Section 1: Passwords

Password policies are a set of rules which govern what types of passwords users can use, and how often they need to be changed. A strong password policy is paramount towards maintaining the security and integrity of user data. In order to ensure this, it is recommended that all passwords should meet the following criteria

- Adopt the 8 + 4 rule. Use eight characters with one upper and one lower case, a special character like as asterisk and a number. The more random the better. This would help to significantly reduce the risk of a hack from a mask attack
- Create a password blacklist. Most attacks start with hackers attempting to guess a password by using a database of the most popular passwords, dictionary words, or passwords that have already been cracked. Comparing every new password with a blacklist of such passwords will therefore help to prevent the usage of weak passwords by users.
- Do not allow sequence of characters, such as 'abcd' or '1234', as this would make the passwords vulnerable to combinator attacks
- Passwords should not contain any part of the user's personal identification details, such as username, name or date of birth, as this information can be known by many people

There are a few other traditional password security practices which had been enforced over the years which have now been deemed ineffective by recent studies and research. For example, many organizations enforce a password expiry policy, which is usually 90 days. Research however shows that this makes users more likely to select predictable passwords, composed of sequential words and numbers which are closely related to each other. In these cases, the next password can be predicted based on the previous password. Password expiration requirements offer no containment benefits because cyber criminals almost always use credentials as soon as they compromise them. Another common misconception is that longer passwords are more secure. Password length requirements of greater that 10 characters can result in user behaviour that is predictable and undesirable, such as using repeating pattern like 'password123password123', and encourage bad practices such as writing down passwords in unencrypted documents.

Apart from using strong password policies, other measures such as multi-factor authentication can also be enforced to enhance security. This involves making users update contact and security information, such as an alternative email address, phone number, or a device registered for push notifications, where they can respond to security challenges or receive security alerts. This feature can be used to verify suspicious log in attempts, or to recover a forgotten password.

Lastly, it is also vital that passwords are stored in the system in a secure manner. It is bad practice to store plain text passwords, as they can easily be read by administrators, or even worse, by anyone if the system is hacked. Password salting used simultaneously with a one-way hash function can be an effective solution. A random salt is added to the password before it is hashed, and the salt is stored unencrypted with the hash. Hashing ensures that plain text passwords are not visible to anyone at any point. By using a different random salt for each user, we can ensure that cracking multiple passwords is slower and also prevent rainbow table attacks, as we can't precompute that many password combinations.

Section 2: Firewall

Administrators might often block certain ports to reduce vulnerability from malwares, as open ports allow malwares to easily run a service on those ports. Certain ports can also be blocked to stop users from running their own servers, for file sharing, gaming, or anything else they shouldn't be doing on the network. Administrators may also choose to restrict, but not prevent access, for example by allowing only a range of IP addresses through a port. In some cases, ports can also be blocked to prevent outgoing traffic from a network. This is usually used to limit what an attacker can do when a system has been compromised. For example, if a malware has been installed on the system, the malware might try to send packets back to a command and control system to get additional code downloaded, or to accept new tasks. Blocking outgoing traffic can prevent this from happening.

However, it is common practice nowadays to circumvent these restrictions using SSH and proxies. For example, let's say that a firewall on machine A blocks outgoing packets through port 80 and port 443, which by default handles HTTP and HTTPS packets respectively. However, outgoing packets can be redirected through an encrypted ssh connection on port 22, which is usually open. If we wanted to communicate with a webserver C via the blocked port 80, we can embed an HTTP communication channel within an encrypted SSH tunnel, which would pass the firewall on the open port 22. We can assign port 9000 to run this SSH service, and connect to our proxy server, '10.0.2.4', with the following command:

ssh -D 9000 -f -C -N 10.0.2.4

-D tells SSH to create a SOCKS tunnel on port 9000, whereas -C compresses the data before sending it. -f forks the process to the background, and -N tells SSH that not command will be sent once the tunnel is up. In this case, dynamic port forwarding is being used. In contrast to local port forwarding, the destination port doesn't need to be specified, as SSH can handle connections from multiple ports. It analyses the traffic to determine the destination for given connection. The SSH acts as SOCKS proxy server. A SOCKS proxy is a simple SSH tunnel in which specific applications forward their traffic through the tunnel to the remote server, then the proxy forwards the traffic out to the general internet.

The proxy server that we have connected to through ssh will receive and forward the packets to the webserver C, and then return the response from the webserver C. Therefore, we are essentially browsing the web from a different proxy machine. All we need is to have access to a proxy server that isn't behind our firewall, and the ability to connect to it via ssh.

Use of ssh tunnelling are not necessarily malicious. For example, an administrator might use ssh tunnelling to execute necessary software updates on workstations remotely. On the other hand, ssh tunnelling may sometimes be used by users to access dangerous and insecure sites, which would otherwise have been blocked by the firewall. This can often result in serious security breaches across the entire network if a malware or other malicious files are inadvertently downloaded and installed into the system.

Section 3: Securing Server

In order to secure the server, first and foremost it was necessary to change the password to a more secure one, as the password of the server was very quickly obtained by a brute force attack. As discussed in Section 1, it is absolutely vital that the password used isn't a commonly known weak password, contains a random mix of different types of characters and cannot be easily guessed or associated with other common knowledge.

It was also found that the server was prone to attacks from outsiders, as no firewall has been set up to filter incoming and outgoing traffics from the network. UFW (Uncomplicated Firewall) was installed and configured with Firestarter GUI. Whereas HTTP and HTTPS packets at ports 80 and 443 were allowed to send and receive packets, an IP Table was used to compare source IPs of all incoming packets against an IP blacklist. If a match is found, that packet is dropped. All Telnet traffic into port 23 were also tunnelled through SSH, as telnet sessions between clients and servers are not encrypted. An attacker can easily intercept TCP/IP packet flow between the machines and reconstruct the data that flows between the endpoints and read the messaging, exposing sensitive information such as usernames and passwords. All FTP communications were also tunnelled through SSH, as like Telnet, FTP packets are not encrypted.

The root account for the server was enabled. When Ubuntu is installed, the standard root account is created, but no password is assigned to it. Therefore, a user can't log in as root until a password is assigned to the root account. Users can however use the sudo command to run a single command as root privileges. They would be prompted to enter their passwords before they can a command as a root user. By default, Ubuntu would remember their passwords for fifteen minutes, and until fifteen minutes are expired, no further requests for password entry would be made. Unlike the su command, which switches the user to the root account and requires the root account's password, the sudo command does not switch to the root account or require a separate password. This has a couple of benefits, as users will only have to set and remember a single password. This also discourages users from using the root account and keeping root shell open for their normal work. Reducing the number of commands as root enhances security and prevents unintentional system-wide changes. The root account was therefore locked using the command '*sudo passwd -l root'.* It is also important to note that only administrator accounts can execute sudo commands in Ubuntu. Therefore, it can easily be ensured that standard users who doesn't require any root privileges can't inadvertently run root commands which can have dangerous security implications.

The default SSH port was changed to a higher port from the standard 22/TCP port, as that port is continuously targeted for vulnerabilities by hackers and bots. Root log in using SSH was also disabled. In the event that the root account is inadvertently unlocked, this would still ensure that users are not able to log in as root. SSH access was also restricted to certain known users on the system. Even with right password, a user not in the 'whitelist' will not be able to log in. This will help to limit damage in the event the server is hacked and the attacker tries to create a 'backdoor user', by preventing them from gaining any remote access.

Apache support for the SSL v3 protocol has also been disabled, and instead forced to use newer protocols. SSL v3 has been proven to be insecure, as it has a vulnerability that allows the plain text of secure connections to be calculated by a network attacker.

To prevent attackers from gaining access network, the Snort intruder detection system (IDS) was installed. Snort can monitor network traffic for intrusion attempts, keep logs of attempted intrusions and take pre-specified actions when an attempt is detected.

DenyHost was installed to block SSH attacks. DenyHost monitors log in attempts to SSH, and if numerous unsuccessful logins from the same IP address are detected, DenyHost will block further log in attempts from the IP address by putting it into /etc/hosts.deny. Since DenyHost is written in python, the python environment also had to be installed.

Attackers often use IP spoofing to gain unauthorized entry into servers, by illicitly impersonating another machine by manipulating IP packets. The attacker alters the IP header with a forged IP

address, a checksum and the order value. The process starts with the attacker identifying the targeted machine to which the packet will be sent into, and finding IP addresses trusted by that machine. The data packet will then be spoofed to appear as if it's being sent from one of these whitelisted IPs. Once the packet has been accepted by the recipient, the hackers can use it for malicious or illegal activities. To prevent IP spoofing on the server, the following command was executed in the terminal: sudo vi /etc/host.conf

This opened the host.conf file, where the following lines were added:

order bind,hosts

nospoof on

Apart from the actions mentioned above, a few other security fixes would have been desirable. For example, the server has Ubuntu 12.04 installed, but the latest Ubuntu version is 19.04. It is good practice to regularly update to the latest version of the operating systems, as older versions may have security vulnerabilities and will inevitably lead to software compatibility issues. For example, encryption technologies used in older versions of Ubuntu may no longer be up to speed with modern hacking techniques. Security critical elements of a software may not be able to function effectively on an outdated operating system and might therefore make sensitive data more vulnerable to intruders.