Apr 27 15:23 - 15:25  (00:01)

RULE host Socket information
root     smbd       45415 35 tcp4   136.186.230.61:445    *:*
root     smbd       45415 36 tcp4   136.186.230.61:139    *:*
root     nmbd       45411 16 udp4   136.186.230.61:137    *:*
root     nmbd       45411 17 udp4   136.186.230.61:138    *:*
root     nmbd       45411 18 udp4   136.186.230.61:137    *:*
root     nmbd       45411 19 udp4   136.186.230.61:137    *:*
root     nmbd       45411 20 udp4   136.186.230.61:138    *:*
root     nmbd       45411 21 udp4   136.186.230.61:138    *:*
www      httpd      37620 3  tcp4   136.186.230.61:80     *:*
www      httpd      37617 3  tcp4   136.186.230.61:80     *:*
tempuser sshd       35750 3  tcp4   136.186.230.61:22     136.186.247.162:49949
tempuser sshd       35750 7  tcp4   136.186.230.61:6010   *:*
root     sshd       35741 3  tcp4   136.186.230.61:22     136.186.247.162:49949
root     cupsd      73173 5  tcp4   136.186.230.61:631    *:*
www      httpd      59579 3  tcp4   136.186.230.61:80     *:*
www      httpd      59578 3  tcp4   136.186.230.61:80     *:*
www      httpd      59577 3  tcp4   136.186.230.61:80     *:*
www      httpd      59575 3  tcp4   136.186.230.61:80     *:*
www      httpd      59574 3  tcp4   136.186.230.61:80     *:*
www      httpd      59573 3  tcp4   136.186.230.61:80     *:*
www      httpd      59572 3  tcp4   136.186.230.61:80     *:*
www      httpd      59571 3  tcp4   136.186.230.61:80     *:*
root     httpd      52413 3  tcp4   136.186.230.61:80     *:*
bind     named      41933 20 tcp4   136.186.230.61:53     *:*
bind     named      41933 21 tcp4   136.186.230.61:953    *:*
bind     named      41933 512 udp4  136.186.230.61:53     *:*
bind     named      41933 513 udp4  136.186.230.61:53     *:*
bind     named      41933 514 udp4  136.186.230.61:53     *:*
bind     named      41933 515 udp4  136.186.230.61:53     *:*
bind     named      41933 516 udp4  136.186.230.61:53     *:*
bind     named      41933 517 udp4  136.186.230.61:53     *:*
bind     named      41933 518 udp4  136.186.230.61:53     *:*
bind     named      41933 519 udp4  136.186.230.61:53     *:*
bind     named      41933 520 udp4  136.186.230.61:53     *:*
bind     named      41933 521 udp4  136.186.230.61:53     *:*
```

```
bind     named      41933 522 udp4  136.186.230.61:53     *:*
bind     named      41933 523 udp4  136.186.230.61:53     *:*
bind     named      41933 524 udp4  136.186.230.61:53     *:*
bind     named      41933 525 udp4  136.186.230.61:53     *:*
bind     named      41933 526 udp4  136.186.230.61:53     *:*
bind     named      41933 527 udp4  136.186.230.61:53     *:*
bind     named      41933 528 udp4  136.186.230.61:53     *:*
bind     named      41933 529 udp4  136.186.230.61:53     *:*
bind     named      41933 530 udp4  136.186.230.61:53     *:*
bind     named      41933 531 udp4  136.186.230.61:53     *:*
bind     named      41933 532 udp4  136.186.230.61:53     *:*
bind     named      41933 533 udp4  136.186.230.61:53     *:*
bind     named      41933 534 udp4  136.186.230.61:53     *:*
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*

TEMPORARILY EDITING nsmb.conf
============================
Setting access to RULE61
  Username: SAMBA
  Password: samba

RUNNING SAMBA ACCESS TEST
=========================

Checking access to //samba@RULE61/public...

Mounting (//samba@RULE61/public) to (/var/tmp/rule/collect/mnt)...

Checking Mount Status...
//SAMBA@RULE61/PUBLIC on /var/tmp/rule/collect/mnt (smbfs)

Creating file (auto_collect_test)...

Checking file access (read)...
=< contents >==============================================
[RULE61]
addr=rule61

[RULE61:SAMBA]
password=samba
============================================================

Checking file properties on actual RULE Host...
-rwxrwxrwx  1 1002  1002  52 Apr 29 22:36 /usr/jails/unix/lab/rule61/usr/home/samba/pubStuff/auto_co
llect_test

Checking contents of written file...

Unmounting (/var/tmp/rule/collect/mnt)


Testing summary:
  Mounting share:           Pass
  File creation:            Pass
  Accessing mounted file:   Pass
  File contents correct:    Pass
  File location on host:    Pass
  File Owner permissions:   Pass
  File Group permissions:   Pass
```

```
  File Public permissions:  Pass

REINSTATING OLD nsmb.conf
=========================

TEMPORARILY EDITING nsmb.conf
=============================
Setting access to RULE61
  Username: AUTOCOLLECTOR
  Password: autocollector

RUNNING SAMBA ACCESS TEST
=========================

Checking access to //autocollector@RULE61/public...

Mounting (//autocollector@RULE61/public) to (/var/tmp/rule/collect/mnt)...

Checking Mount Status...
//AUTOCOLLECTOR@RULE61/PUBLIC on /var/tmp/rule/collect/mnt (smbfs)

Creating file (auto_collect_test)...

Checking file access (read)...
=< contents >==============================================
[RULE61]
addr=rule61

[RULE61:AUTOCOLLECTOR]
password=autocollector
============================================================

Checking file properties on actual RULE Host...
-rwxrwxrwx  1 1003  1002  68 Apr 29 22:36 /usr/jails/unix/lab/rule61/usr/home/samba/pubStuff/auto_co
llect_test

Checking contents of written file...

Unmounting (/var/tmp/rule/collect/mnt)


Testing summary:
  Mounting share:           Pass
  File creation:            Pass
  Accessing mounted file:   Pass
  File contents correct:    Pass
  File location on host:    Pass
  File Owner permissions:   Pass
  File Group permissions:   Pass
  File Public permissions:  Pass

REINSTATING OLD nsmb.conf
=========================

********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 27 Practical - Lab 9: Samba (P)

Lab task - Samba

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦♦♦◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 9: Samba (P)

*Submitted By:*
Michael Reeb
4936094
2019/04/29 15:29

*Tutor:*
Jonathan Kua

April 29, 2019

```
Marking P-Lab-09-Samba-P on RULE host rule61
TNE30019/TNE80014 - Laboratory - Samba
=====================================
********************************************************************************
 Portfolio Task: P-Lab-09-Samba, Pass Task

 Configure your rule host (rule61) with Samba running private home shares

 Tasks:
  Two Samba users: samba(password=samba) and autocollector(password=autocollector)
  Samba share available to both users: //rule61/home
  Each share maps to /home/&ltusername>
  Users access their own (respective) home directories only
********************************************************************************

Date: 20190429_1529
Last 5 Logons:
student    pts/5    atc330-20.ict.swin.edu Mon Apr 29 14:31   still logged in
student    pts/7    vpn247-53.cc.swin.edu. Sat Apr 27 15:29 - 18:39  (03:09)
student    pts/7    vpn247-53.cc.swin.edu. Sat Apr 27 15:23 - 15:25  (00:01)
student    pts/8    vpn247-190.cc.swin.edu Wed Apr 24 23:01 - 01:12  (02:10)
student    pts/1    vpn247-58.cc.swin.edu. Tue Apr 23 22:13 - 22:55  (00:41)

RULE host Socket information
root     smbd       61871 39 tcp4   136.186.230.61:445    136.186.15.216:53343
root     smbd       61866 35 tcp4   136.186.230.61:445    *:*
root     smbd       61866 36 tcp4   136.186.230.61:139    *:*
root     nmbd       61862 16 udp4   136.186.230.61:137    *:*
root     nmbd       61862 17 udp4   136.186.230.61:138    *:*
root     nmbd       61862 18 udp4   136.186.230.61:137    *:*
root     nmbd       61862 19 udp4   136.186.230.61:137    *:*
root     nmbd       61862 20 udp4   136.186.230.61:138    *:*
root     nmbd       61862 21 udp4   136.186.230.61:138    *:*
www      httpd      59579 3  tcp4   136.186.230.61:80     *:*
www      httpd      59578 3  tcp4   136.186.230.61:80     *:*
www      httpd      59577 3  tcp4   136.186.230.61:80     *:*
www      httpd      59576 3  tcp4   136.186.230.61:80     *:*
www      httpd      59575 3  tcp4   136.186.230.61:80     *:*
www      httpd      59574 3  tcp4   136.186.230.61:80     *:*
www      httpd      59573 3  tcp4   136.186.230.61:80     *:*
www      httpd      59572 3  tcp4   136.186.230.61:80     *:*
www      httpd      59571 3  tcp4   136.186.230.61:80     *:*
www      httpd      59570 3  tcp4   136.186.230.61:80     *:*
tempuser sshd       49718 3  tcp4   136.186.230.61:22     136.186.15.216:56518
root     sshd       49711 3  tcp4   136.186.230.61:22     136.186.15.216:56518
root     httpd      52413 3  tcp4   136.186.230.61:80     *:*
bind     named      41933 20 tcp4   136.186.230.61:53     *:*
bind     named      41933 21 tcp4   136.186.230.61:953    *:*
bind     named      41933 512 udp4  136.186.230.61:53     *:*
bind     named      41933 513 udp4  136.186.230.61:53     *:*
bind     named      41933 514 udp4  136.186.230.61:53     *:*
bind     named      41933 515 udp4  136.186.230.61:53     *:*
bind     named      41933 516 udp4  136.186.230.61:53     *:*
bind     named      41933 517 udp4  136.186.230.61:53     *:*
bind     named      41933 518 udp4  136.186.230.61:53     *:*
bind     named      41933 519 udp4  136.186.230.61:53     *:*
bind     named      41933 520 udp4  136.186.230.61:53     *:*
bind     named      41933 521 udp4  136.186.230.61:53     *:*
bind     named      41933 522 udp4  136.186.230.61:53     *:*
bind     named      41933 523 udp4  136.186.230.61:53     *:*
```

```
bind     named      41933 524 udp4  136.186.230.61:53     *:*
bind     named      41933 525 udp4  136.186.230.61:53     *:*
bind     named      41933 526 udp4  136.186.230.61:53     *:*
bind     named      41933 527 udp4  136.186.230.61:53     *:*
bind     named      41933 528 udp4  136.186.230.61:53     *:*
bind     named      41933 529 udp4  136.186.230.61:53     *:*
bind     named      41933 530 udp4  136.186.230.61:53     *:*
bind     named      41933 531 udp4  136.186.230.61:53     *:*
bind     named      41933 532 udp4  136.186.230.61:53     *:*
bind     named      41933 533 udp4  136.186.230.61:53     *:*
bind     named      41933 534 udp4  136.186.230.61:53     *:*
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*
?        ?          ?     ?  tcp4   136.186.230.61:57615  136.186.230.61:139
?        ?          ?     ?  tcp4   136.186.230.61:57664  136.186.230.61:139

TEMPORARILY EDITING nsmb.conf
=============================
Setting access to RULE61
  Username: SAMBA
  Password: samba

RUNNING SAMBA ACCESS TEST
=========================

Checking access to //samba@RULE61/home...

Mounting (//samba@RULE61/home) to (/var/tmp/rule/collect/mnt)...

Checking Mount Status...
//SAMBA@RULE61/HOME on /var/tmp/rule/collect/mnt (smbfs)

Creating file (auto_collect_test)...

Checking file access (read)...
=< contents >==============================================
[RULE61]
addr=rule61

[RULE61:SAMBA]
password=samba
============================================================

Checking file properties on actual RULE Host...
-rwxr--r--  1 1002  1002  52 Apr 29 15:29 /usr/jails/unix/lab/rule61/usr/home/samba/auto_collect_tes
t

Checking contents of written file...

Unmounting (/var/tmp/rule/collect/mnt)


Testing summary:
  Mounting share:           Pass
  File creation:            Pass
  Accessing mounted file:   Pass
  File contents correct:    Pass
  File location on host:    Pass
  File Owner permissions:   Pass
  File Group permissions:   Pass
```

```
  File Public permissions:  Pass

REINSTATING OLD nsmb.conf
=========================

TEMPORARILY EDITING nsmb.conf
=============================
Setting access to RULE61
  Username: AUTOCOLLECTOR
  Password: autocollector

RUNNING SAMBA ACCESS TEST
=========================

Checking access to //autocollector@RULE61/home...

Mounting (//autocollector@RULE61/home) to (/var/tmp/rule/collect/mnt)...

Checking Mount Status...
//AUTOCOLLECTOR@RULE61/HOME on /var/tmp/rule/collect/mnt (smbfs)

Creating file (auto_collect_test)...

Checking file access (read)...
=< contents >==============================================
[RULE61]
addr=rule61

[RULE61:AUTOCOLLECTOR]
password=autocollector
============================================================

Checking file properties on actual RULE Host...
-rwxr--r--  1 1003  1003  68 Apr 29 15:29 /usr/jails/unix/lab/rule61/usr/home/autocollector/auto_col
lect_test

Checking contents of written file...

Unmounting (/var/tmp/rule/collect/mnt)


Testing summary:
  Mounting share:           Pass
  File creation:            Pass
  Accessing mounted file:   Pass
  File contents correct:    Pass
  File location on host:    Pass
  File Owner permissions:   Pass
  File Group permissions:   Pass
  File Public permissions:  Pass

REINSTATING OLD nsmb.conf
=========================

********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 28 Communications - Project Presentation

New Description

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦♦♦◊ |

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Communications - Project Presentation

*Submitted By:*
Michael Reeb
4936094
2019/05/29 12:43

*Tutor:*
Jonathan Kua

May 29, 2019

# Issues with current system

- Human error in entries breaking the DNS
  - Long downtime to fix this
    - Can cause our other customers to also experience down time
- Customers must contact our representatives to perform DNS update
  - Annoying for customer
  - Annoying for our service representatives
- Have to manage reverse DNS separately
  - Human errors
  -

# New Implementation

- Removing the potential for human error with automated scripts
  - Script handles all request formatting and execution
- Allow customers to self-managed their resources
  - Implemented with a login system
  - Customers can only change their allocated domain
  - Customers managed with Admin tools.
  - Customers only need to contact Admin for special requests, ie creating of whole domains
- Automatically manage the reverse DNS
  - Remove human error
  - Ensure that domains are not “forgotten” to be added if they are within the managed domain.

# Where the project is at

- Proof of concept is complete. However, this software is not ready for web facing customer service
  - Working demonstration
  - Functional (albeit feature limited)
  - Web-interface & console interface
- To go forward to create commercial product needs complete rebuild to ensure robustness
  - Including all record types (AAAA, PTR).
  - More security – Web service must be secure
    - Encryption of databases
    - Authentication of users
  - Better menuing system (current one is not scalable)
  - Improved UX design for both Web Interface and Console interface
  - As many features implemented into the web interface as possible as this the easiest service for both parties to manage.
    - Customer security issue with using SSH ports and commands

# Risk Analysis – Likely hood x Impact

- Human error in entries breaking the DNS
  - Current system - 5 x 4 = 20
  - Proposed system – 1 x 4 = 4
- Customer dissatisfaction with our service
  - Current system – 3 X 4 = 12
  - Proposed system – 1 X 3 = 3
- Issue with managing reverse DNS
  - Current system 5 X4 =20
  - Proposed system – 1 x 3 = 3

# 29 Communications - Speaker Reflection 2

Reflection of presentation of second Industry Invited Speaker

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦♦◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Communications - Speaker Reflection 2

*Submitted By:*
Michael Reeb
4936094
2019/05/27 16:27

*Tutor:*
Jonathan Kua

May 27, 2019

# Unix for Telecommunications

Portfolio Task – C-Speaker-Reflection-Form

Name: Michael Reab Date: 27/5/19

1) Reflect on one thing you learnt today that you were completely surprised by

change in view regarding security; ie attacks are now assumed to happen; and that your systems is assumed to be going to be comprimised..

2) How do you think todays presentation changed the way you look at your career following graduation?

cyber kill chain is an interseting tool for security design.

"Non-Technical" Mangment → avoid whenever possible.

NIST cyber security frame work.

## 30 Theory - Test 2 (P)

In class test run during tutorial in week 11 of semester

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO6 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO2 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO5 | ♦♦◊◊◊ |

| Outcome | Weight |
|---|---|
| ULO1 | ♦♦◊◊◊ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Theory - Test 2 (P)

*Submitted By:*
Michael Reeb
4936094
2019/05/27 15:21

*Tutor:*
Jonathan Kua

May 27, 2019

# Unix for Telecommunications

Portfolio Task – T-Test-2
**Pass Level Task**

| | |
|---|---|
| **Student ID** | 4936094 |
| **Student Name** | Michael Lees |
| **Test Date** | Week 11 |
| **Time Allowed** | 30 minutes |

## Assessment – Staff Use Only

In order to pass the test you **MUST** score a minimum of **60%**

| Question: | 1 | 2 | 3 | 4 | Total |
|---|---|---|---|---|---|
| Points: | 4 | 10 | 7 | 8 | 29 |
| Score: | 4 | 7½ | 7 | 6 | 24½ |

P

# 31 Practical - Lab 10: CUPS (D)

Lab task - CUPS (Distinction)

| Outcome | Weight |
|---|---|
| ULO6 | ♦♦♦♦♦ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 10: CUPS (D)

*Submitted By:*
Michael Reeb
4936094
2019/05/16 22:38

*Tutor:*
Jonathan Kua

May 16, 2019

```
Marking P-Lab-10-CUPS-D on RULE host rule61
TNE30019/TNE80014 - Laboratory - CUPS
====================================
********************************************************************************
 Portfolio Task: P-Lab-10-CUPS, Distinction Task

 Configure your rule host (rule61) with CUPS with a printer called PDFPrinter

 Tasks:
  Server accepts print jobs to printer PDFPrinter from user student and root
  PDF files are created in correct locations AND with correct permissions
  PDF files are valid and contain correct information
********************************************************************************

Date: 20190516_2237
Last 5 Logons:
root       pts/20                       Thu May 16 16:43 - 16:43  (00:00)
root       pts/5                        Thu May 16 13:59 - 20:08  (06:09)
root       pts/2                        Thu May 16 13:52   still logged in
student    pts/17   atc330-17.ict.swin.edu Thu May 16 12:54 - 13:43  (00:49)
root       pts/16                       Tue May 14 13:42 - 19:45  (06:02)

RULE host Socket information
root     cupsd      48320 5  tcp4   136.186.230.61:631    *:*
www      httpd      66695 3  tcp4   136.186.230.61:80     *:*
root     smbd       45415 35 tcp4   136.186.230.61:445    *:*
root     smbd       45415 36 tcp4   136.186.230.61:139    *:*
root     nmbd       45411 16 udp4   136.186.230.61:137    *:*
root     nmbd       45411 17 udp4   136.186.230.61:138    *:*
root     nmbd       45411 18 udp4   136.186.230.61:137    *:*
root     nmbd       45411 19 udp4   136.186.230.61:137    *:*
root     nmbd       45411 20 udp4   136.186.230.61:138    *:*
root     nmbd       45411 21 udp4   136.186.230.61:138    *:*
www      httpd      37620 3  tcp4   136.186.230.61:80     *:*
www      httpd      59579 3  tcp4   136.186.230.61:80     *:*
www      httpd      59578 3  tcp4   136.186.230.61:80     *:*
www      httpd      59577 3  tcp4   136.186.230.61:80     *:*
www      httpd      59575 3  tcp4   136.186.230.61:80     *:*
www      httpd      59574 3  tcp4   136.186.230.61:80     *:*
www      httpd      59573 3  tcp4   136.186.230.61:80     *:*
www      httpd      59572 3  tcp4   136.186.230.61:80     *:*
www      httpd      59571 3  tcp4   136.186.230.61:80     *:*
root     httpd      52413 3  tcp4   136.186.230.61:80     *:*
bind     named      41933 20 tcp4   136.186.230.61:53     *:*
bind     named      41933 21 tcp4   136.186.230.61:953    *:*
bind     named      41933 512 udp4  136.186.230.61:53     *:*
bind     named      41933 513 udp4  136.186.230.61:53     *:*
bind     named      41933 514 udp4  136.186.230.61:53     *:*
bind     named      41933 515 udp4  136.186.230.61:53     *:*
bind     named      41933 516 udp4  136.186.230.61:53     *:*
bind     named      41933 517 udp4  136.186.230.61:53     *:*
bind     named      41933 518 udp4  136.186.230.61:53     *:*
bind     named      41933 519 udp4  136.186.230.61:53     *:*
bind     named      41933 520 udp4  136.186.230.61:53     *:*
bind     named      41933 521 udp4  136.186.230.61:53     *:*
bind     named      41933 522 udp4  136.186.230.61:53     *:*
bind     named      41933 523 udp4  136.186.230.61:53     *:*
bind     named      41933 524 udp4  136.186.230.61:53     *:*
bind     named      41933 525 udp4  136.186.230.61:53     *:*
bind     named      41933 526 udp4  136.186.230.61:53     *:*
```

```
bind     named      41933 527 udp4  136.186.230.61:53     *:*
bind     named      41933 528 udp4  136.186.230.61:53     *:*
bind     named      41933 529 udp4  136.186.230.61:53     *:*
bind     named      41933 530 udp4  136.186.230.61:53     *:*
bind     named      41933 531 udp4  136.186.230.61:53     *:*
bind     named      41933 532 udp4  136.186.230.61:53     *:*
bind     named      41933 533 udp4  136.186.230.61:53     *:*
bind     named      41933 534 udp4  136.186.230.61:53     *:*
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*

SENDING PRINT JOBS
==================
Removing existing output print jobs files on rule host for (student)...

Sending print job as (student)

Pausing to allow output file to be created...

Checking output directory (/usr/jails/unix/lab/rule61/usr/home/PDF/student), owner (1001:student), p
ermissions(rw???????)...
drwxr-x---  2 1001  0  3 May 16 22:38 student


Checking output file - expected location(/usr/jails/unix/lab/rule61/usr/home/PDF/student), owner(100
1:student), permissions(rw???????)...
total 1
-rw-r-----  1 1001  0  15157 May 16 22:38 cups_test.txt-16052019-22:38:00.pdf


Checking PDF...
  IF THE FOLLOWING IS EQUAL TO 'PDF%' (or similar) PLEASE LET ME KNOW
====================
 %-1
====================
------- No current checking for PDF validity

Checking PDF contents...
==</usr/jails/unix/lab/rule61/usr/home/PDF/student/cups_test.txt-16052019-22:38:00.pdf>==========
################################################################################
## TEST OUTPUT FILE FOR CUPS PRINT JOB ##
################################################################################

Hello world...

==============================

Testing summary:
  Print job accepted:               Pass
  Correct output directory created:  Pass
  Output file created:              Pass
  Output directory owner:           Pass
  Output directory permissions:     Pass
  Output file owner:                Pass
  Output file permissions:          Pass
  Output file is a valid PDF:       Pass
  Output file contents:             Pass

Removing existing output print jobs files on rule host for (root)...
```

```
Sending print job as (root)

Pausing to allow output file to be created...

Checking output directory (/usr/jails/unix/lab/rule61/usr/home/PDF/remroot), owner (0:root), permiss
ions(rw???????)...
drwxr-x---  2 0     0  3 May 16 22:38 remroot


Checking output file - expected location(/usr/jails/unix/lab/rule61/usr/home/PDF/remroot), owner(0:r
oot), permissions(rw???????)...
total 33
-rw-r-----  1 0  0  15157 May 16 22:38 cups_test.txt-16052019-22:38:13.pdf


Checking PDF...
  IF THE FOLLOWING IS EQUAL TO 'PDF%' (or similar) PLEASE LET ME KNOW
====================
 %-1
====================
------- No current checking for PDF validity

Checking PDF contents...
==</usr/jails/unix/lab/rule61/usr/home/PDF/remroot/cups_test.txt-16052019-22:38:13.pdf>==========
################################################################################
## TEST OUTPUT FILE FOR CUPS PRINT JOB ##
################################################################################

Hello world...

==============================

Testing summary:
  Print job accepted:                Pass
  Correct output directory created:  Pass
  Output file created:               Pass
  Output directory owner:            Pass
  Output directory permissions:      Pass
  Output file owner:                 Pass
  Output file permissions:           Pass
  Output file is a valid PDF:        Pass
  Output file contents:              Pass

********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 32 Practical - Lab 10: CUPS (P)

Lab task - CUPS

| Outcome | Weight |
|---|---|
| ULO6 | ♦♦♦♦♦ |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Lab 10: CUPS (P)

*Submitted By:*
Michael Reeb
4936094
2019/05/02 12:44

*Tutor:*
Jonathan Kua

May 2, 2019

```
Marking P-Lab-10-CUPS-P on RULE host rule61
TNE30019/TNE80014 - Laboratory - CUPS
====================================
********************************************************************************
 Portfolio Task: P-Lab-10-CUPS, Pass Task

 Configure your rule host (rule61) with CUPS with a printer called PDFPrinter

 Tasks:
  Configure a CUPS server on your RULE host
  Create a printer called PDFPrinter
  Configure this device to accept print jobs
********************************************************************************

Date: 20190502_1243
Last 5 Logons:
student    pts/8    atc330-19.ict.swin.edu Thu May  2 11:45   still logged in

/usr/jails/unix/lab/rule61/var/log/utx.log begins Thu May  2 11:45:00 AEST 2019

RULE host Socket information
root     cupsd      29213 5  tcp4   136.186.230.61:631    *:*
tempuser sshd       16660 3  tcp4   136.186.230.61:22     136.186.15.215:56311
root     sshd       16217 3  tcp4   136.186.230.61:22     136.186.15.215:56311
root     smbd       45415 35 tcp4   136.186.230.61:445    *:*
root     smbd       45415 36 tcp4   136.186.230.61:139    *:*
root     nmbd       45411 16 udp4   136.186.230.61:137    *:*
root     nmbd       45411 17 udp4   136.186.230.61:138    *:*
root     nmbd       45411 18 udp4   136.186.230.61:137    *:*
root     nmbd       45411 19 udp4   136.186.230.61:137    *:*
root     nmbd       45411 20 udp4   136.186.230.61:138    *:*
root     nmbd       45411 21 udp4   136.186.230.61:138    *:*
www      httpd      37620 3  tcp4   136.186.230.61:80     *:*
www      httpd      37617 3  tcp4   136.186.230.61:80     *:*
www      httpd      59579 3  tcp4   136.186.230.61:80     *:*
www      httpd      59578 3  tcp4   136.186.230.61:80     *:*
www      httpd      59577 3  tcp4   136.186.230.61:80     *:*
www      httpd      59575 3  tcp4   136.186.230.61:80     *:*
www      httpd      59574 3  tcp4   136.186.230.61:80     *:*
www      httpd      59573 3  tcp4   136.186.230.61:80     *:*
www      httpd      59572 3  tcp4   136.186.230.61:80     *:*
www      httpd      59571 3  tcp4   136.186.230.61:80     *:*
root     httpd      52413 3  tcp4   136.186.230.61:80     *:*
bind     named      41933 20 tcp4   136.186.230.61:53     *:*
bind     named      41933 21 tcp4   136.186.230.61:953    *:*
bind     named      41933 512 udp4  136.186.230.61:53     *:*
bind     named      41933 513 udp4  136.186.230.61:53     *:*
bind     named      41933 514 udp4  136.186.230.61:53     *:*
bind     named      41933 515 udp4  136.186.230.61:53     *:*
bind     named      41933 516 udp4  136.186.230.61:53     *:*
bind     named      41933 517 udp4  136.186.230.61:53     *:*
bind     named      41933 518 udp4  136.186.230.61:53     *:*
bind     named      41933 519 udp4  136.186.230.61:53     *:*
bind     named      41933 520 udp4  136.186.230.61:53     *:*
bind     named      41933 521 udp4  136.186.230.61:53     *:*
bind     named      41933 522 udp4  136.186.230.61:53     *:*
bind     named      41933 523 udp4  136.186.230.61:53     *:*
bind     named      41933 524 udp4  136.186.230.61:53     *:*
bind     named      41933 525 udp4  136.186.230.61:53     *:*
bind     named      41933 526 udp4  136.186.230.61:53     *:*
```

```
bind     named      41933 527 udp4  136.186.230.61:53     *:*
bind     named      41933 528 udp4  136.186.230.61:53     *:*
bind     named      41933 529 udp4  136.186.230.61:53     *:*
bind     named      41933 530 udp4  136.186.230.61:53     *:*
bind     named      41933 531 udp4  136.186.230.61:53     *:*
bind     named      41933 532 udp4  136.186.230.61:53     *:*
bind     named      41933 533 udp4  136.186.230.61:53     *:*
bind     named      41933 534 udp4  136.186.230.61:53     *:*
root     sshd       97762 3  tcp4   136.186.230.61:22     *:*
root     ntpd       1207  61 udp4   136.186.230.61:123    *:*

CHECKING PRINTER INSTALLATION/AVAILABILITY
==========================================

Listing installed printers on rule61...

PDFPrinter accepting requests since Thu May  2 12:26:13 2019

Testing...

Printer (PDFPrinter) is installed: Pass
Printer is accepting jobs:         Pass

********************************************************************************
PORTFOLIO TASK RESULT: PASSED
```

# 33 Practical - Project

project submission

| Outcome | Weight |
|---|---|
| ULO3 | ♦♦♦◊◊ |

| Outcome | Weight |
|---|---|
| ULO6 | ♦♦♦♦◊ |

| Outcome | Weight |
|---|---|
| ULO7 | ♦♦♦♦◊ |

| Outcome | Weight |
|---|---|
| ULO2 | ♦♦♦♦◊ |

| Outcome | Weight |
|---|---|
| ULO4 | ♦♦♦♦◊ |

| Outcome | Weight |
|---|---|
| ULO5 | ♦♦♦♦◊ |

| Outcome | Weight |
|---|---|
| ULO1 | ♦♦♦♦◊ |

| Date | Author | Comment |
|---|---|---|
| 2019/06/05 12:30 | Jonathan Kua | Hi Michael, maybe I missed something in your document, but I can't seem to find the username and password to login via the Web interface. |
| 2019/06/05 15:26 | Michael Reeb | Hi Jon, To launch the webserver, simply login to the webserver account. username is <webserver>; password (should) also be <webserver>. To connect to the webpage (I use Chrome). The webserver is located on: http://136.186.230.101:8000, ie port 8000 (due to prvilidge issues when using well known ports). I have just connected to the webserver via VPN and it seems to work fine.<br>Thanks, mike |
| 2019/06/05 15:34 | Michael Reeb | as for login into the web interface, the accounts you have created in the Admin tools should work to login here |
| 2019/06/05 15:43 | Jonathan Kua | Great, thanks Michael. |

Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

# Practical - Project

*Submitted By:*
Michael Reeb
4936094
2019/05/29 21:20

*Tutor:*
Jonathan Kua

May 29, 2019

SWINBURNE UNIVERSITY OF TECHNOLOGY

# TNE30019 - C-Project-Bind - Project Report

Michael REEB
4936094

May 29, 2019

# Contents

# 1 Aim

The aim of this project was to implement a fully operable DDNS server with easy to use User front-end, in both command line interface and a web interface. This would allow for Users to be self-sufficient with their DNS needs and thus would not have to rely upon Administrators to perform basic tasks.
The second part of this project was to implement a suite of Admin tools that would allow for: creating and deleting of User accounts, creating and removing DNS zones, and creating and removing reverse DNS zones. (Note that the reverse DNS control was only partially implemented and thus removed from the demonstration.

# 2 Setting Up and Running Project

This whole project was written in Python3.6, and used a number of libraries to implement the web interface, database, and IP ranges. Python3.6 was chosen throughout the project such that the scripts would be able to be directly called by the differing interfaces reducing the need for wrapper classes; and so that behaviour would be consistent thought out.

## 2.1 Dependencies

The dependencies for this project are listed below in "pip install" format for ease of replication for the future user. Note that this list may not be comprehensive and is upon the reader to recognise missing packages and install them. It is also assumed that Python3.6 is installed with the PIP package installer.
pip install flask-login
pip install flask-sqlalchemy
pip install flask-migrate
pip install flask-wtf
pip install python-dotenv
pip install flask
pip install ipaddress

## 2.2 List of freeBSD accounts and Passwords

The following is a list of freeBSD users, passwords, and a description of the account usage. The username/password is given in the form of: **username - password**.
**student - z2rx3gyp**: The student account is the account to log in to to access FreeBSD root privileges. Also has a working copy of all the files
**root - bkxplus1** Root can be accessed from If anything breaks, use this account to fix it
**ddns - ddns** - Login into this account will automatically run the User command line interface.
**admin - admin** - Login into this account will automatically run the Admin tools command line interface. This account can also be used to gain root privileges.
**webserver - webserver** - Login into this account will automatically run the Webserver in a background process that will persist after the terminal has been closed.

## 2.3 List of created files

A large amount of file was created for this project, so I will briefly explain the files I wrote and what they contain. Apart from the Named configuration files and zones files, and the login scripts, all files are found within the `/usr/local/share/webInterface/venv/microblog` directory.

### 2.3.1 Main python files

- `pythonDNS.py` - This contains all the nsupdate functions, reverse update, and basic querying . This is where checking domains, ip ranges, creating the nsupdate commands and sending them to the server. This also contains all the User console functions, and is responsible for User login, and command parsing.
- `pythonDNS_Admintools.py` - This contains all the Administration scripts - Creating zone files, updating Named.conf, adding Users to the database, adding Domains to the Database, etc. This script also contains all the Admin console functions, and is also able to call the User tools.
- `microblog.py` - The entry point for the web-server. The web-server was built using the Flask framework , and this file is where we can specify the port and ip address that the server will run on.
- `__init__.py` - Initialises the Flask web-server data.
- `config.py` - Contains the configuration info for the database Flask-Alchemy (which uses SQL-Lite)
- `forms.py` - Specifies the forms used in the dynamically generated HTML pages that is used in the User web interface. Allows the reading and writing of data
- `models.py` - Specifies the SQL tables and databases that will be used throughout the programs to query from.
- `templates.py` - Contains the information related to the operation of the html pages, including forms, redirections, database checking, user login etc. This file directly uses the HTML template pages located in the `./templates/` directory

### 2.3.2 Database files

- `app.db` - contains all the User login data. Passwords are hashed with sha256.
- `domainDb.db` - contains all the Domain data.
- `reverseDb.db` - contains the reverse DNS info (ip range)

#### 2.3.3 Other Files

- `./templates/` - contains all the HTML webpage templates that are then filled in by Python in `templates.py` before being served. They are basic and straightforward, and need no further explanation.

- Each user account has an script in their respective home directory that begins their service.
  A template file in `../named/dynamic` directory where the Dynamic Zones are located. This template is used to generate new Zones.

### 2.4 Running and troubleshooting the Web-server

To begin the web-server simply login to the **webserver** account, and the login script should automatically run an instance of this web-server.
the server can than be accessed on `136.186.230.101:8000`.
If this fails for whatever reason, exit to the shell (ctrl+c) then try manually running the shell script (`./startWebserver`)
If this script fails, then navigate and run `/usr/local/share/webInterface/venv/microblog/microblog.py`
Also check that there in no current running background web server (easiest I found was with sockstat -4) and kill if nessisary.
Operating the website is self explanatory. Please be mindful as this is not very well sanity checked, so it is not overly difficult to create requests and.or queries that can break or lockup the script.

### 2.5 Running and troubleshooting the User CLI interface

To begin the User command line script, simply login to the DDNS account, and then after a pause the Login prompt should begin. Here you will be able to enter valid User credentials. Once logged in, you can add and remove DNS entries from the respective domains by following the console prompts.
Likewise, if this fails try executing the login script `loginScript`. If the login script is broken, then run the program directly from `/usr/local/share/webInterface/venv/microblog/pythonDNS.py`. Please be mindful as this is not very well sanity checked, so it is not overly difficult to create requests and.or queries that can break or lockup the script.

### 2.6 Running and troubleshooting the Admin CLI interface

Like wise, to run the Admin tools, simply sshing to the Admin user should open the Admin-tools CLI. If that doe not work, execute the script `loginScript` in the gome directory. Finally, attempt to run the `/usr/local/share/webInterface/venv/microblog/pythonDNS_Admintools.py` script if nothing else works.
Also worth noting, be careful with these Admin tools as it is possible to break thing peculiarly. Currently there is no database referencing, so deleting an entry (say a Domain) will not update to other tables (say the User who contains a field of Domains that they are allowed to update to), and thus the User will be able to attempt to NSUpdate non existent

Zones.

# 3 Self Reflection

To continue on the is project would require a complete rewrite, as this project was built by tacking on more and more features which has caused a lot of clumsy practises.
Better and structured value checking should be implemented through this program, as often it is done in the lower underlying classes and functions, where values should be checked near to where they are read in from user input.
The database should be constructed in such way that the tables are properly join, as currently information is saved as a string which is parsed and queried in the other database table.
The reverse DNS lookup should be properly implemented, currently it was a huge hassle to attempt to implement, as through out the program the reverse IP range was statically defined when needed, in hind sight this was NOT an easy way of doing this.
The console and command parsing is clumsy, and in the next version a proper command parser and menuing system should be implemented.
The Web interface is rather plain, and could use some enhancements.
Big Brother is Watching You
All in all, I am happy with what I achieved. I entered this subject never having used a Unix os before, and never have done any BASH, Python, or HTML before. This subject has made me familiar with some vital knowledge that his filled a lot of holes I had regarding Operating Systems, networking, computer systems in general. Whilst a lot of work, I thoroughly enjoyed this Unit. Also printers suck