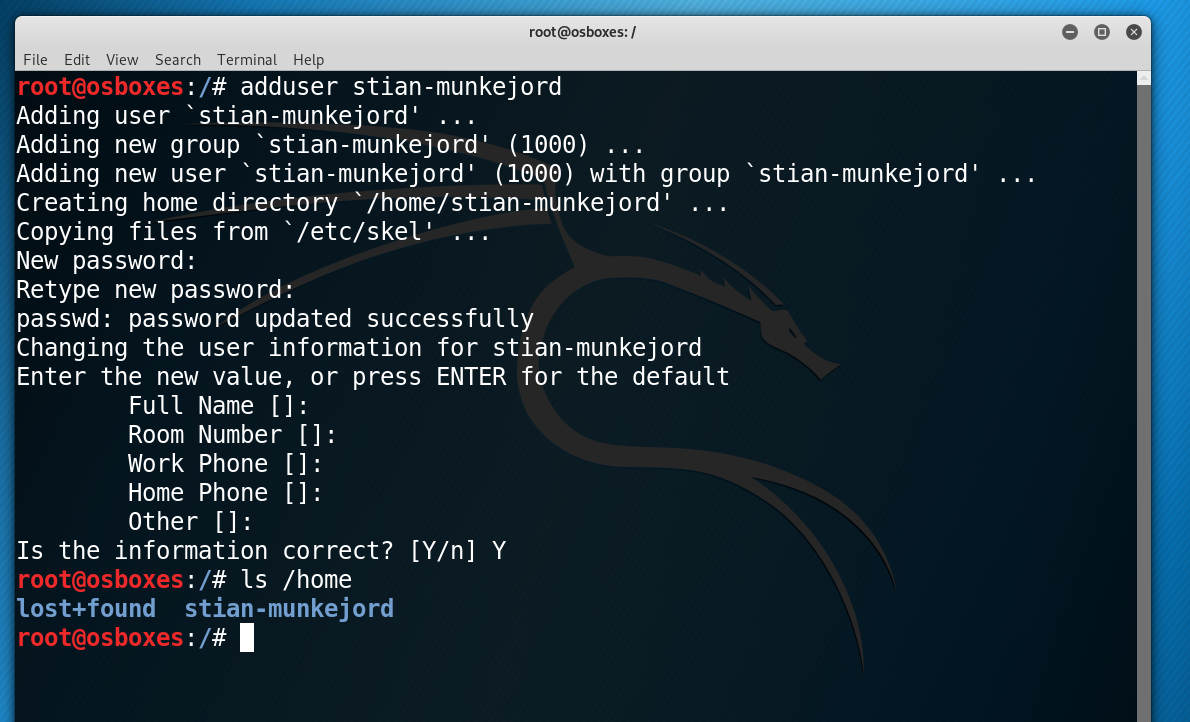
**Mandatory Assignment 2**

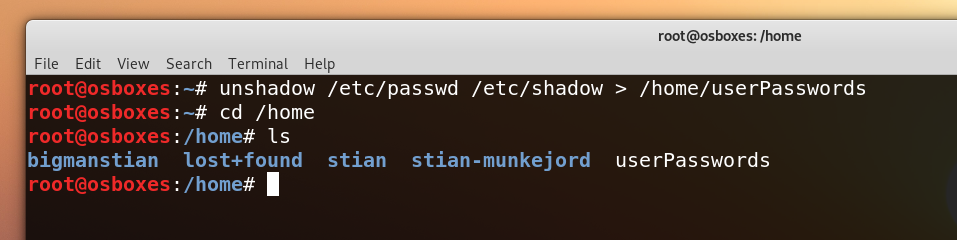
**User Authentication (15 pts)**

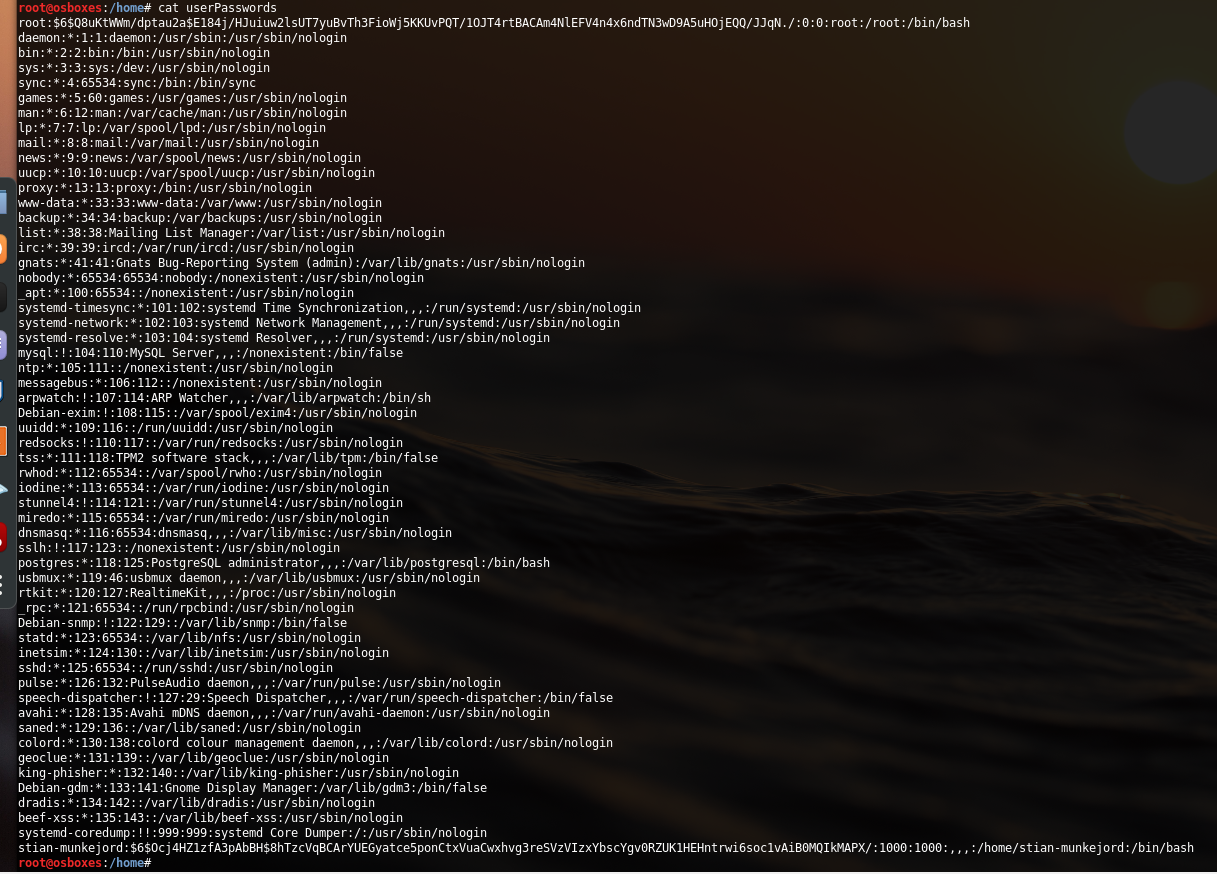
**Question 1.**

1. *Add yourself (firstname-lastname) as a new user with a weak password in Kali Linux (Take a screenshot of the home directory);*



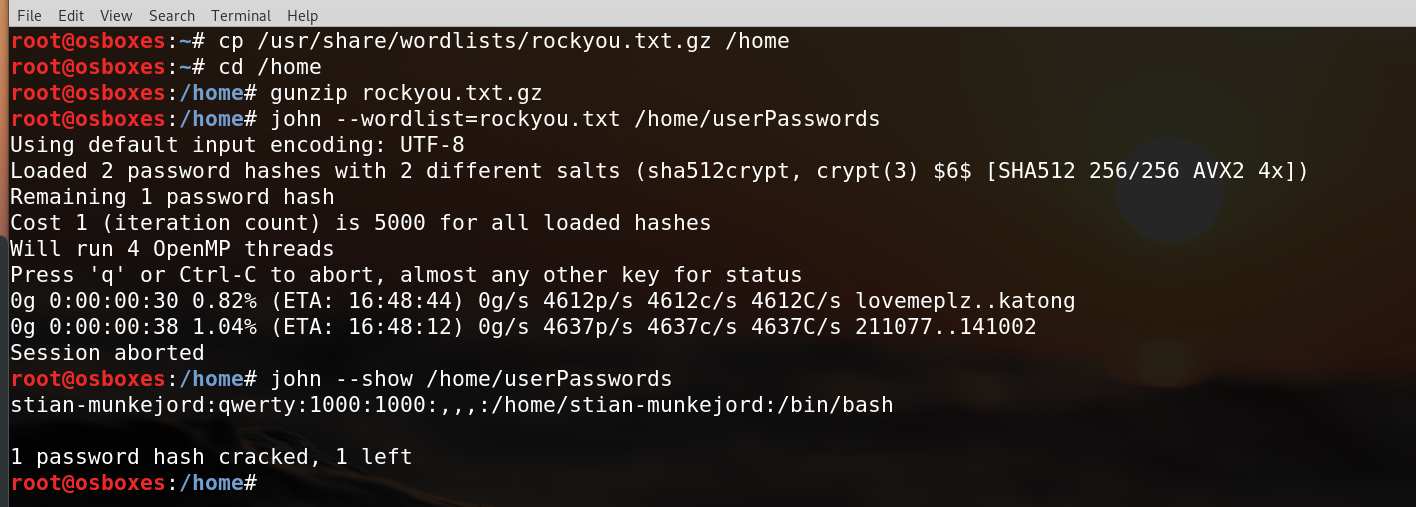
1. *Locate the password file of all users in Kali Linux and print its content (Take a screenshot of the home directory);*





1. *Try to use the tool John Ripper (or Hashcat) to crack your password file. If it takes too long, try to re-create the user with even weaker password so that John Ripper (or Hashcat) can crack within one hour. Take a screenshot of the output of John Ripper (or Hashcat).*

I only decrypted the hash to the user “stian-munkejord”, since the password: “osboxes.org” for the root user probably isn’t in the wordlist: “rockyou.txt”



**Access Control (35 pts)**

**Question 2.**

1. *Describe the differences between discretionary access control model and mandatory access control model*

The difference between discretionary access control (DAC) and the mandatory access control (MAC) model, is where the restrictions are set. In a DAC model the owner of the object specifies who is granted access, this is commonly found most operating systems such as Windows and Linux. On the other side the MAC model gives each subject a security clearance where only the people with high enough clearance can view specific files.

1. *File permissions in Linux can be also represented in digits from 0-7 for the owner, group and others with reading as the most significant bit (E.g., the value 6 represents the permission right rw- for a file). Suppose a file in Linux has the permission as the digits 764.*
   * *What does this permission right indicate for the owner/user, group and others?*

The number indicate that the: User can: read, write, execute; group can: read, write; others can: read.

* + *What is the letter representation of this permission right?*

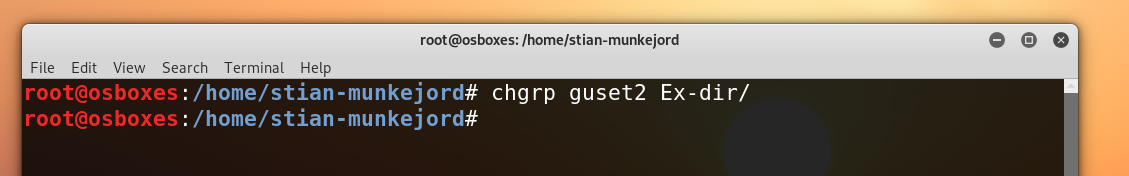
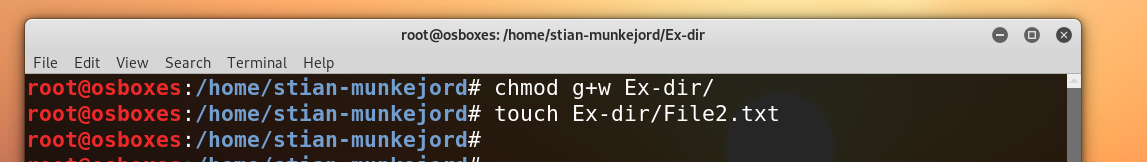
The letter representation for the number 764 is: “rwxrw-r--"

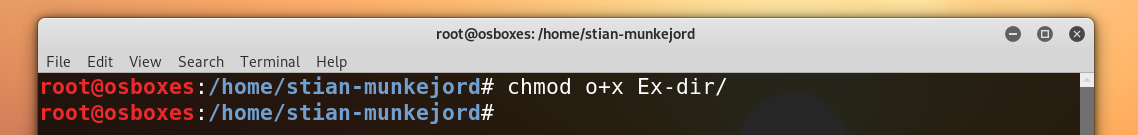
**Question 3.**

*In Kali Linux, create two users Guest2 and Guest3. Use the user (for yourself in Question 1) to create a directory Ex-dir containing a text file File1.txt.*

1. *Display the owner and group for the directory Ex-dir; (Take a screenshot of the command and its output)*

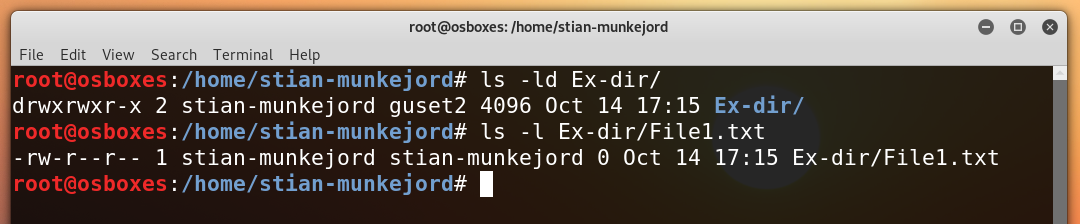


1. *Change the directory’s group as Guest2; (Take a screenshot of the command)*
2. *Assign Guest2 the* ***read*** *and* ***write*** *permission to the directory. Create a file File2.txt in the directory;*
3. *Assign Guest3 (as other users) only execute permission to the directory*

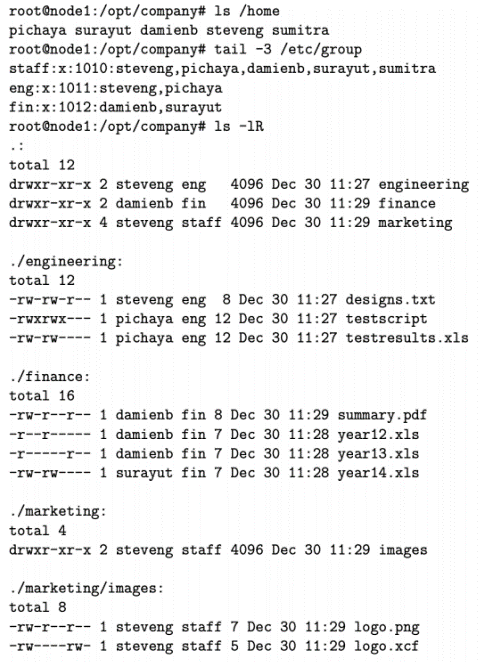


1. *Can Guest3 display the content of File1.txt in the directory? Justify your answer.*

As default all “other” users can read the file when it’s created, and since I haven’t changed any permissions to File1.txt. The permission to the directory where the file is located is also readable for “others” and therefore the file is also readable for guest3.

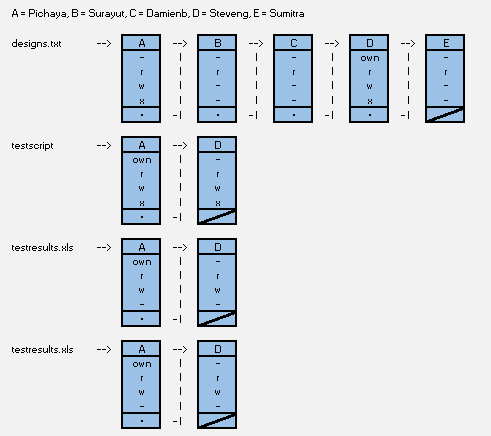
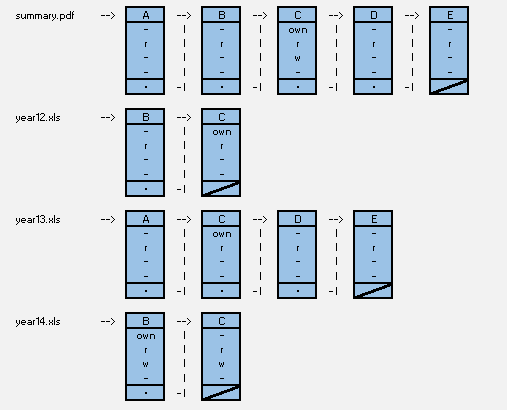


**Question 4.**

*A company has implemented a discretionary access control system for important documents and programs on a shared Linux server. Figure 1 shows the current implementation of the DAC system. It only applies to the directory /opt/company (you can ignore all files/directories that are not within this directory). The list of users is shown in the output from the first ls command; the members of each group is shown in the output of the tail command; the permissions are shown in the output of the second ls command.*

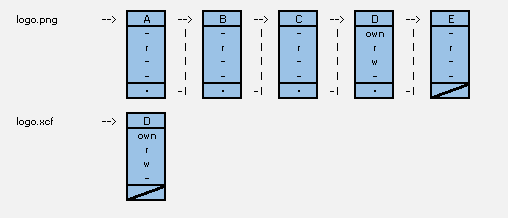
*Assume the files are the objects, the subjects are the five users and there are four access rights: own, read, write and execute. Refer to Figure 4.2 and Table 2 in the textbook and complete the following sub questions.*

*Figure 1: Implementation of a DAC system*

1.   *Draw the Access Control Lists that illustrate the implementation on the Linux server;*

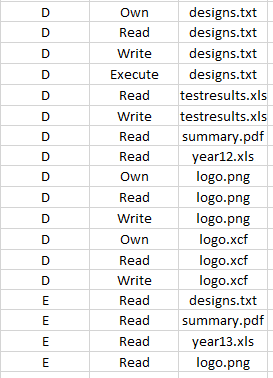
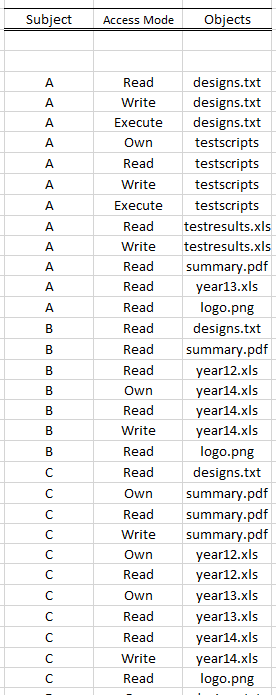
*Figure 1.2*

*Figure 1.1*



*Figure 1.3*

1. *Draw the Capability Lists that illustrate the implementation on the Linux server;*
2. *Build the Authorization Table that illustrate the implementation on the Linux server.*



*Figure 3.2*

*Figure 3.1*

**Malware (20 pts)**

**Question 5.**

1. *Malicious software can be classified by propagation method or payload. Explain the difference between the three common propagation methods: worm, virus and social engineering;*

The three common propagation methods are viruses, worms and social engineering. The difference of these different methods will I explain under:

Viruses hides inside another program and propagates itself to other programs, or even computers itself. Often the virus also includes some destructive functions to the computer.

Worms, on the other hand, can run independently and can propagate a complete working version of itself onto another host on the network. The worm usually consumes computer resources destructively.

Last, but not least we have social engineering which is all about tricking someone to give private information to compromise a computer system. An example on this is trojan horses. The trojan horse is a useful or seemingly useful program that contains hidden code of a malicious nature that executes when the program is invoked.

1. *Explain the difference between a normal virus, a metamorphic virus and a*

*polymorphic virus, including discussing how easy they are to detect by anti-virus software*

A virus is a computer program, and its sole purpose to infect other computer programs. The two types of viruses we’ll look at here are so called stealth viruses. These viruses are designed to slip under the installed anti-virus program of the host computer.

Firstly, we have **Polymorphic viruses**. These viruses inflict programs with an encrypted version of itself, and to decode it the virus also copy over a decryption module. Unlike normal encrypted viruses a polymorphic virus can modify the decryption module every time it infects a new file.

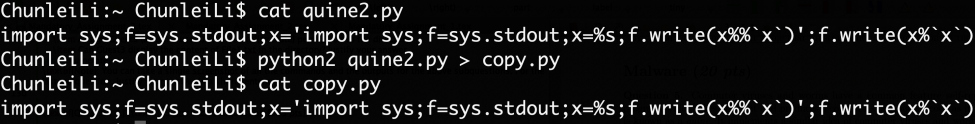
Secondly, we have **Metamorphic viruses**. Unlike polymorphic viruses the metamorphic viruses don’t use encryption, therefore it doesn’t need a decryption module. To achieve its ability to avoid antivirus detection most of the virus consists of a metaphoric engine. This engine completely rewrites itself every time it copies.

Finally, about the detection of the different types of viruses. Since polymorphic viruses changes it code, it makes the job for signature-based antivirus programs very difficult. However, the antivirus programs can decrypt the viruses using an emulator, or it can analyze its behavior to figure out the virus. The polymorphic virus therefore has some countermeasures to these solutions by the antivirus program. These involve reducing the speed of mutation and not to copy itself when there is already a copy on the computer. On the other hand, metamorphic viruses are just hard to detect in general because of its ever changing code of the virus.

**Question 6.**

*Computer viruses and worms have a common feature self-replication. A program, which takes no input and produces a copy of its own source code as its only output, is known as a quine. It implements a similar idea of computer viruses and worms. There are examples of quine program in different languages, e.g., one simple python quine example is as* ***follows****:*

*import sys;f=sys.stdout;x=’import sys;f=sys.stdout;x=%s;f.write(x%%‘x‘)’;f.write(x%‘x‘)*



*Write a python quine script by yourself (preferably with a part of your name embedded in the quine). Bonus points:*

* + *Manage to include your first/last name in the quine script (+ 5 pts)*
  + *Write a quine with more than 10 lines (+5 pts)*

Python 3 quine:

#StianMunkejordLoveCybersecurity#StianMunkejordLoveCybersecurity#StianMunkejordLoveCybersecurity

myQuine = 'print("myQuine = " + repr(myQuine) + "\\nName =" + repr("StianMunkejord") + "\\nChunleiFan =" + repr(ChunleiFan) + "\\nif Name ==" + repr("StianMunkejord") + ":" + "\\n    Name =" + repr("#") + "+ Name + ChunleiFan" + "\\nmyList = []" +"\\nfor i in range(3):" +"\\n    myList.append(Name)" +"\\nprint(" + repr("") + ".join(myList))" + "\\neval(myQuine)")'

Name ='StianMunkejord'

ChunleiFan ='LoveCybersecurity'

if Name =='StianMunkejord':

    Name ='#'+ Name + ChunleiFan

myList = []

for i in range(3):

    myList.append(Name)

print(''.join(myList))

eval(myQuine)

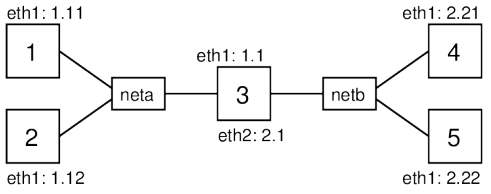
To write this quine I followed the tutorial at[[1]](#footnote-1). To further develop the code, I used the same premise with the repr() and eval() function.

The quine I wrote consists of a variable “myQuine” this variable contains the entire data structure of the program. To achieve this, I started to write a simple code, then copy it into the variable “myQuine”. The result is the code above, and when run it outputs a complete replica of itself.

**Firewalls (20 pts)**

**Question 7.**

*Given the following topology of a network*

**

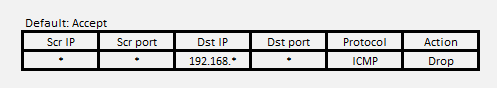
*where the router connecting the two subnets together, Nodes 1 and 2 are ’inside’ of the subnet neta, Nodes 4 and 5 are ’outside’ of neta and Node 3 with a firewall is the gateway of neta.*

*Try to write rules as general as possible. For example, although there are only two nodes outside, try to write rules such that the policy is achieved even if there were more than two nodes outside*

*Suppose the default policy is ACCEPT in the firewall on Node 3. Write packet filtering rules for the following goals (each of which is treated separately):*

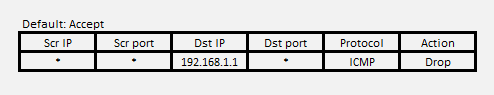
*While writing these rules a treat net a as the “inside” hosts and netb as the “outside” hosts*

***Rule 1.*** *Block ping (icmp) packets being forwarded between the two subnets;*

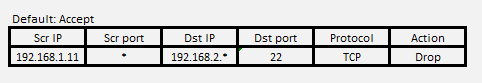


With this rule the two subnets cannot send the packets through the firewall which is the barrier between the two subnets.

***Rule 2.*** *Block ping packets coming into the firewall;*

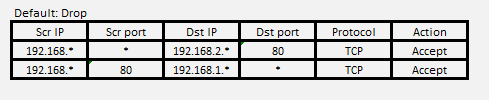


***Rule 3.*** *Prevent Node1 from SSHing to any outside nodes;*



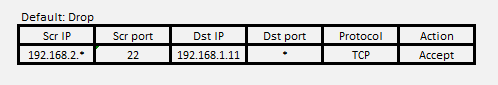
*Change the default policy as DROP and write packet filtering rules for the following goals:*

***Rule 4.*** *Allow inside hosts can access outside websites;*



For simplicity we only write the rules with the port 80 which is HTTP. (HTTPS = port 443).  
In this instance the default policy is DROP, therefore we need to write two rules because when the client sends a TPC SYN the server responds with a TPC SYN ACK. If the SYN ACK is not responded to by the client, the connection would not be established. However, there is a potential security breach when following this system. A hacker can connect to the “netb” network and can bypass the firewall.

***Rule 5.*** *Allow outside hosts can SSH into Node1. No other access should be allowed.*

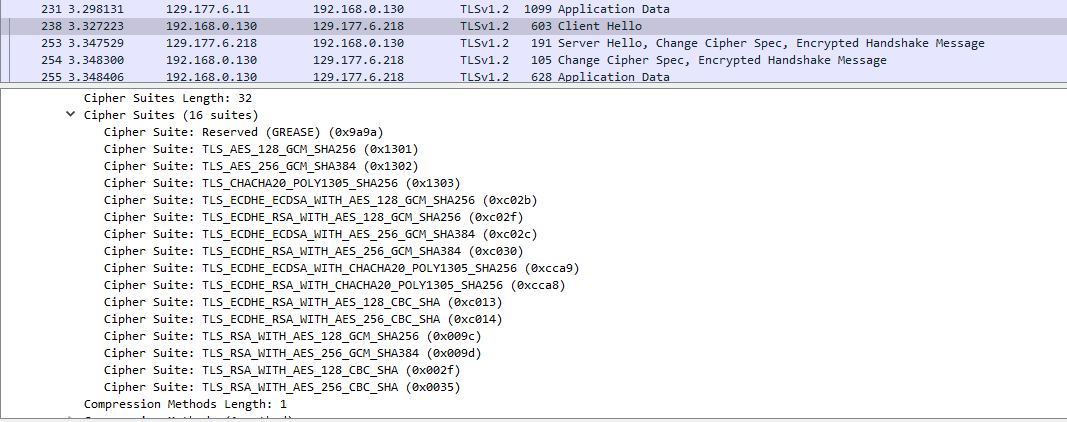


Since Node2 has a connection directly to Node1 without going through the firewall, this Node cannot be blocked to SSH into Node1 by the firewall.

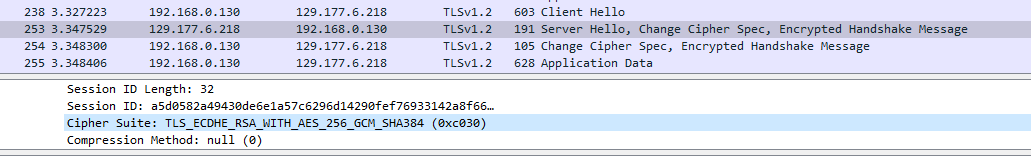
**Transport-Layer Security (10 pts)**

**Question 8.**

1. *Use Wireshark to capture the traffic when you visit the website https://mitt.uib.no/ login/ldap. Open the captured traffic Client Hello of the TLS protocol and take a screenshot of cipher suites initiated by the client (which is your browser)*



1. *Check the traffic Server Hello of TLS protocol. Display the cipher suite offered by the server and explain how the cipher suite will protect subsequent HTTP traffics (Application Data) between your browser and the server.*



The cipher suite contains a set of algorithms which usually include a key exchange algorithm, a bulk encryption, and a message authentication code algorithm. A cipher suite can also include authentication algorithms which help the server and or the client to authenticate the other entity.

In our instance the protocol is Transport Layer Security (TLS), the key exchange is done with Elliptic Curve Diffie-Hellman Ephemeral (ECDHE), the authentication process uses the Rivest Shamir Adleman algorithm (RSA), encryption is accomplished by the Advanced Encryption Standard with 256bit key in Galois/Counter mode (AES 256 GCM) and finally it is all hashed with the Secure Hash Algorithm 385 (SHA385)

With this setup the two devices can exchange a key between themselves which is used to securely deliver the message. Further the bulk encryption algorithm is used to encrypt the data being sent. Last but not least, the message authentication code algorithm provides integrity. This algorithm ensures the receiver that the information is coming from the right person.

1. https://towardsdatascience.com/how-to-write-your-first-quine-program-947f2b7e4a6f [↑](#footnote-ref-1)