G53SEC COURSEWORK 2018/2019

Daniel Kwok Zheng Xian

0135552358

khcy6dkz@nottingham.edu.my

Question 1

The following entails the policies that I would implement to the network. Before proceeding, it should be noted that policies should suite the nature of the application to strive the balance between security and convenience. The policies listed below represents that of the strictest nature with zero compromises for convenience.

1. Enforcing Password History policy

   This policy states how often an old password can be reused for future purposes. This discourages users from alternating between several common passwords which decreases the strength and reliability of the password. For this implementation, users would be prohibited from re-using the last 5-6 previously used passwords.

2. Minimum Password Age policy

   This policy states the minimum period that a password must be kept in use before it can be changed. This would prevent a more tech-savvy user to work around the previous policy by simply reverting back to the old password after renewing. For this implementation, the minimum period would be set to 2 days, unknown to the user, that would create just the right amount of hassle that would prevent the workaround as mentioned above.

3. Maximum Password Age policy

   This policy states the maximum period in the password can be kept in used before it expires, and the user would be required to change it. For this implementation, the maximum period would be 180days (1/2 year) for passwords and 730 days (2 years) for passphrases.

4. Minimum Password Length policy

   This policy states the minimum number of characters needed to create a password. Generally, a longer password would mean a longer brute-force time. For this implementation, a minimum password length of 10 characters would be enforced.

5. Password Composition

   This policy states the type of characters needed to create a password. For this implementation, an alphanumeric, with a mix of upper and low case alphabetic characters, along with special symbols would be required. This would take a conventional computer around 289217 years to brute force through.

6. Strong Passphrases

   A passphrase could be simply understood as a simpler form of a password. Instead of random characters bunched together forming an incredibly hard to remember seemingly random string, a passphrase is usually a sentence, or, a phrase. As such, it is much easier to remember by a user as it is more natural sounding, yet harder to crack due to its long string length. For this implementation, a passphrase of 15characters minimum, with the upper and lower case would be used.

7. Maximum attempts
   This policy states the maximum number of attempts a user can undertake to log in using the password to prevent brute force attempts. For this implementation, a maximum login attempt of 3 times per 24 hours would be implemented.

8. E-Mail Notifications
   E-mail notifications prior to password expiry remind users when passwords are due for a change.

9. Password storage
   This policy states how users' passwords would be stored on the system. All passwords should never be stored as plain texts. Instead, passwords should be irreversibly encrypted into a hash.

Question 2

SSH stands for secure shell, which is a standard for the secure remote logins and file transfer over the network. The application data traffic flows within an encrypted SSH connection so that even if it is intercepted, cannot be extracted. SSH is also a tool that could be used to tunnel traffic that would otherwise be blocked by a network firewall. These are several reasons that an admin might choose to block ports from normal traffic.

1. Prevent unauthorized remote access
   All machines come with the capability to be remotely accessed. Windows-based machines have remote desktop protocol (RDP) while Linux-based machines have secure shell (SSH). However, with this capability, any user who is able to log into the server can access these capabilities – hackers and malware included. By using firewall to block ports, these “entrances” to the machines are closed off, preventing unauthorized remote access.

2. Blocking of content
   Whether if it’s to discourage slacking in the workplace, or to prevent unsuitable content from being viewed by the user, these websites could be blocked off easily with a firewall.

How is SSH used to circumvent firewall restrictions? By simply connecting the client-side computer to an SSH server through port 22, which most firewalls allow for communications to happen over because it’s the port used by HTTPS.

One legitimate cause for using SSH tunneling would be for corporate environments that employ mainframe systems as their backend services to achieve compliance standards such as SOX, HIPAA, and PCI-DSS without the need to modify those very applications, as these modifications are usually impractical or costly. By having SSH tunneling as a security wrapper, a more practical method of adding security to these applications can be achieved. For example, ATM networks run using tunneling for security.

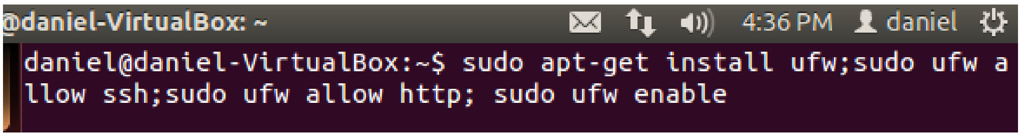
However, due to the very fact that SSH connections are encrypted, this makes the contents invisible to network monitoring and traffic filtering, which if deployed, could be used for malicious purposes such as data exfiltration, while keeping the attacker anonymous. This anonymity can be achieved by simply bouncing the attack of systems and devices that’ll hide the track of the attacker. The problem with this is that most organizations with a server in public cloud permit outgoing SSH connections, which an attacker could connect to via another external SSH server. Most firewalls offer little to no protection against these kinds of attack.

Question 3

A vulnerable server was scanned and accessed in the final lab to improve its security. These are the actions performed to accomplish such goal.

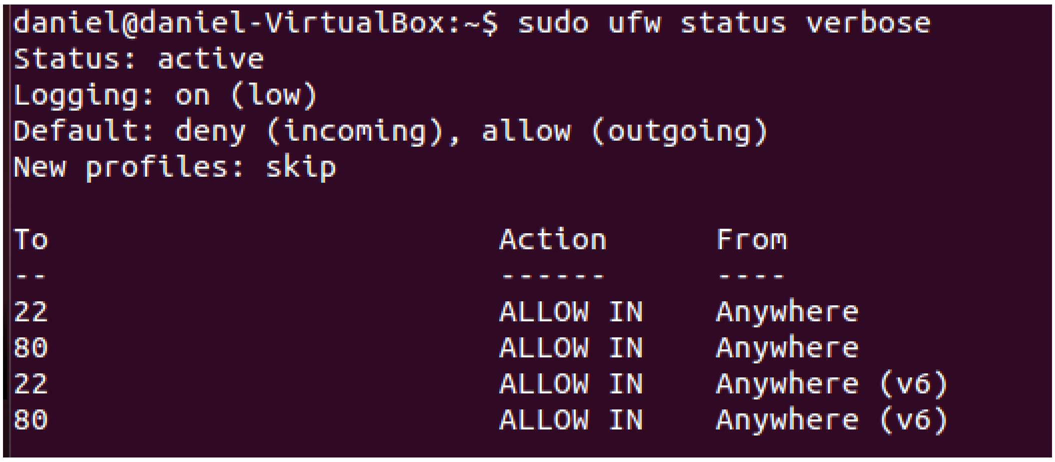
The most basic step would be to first install a firewall. Based on predetermined filtering rules, a firewall acts as a network security system that monitors and controls incoming and outgoing traffic by blocking or restricting access to every port, except those that should be publicly available (80 for HTTP, 3306 for MySQL, etc). By doing so, access to the server can be tightly controlled according to any sort of preset configuration.

For this implementation, Uncomplicated Firewall (UFW) is a basic firewall installed.



```
daniel@daniel-VirtualBox:~$ sudo apt-get install ufw;sudo ufw a
llow ssh;sudo ufw allow http; sudo ufw enable
```

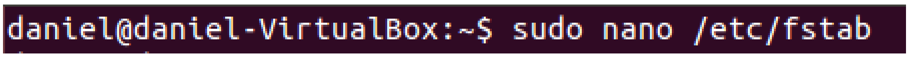
Figure 1: Install ufw; allow for ssh and http; enable ufw service



```
daniel@daniel-VirtualBox:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing)
New profiles: skip

To                         Action      From
--                         ------      ----
22                         ALLOW IN    Anywhere
80                         ALLOW IN    Anywhere
22                         ALLOW IN    Anywhere (v6)
80                         ALLOW IN    Anywhere (v6)
```

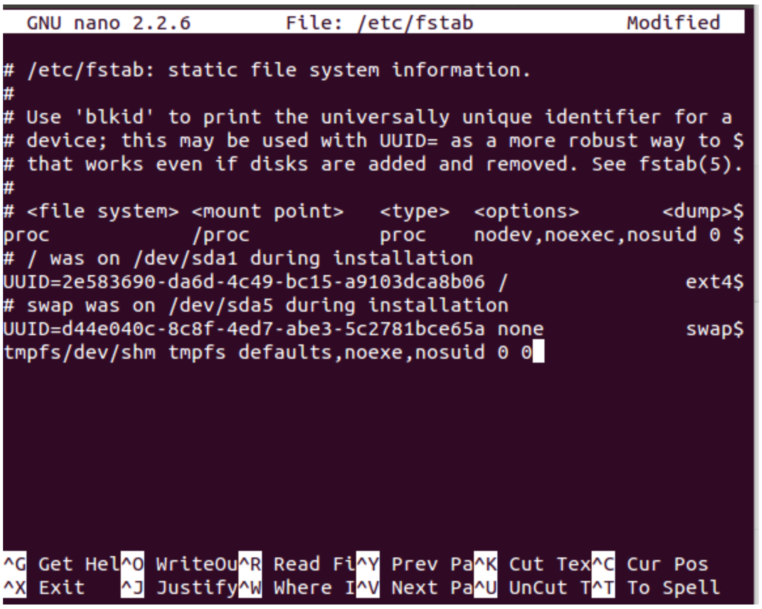
Figure 2: Check THE status of ufw

Shared memory is a feature in the server which allows for efficient passing of data between programs by having both processes utilizing the same memory space. However, due to the very nature of shared memory being mounted as read/write by default, this space can be exploited by malicious parties easily. This memory space can be secured easily by simply having the memory space to be read-only, deprived of any permission to execute or modify running programs. For this implementation, the shared memory space located at /etc/fstab shall be secured.



```
daniel@daniel-VirtualBox:~$ sudo nano /etc/fstab
```

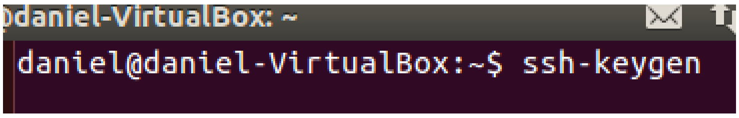
Figure 3: Open memory space file



```
  GNU nano 2.2.6          File: /etc/fstab              Modified

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to $
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>$
proc            /proc           proc    nodev,noexec,nosuid 0 $
# / was on /dev/sda1 during installation
UUID=2e583690-da6d-4c49-bc15-a9103dca8b06 /               ext4$
# swap was on /dev/sda5 during installation
UUID=d44e040c-8c8f-4ed7-abe3-5c2781bce65a none            swap$
tmpfs/dev/shm tmpfs defaults,noexe,nosuid 0 0









^G Get Hel^O WriteOu^R Read Fi^Y Prev Pa^K Cut Tex^C Cur Pos
^X Exit   ^J Justify^W Where I^V Next Pa^U UnCut T^T To Spell
```

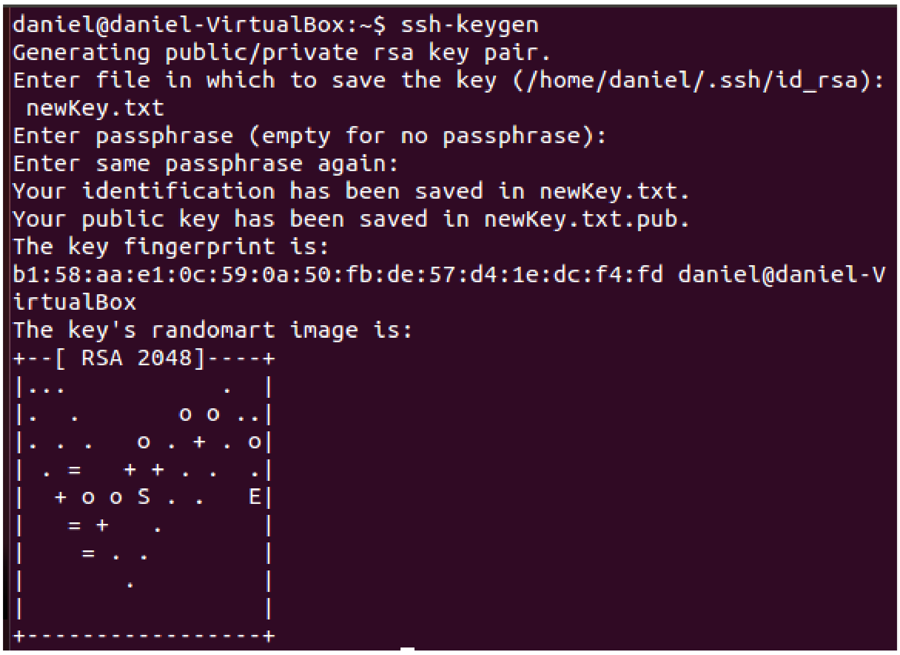
Figure 4: Appended final line to file

SSH is essential to remote server management. It's a more secure form of inter-machine communication, replacing tel-net. Despite this, there still exists options to harden the SSH of a server. For starters, instead of using root, users would be connected with sudo permission, designating heighten permissions only when required. Idle sessions, or long sessions of inactivity could be dangerous as well. These users shall be logged off the network after a pre-determined amount of duration. There are two forms of authentication –passwords, or SSH keys – the latter which is more secure, simply due to the fact that it is much harder to brute force into, compared to the prior. A private and public key pair is created prior to authentication. As the name suggests, private key is kept secret and secure by the user, while public key can be shared out. To connect to the server, user would have to present the corresponding private key. If succeed, user would be able to access the server.



```
)daniel-VirtualBox: ~
daniel@daniel-VirtualBox:~$ ssh-keygen
```

Figure 5: Creating ssh key pairs

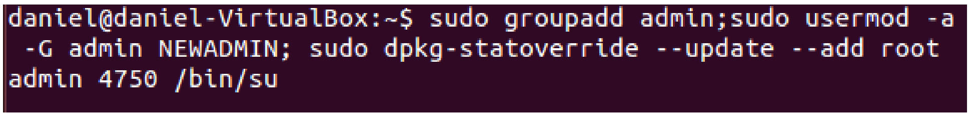


```
daniel@daniel-VirtualBox:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/daniel/.ssh/id_rsa):
 newKey.txt
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in newKey.txt.
Your public key has been saved in newKey.txt.pub.
The key fingerprint is:
b1:58:aa:e1:0c:59:0a:50:fb:de:57:d4:1e:dc:f4:fd daniel@daniel-V
irtualBox
The key's randomart image is:
+--[ RSA 2048]----+
|...           .  |
|.  .       o o ..|
|. . .   o . + . o|
| . =   + + . .  .|
|  + o o S . .   E|
|   = +   .       |
|    = . .        |
|       .         |
|                 |
+-----------------+
```

Figure 6: SSH key pair created

SSL is a technology which secures and safeguards any and all data sent between two systems via the internet. However, version 3 protocol has been proven to be insecure to a form of attack called POODLE (Padding Oracle on Downgraded Legacy Encryption), which is a form of man-in-the-middle exploit. Hence, we would disable Apache support for the protocol to enforce the usage of newer protocols.

SU is a command in Linux-based systems to switch from one user account to another. The usage of this command should be minimized as much as possible. This can be achieved with the creation of an admin group, and proceeding to add users via the admin group. This would limit the usage of su only to that of the admin group.



```
daniel@daniel-VirtualBox:~$ sudo groupadd admin;sudo usermod -a
 -G admin NEWADMIN; sudo dpkg-statoverride --update --add root
admin 4750 /bin/su
```

Figure 7: Creating admin group

SYSCTL is a command available in Ubuntu which allows for the configuration of kernel parameters, at runtime. All parameters available is located in the /etc/sysctl.conf file. Source routing of incoming packets and logging of malformed IP’s could be prevented by editing this file.

Despite all of the actions taken above, a much simpler solution would be to simply upgrade Ubuntu version of the server. With the upgrade, comes along all of the security implementations mentioned above. However, a much more desirable long term solution would be a complete system migration to a more secure distribution. Linux distributions such as Arch Linux or Fedora provides much more features that are more “server-friendly”.