Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Programming Project 3 - DVWA

Abstract

The goal of this assignment is to work through the Damn Vulnerable Web App (DVWA) and report on the vulnerabilities as well as their potential fixes. Some of these challenges required assistance from online guides and walkthroughs. These sources are listed alongside sources for the background and description of the vulnerabilities and their fixes. Unless stated otherwise, all vulnerabilities were assessed in the application's Low security mode.

Contents

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Setup

Kali Linux was used to set up the DVWA in this project. This decision was made on the fact that Kali had been previously installed in VirtualBox on the development computer and the installation of DVWA in Kali appeared to be simple and straightforward.

Installation of DVWA was indeed simple and straightforward on Kali. This was done by opening a terminal and entering the following command:

```
sudo apt install dvwa
```

In addition to installing the DVWA directory and configuration, this command also installed the PHP8 and Apache2 dependencies. This can be seen in Figures 2 and 3.

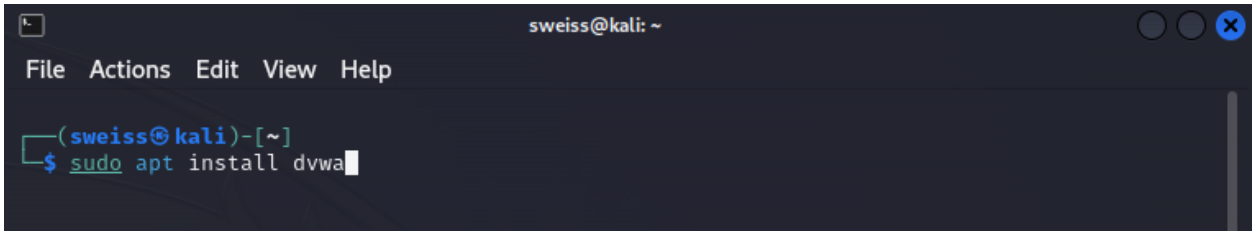

Figure 1: DVWA installation command

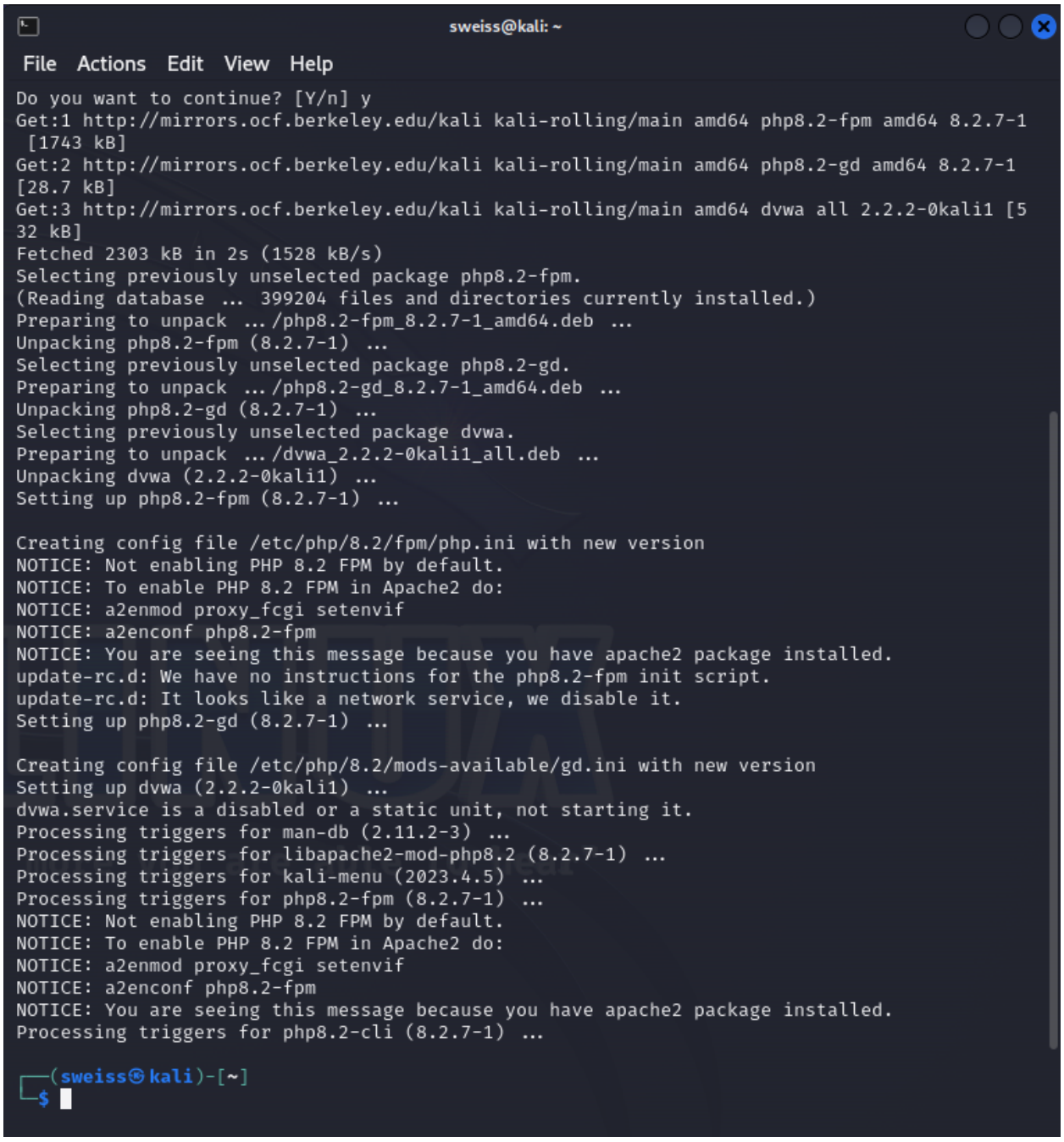

Figure 2: Installation report in terminal.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

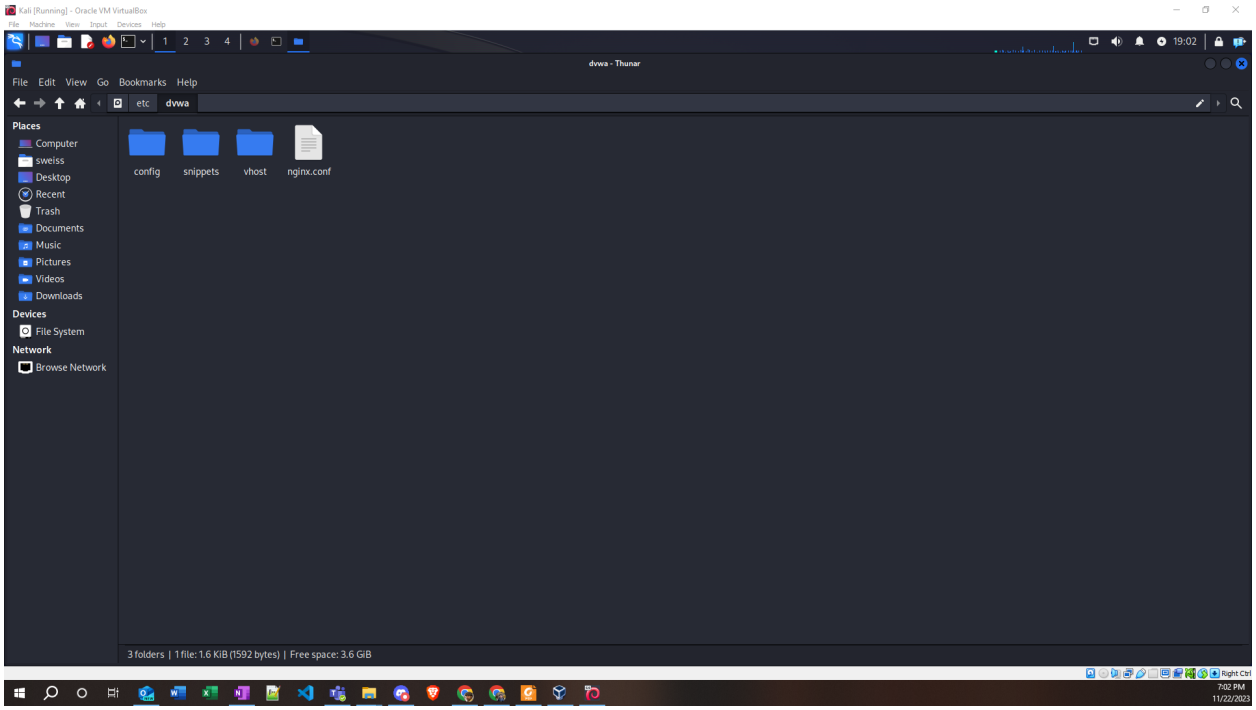

Figure 3: Wall-to-wall screenshot of /dvwa directory after installation.

The command `dvwa-start` was then entered in the terminal to start the application, which automatically opened in a browser. The Setup / Reset DB task was then executed to finish installation and setup. The security setting was then set to “Low” to ease into the vulnerability exploration.

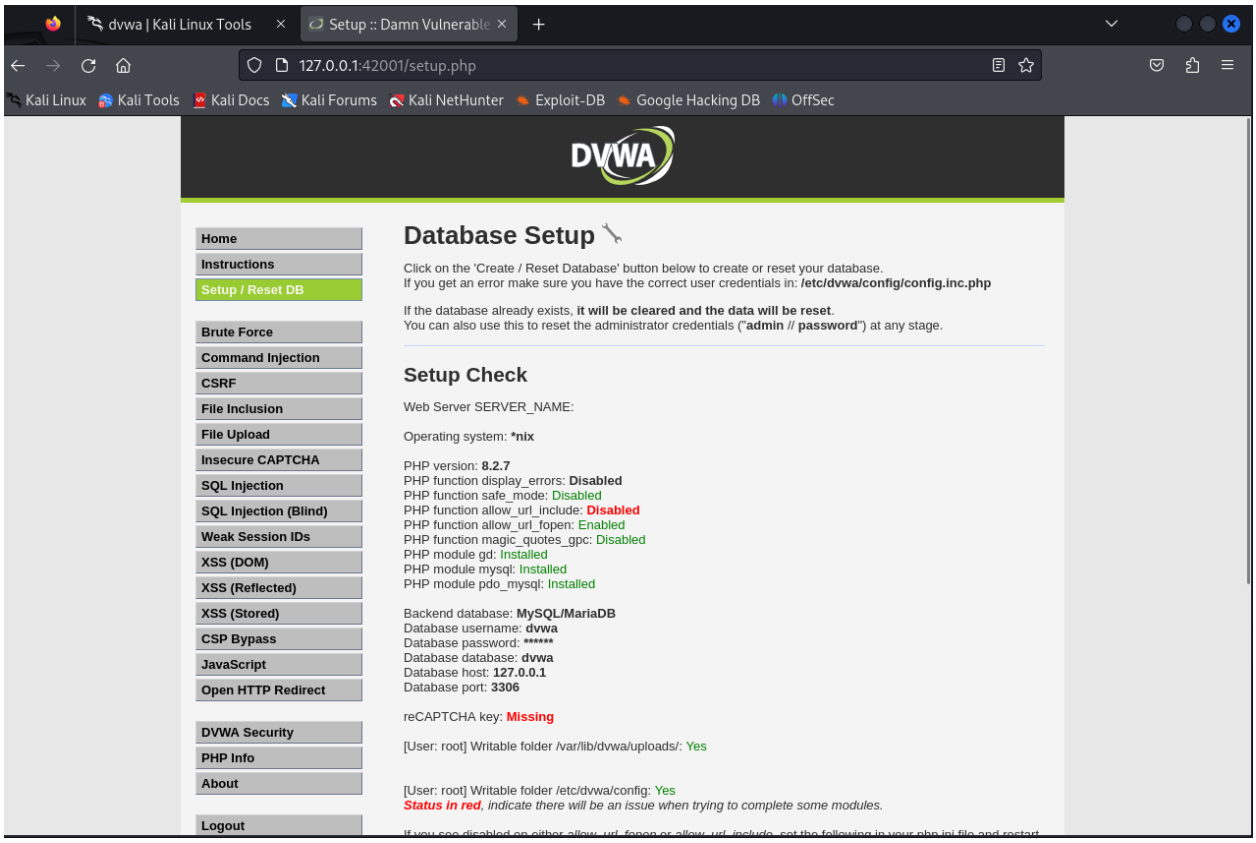

Figure 4: Database Setup page.

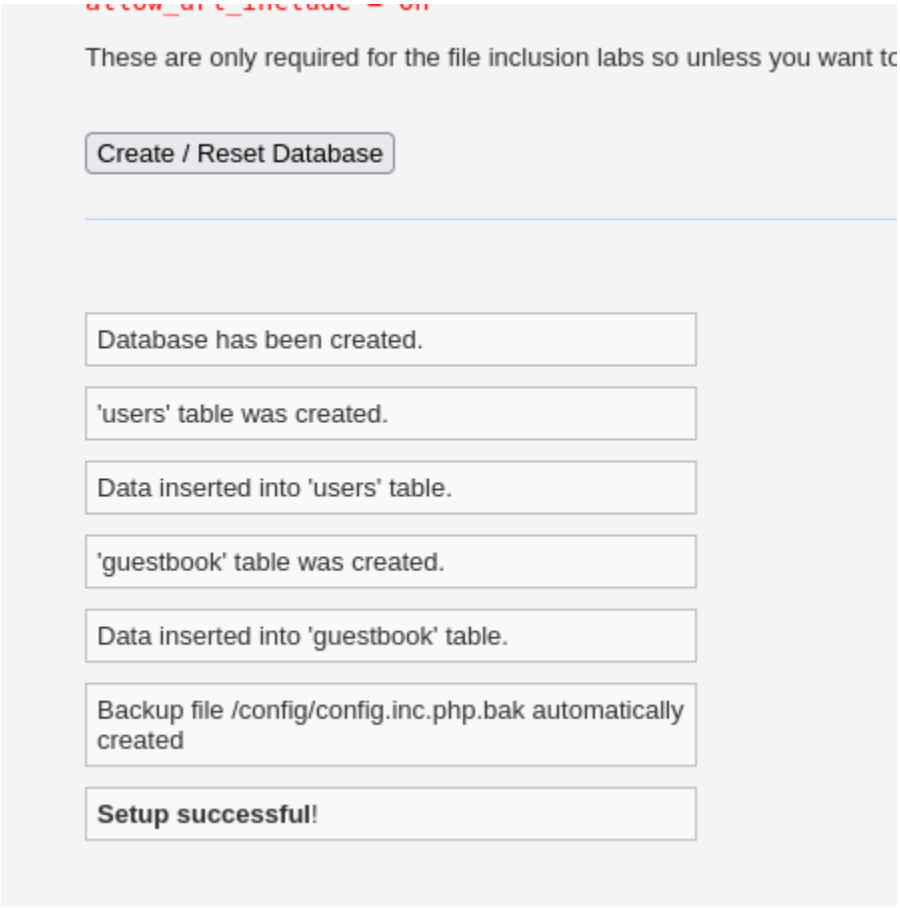

Figure 5: Database setup confirmation.

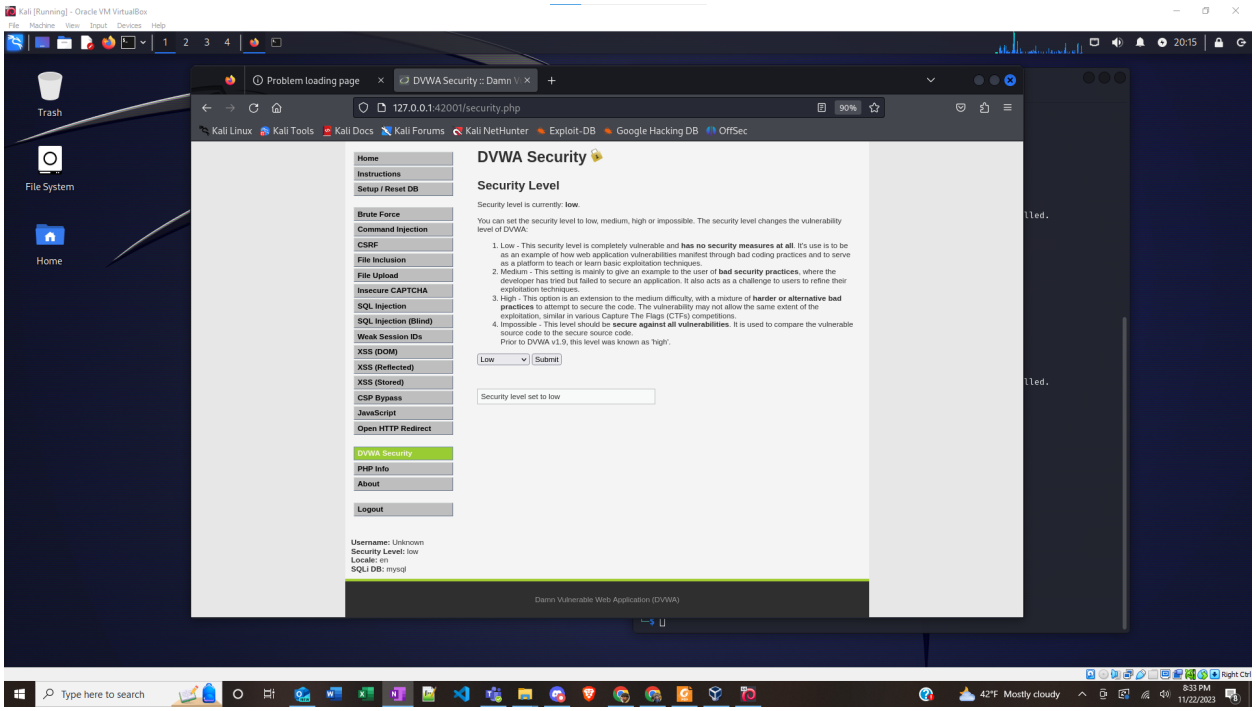

Figure 6: Setting the security setting to “Low”.

Vulnerability 1 - Brute Force

How does this feature normally work?

This feature is a simple login screen. If the correct username and password is entered, an image and a message saying “Welcome to the password protected area.” is displayed. If an incorrect username or password is entered, the message “Username and/or password incorrect.” is displayed.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

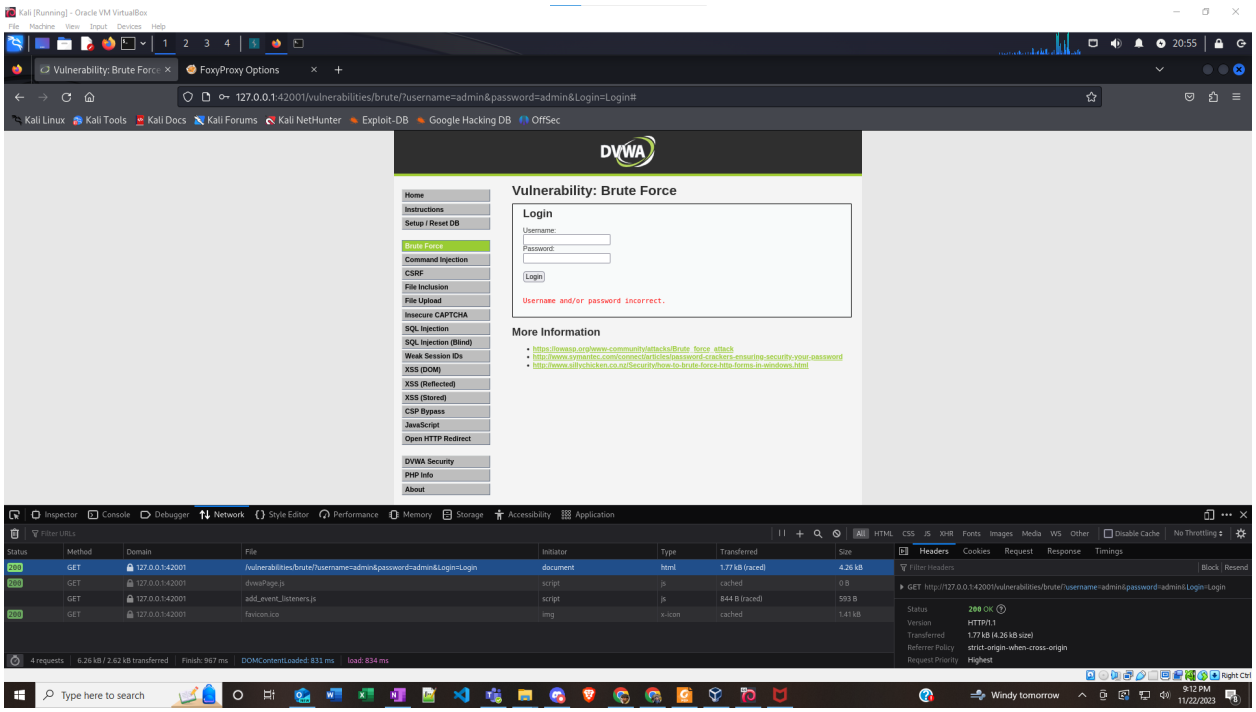

Figure 7: Incorrect username or password screen in Brute Force task.

What does it take to exercise the vulnerability?

Brute Force vulnerability takes time and computing power. It involves checking every possible known combination of characters for a password for each user. This process can be sped up by using a dictionary attack which uses lists of commonly used usernames and passwords to try before checking every combination of characters.

How did the feature work differently than normal use?

Once the correct username and password had been retrieved, this feature worked no differently than normal. However, assuming that “normal use” means a user entering an incorrect username and password and receiving the message “Username and/or password incorrect.”, this feature works differently once the correct username and password are entered by displaying an image and the message “Welcome to the password protected area <username>.” where <username> is the username used to log in.

Why did this work differently?

The page displays a different response because the correct username and password have been verified in the application’s database.

Why we should care about this vulnerability and potential loss

This is an important vulnerability to be aware of because unauthorized users can access sensitive data with malicious intent. The entire point of requiring a username and password is to regulate who has access to what information. If this login page was protecting a list of names and associated credit card numbers, those could be stolen and used illegally.

Briefly describe how to fix the vulnerability

The “View Help” button on the application page explains the difference between the various levels of security. The Low level only requires a username and password. The Medium level adds a delay between allowed login attempts. This will not stop a brute force attack, but it will slow it down and in some cases might make it computationally inefficient. The High level uses an “anti Cross-Site Request Forgery (CSRF) token” which evades timing predictions by brute force programs, making it further less efficient but not impossible to break. The Impossible level uses a “lock out” which locks the account after a predetermined number of failed attempts. Unless the brute force attack is able to guess the password within the attempt limit, it will fail. Often, these limits are set to 3-5 attempts.

Additionally, enforcing a password policy to disallow default and weak passwords can help prevent against the dictionary version of a brute force attack.

A significant way to fix this vulnerability is to enforce multi-factor or multi-step authentication. Brute force attacks cannot handle this because they do not have access to the other authentication factors and only repeatedly try possible passwords.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Try username: admin, password: admin (based on previous experience with default account info.) with the inspector window open.

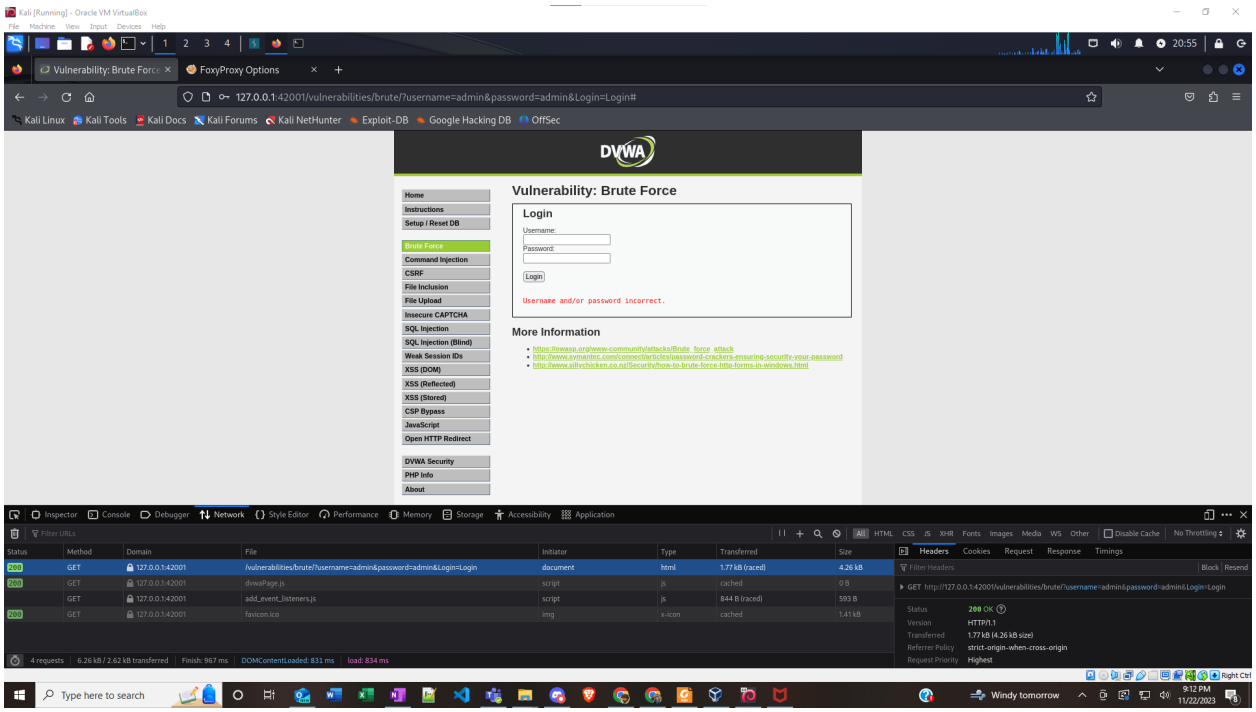

Figure 8: Initial login attempt.

Step 2: Install Foxyproxy and configure with Burp Suite.

Step 3: Enable “Intercept” on Burp Suite and send another login request. Capture the login request and send it to the “Intruder” feature in Burp.

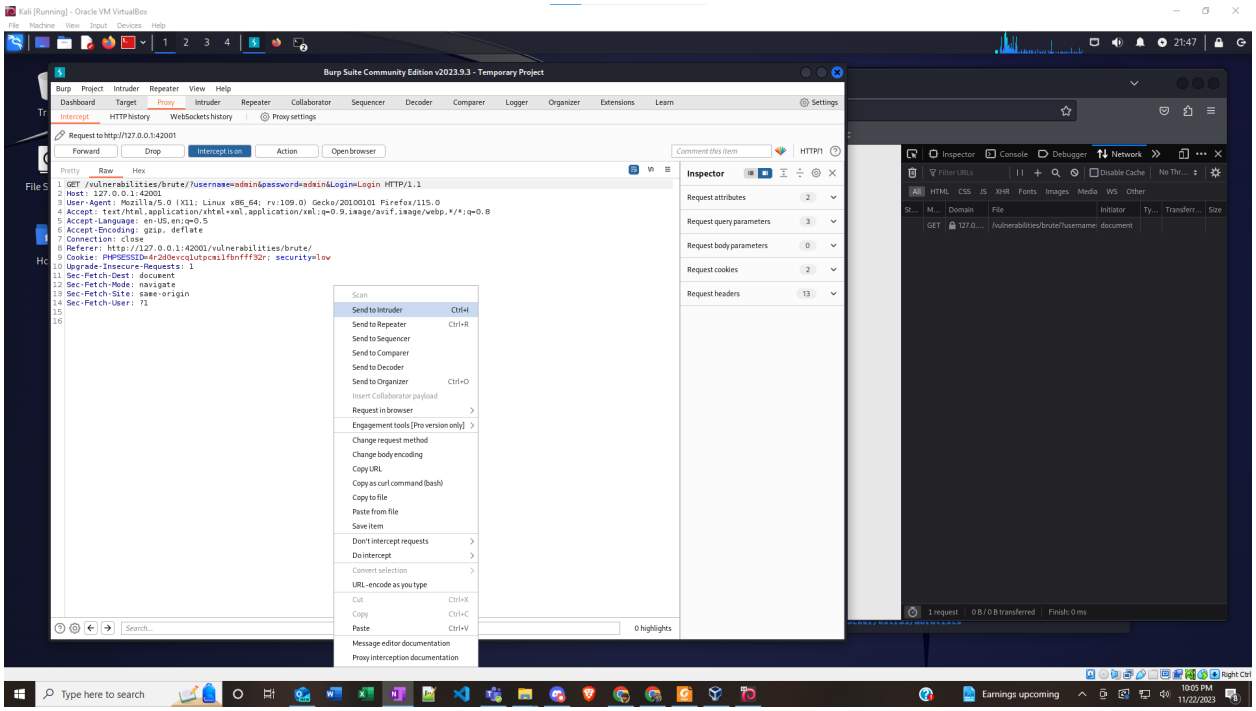

Figure 9: Capture login request in Burp and send it to “Intruder”

Step 4: Search for word lists that come with Kali Linux:

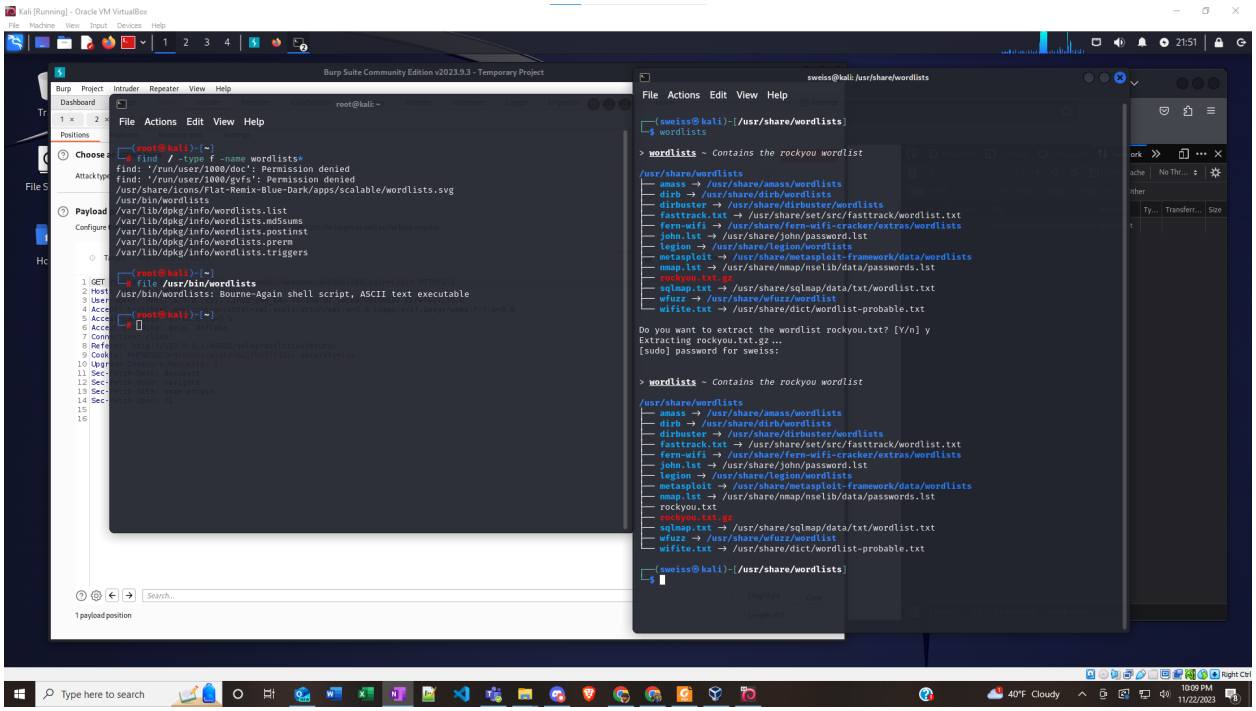

Figure 10: Search for Kali Linux wordlists.

Step 5: Select john.lst because it lists commonly used passwords in order of frequency rather than alphabetically.

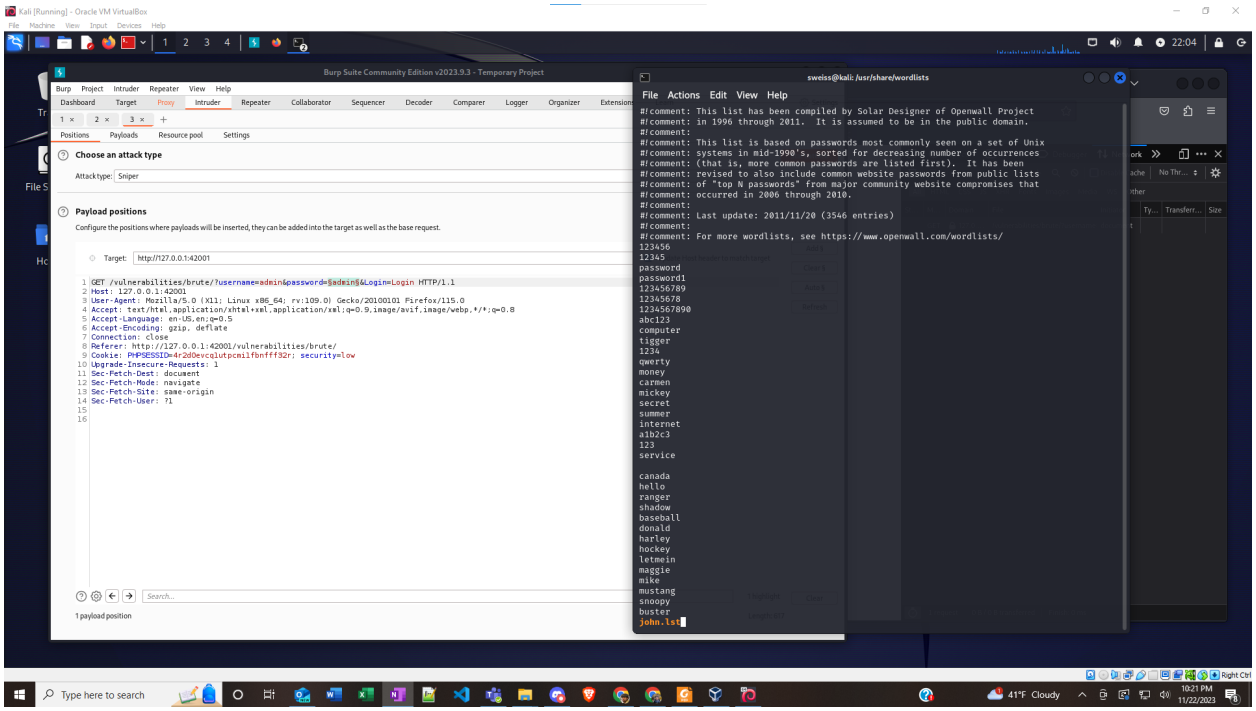

Figure 11: Display contents of john.lst

Step 6: Use john.lst as password source for Burb brute force attack.

Step 7: Order results by “Length”. This displays the successful password at the top.

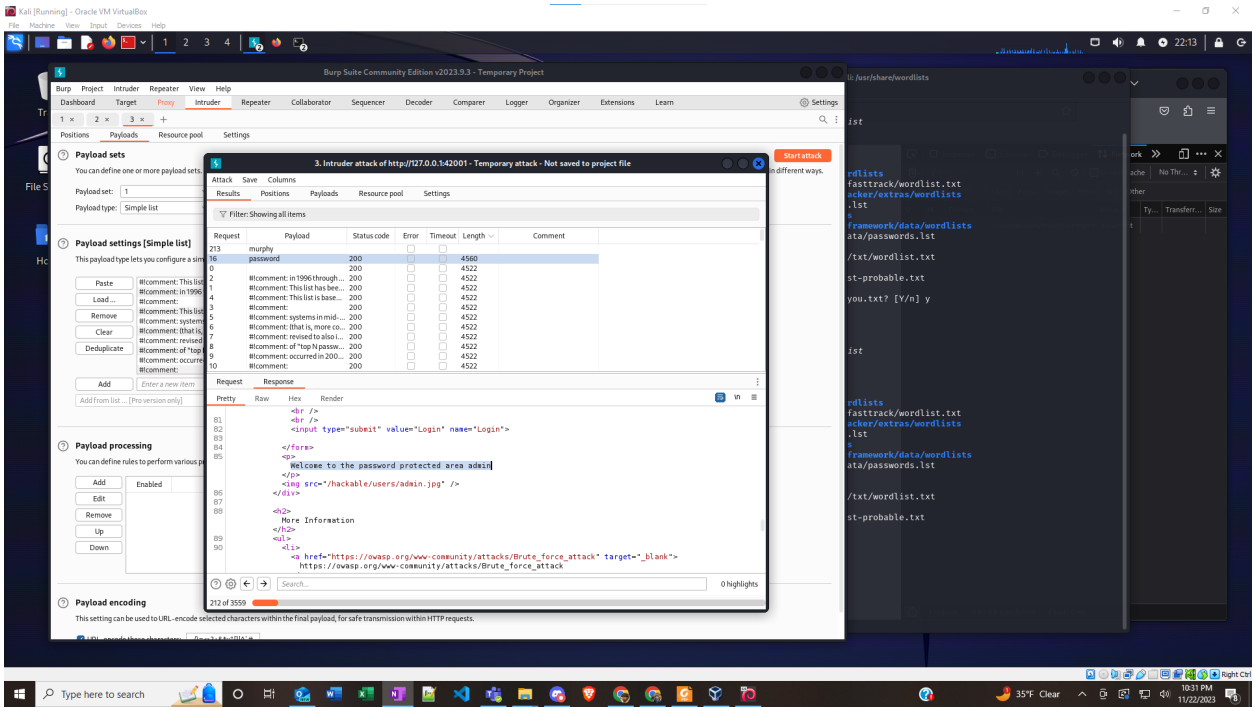

Figure 12: Burp displays a successful password.

Step 8: Stop the brute force attack. Disable ”Intercept” on Burp and attempt to log in with the discovered password.

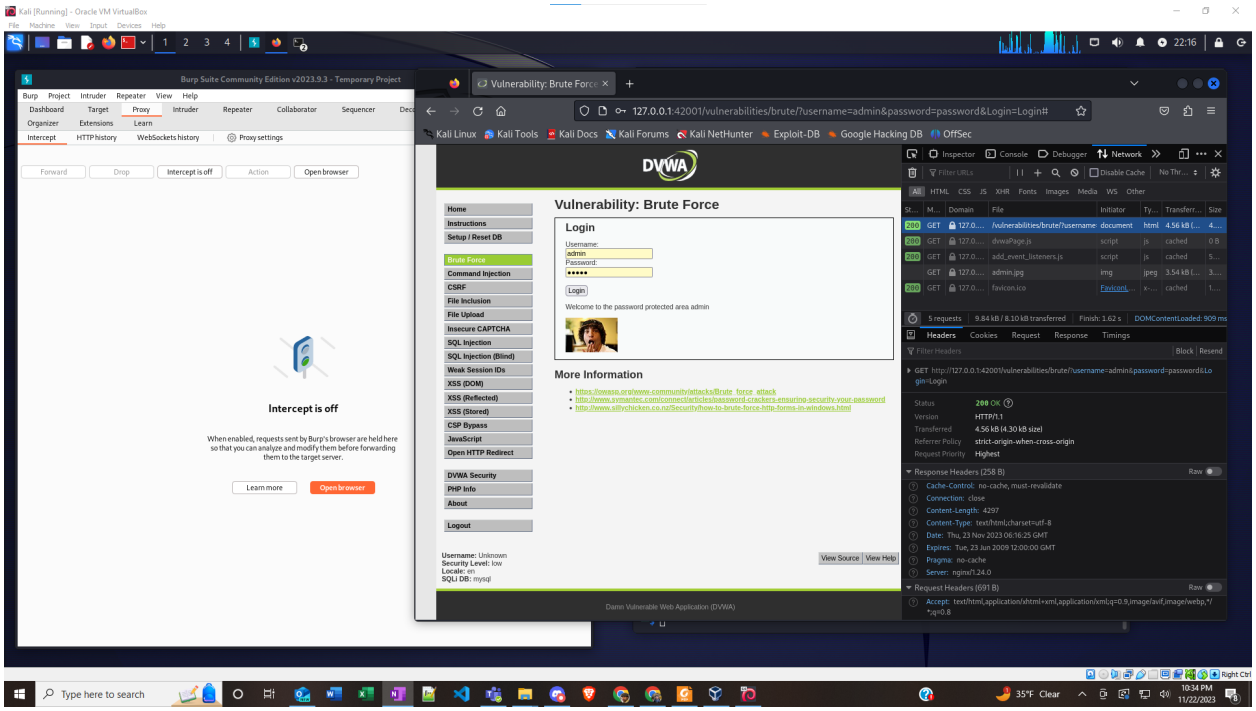

Figure 13: Successful login with username: admin and password: password.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A07:2021 – Identification and Authentication Failures
- A05:2021 – Security Misconfiguration

Resources

- Rezos, G. (n.d.) *Brute Force Attack*. OWASP. https://owasp.org/www-community/attacks/Brute_force_attack
- CavemenTech. (2023, Jan 7). *1- DVWA Brute forcing Walkthrough with Burp and Hydra* [Video]. YouTube. https://www.youtube.com/watch?v=FAzRMqNGScs&list=PL-Fa25Pu8l6xWiqWwStfxxjgdRx0hQCi-&index=4
- Configuring Burp Suite with FoxyProxy
- Jim Schultz. (2023, Feb 1). *Configuring Burp Suite with FoxyProxy* [Video]. YouTube. https://www.youtube.com/watch?v=2UMwYmKhu2w

Vulnerability 2 - Command Injection

How does this feature normally work?

The feature has a text entry where a user can enter an IP address, and a Submit button to initiate the ping command. If an invalid IP address is entered, there appears to be no changes to the page in the response. If a valid IP address is entered, the results of the ping command are displayed in the response. See Figure 14.

What does it take to exercise the vulnerability?

On the Low security level, any linux operator to tag on a second command can be used to inject the command in the text entry after the IP. The source code concatenates the user entry to the ping command (see the source code in Figure 15). For example, entering the IP address 192.168.0.211 && ls -al returns the results of the ping command as well as the results of ls (see Figure 16).

How did the feature work differently than normal use?

In addition to displaying the results of the ping command, the results of whatever other command is also displayed. For example, if the `ls -al` is tagged on to the end of the IP, the results of `ls -al` are displayed on the application page response after the ping results.

Why did this work differently?

This altered functionality is a result of the source code simply executing ping on whatever it is passed. If it is passed the correct syntax to initiate a second command after ping, it does so.

Why we should care about this vulnerability and potential loss

We should care about this vulnerability because the commands that are executed after ping are done so with the privileges of the application. If the application has root privileges, then it can execute any command on any file, potentially exposing sensitive information. It can also be used to initiate a remote shell for further exploitation at the privilege level of the application.

Briefly describe how to fix the vulnerability

Rather than adding forbidden characters such as “&&” or “;” as is done in the Medium and High security levels, the best practice is to “whitelist” characters. The impossible security level demonstrates this by parsing the input string and checking each chunk to make sure it contains only int values. It then puts the IP back together and runs the ping command on it. The links below provide further information.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Enter a valid IP address.

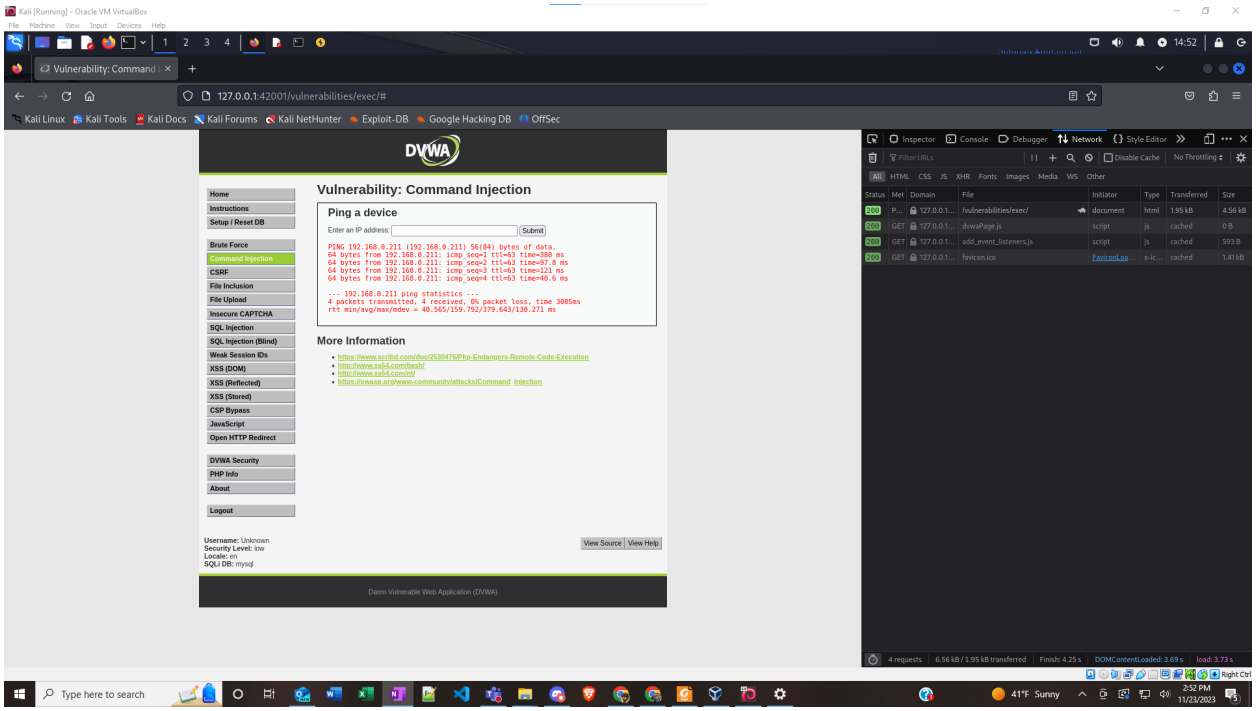

Figure 14: Response from a valid IP address.

Step 2: View the source code.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

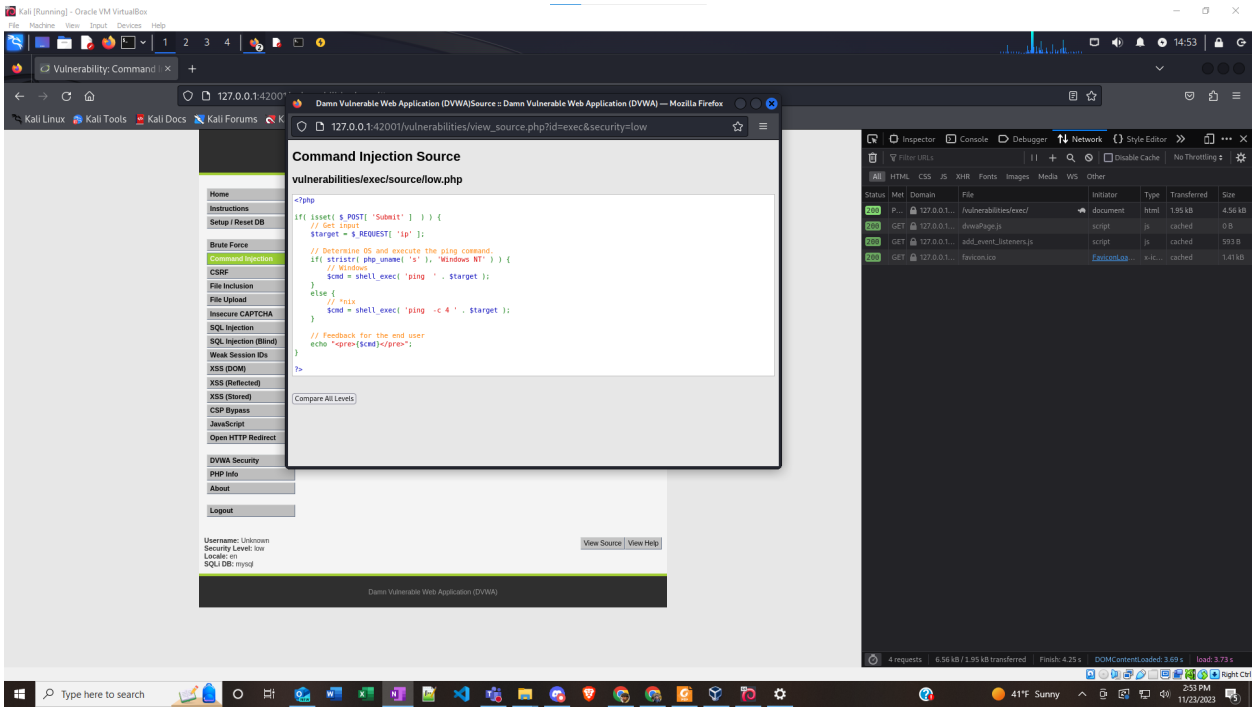

Figure 15: Source code of Low security mode.

Step 3: Enter the valid IP address again, but tag on an additional linux command.

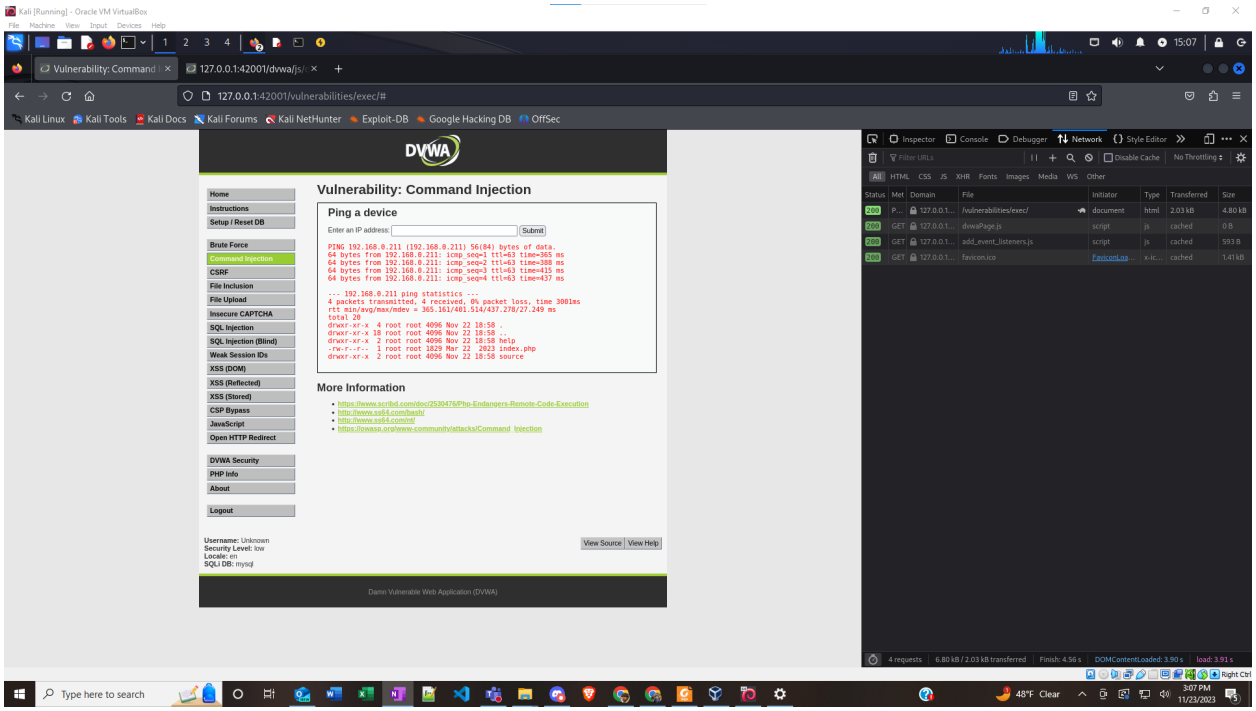

Figure 16: Results of entering 192.168.0.211 && ls -al

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Step 4: Use the injection technique to display the contents of /etc/passwd

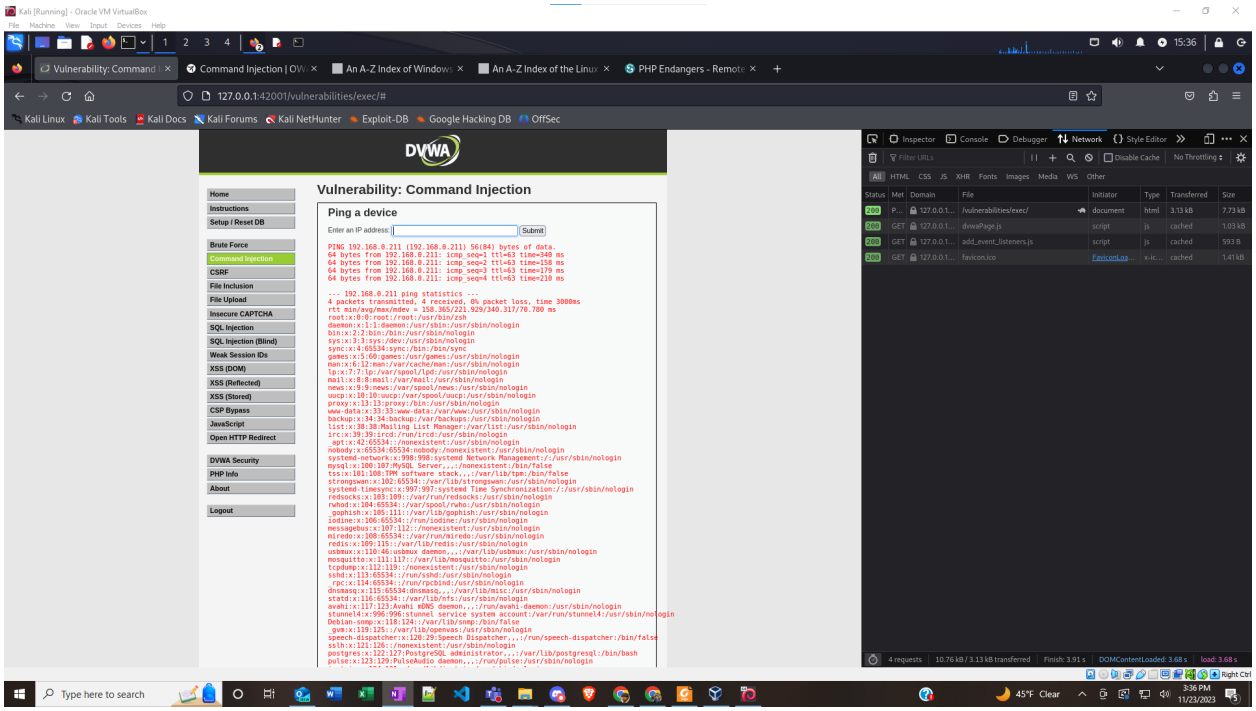

Figure 17: Contents of /etc/passwd displayed via command injection.

Step 5: Use command injection to open a remote shell (Figures 18-19)

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

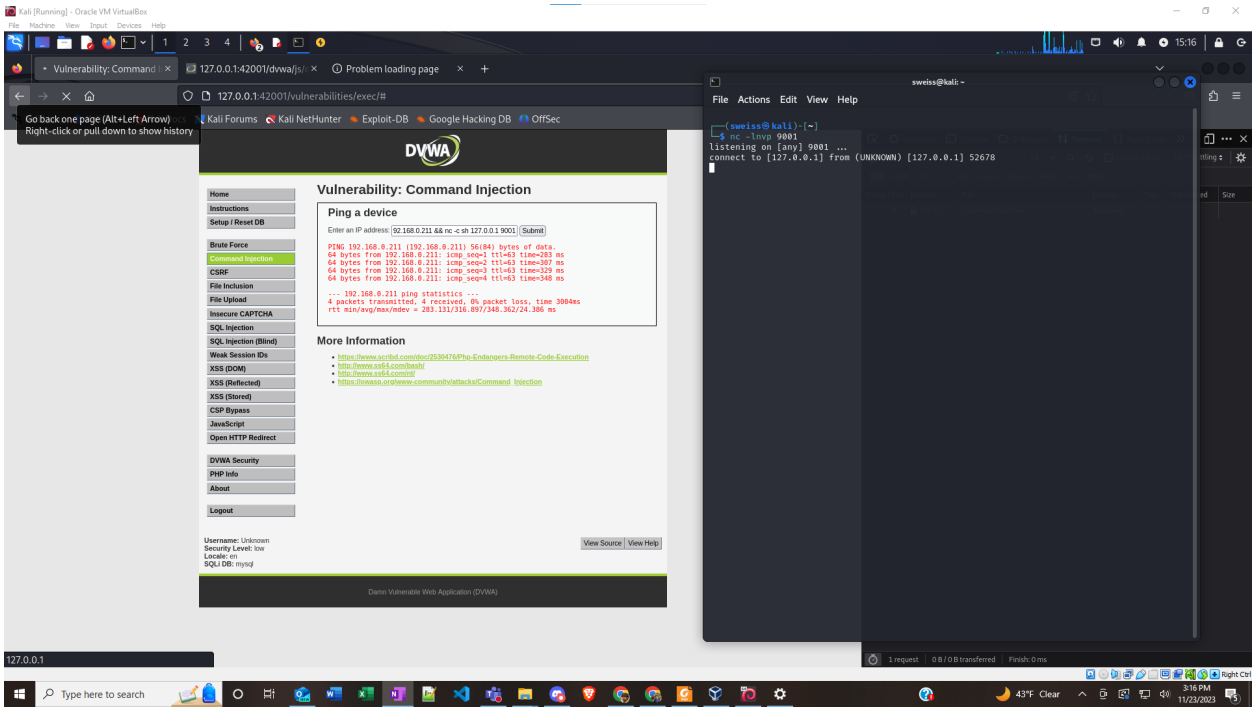

Figure 18: Utilize command injection to initiate a remote shell.

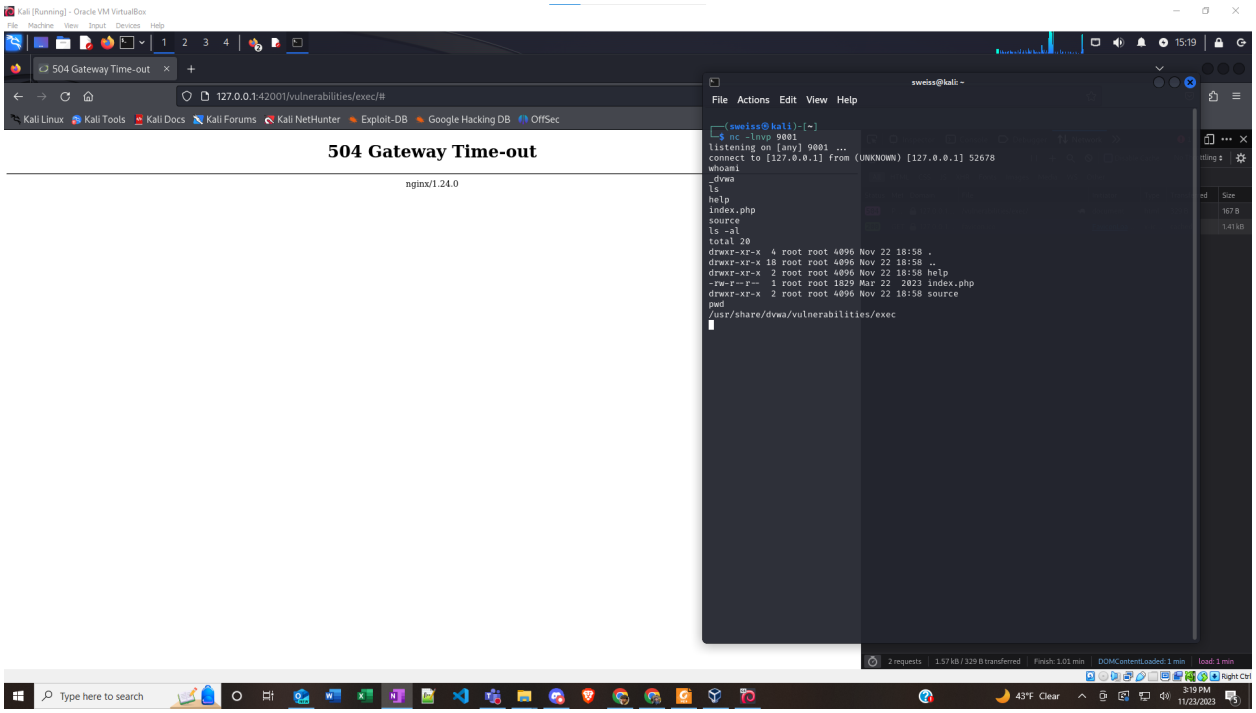

Figure 18: Use the remote shell to explore the file system as the application user (_dvwa). Note that the web page times out but the shell is still active.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection

Resources

- CavemenTech. (2023, Jan 10). *2. Command Execution Walkthrough DVWA | Web security Basics for Beginners* [Video]. YouTube. https://www.youtube.com/watch?v=bEWQevp7ZO4&list=PL-Fa25Pu8l6xWiqWwStfxxjgdRx0hQCi-&index=5
- Zhong, W. (n.d.). *Command Injection*. OWASP. https://owasp.org/www-community/attacks/Command_Injection

Vulnerability 3 - CSRF

How does this feature normally work?

The page displays a Test Credentials button which opens a new window when clicked. The user can enter a username and password and check if it is valid (see Figure 21). Underneath the button are entry fields to change the password. If the two entries for the new password match, the message “Password Changed” is displayed (see Figure 22). The new password is displayed in the URL query string. The user must then log out for the password change to take effect. This is done by clicking the Logout button at the bottom left of the application. The User can then log in using the updated password (Figure 25). The Test Credentials button will then reflect the password change.

What does it take to exercise the vulnerability?

Save the query string, either copied from the URL address bar or retrieved from the Inspector window. This Query string can now be added to a link or image tag that can be used to phish a user. For this to work, the user must already be logged into the site when they click on the link. The link then sends the password change request to the site recognizing that it is from the legitimate user.

How did the feature work differently than normal use?

The password will change based on the URL query string rather than the entry form.

Why did this work differently?

This works because there is no encryption in the query string, and it is displayed in plain text. Note: this can also be done via POST request by including the necessary information in the request body.

Why we should care about this vulnerability and potential loss

This vulnerability can allow a phished user to unknowingly change their password to a value that the attacker knows but the user does not. The attacker can then take control of the user account. If the account contains sensitive information such as credit card numbers, that information is now in the control of the attacker rather than the user.

Briefly describe how to fix the vulnerability

Browsers have made steps to add a nonce to the URL and forms, acting as a non-predictable challenge token to requests. This helps to verify that the request came from the expected page/form. The token should be attached to a session cookie with a limited time of validity. Forms should also be hashed for additional security. Password changes should also require entry of the old password. The links below provide further information.

A step the user can take to fix this vulnerability is to always remember to log out when done using a website or application.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Read the content of the View Help and View Source Code buttons.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

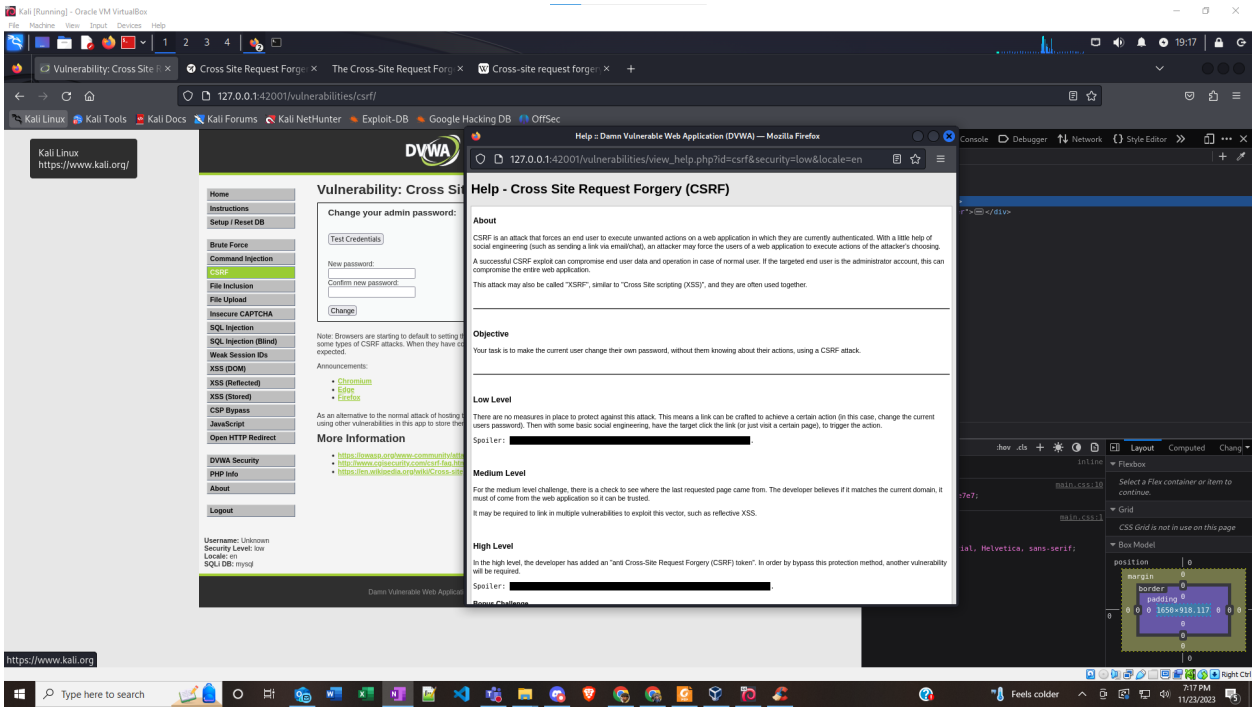

Figure 19: CSRF View Help button content.

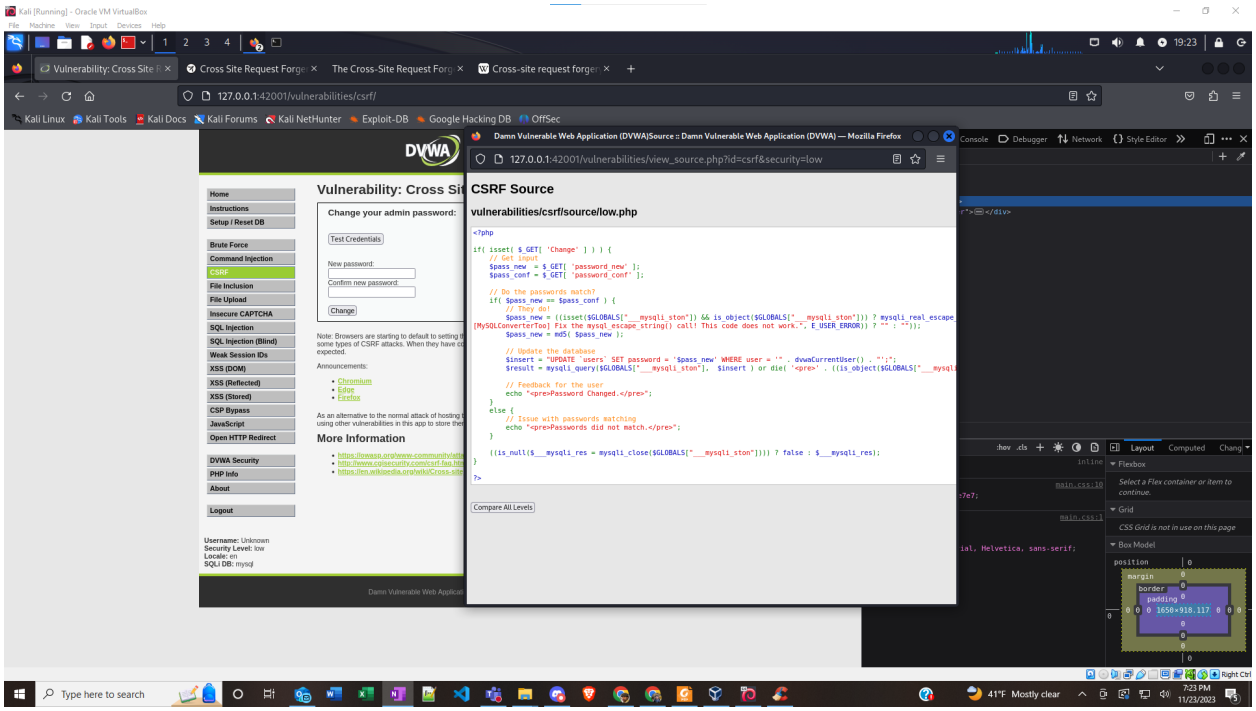

Figure 20: CSRF View Source Code button content.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Step 2: Explore the intended features. Verify the current password, change the password, and verify the changes.

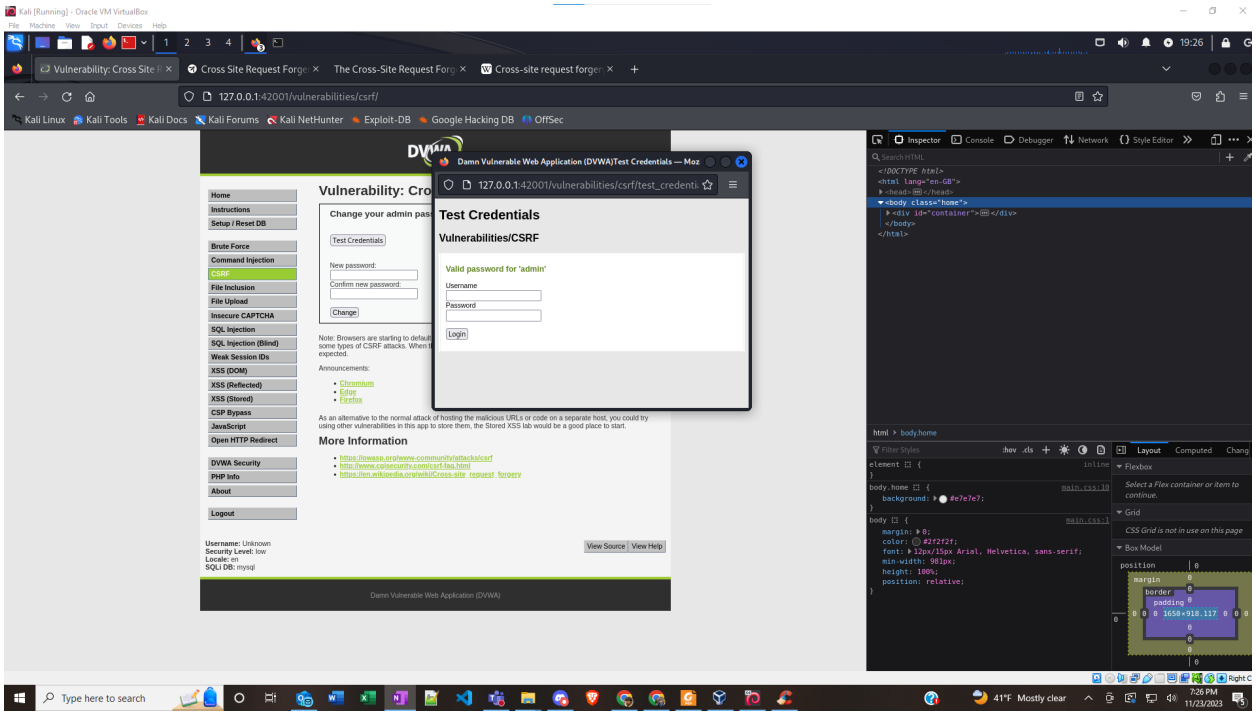

Figure 21: Verify Credentials

Step 3: Get the URL query string from the password change.

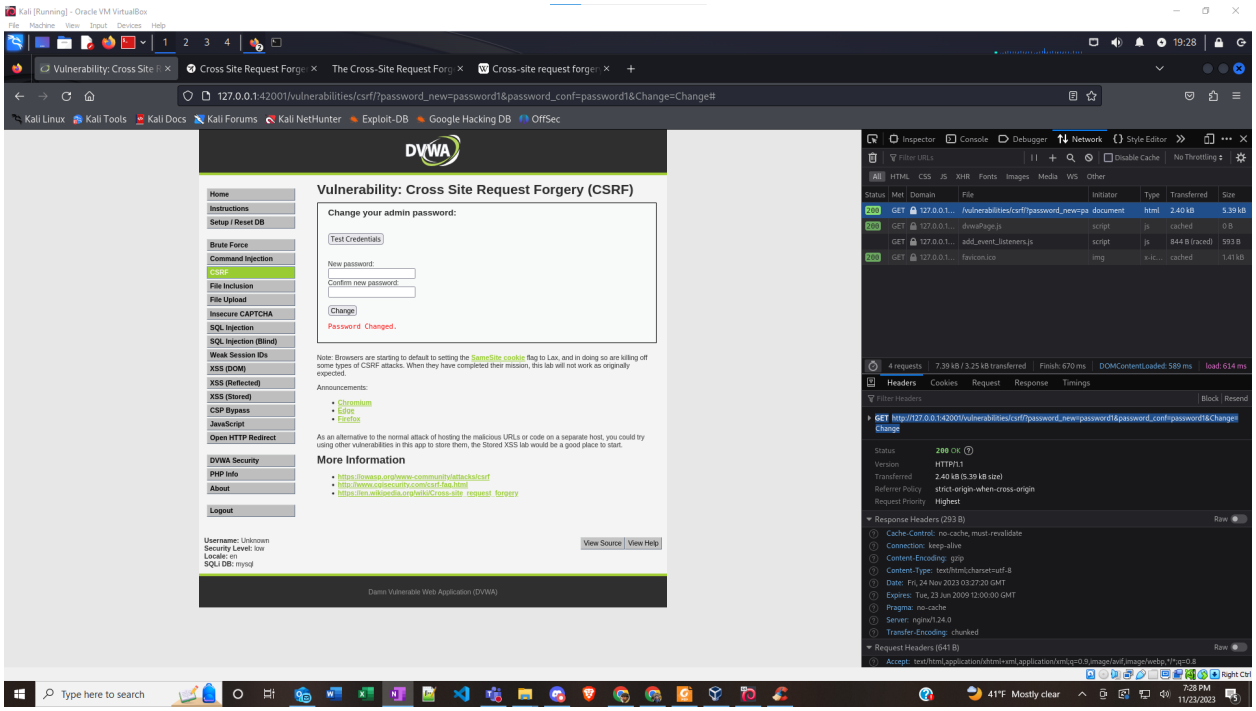

Figure 22: The new password is displayed in the URL query string.

Step 4: Create a new query string with a new password and embed it in a phishing link. In this example, a link on the application page was temporarily modified using the edit feature of the Inspector window. In a real-life scenario, this link would be stored in a more permanent location such as an attacker's website, or sent to a user via email.

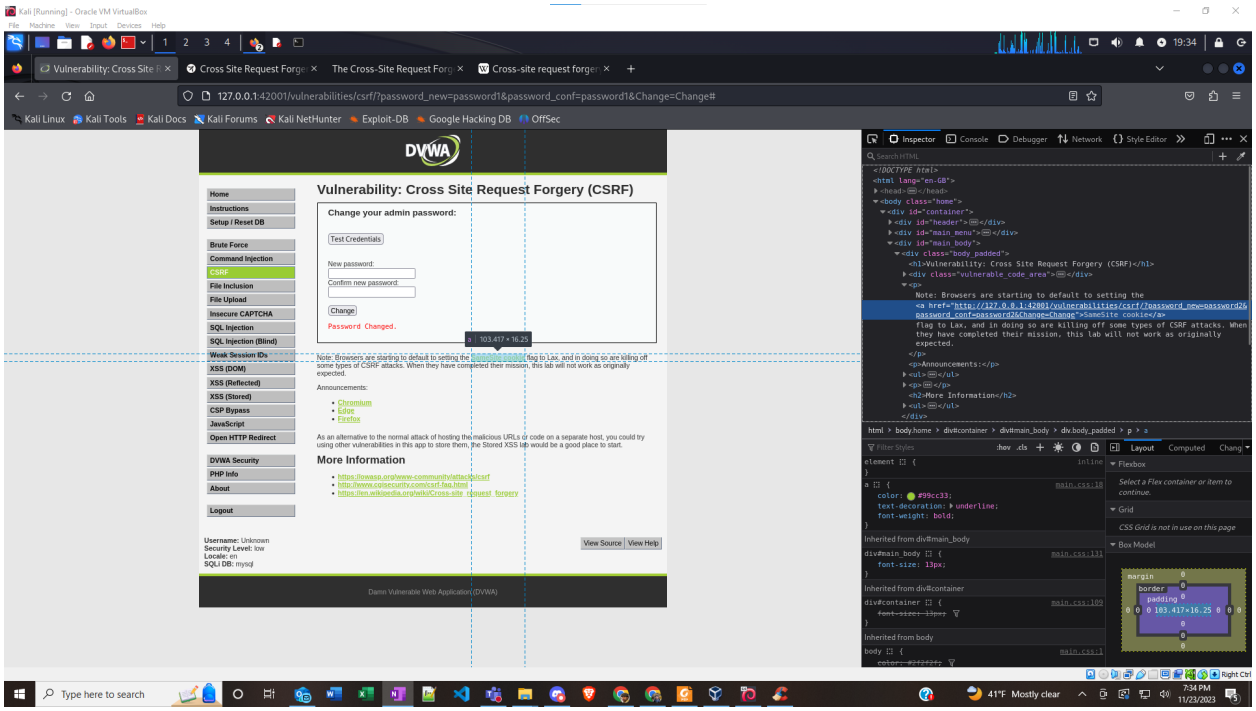

Figure 23: Create a malicious link with the password change query string.

Step 5: Click on the link, confirm the password change in the address bar. Log out of the application and log in with the new password.

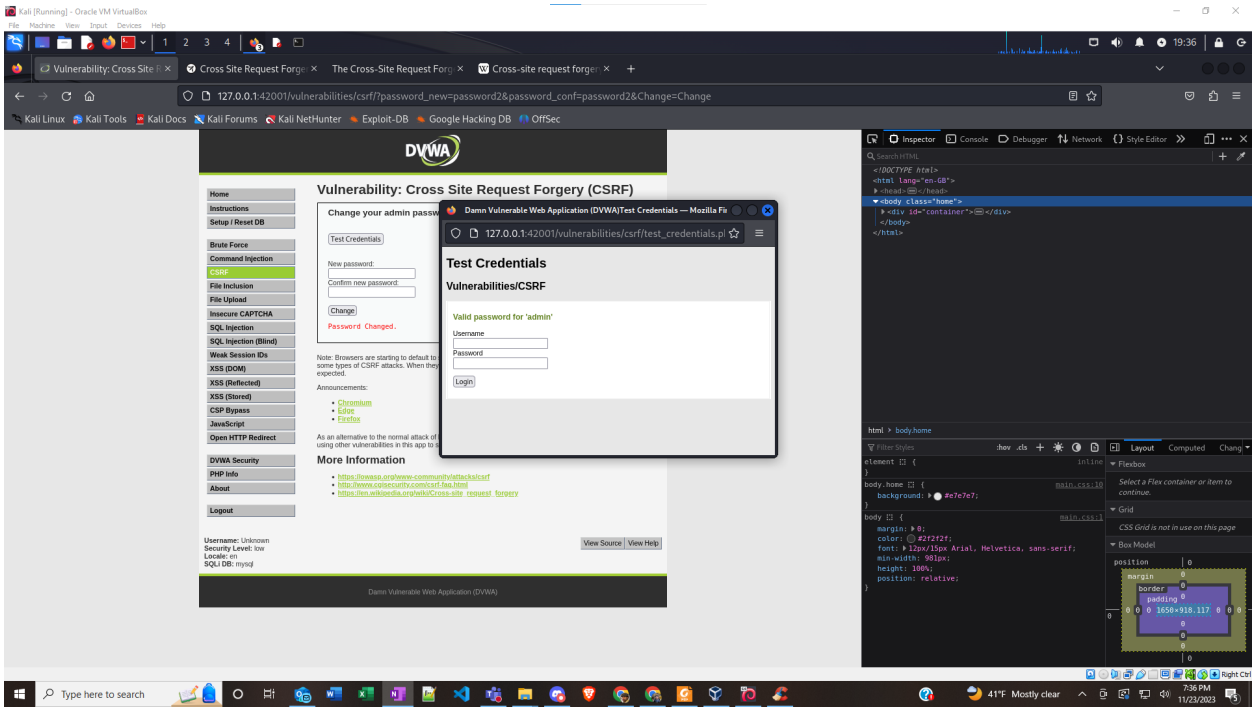

Figure 24: Password change confirmed in the address bar.

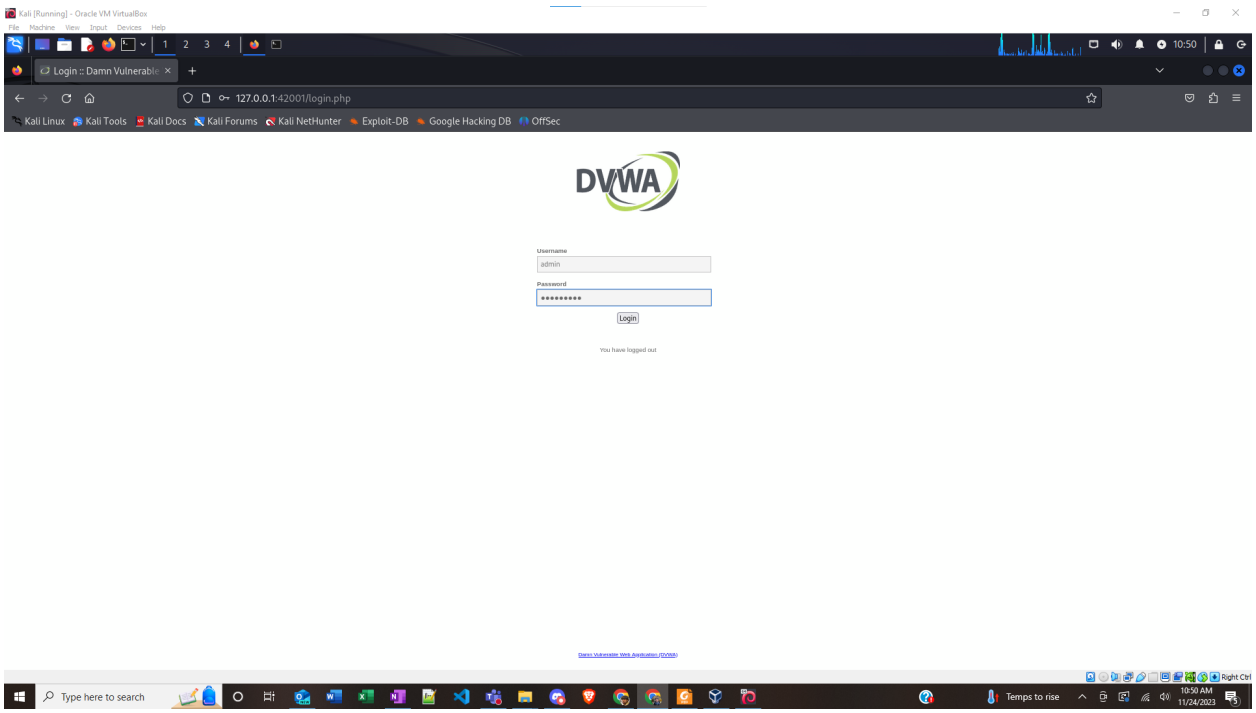

Figure 25: New password used to log in.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A01:2021 – Broken Access Control

Resources

- KirstenS. (n.d.). *Cross Site Request Forgery (CSRF)*. OWASP. https://owasp.org/www-community/attacks/csrf
- Auger, R. (2010, April 4). *The Cross-Site Request Forgery (CSRF/XSRF) FAQ*. Cgisecurity. https://www.cgisecurity.com/csrf-faq.html
- CavemenTech. (2023, Jan 12). *3. CSRF DVWA Low Difficulty Walkthrough | Web Security for Beginners [Video]*. YouTube. https://www.youtube.com/watch?v=Z_xH751F5Lw&list=PL-Fa25Pu8l6xWiqWwStfxxjgdRx0hQCi-&index=6

Vulnerability 4 - File Inclusion

How does this feature normally work?

The application page has three file links: file1.php, file2.php, and file3.php. Each link can be clicked and the page then displays the contents of that file. The file is included in the URL query string. A back button can be clicked to return to the page with the file listings.

What does it take to exercise the vulnerability?

Enter the location of a known file in the URL query string.

How did the feature work differently than normal use?

The unintended file is displayed on the page via URL rather than via a link.

Why did this work differently?

The file include returns a file that is being hosted on the server, whether or not that file was intended to be displayed. The simple nature of the request/response protocol works such that if a file is requested, and that file exists, it will be included in the response.

Why we should care about this vulnerability and potential loss

If a file with sensitive data is stored on the server without any permissions checks or authentication, and an attacker knows of the existence of this file, it can be requested in a GET request via the URL address bar. Additionally, the request can also be used for remote file inclusion, a scenario in which the attacker includes the location of a malicious file hosted on a

different server. The server processing the request then retrieves this file and includes it in the response. This malicious file could contain something like a reverse shell which would then give the attacker access to the server's file system.

Briefly describe how to fix the vulnerability

The best way to fix this vulnerability is to avoid passing user-submitted input into any file system, framework, or API. Another method is checking inputs for filenames, and only allowing requests for the intended files. As described in the View Help button, this is done in the DVWA Impossible mode by hard-coding the files intended for viewing.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore the intended functionality.

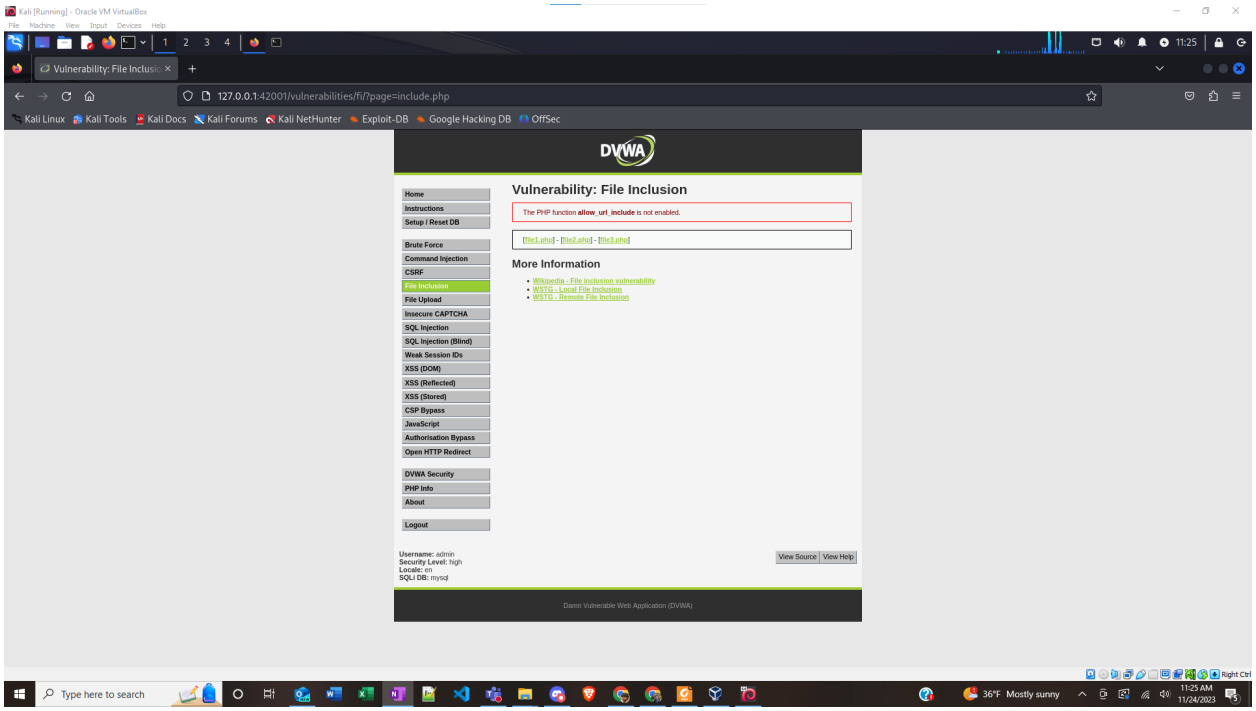

Figure 26: File Inclusion main page.

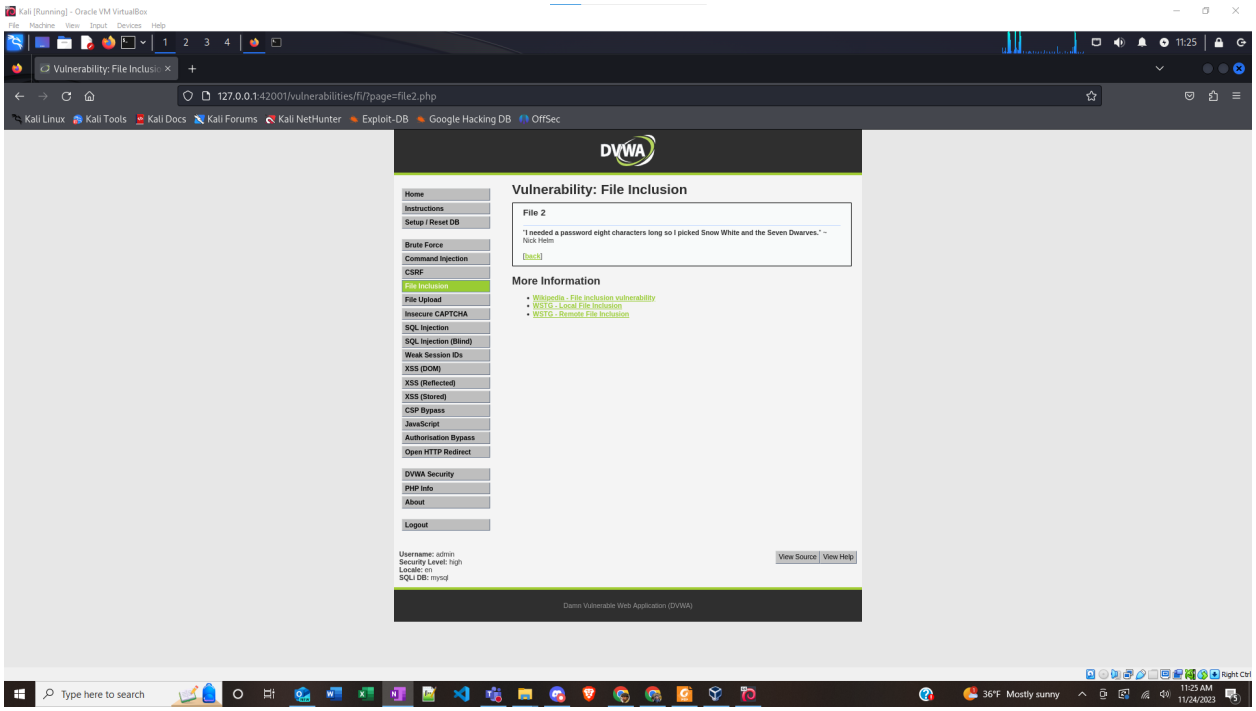

Figure 27: File 2 as requested via link.

Step 2: The target file ../hackable/flags/fi.php was not displayed with the first attempt. A reverse shell was used in the Command Injection challenge to browse the file system for it.

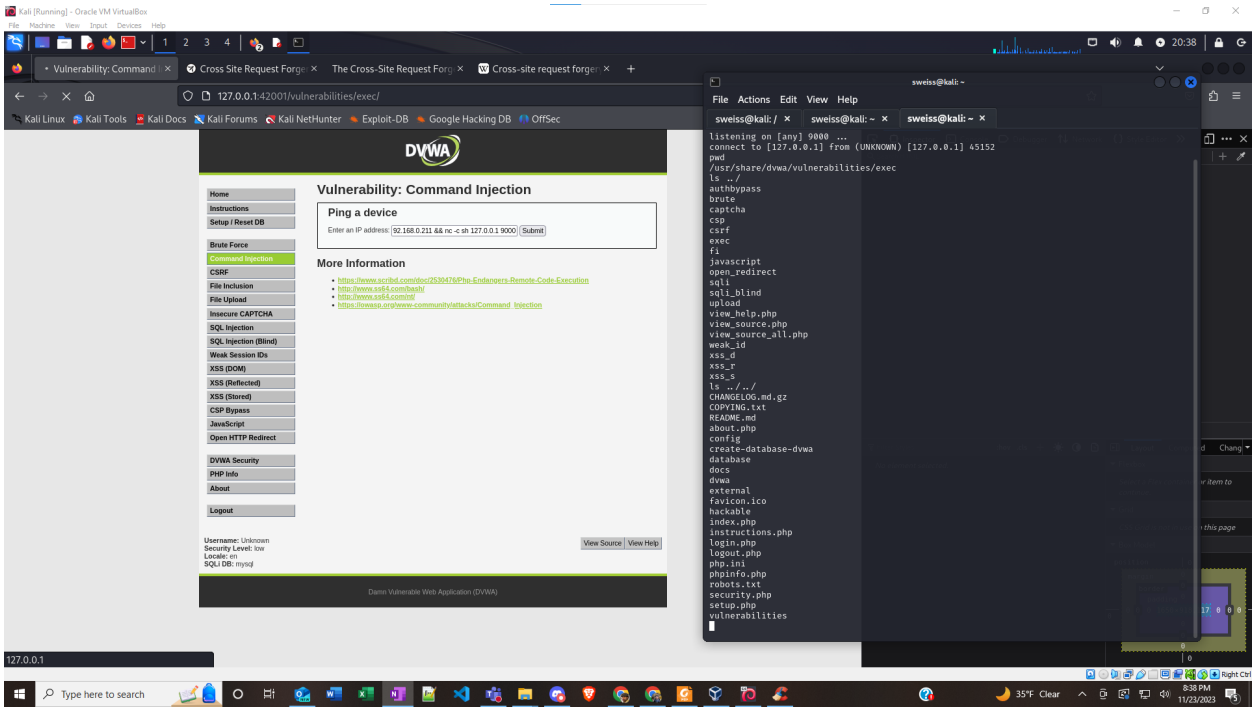

Figure 28: Using a reverse shell from the Command Injection challenge to locate the target file path.

Step 3: Enter the correct file path (../../hackable/flags/fi.php) in the URL query string with the Inspector tab open. The Inspector tab reveals one of the target quotes is commented out and therefore is not displayed on the page with the others.

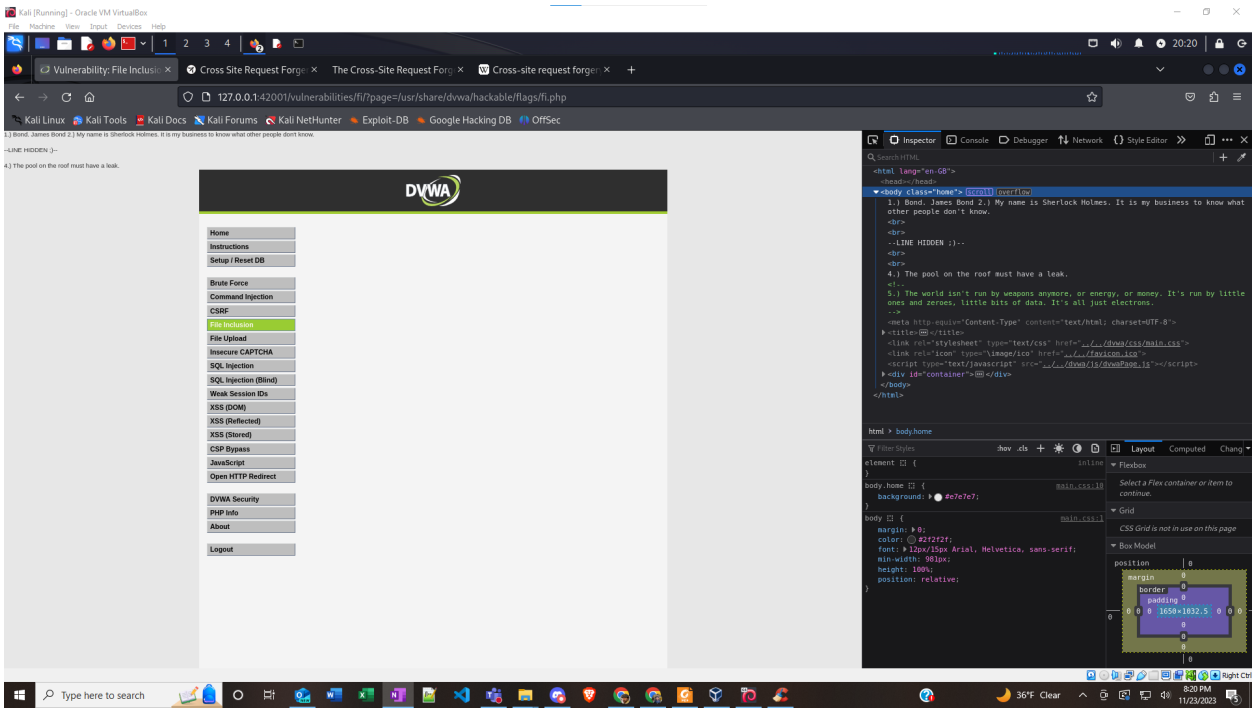

Figure 29: Successful retrieval of target file through file inclusion method.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection

Resources

- (n.d.) *Testing for Remote File Inclusion*. OWASP https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.2-Testing_for_Remote_File_Inclusion
- Mark R. (2020, January 12). *Challenge 04: File Inclusion.md*. GitHub. https://github.com/keewenaw/dvwa-guide-2019/blob/master/low/Challenge%2004%3A%20File%20Inclusion.md
- CavemenTech. (2023, February 5). *4. File Inclusion DVWA Low Difficulty Walkthrough | Web Security for Beginners* [Video]. YouTube. https://www.youtube.com/watch?v=yR5eu8SCEdg&list=PL-Fa25Pu8l6xWiqWwStfxxjgdRx0hQCi-&index=7

Seth Weiss
weissse@oregonstate.edu

Vulnerability 5 - File Upload

How does this feature normally work?

This feature has two buttons. One allows the user to select a file (such as an image) to upload. The other allows the user to upload that file. On successful upload, there is a confirmation message displayed with the path to the file. If the user then goes to that URL, the file contents is displayed.

What does it take to exercise the vulnerability?

The Low security level has no restrictions on what type of file the user can upload. This allows the user to upload a file containing php code that invokes the php method `phpinfo()` to display system information, or code to initiate a reverse shell.

How did the feature work differently than normal use?

If a file containing code to initiate a reverse shell is uploaded, the feature will act the same as any other file upload by displaying the confirmation message. However, when the user now goes to the URL path, instead of displaying the image of a cat, for example, the application will instead hang in a loading phase and eventually display a 503 error. Meanwhile, the reverse shell is active and the attacker can explore the hosting server's file system.

Why did this work differently?

Because there are no restrictions on what type of file can be uploaded, a user can upload files containing php code. The application will include the file contents in the response, unknowingly allowing the activation of a shell or display of sensitive information.

Why we should care about this vulnerability and potential loss

We should care about this vulnerability because if there is sensitive data on the hosting server, an attacker would now have access to it. This sensitive data could include credit card information, names and passwords or user accounts, company trade secrets, etc.

Briefly describe how to fix the vulnerability

The View Help button on the application explains a few different measures to fix this vulnerability. Restricting file types is one measure, but does not fully eliminate the problem. An application can also resize any image that was included in the request. This, however, is also not a complete solution. The Impossible level re-encodes the image file which strips out any "non-image" code, including metadata, and creates a new image file.

Other measures according to OWASP include limiting file size, only allowing authorized users to upload files, storing files on a different server or in a location other than webroot, and run the file through an antivirus or sandbox on upload.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality.

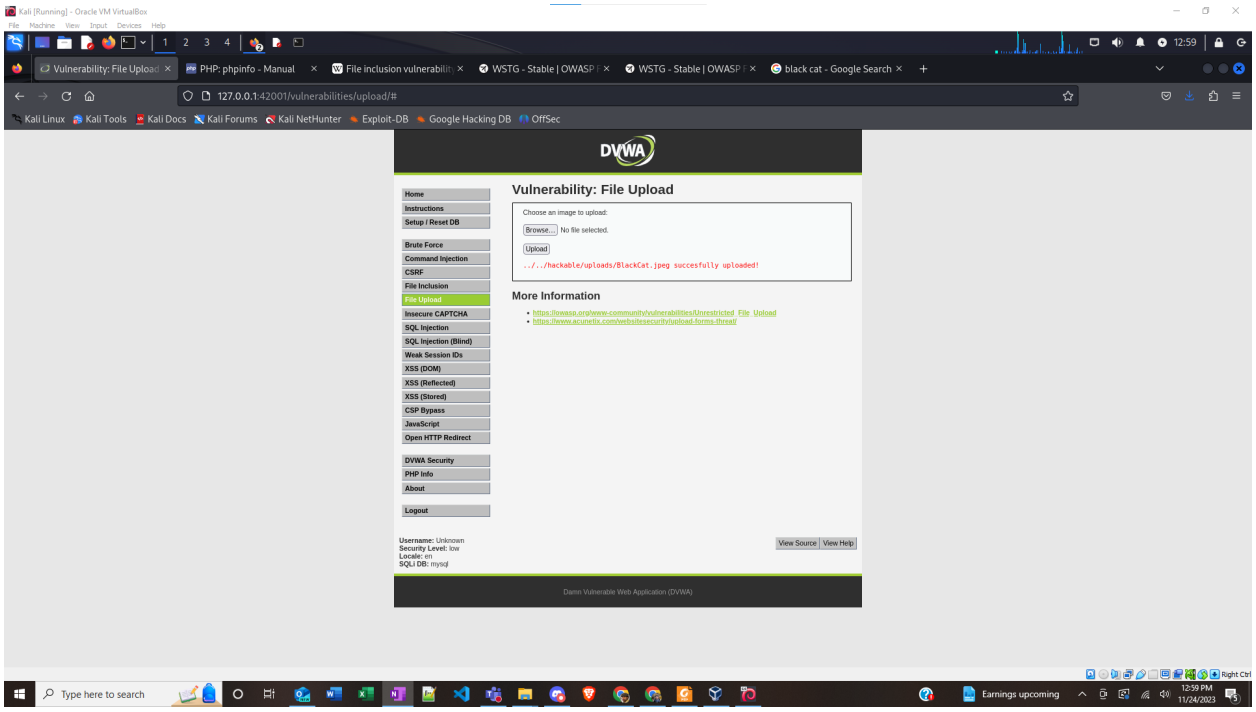

Figure 30: Confirmation of uploaded image file.

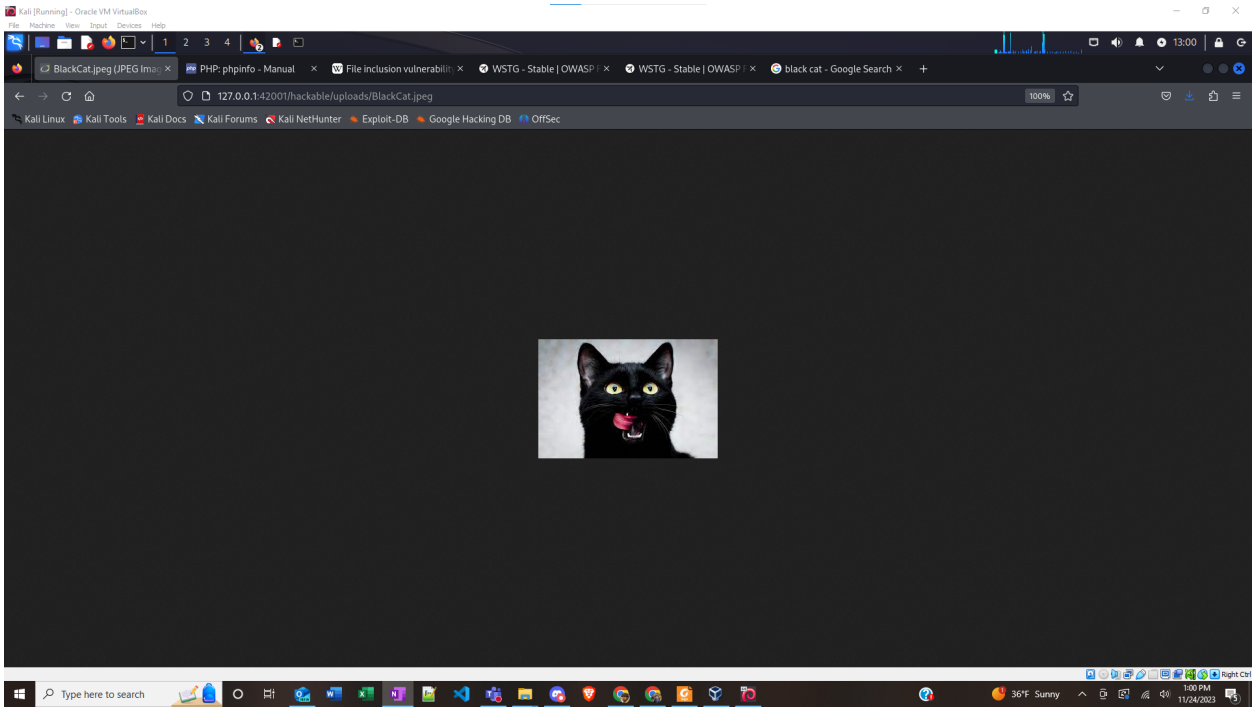

Figure 31: Uploaded image file (intended use).

Step 2: Create and upload a php file that evokes `phpinfo()`.

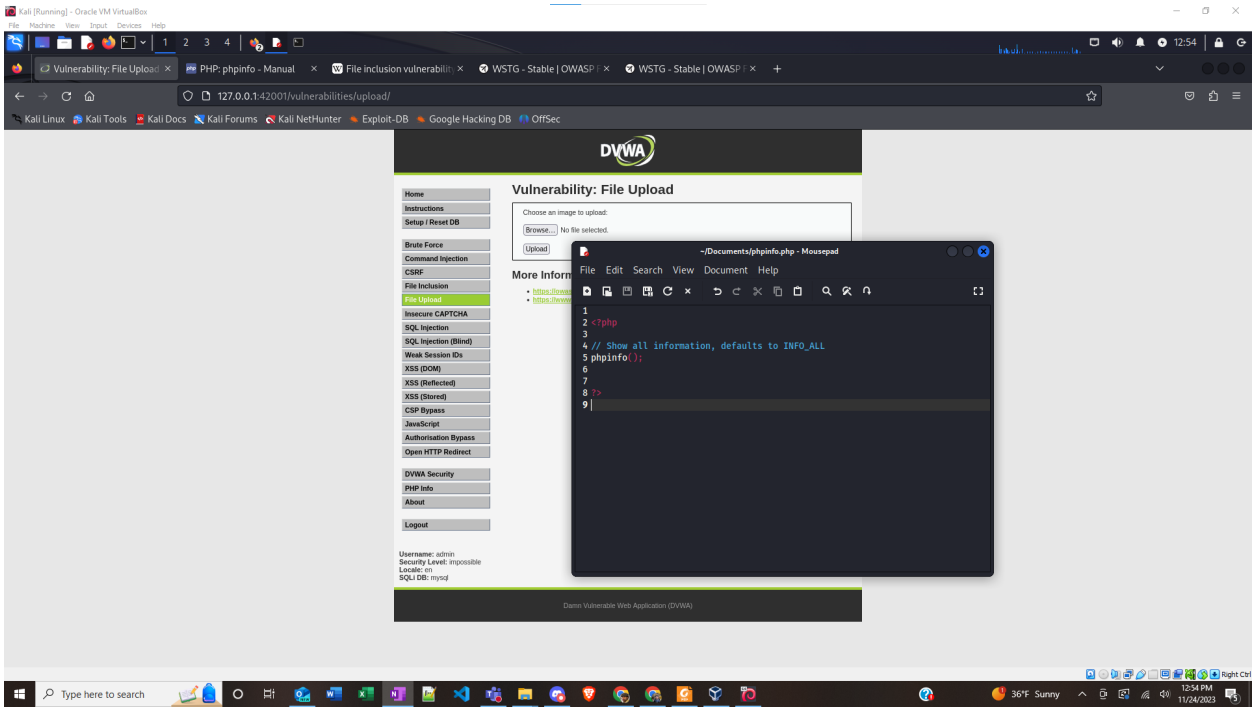

Figure 32: A php file to exploit the vulnerability.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

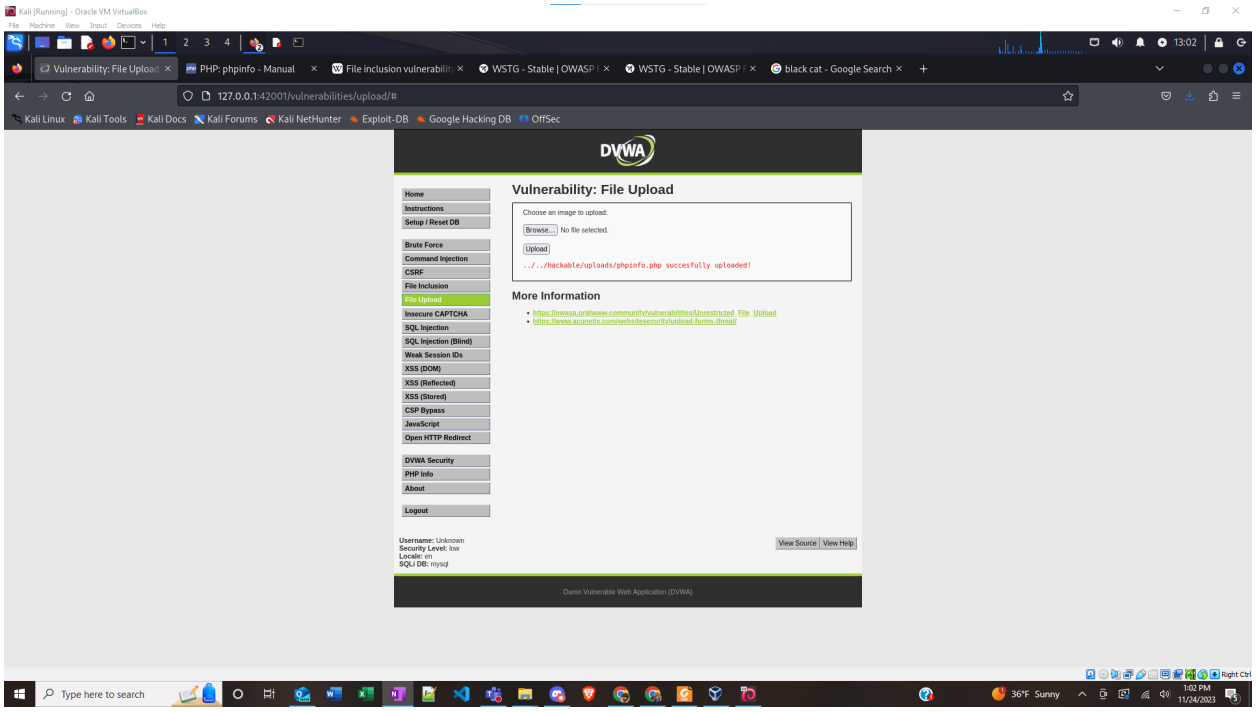

Figure 33: Confirmation message for php file upload.

Step 3: View php file location URL. The `phpinfo()` method is called and the page displays system information.

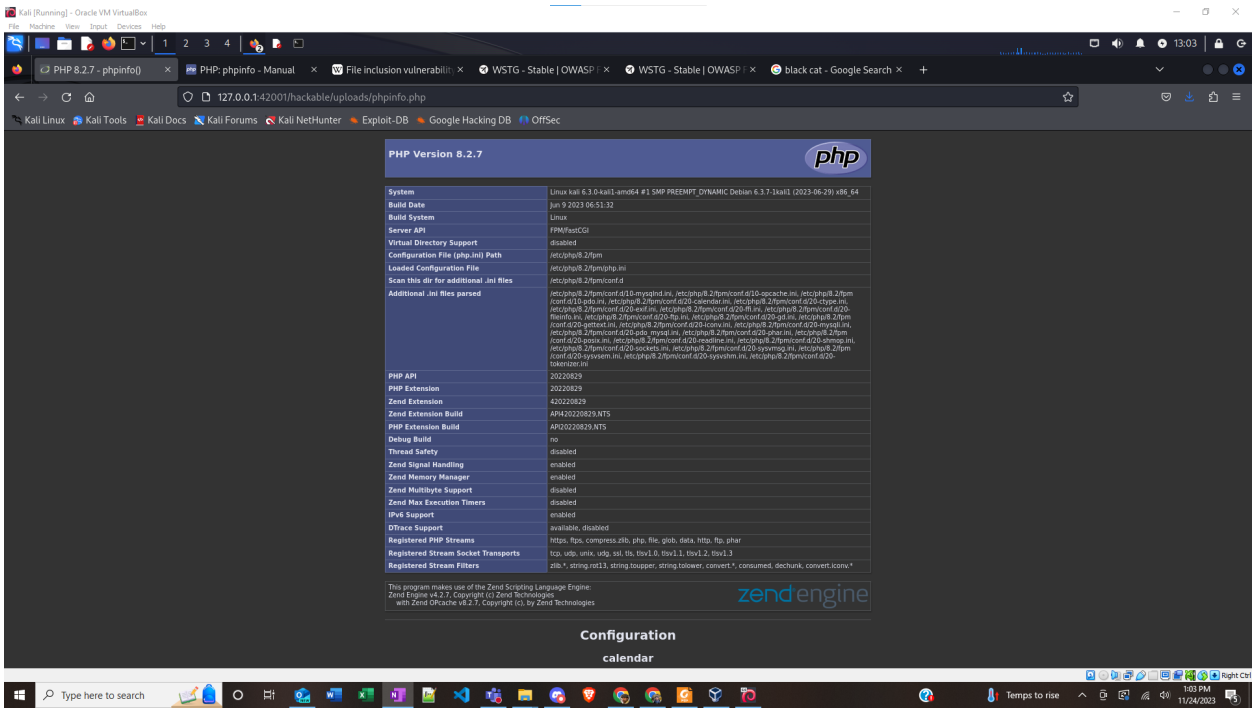

Figure 34: `phpinfo()` displays the system information.

Step 4: Upload a php file initiating a reverse shell. Confirm successful initialization with the `whoami` and `pwd` commands.

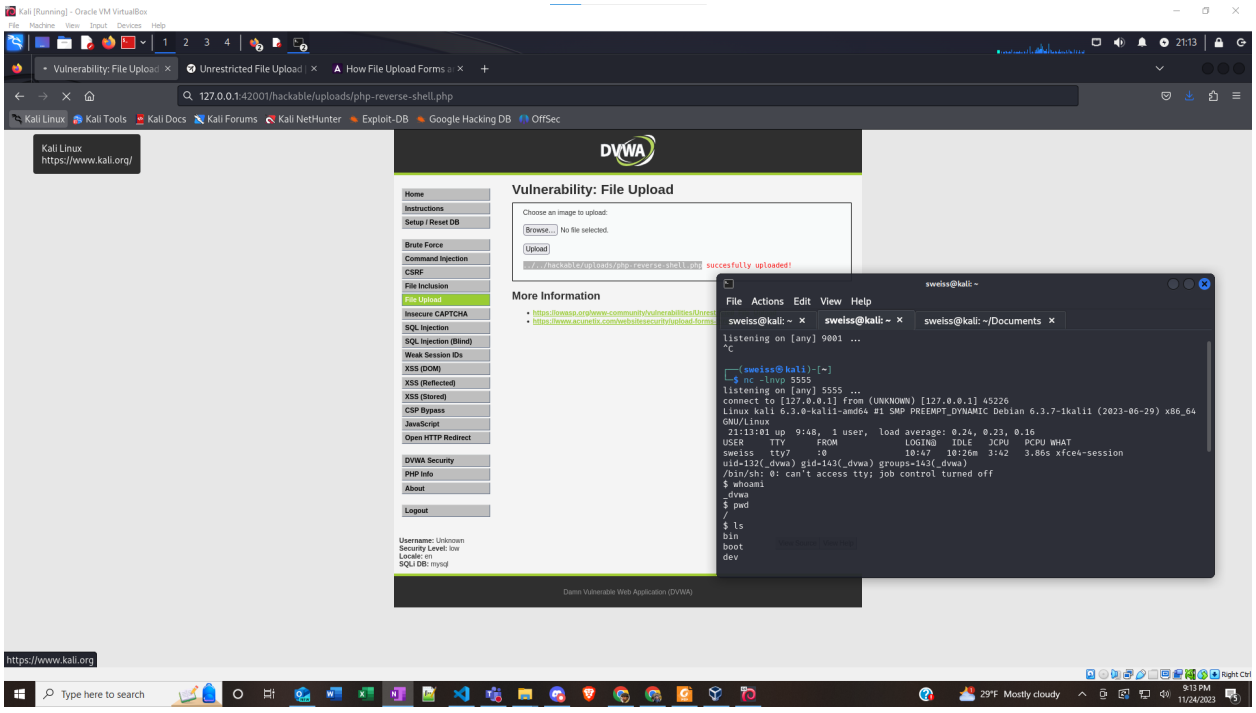

Figure 35: Reverse shell initiated through file upload.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A04:2021 – Insecure Design

Resources

- (n.d.) *File Upload Cheat Sheet*. OWASP. https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html
- Dalili, S. et.al. (n.d.). *Unrestricted File Upload*. OWASP. https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- (n.d.) *Why File Upload Forms are a Major Security Threat*. Acunetix. https://www.acunetix.com/websitesecurity/upload-forms-threat/
- Mark R. (2020, January 12). *Challenge 05: File Upload.md*. GitHub. https://github.com/keewenaw/dvwa-guide-2019/blob/master/low/Challenge%2005%3A%20File%20Upload.md

- CavemenTech. (2023, March 26). *6. Exploiting File upload Vulnerabilities DVWA walkthrough* [Video]. YouTube. https://www.youtube.com/watch?v=GtZ_7GUXVJ8&list=PL-Fa25Pu8l6xWiqWwStfxxjgdRx0hQCi-&index=11

Vulnerability 6 - SQL Injection

How does this feature normally work?

On the Low security level, this feature has a text entry box with a submit button. If the user enters the correct ID for any user registered in the database (1 through 5), that user's ID, FIsrt name, and Surname are displayed on the screen. If the user enters something other than a registered ID, then nothing is displayed.

What does it take to exercise the vulnerability?

To exercise this vulnerability, it takes a bit of knowledge of SQL queries and some guessing/trial and error to determine the table and column names in the database as well as what security checks may or may not be in place.

How did the feature work differently than normal use?

With the correct syntax, an attacker can retrieve other information from the database and it will be displayed on the screen (seen figures below). This can contain information about how the database is structured or it can contain other information inside the database such as passwords.

Why did this work differently?

If the SQL query uses a raw input that directly queries the database, the attacker can use SQL syntax to trick the query into adding additional commands by closing the input string early by entering a quotation mark `'`. Additionally, the attacker can add an SQL comment `--` or `#` at the end of the entry to cancel out the rest of the intended query.

Why we should care about this vulnerability and potential loss

This is a potentially destructive vulnerability. Not only can it allow an attacker access to sensitive information such as passwords or credit card numbers, but an attacker can also modify the database in this way. This can take the form of entering unauthorized information, or removing entire tables from the database.

Briefly describe how to fix the vulnerability

The Impossible level on the DVWA app solves this problem by parameterizing the queries. This means that instead using the user input to form the query, the query has been done by the developer and the user input is added to it after the fact. There is also a value check on the input to make sure that it is an integer and not an expression.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality.

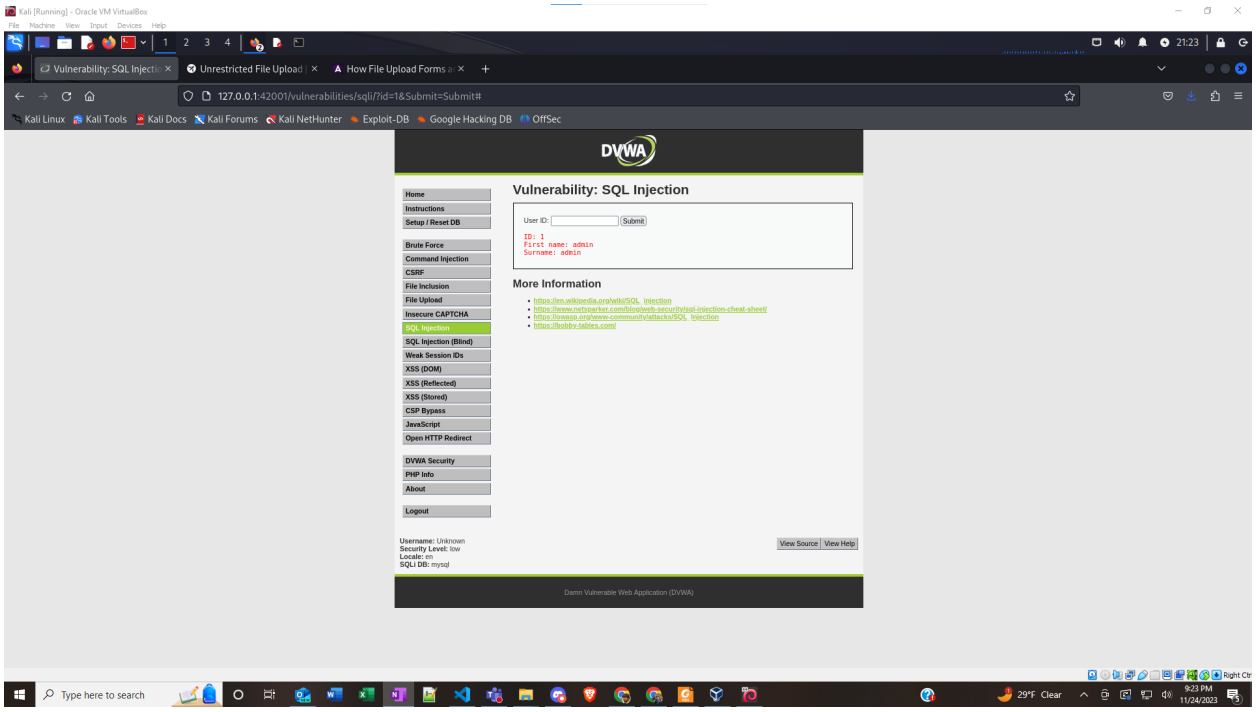

Figure 36: Intended functionality. User enters the User ID “1” and the information is displayed.

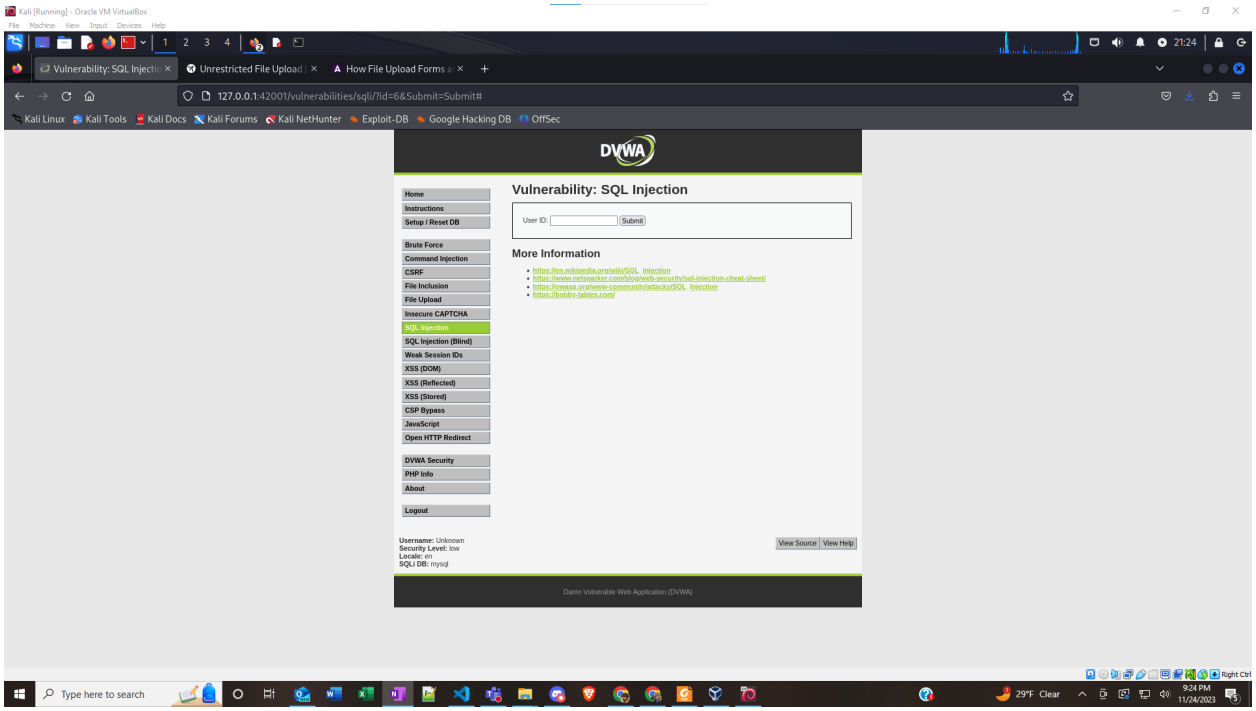

Figure 37: Unrecognized user ID returns no response.

Step 2: Try the classic SQL injection command `*' OR '1'='1`.

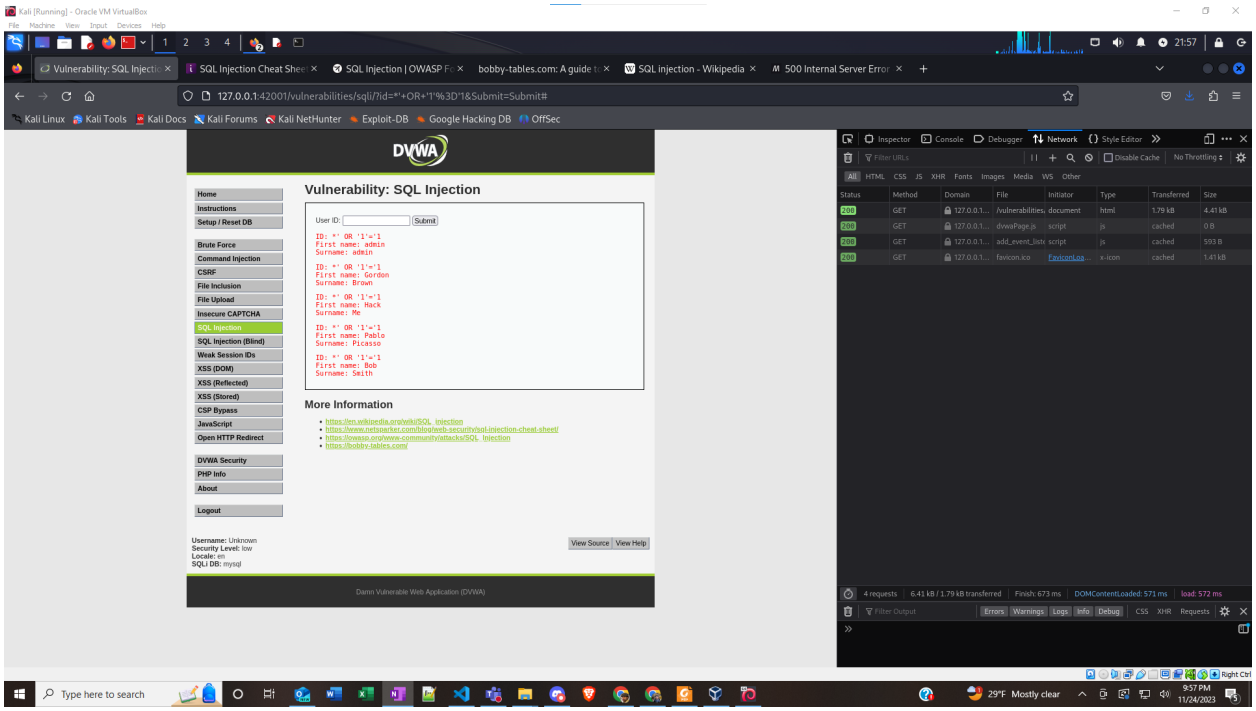

Figure 38: Results of SQL injection `*' OR '1'='1`. First name and Surname are displayed for all users.

Step 3: Get column names.

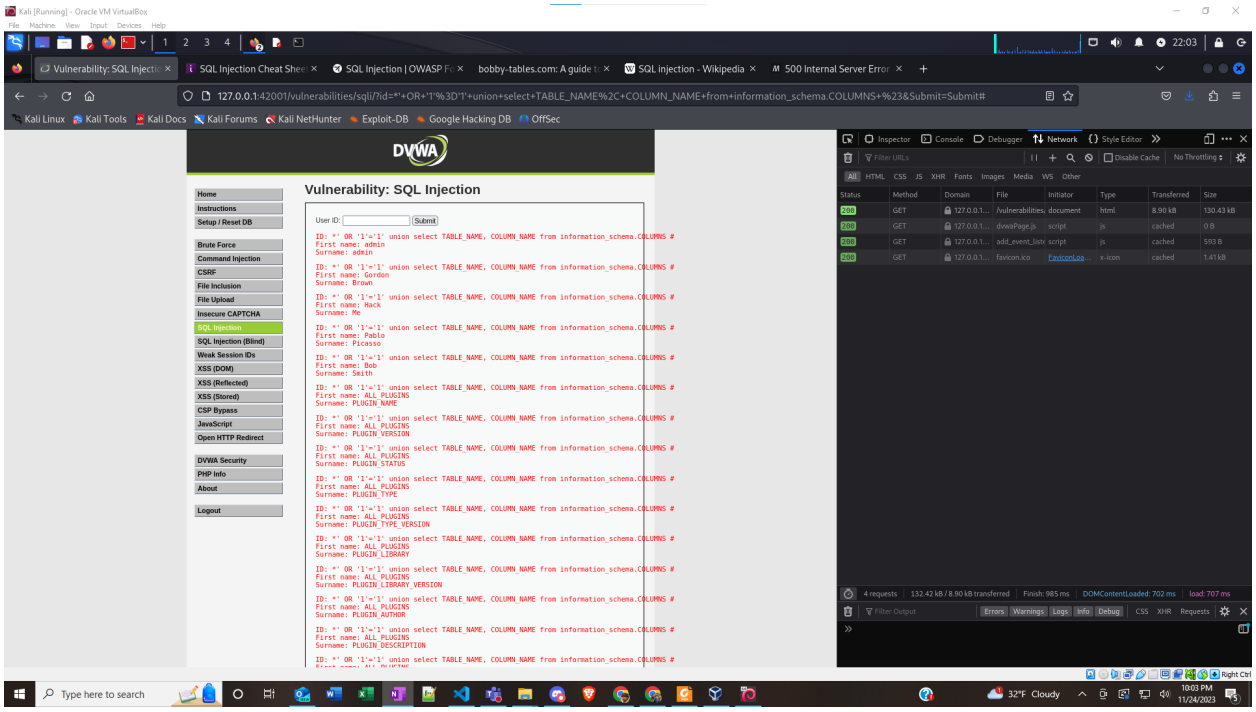

Figure 39: Results of `%' or '0'='0' union select TABLE_NAME, COLUMN_NAME from information_schema.COLUMNS #`

Step 4: Get column names of the users table.

Seth Weiss
weissse@oregonstate.edu

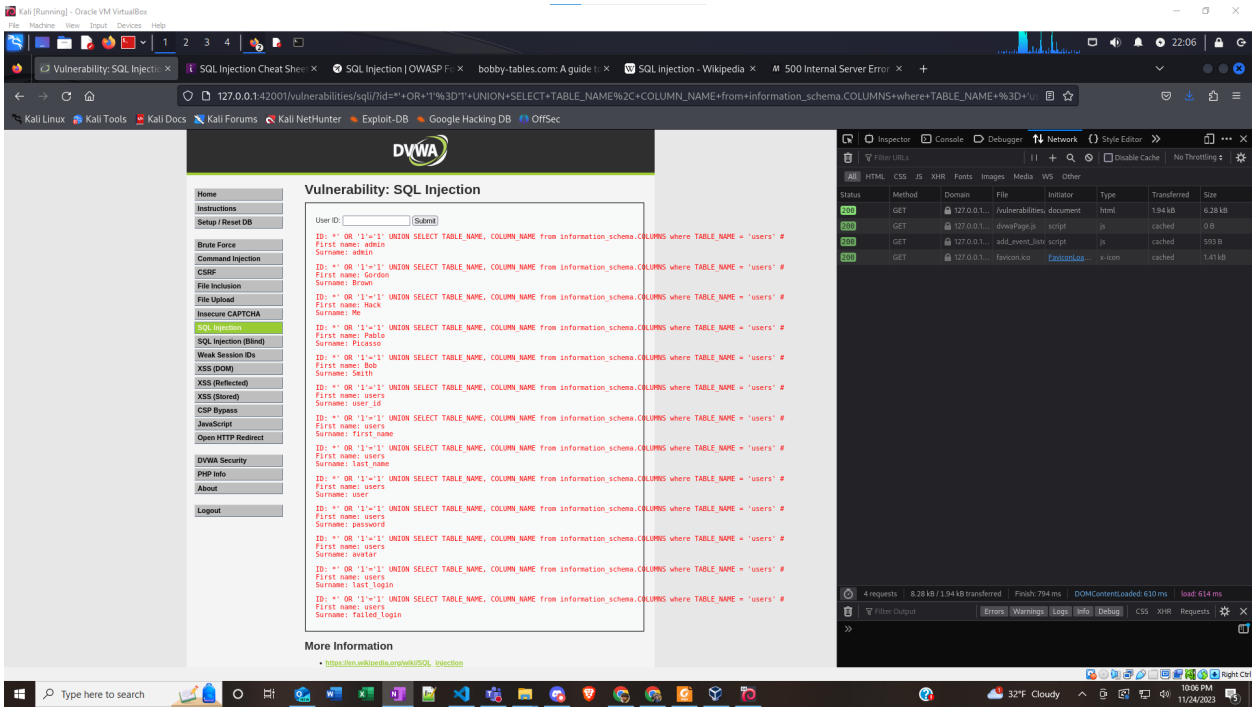

Figure 40: users table columns displayed by SQL injection.

Step 5: Enter inquiry for usernames and passwords of users table.

Seth Weiss
weissse@oregonstate.edu

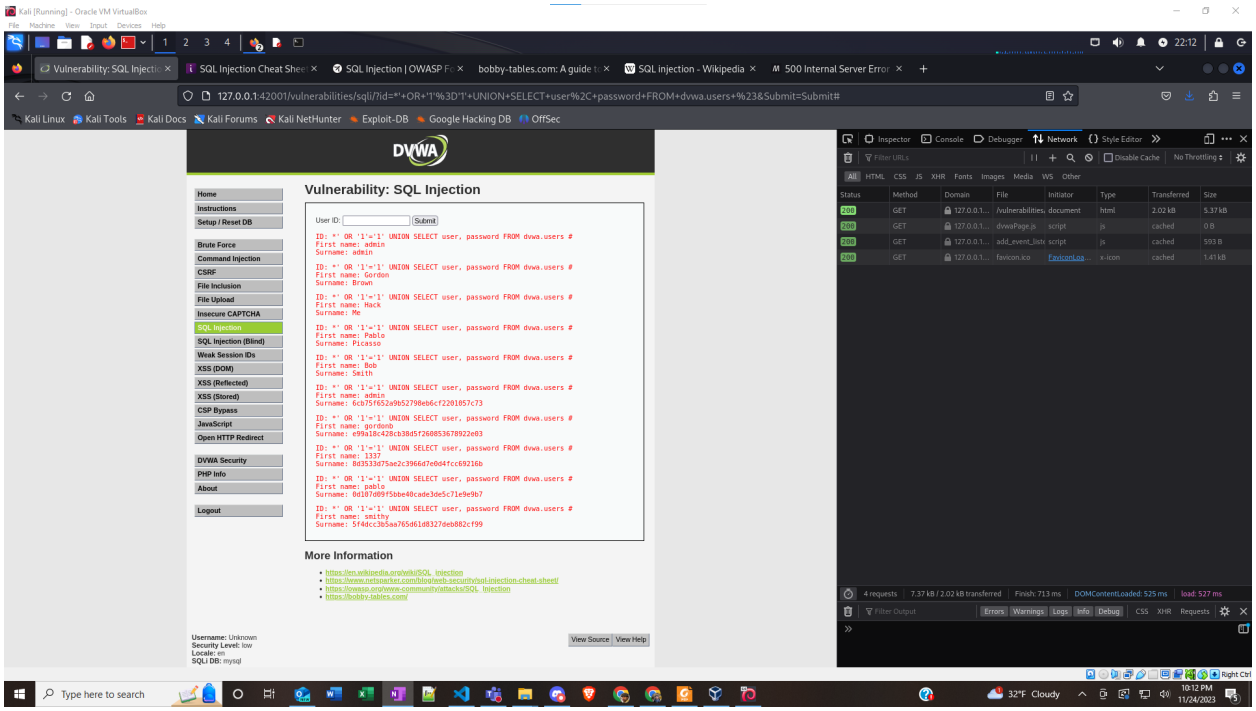

Figure 41: Results of SQL injection `%' or '0'='0' union select user, password from dvwa.users #`.

Step 6: Crack the password hashes with john the ripper.

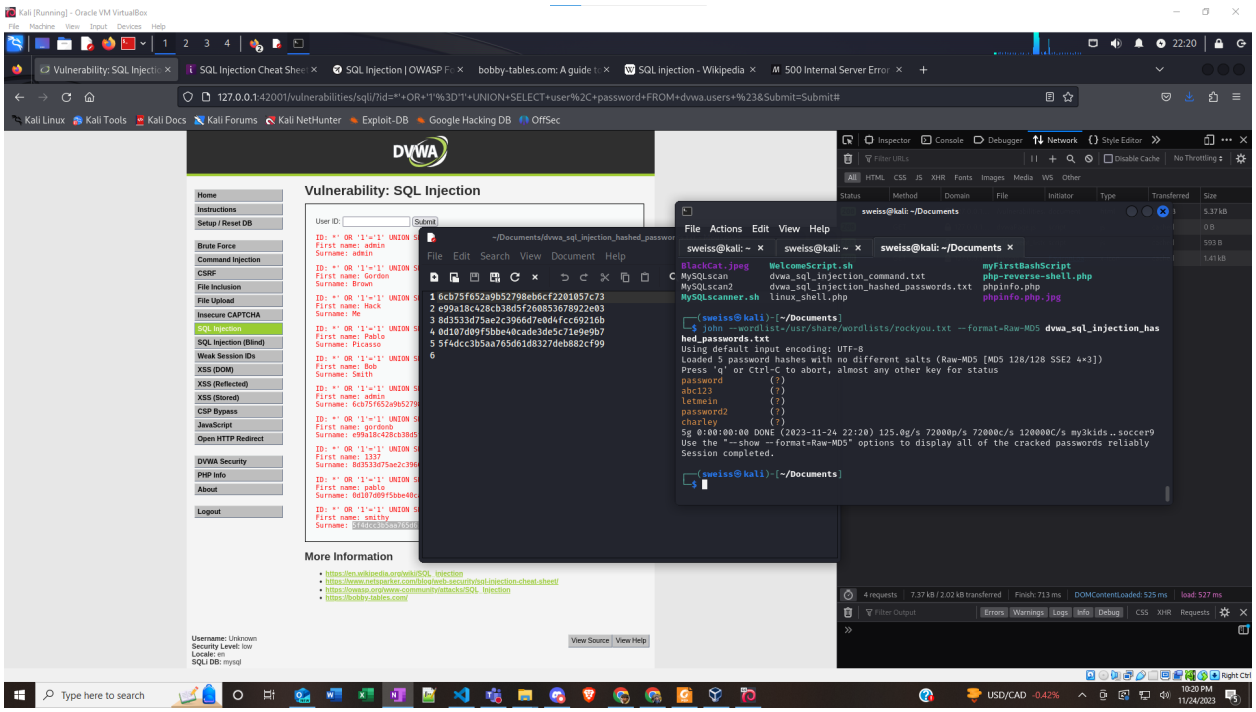

Figure 42: Cracking password hashes retrieved by SQL injection.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection

Resources

- (2023, November 27). *SQL injection*. Wikipedia. https://en.wikipedia.org/wiki/SQL_injection
- Mark R. (2020, January 21). *Challenge 07: SQL Injection.md*. GitHub. https://github.com/keewenaw/dvwa-guide-2019/blob/master/low/Challenge%2007%3A%20SQL%20Injection.md

Vulnerability 7 - Weak Session IDs

How does this feature normally work?

This application feature is very simple and only has a button to generate a new session cookie. The session cookie is only used for this feature and can be viewed in the request headers. In the Low security mode, the cookie value is a simple incrementation value starting at 0. On

Medium, it uses a timestamp (unix time). On High, it uses hashed values, but values are still incremented.

What does it take to exercise the vulnerability?

Any sort of snooping or man in the middle attack that could view the cookie in the request header can be used to extract and predict the cookie. A tool such as burp suite can intercept the request and increment the cookie through repeated requests.

How did the feature work differently than normal use?

There will be a gap between the response cookie and the request cookie.

Why did this work differently?

There is a gap between the response cookie and the request cookie because Burp suite was sending requests while the browser was hung up during intercept. The browser did not see the sequential incrementation between the server and Burp suite.

Why we should care about this vulnerability and potential loss

Simple incrementation or any level of predictability in session cookies is dangerous because it could allow an attacker to forge a session cookie. In this example, the session cookie was only used on this particular feature. However, if it was used for access to more sensitive information, an attacker could fake a valid session and gain access to that information. Similar to other vulnerabilities, this information could be passwords, credit card numbers, proprietary company documents, etc.

Briefly describe how to fix the vulnerability

As described in the View Help button, the Impossible level resists predictability and, more importantly, forgery by including extra glass as well as tieing the domain and path to the cookie challenge.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality with the Inspector window open.

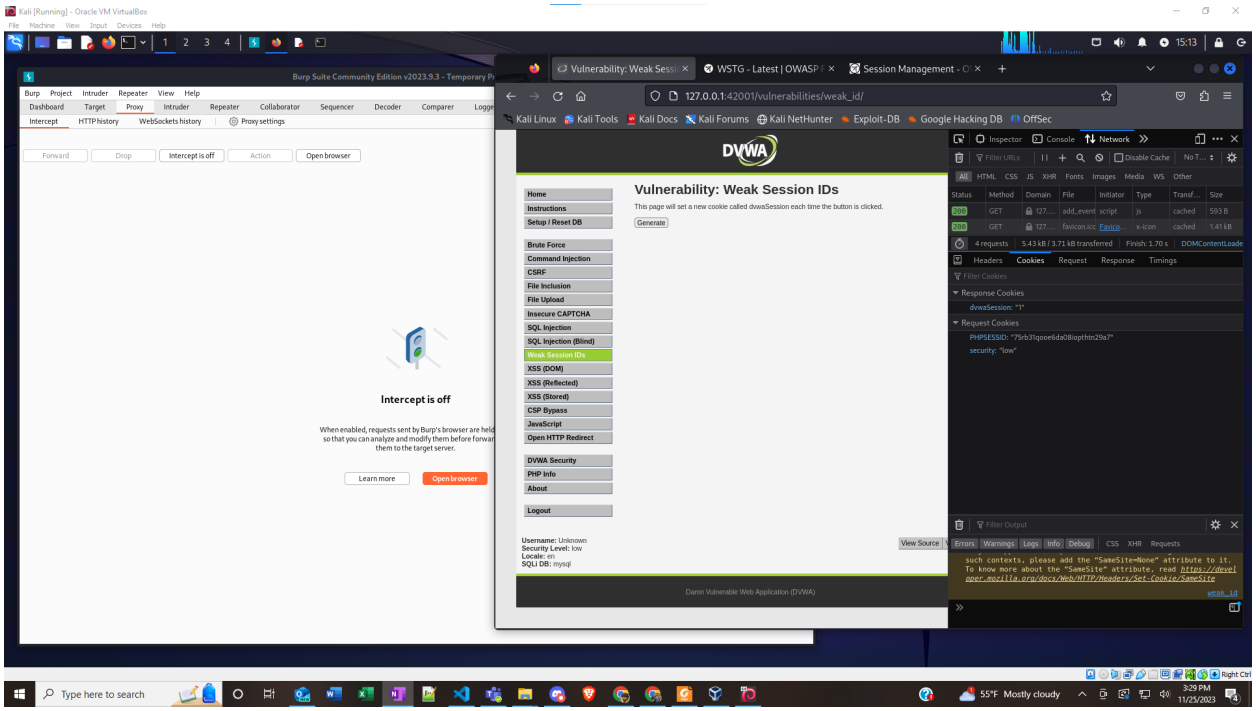

Figure 43: Initial cookie generation.

Step 2: Turn on Burp Suite Intercept and capture the request for a new cookie.

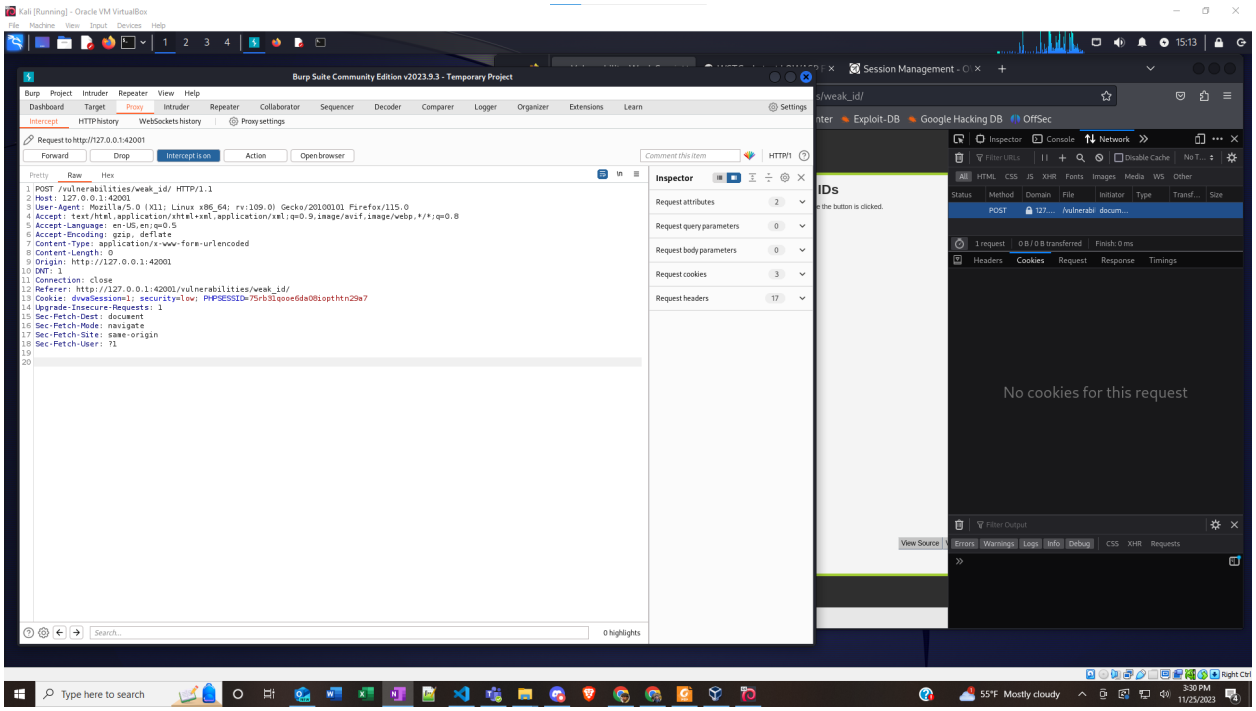

Figure 44: 2nd cookie generation request captured in Burp Suite.

Step 3: Send captured request to Repeater.

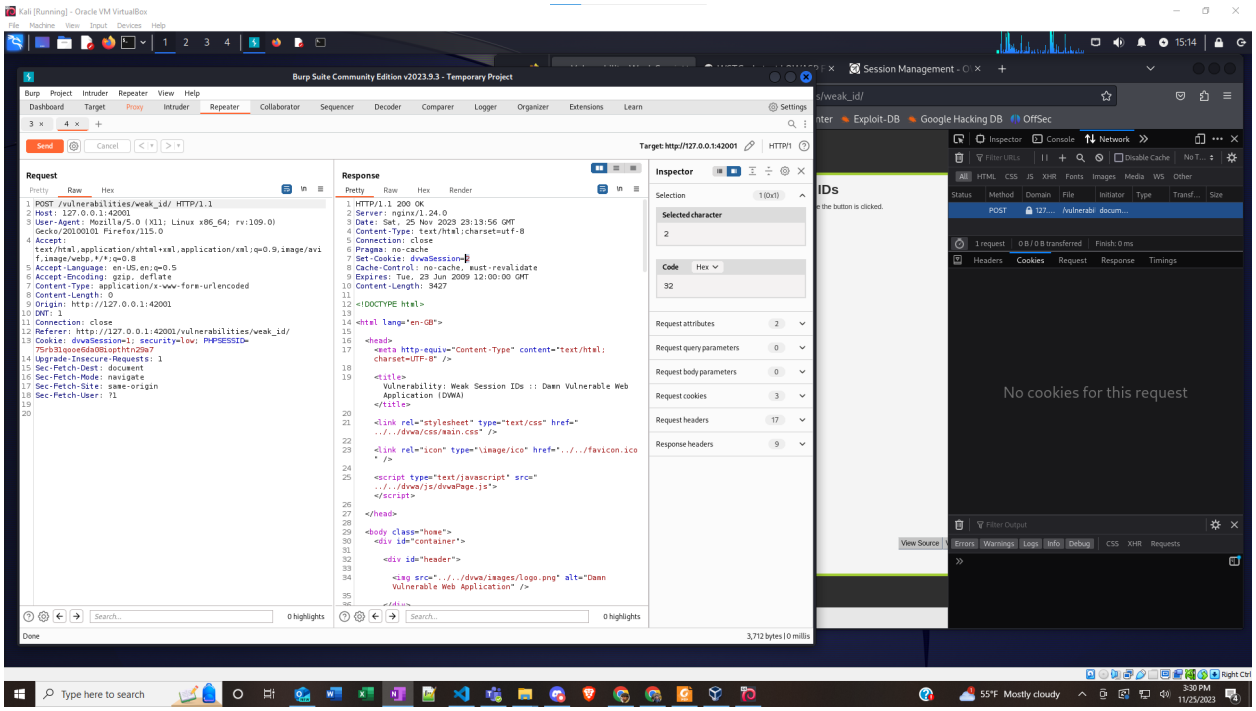

Figure 45: Captured request reproduced in Repeater. The dvwaSession cookie is incremented from 2 to 3.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

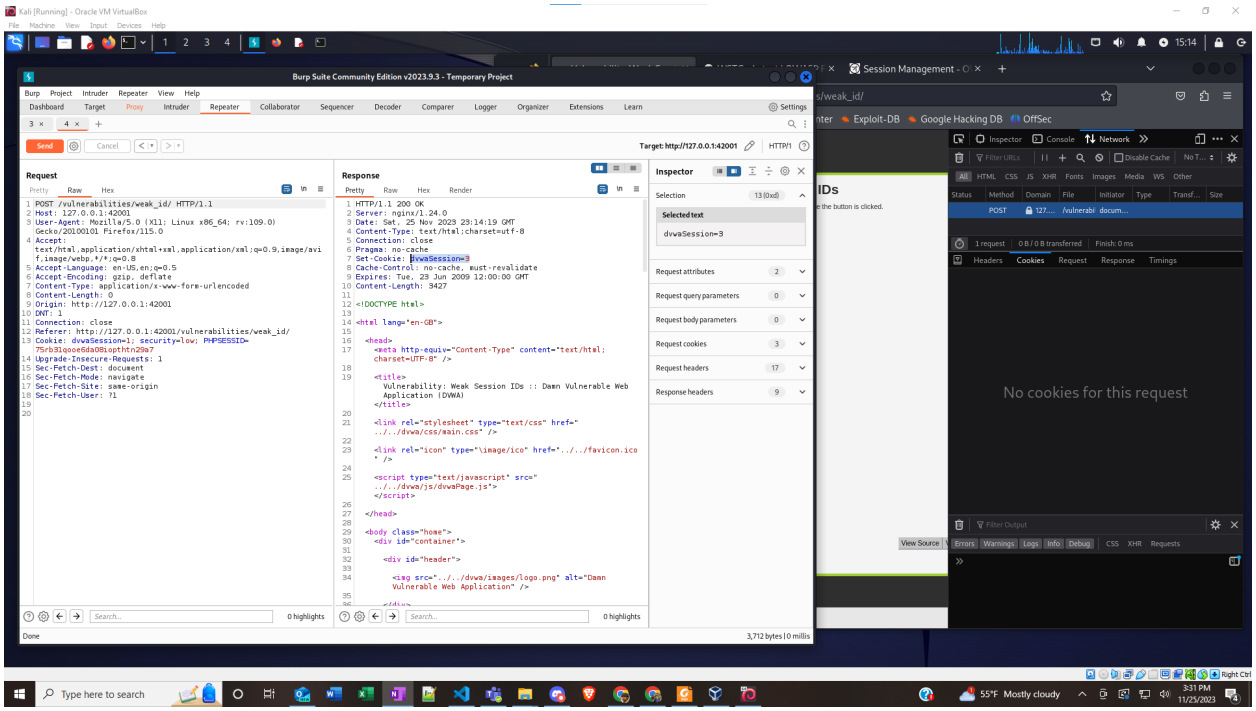

Figure 46: Additional request sent in Repeater. The dvwaSession cookie is incremented to 3.

Step 4: Turn off Burp Suite Intercept and analyze the session cookie in the Inspector window.

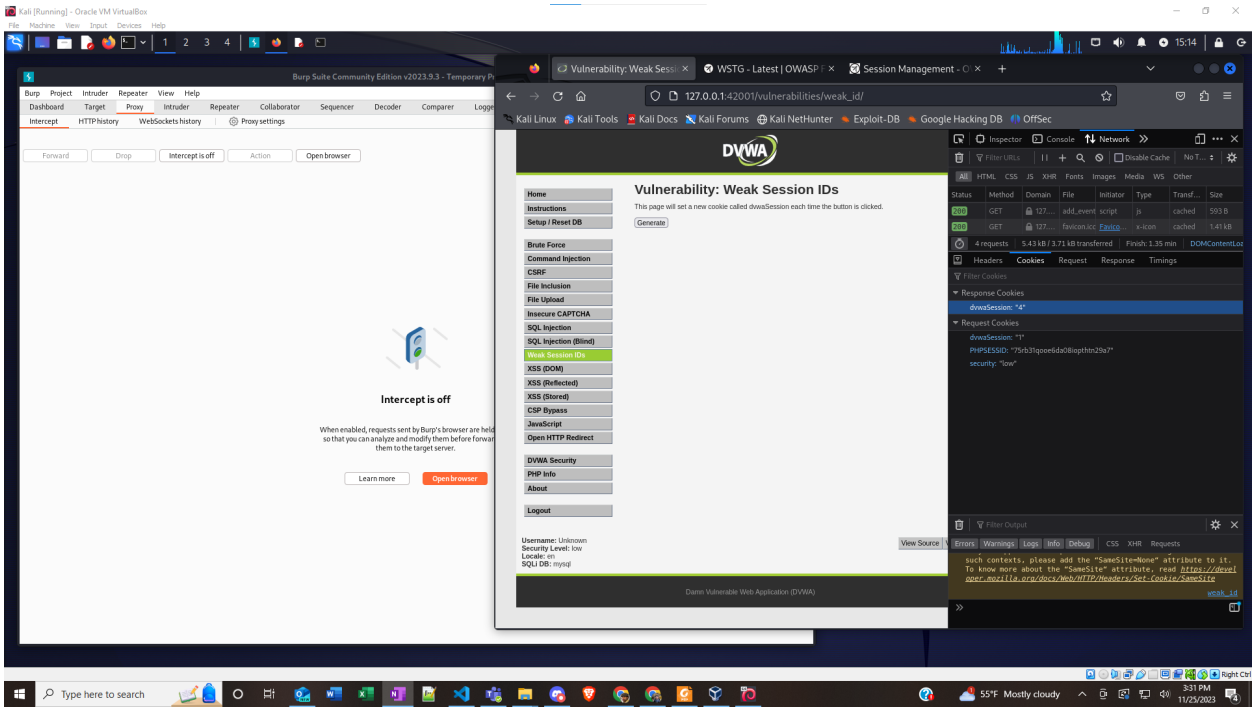

Figure 47: Response Cookies shows a gap in sequential increments from Request Cookies.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A01:2021 – Broken Access Control
- A02:2021 – Cryptographic Failures
- A04:2021 – Insecure Design
- A05:2021 – Security Misconfiguration
- A08:2021 – Software and Data Integrity Failures

Resources

- Mark R. (2020, January 12). *Challenge 09: Weak Session IDs.md*. GitHub. https://github.com/keewenaw/dvwa-guide-2019/blob/master/low/Challenge%2009%3A%20Weak%20Session%20IDs.md
- CryptoCat. (2021, February 27). *9 - Weak Session IDs (low/med/high) - Damn Vulnerable Web Application (DVWA)* [Video]. YouTube. https://www.youtube.com/watch?v=xzKEXAdlxPU&list=PLHUKi1UlEgOJLPSFZaFKMoexpM6qhOb4Q&index=11
- (n.d.). *Testing for Session Management Schema*. OWASP. https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/01-Testing_for_Session_Management_Schema
- (n.d.). *Session Management Cheat Sheet*. OWASP. https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Vulnerability 8 - XSS (DOM)

How does this feature normally work?

This feature has a dropdown menu to select a language and a “Select” button. When the button is pressed, a request is generated and the selected option is included in the query string.

What does it take to exercise the vulnerability?

All it takes is to use a valid selection URL and query string and add some scripting language (e.g. JavaScript) to the end of the query. To bypass the High security settings, this is done after a fragment (`#`).

How did the feature work differently than normal use?

When passing in a short script to log the session cookie to the console, the feature works exactly the same other than the fact that the cookie is now logged to the console and viewable in the Inspector window (see Figure 3).

Why did this work differently?

This works because the server is set up to execute whatever is in the query string. In the High security mode, the server is set up to not allow this. However, since JavaScript works client side, if the fragment (`#`) is included before the script, the server never sees the script and the browser executes it when rendering the server's response.

Why we should care about this vulnerability and potential loss

We should care about this vulnerability because this technique can be used to retrieve more valuable information than a session cookie that says "security=low". A browser may be storing more valuable cookies. Additionally, this technique can be used to launch more malicious software such as trojans, ransomware, or programs to take over a user account.

Briefly describe how to fix the vulnerability

As explained in the application's View Help button and by Acunetix, the server should include checks against scripting languages and client-side code should be sanitized by inspecting references to DOM objects that pose a threat.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality.

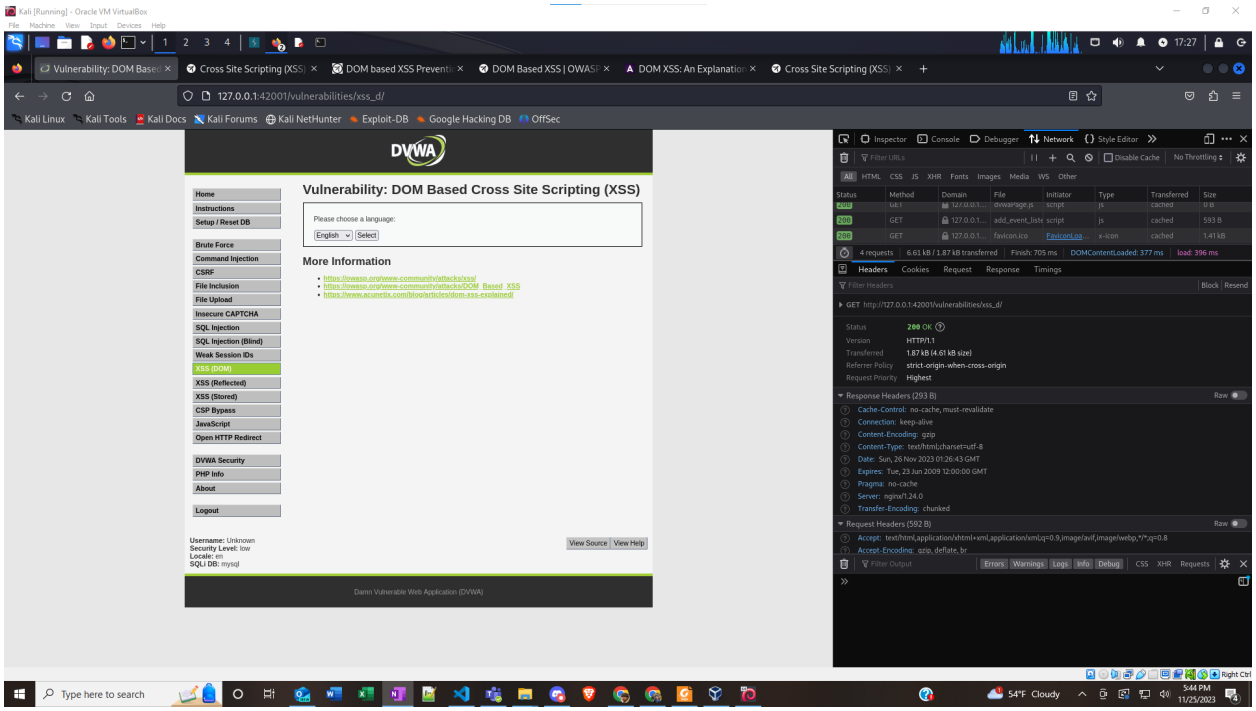

Figure 48: Initial screen for DOM based XSS feature.

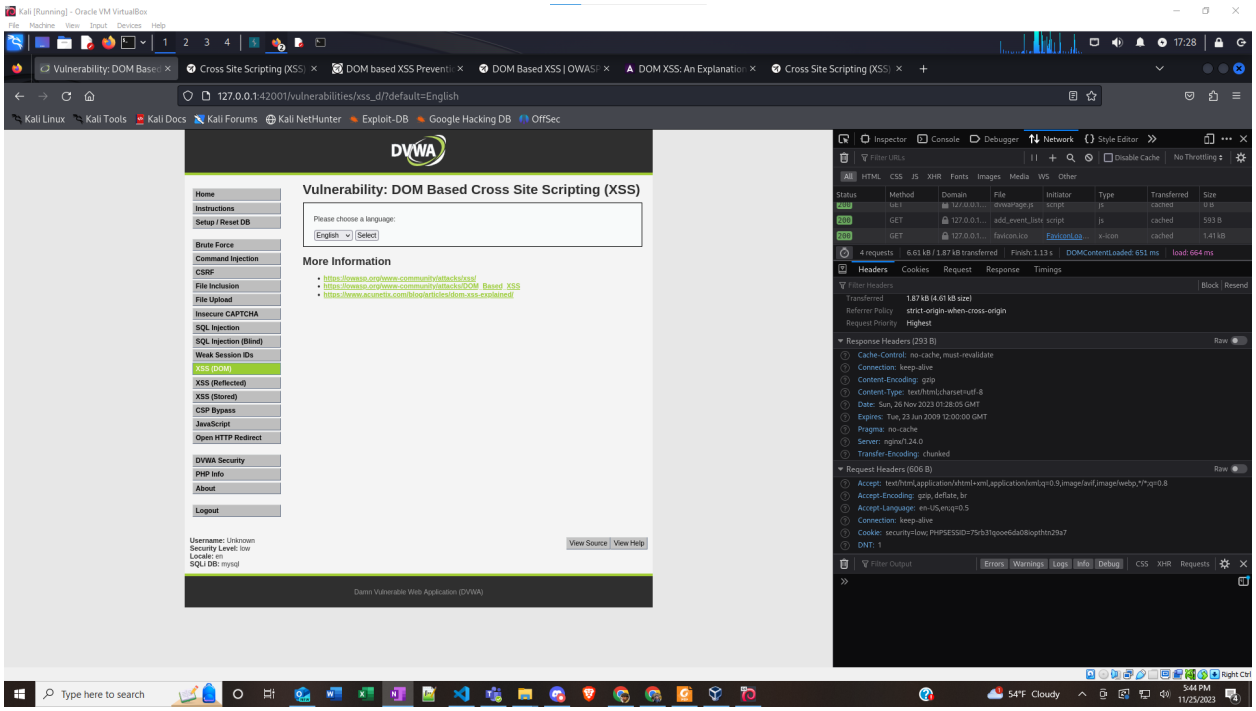

Figure 49: “English” selected and included in the query string.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Step 2: Add a script to the end of the URL query string and resend the request.

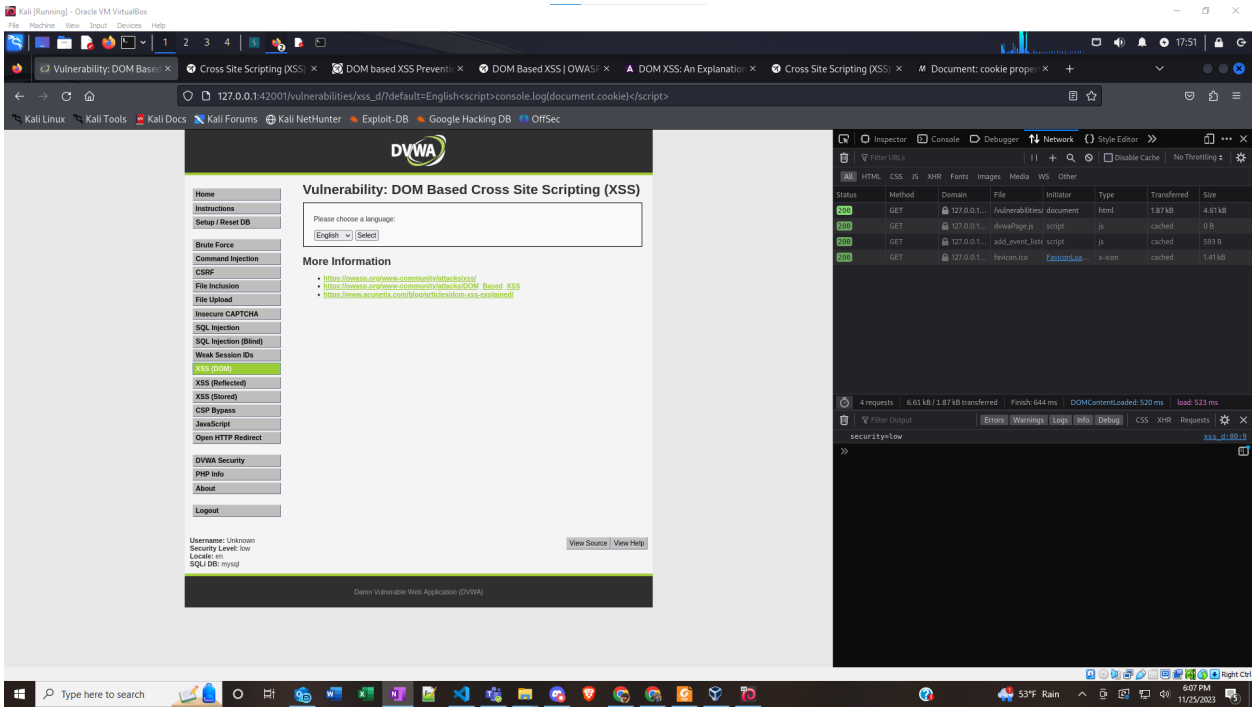

Figure 50: A script to console log the cookie is included in the query string.

Step 3: Utilize a fragment to execute the script exclusively on the client side.

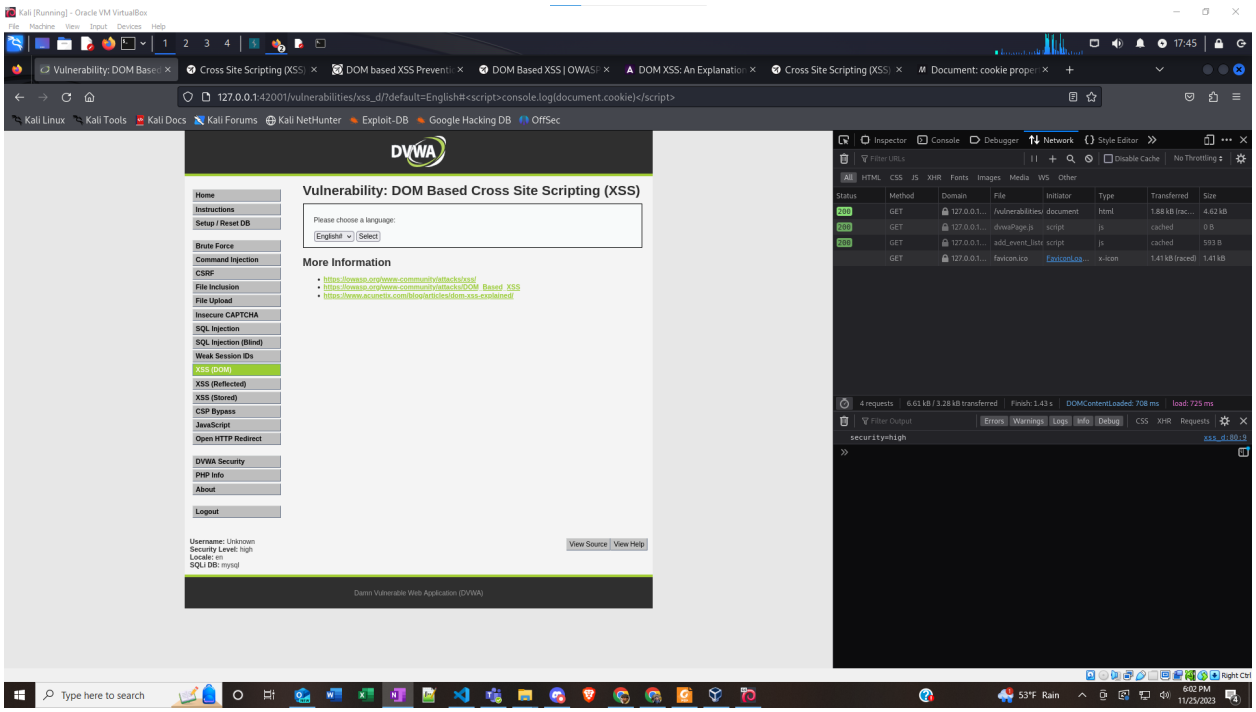

Figure 51: Fragment is used to execute the script client side.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection

Resources

- KirstenS. (n.d.). *Cross Site Scripting (XSS)*. OWASP. https://owasp.org/www-community/attacks/xss/
- (n.d.). *DOM Based XSS*. OWASP. https://owasp.org/www-community/attacks/DOM_Based_XSS
- Nidecki, T. A. (2019, March 3). *DOM XSS: An Explanation of DOM-based Cross-site Scripting*. Acunetix. https://www.acunetix.com/blog/articles/dom-xss-explained/

Vulnerability 9 - XSS (Reflected)

How does this feature normally work?

This feature has a text entry box where the user is prompted to enter a name. When a name is entered and submitted, the response page displays “Hello <name>.”

What does it take to exercise the vulnerability?

A vulnerable site can be used to inject a script into the URL query string and create a malicious link. This link can be hosted on a malicious site or sent to someone via a phishing email. When the user clicks on the link, their browser will execute the script because the response is coming from a “trusted” site (the one with the vulnerability).

How did the feature work differently than normal use?

In this example, a link was created to use the JavaScript `alert()` method to send the user a message warning them that they could have been compromised when they clicked on the link. The browser opened a new tab and navigated to the vulnerable site and displayed the alert.

Why did this work differently?

Because this script was placed in the value of `name` in the query string, the vulnerable site thought it was returning a name to display. When the browser goes to display this name in the response, the script is executed in its place.

Why we should care about this vulnerability and potential loss

Because XSS comes from a “trusted” site and the browser will execute the script. This script can be a payload for malicious code which could reveal sensitive user information such as session

cookies which could allow the attacker to hijack the user account. Some examples of XSS don't even require the user to click on a link. In the second example below, the embedded script uses the `mouseover` attribute of the `a` tag to reveal the user's cookie in an `alert()`.

Briefly describe how to fix the vulnerability

According to the OWASP Cross Site Scripting Prevention Cheat Sheet, there is no single technique to protect against XSS. Using a combination of techniques is necessary, including using modern frameworks (with less bugs), using output encoding to safely display user typed data, using HTML sanitization, using "safe sinks" (treating variables as text and never executing them), using cookie attributes to change how JavaScript and browsers interact with cookies, and using web application firewalls.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended use of the feature.

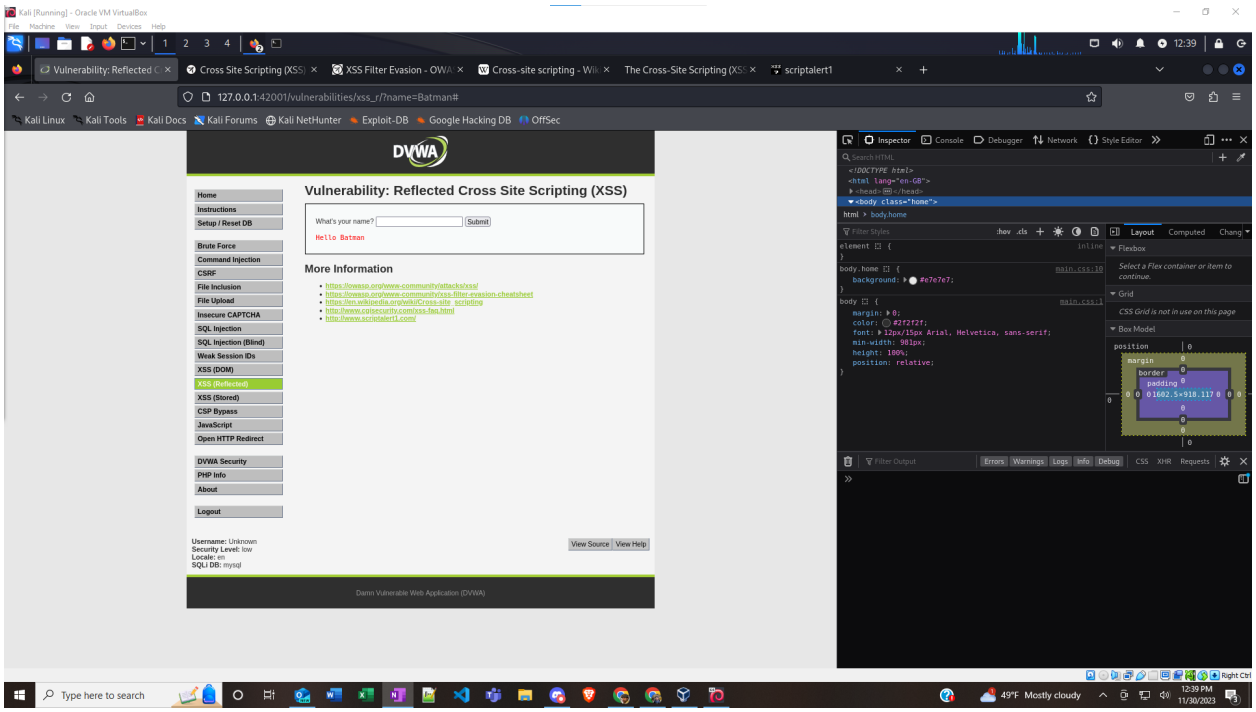

Figure 52: Intended use of feature. User entered name is displayed in response page.

Step 2: Use the inspector window to create a malicious link. Note that in a more realistic scenario, the link might not be on the vulnerable site itself, but might be on a malicious site or in a phishing email. The link directs the user's browser to the vulnerable site where the script is then executed in the response.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

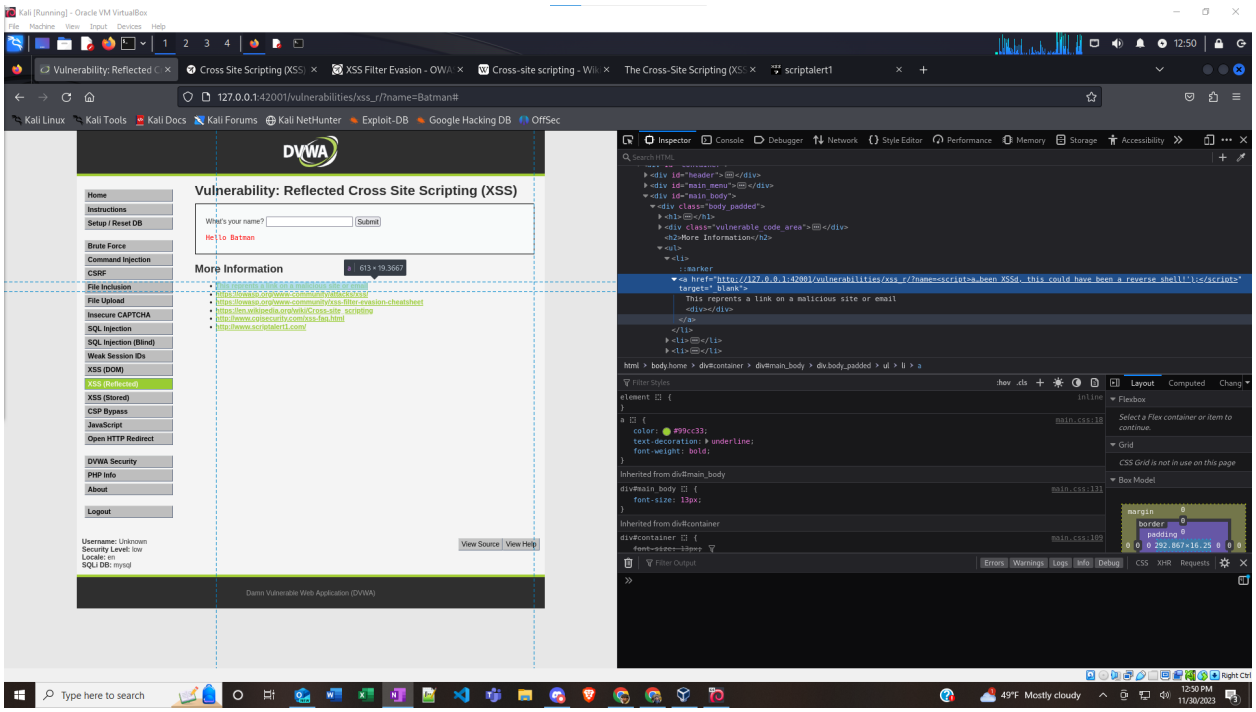

Figure 53: Create a malicious link

Step 3: Click on the malicious link.

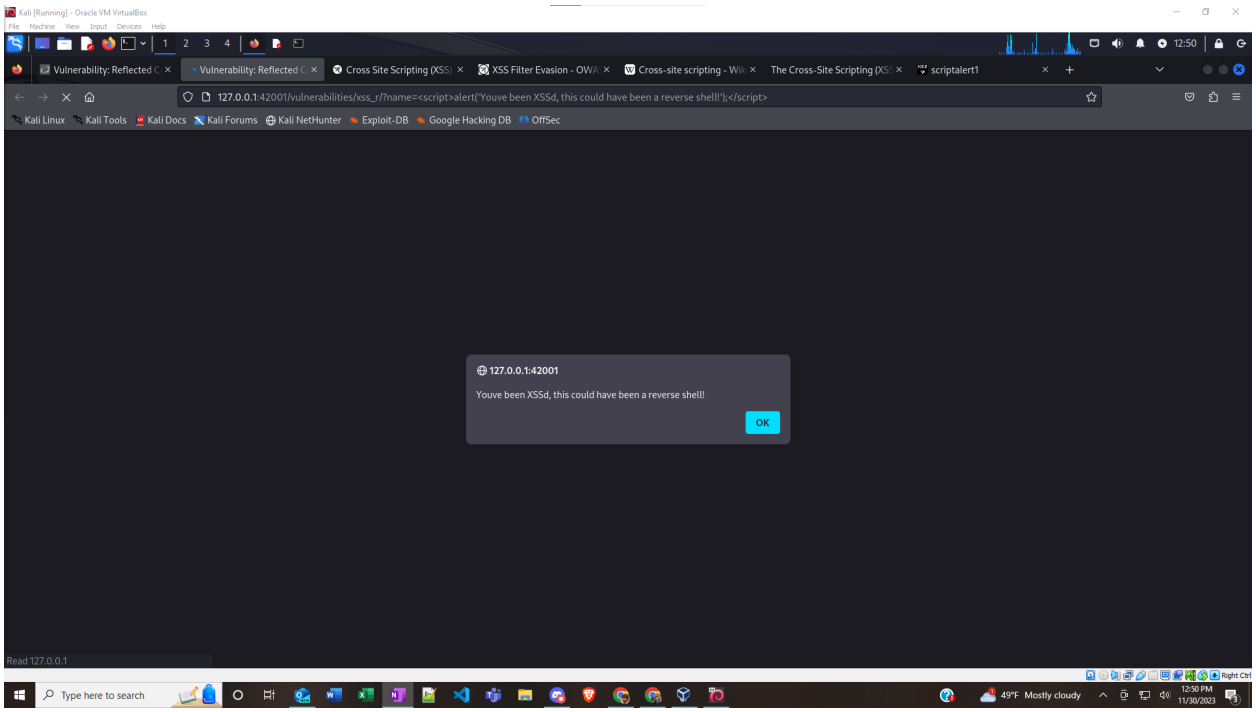

Figure 54: The malicious link takes the user to the vulnerable site which executes the script.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Step 4: Create a new malicious link using the mouseover attribute.

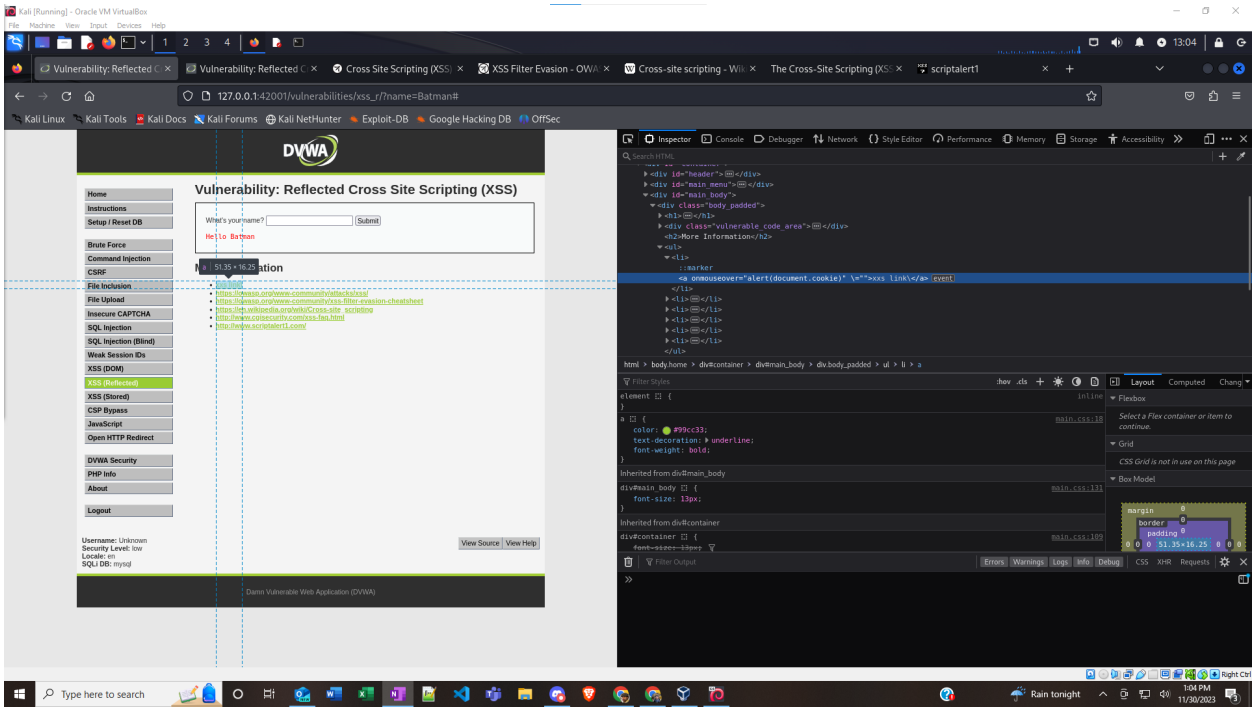

Figure 55: A second malicious link uses the mouseover attribute.

Step 5: Pass over the malicious link with the mouse pointer.

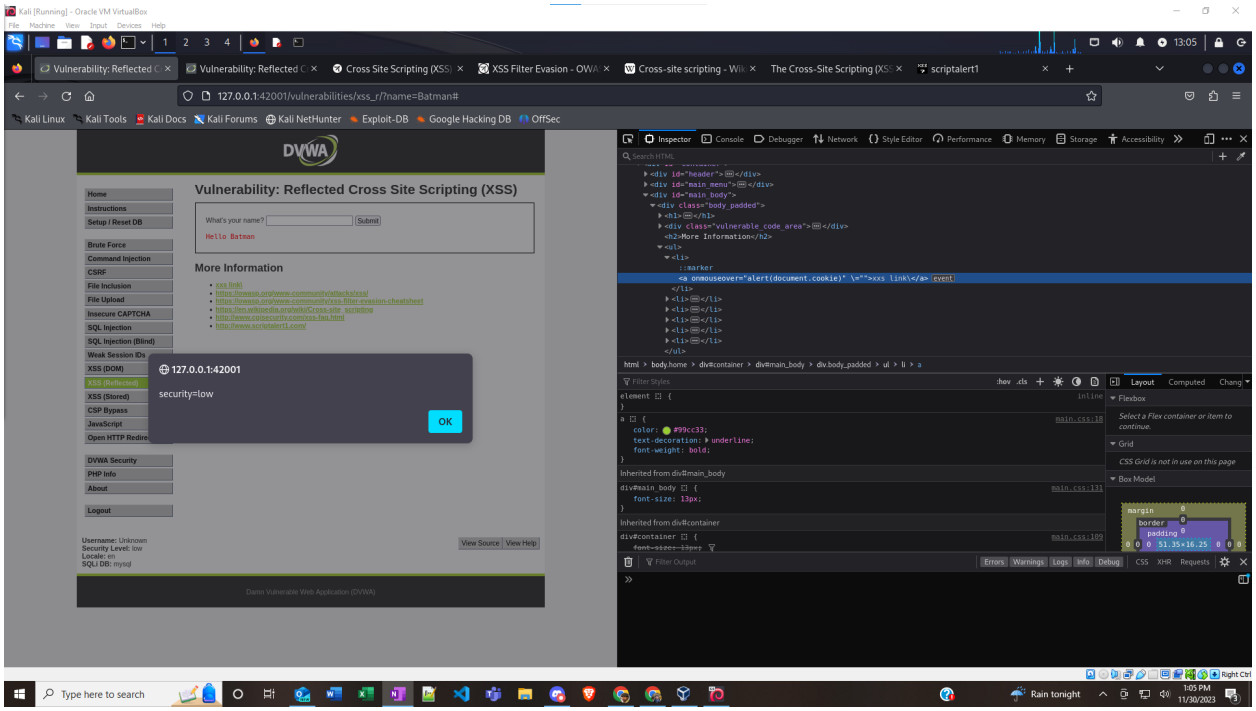

Figure 56: The malicious script is executed when the mouse pointer passes over the link.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection

Resources

- KirstenS. (n.d.). *Cross Site Scripting (XSS)*. OWASP. https://owasp.org/www-community/attacks/xss/
- (2023). *Cross Site Scripting Prevention Cheat Sheet*. OWASP. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- (2023). *XSS Filter Evasion Cheat Sheet*. OWASP. https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html
- Application “View Help” button

Seth Weiss
weissse@oregonstate.edu

Vulnerability 10 - XSS (Stored)

How does this feature normally work?

This feature has a guest book composed of a text entry form where users can enter a name and a message. When the form is submitted, the contents are displayed on the page of others to see.

What does it take to exercise the vulnerability?

Viewing the source code reveals that there are no sanitation checks on the form entry fields. Therefore, an attacker can enter a malicious script in either field.

How did the feature work differently than normal use?

Once the form with the malicious code is submitted, it will execute any time the browser loads the page (since the code is supposed to be “displayed” in the page).

Why did this work differently?

Because the code is in a file that is in the database, this is a persistent (or stored) form of XSS. The code will execute as long as the browser tries to display the code contained in the file storage.

Why we should care about this vulnerability and potential loss

Much like the other forms of XSS, this can be used to reveal user data and hijack accounts. However, unlike the other types, since it is persistent in the site’s database it may have an ability to reach a wider range of users.

Briefly describe how to fix the vulnerability

The prevention methods for Stored XSS are the same as for the other types of XSS. User input should be sanitized as well as data output to be displayed.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality.

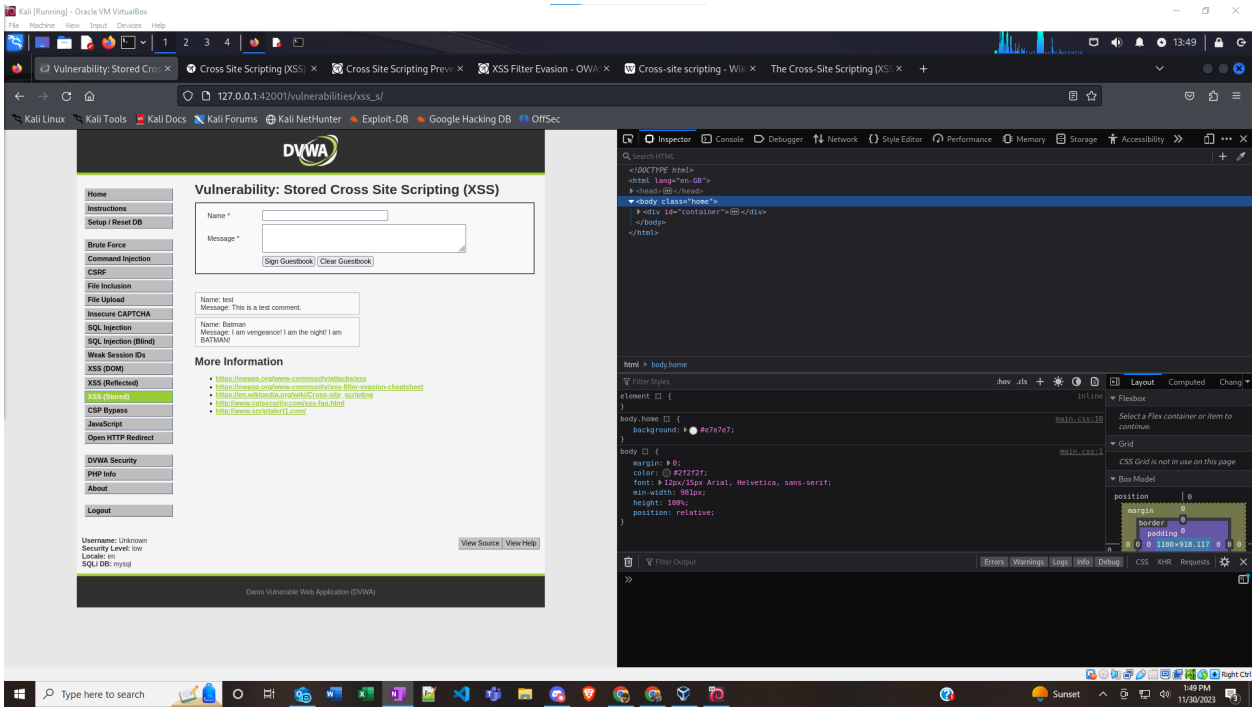

Figure 57: Enter a username and a message. The form content is displayed after submittal.

Step 2: Enter a script to reveal the user cookie in the Message field.

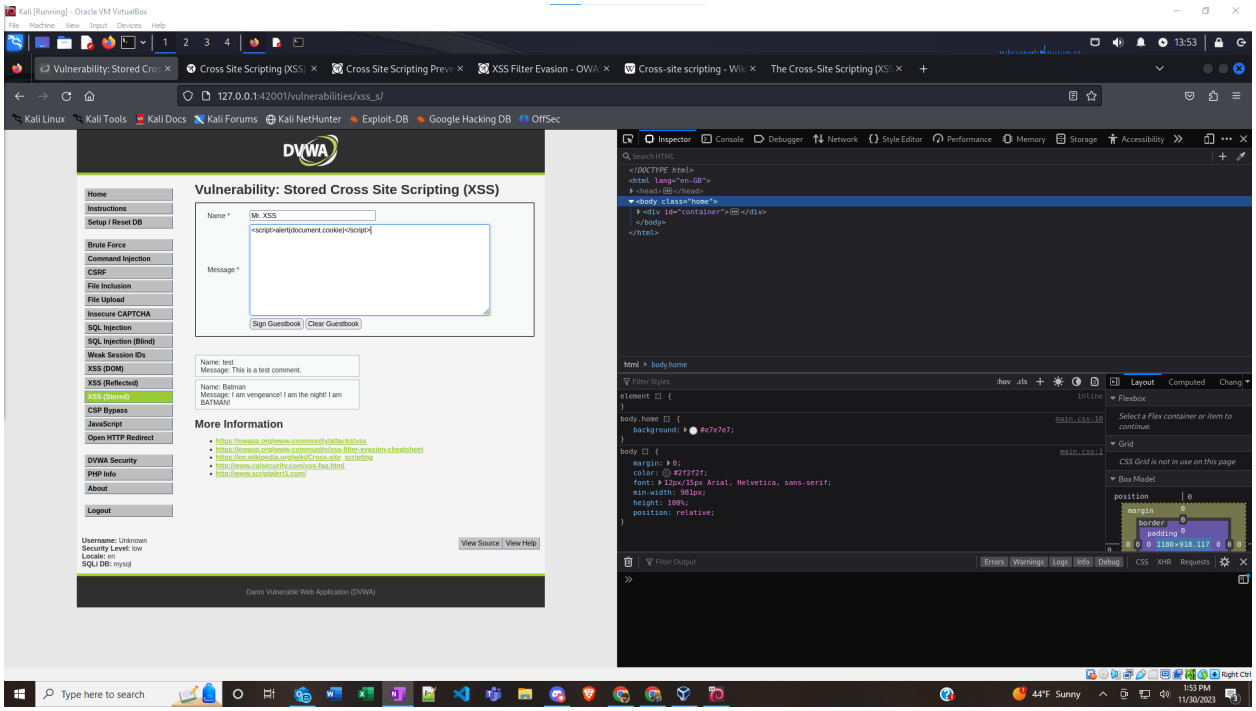

Figure 58: Malicious script entered in Message field.

Step 3: Submit the form and watch the code execute.

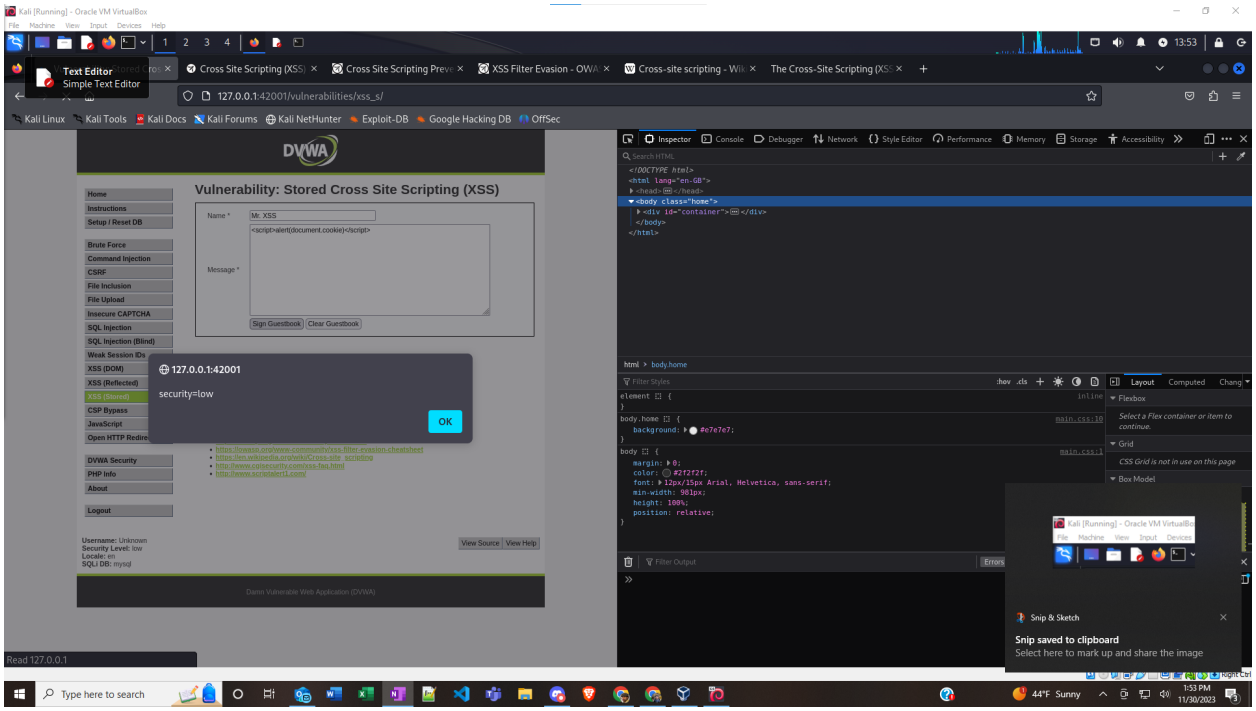

Figure 59: Malicious code executes when the page tries to display the Message content.

Step 4: Confirm persistence by refreshing the page.

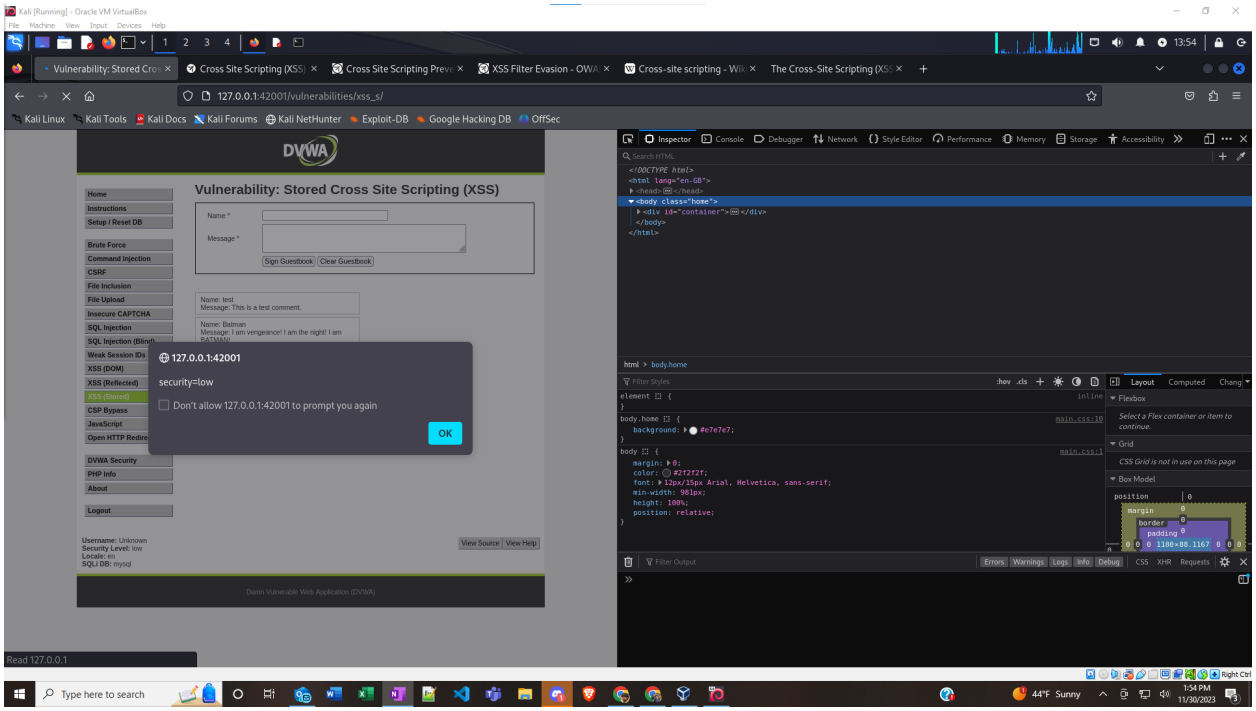

Figure 60: The malicious code executes again when the page is refreshed.

Step 5: Close the alert window. Notice the duplicate entry.

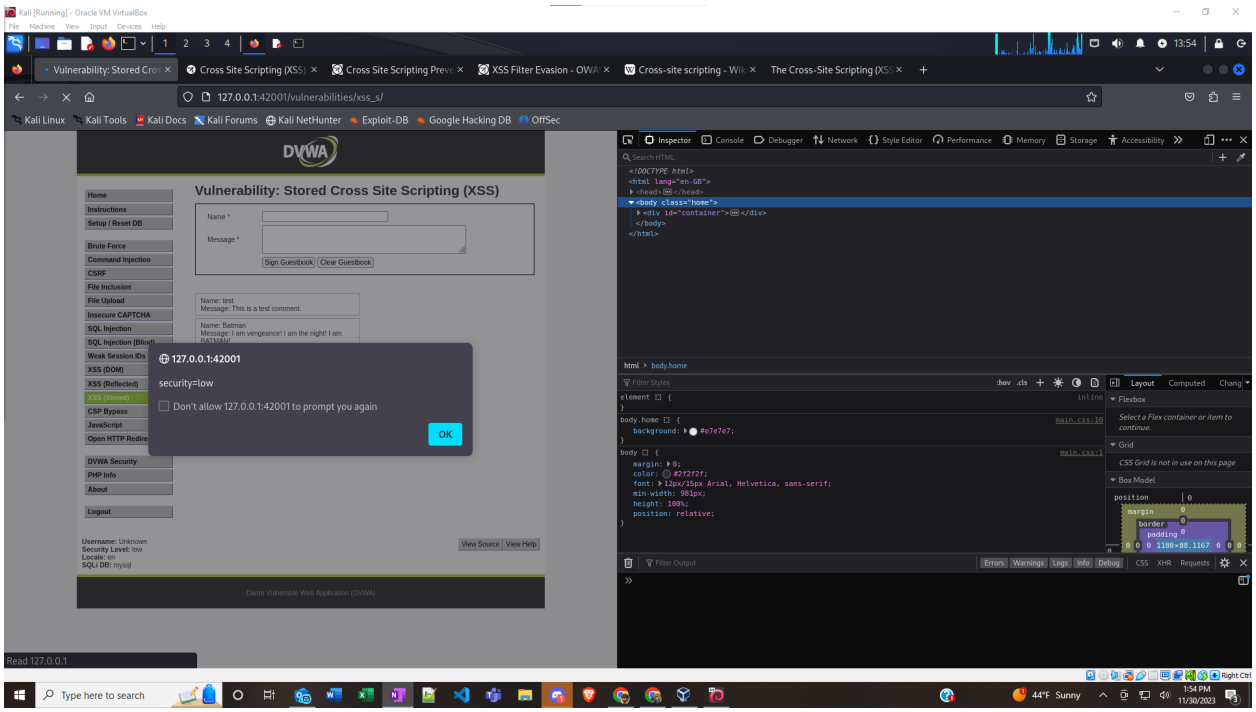

Figure 61: The entry is re-submitted on page refresh.

Step 6: Navigate to another tab on the application and then return to this feature to confirm persistence.

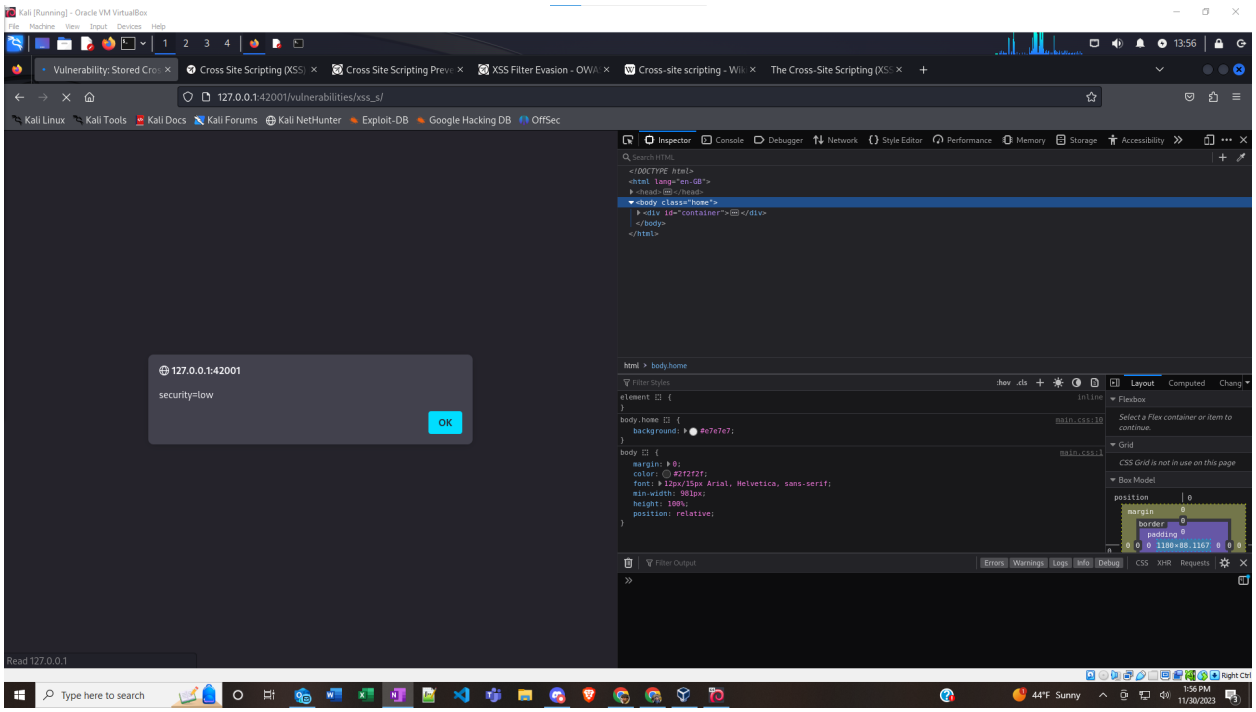

Figure 62: When return to the page, the malicious code still executes.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection

Resources

- KirstenS. (n.d.). *Cross Site Scripting (XSS)*. OWASP. https://owasp.org/www-community/attacks/xss/
- (2023). *Cross Site Scripting Prevention Cheat Sheet*. OWASP. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- (2023). *XSS Filter Evasion Cheat Sheet*. OWASP. https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html
- Application “View Help” button

Vulnerability 11 - CSP Bypass

How does this feature normally work?

This feature has a simple text entry box with a submit button labeled “Include.” It is intended to accept URL addresses from sites approved by the CSP. Any URL that is not approved will be denied and the page display will remain the same.

What does it take to exercise the vulnerability?

Examining the response header reveals the CSP, which is supposed to allow URLs from the following sources:

- Self
- https://pastebin.com
- Hastebin.com
- www.toptal.com
- Example.com
- Code.jquery.com
- https://ssl.google-analytics.com

However, there has since been an update to the application that denies all of these except “self”. This update is `x-content-type-options: nosniff`, which blocks the websites because of a MIME type mismatch.

The way around this is to chain vulnerabilities. Since a URL from “self” is still allowed under the CSP and is not blocked by the new header, a script can be loaded to the site and the URL of that script can then be included in the feature.

How did the feature work differently than normal use?

On successful inclusion, the page runs and executes the contents of the script rather than returning the same response screen.

Why did this work differently?

As aforementioned, this works because the URL source is now “self” which is approved under the CSP.

Why we should care about this vulnerability and potential loss

Bypassing the CSP can undermine the whole purpose of the CSP, which allows malicious code to be executed on the site. This malicious code could steal sensitive information, hijack accounts, cause defacement/vandalism to the site, or spread malware.

Briefly describe how to fix the vulnerability

The Impossible security level fixes this vulnerability by creating a JSONP file to call hardcoded scripts from and locks down the CSP to only allow external scripts and deny inline scripting.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality with the Inspector window open.

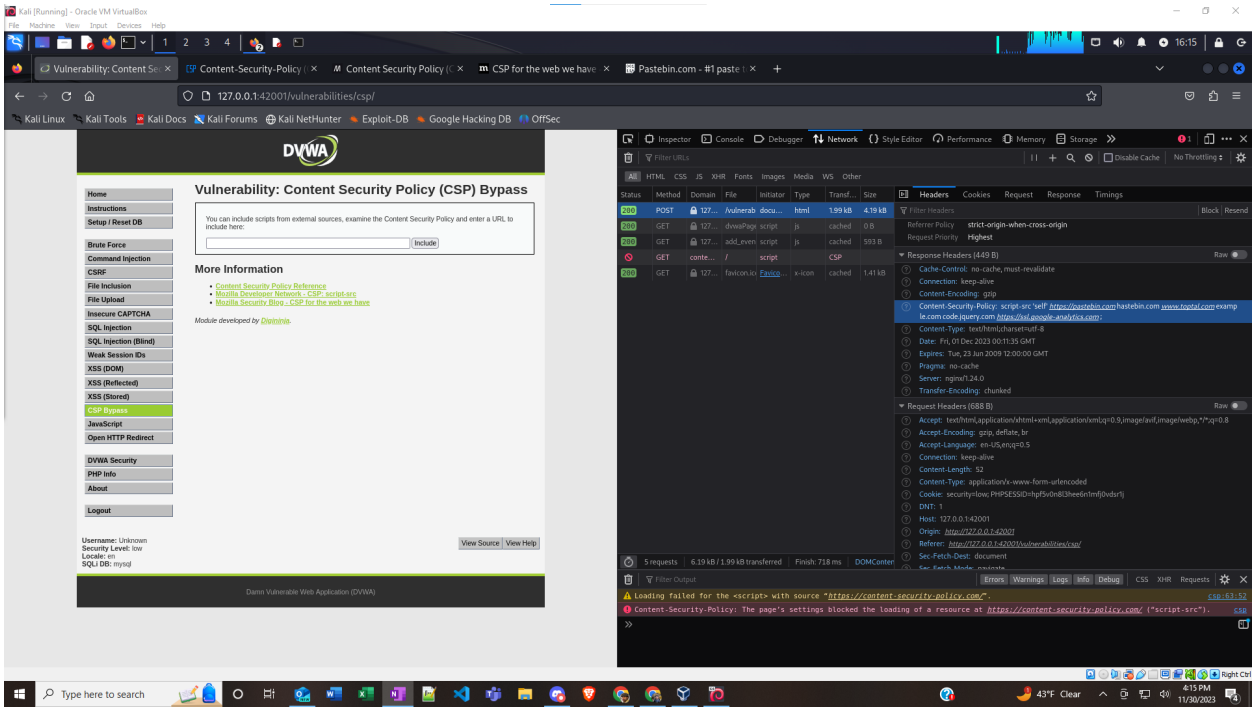

Figure 63: Include a random URL and use the Inspector window to view the CSP.

Step 2: Create a paste on pastebin.com since it appears to be allowed in the CSP. Include the past URL.

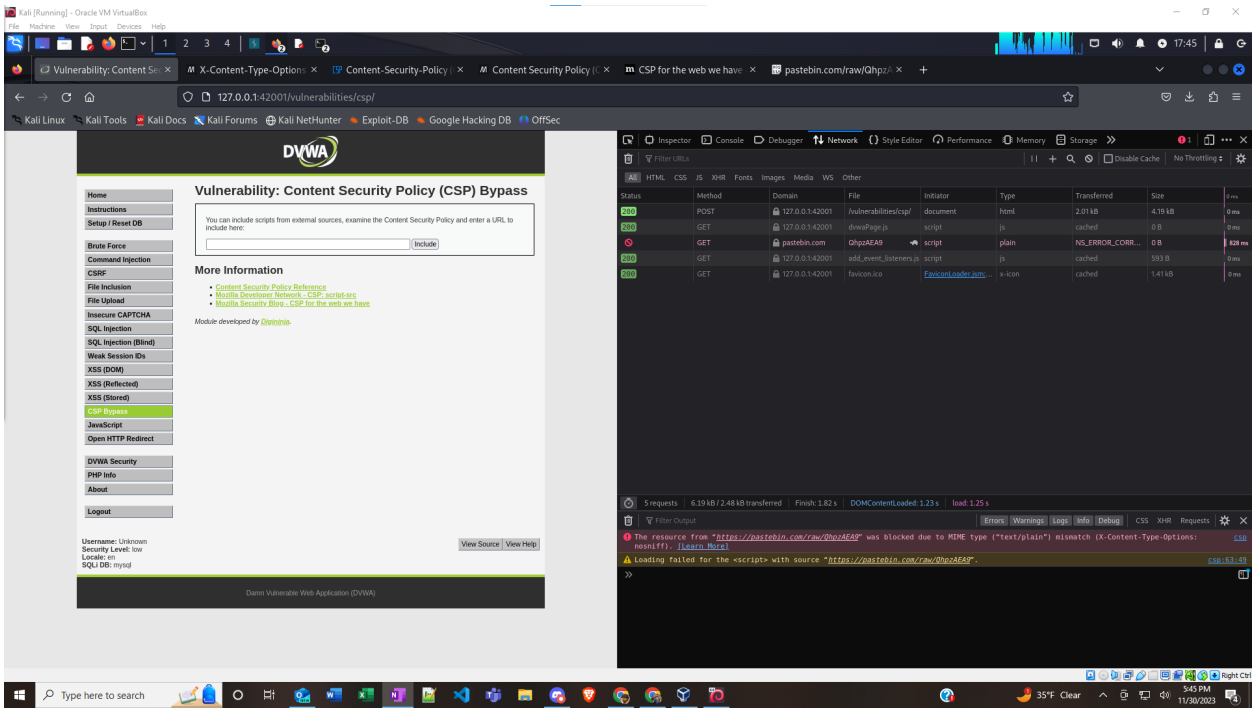

Figure 64: Pastebin URL is denied despite being allowed in CSP due to updated headers.

Step 3: Create a JavaScript file and use the File Upload feature to load it into the website's database.

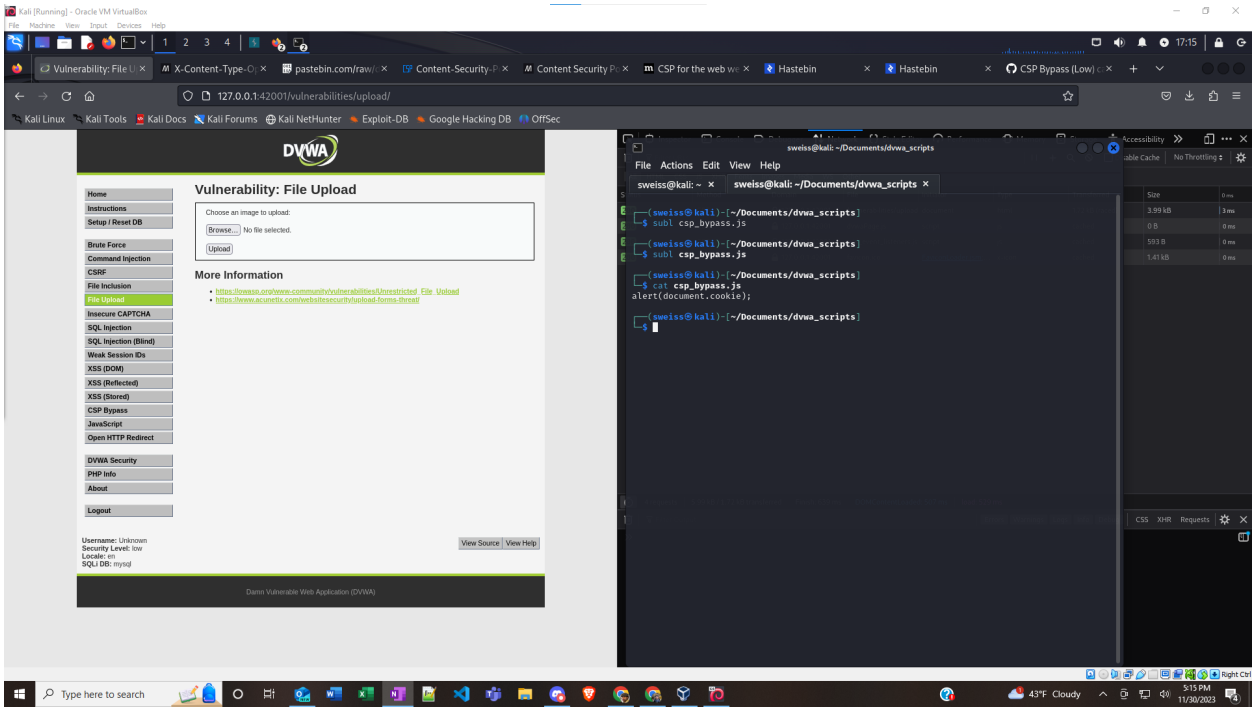

Figure 65: A simple JavaScript file

Figure 66: Successfully uploaded JavaScript file to the application.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Step 4: Include new JavaScript file URL on CSP Bypass page.

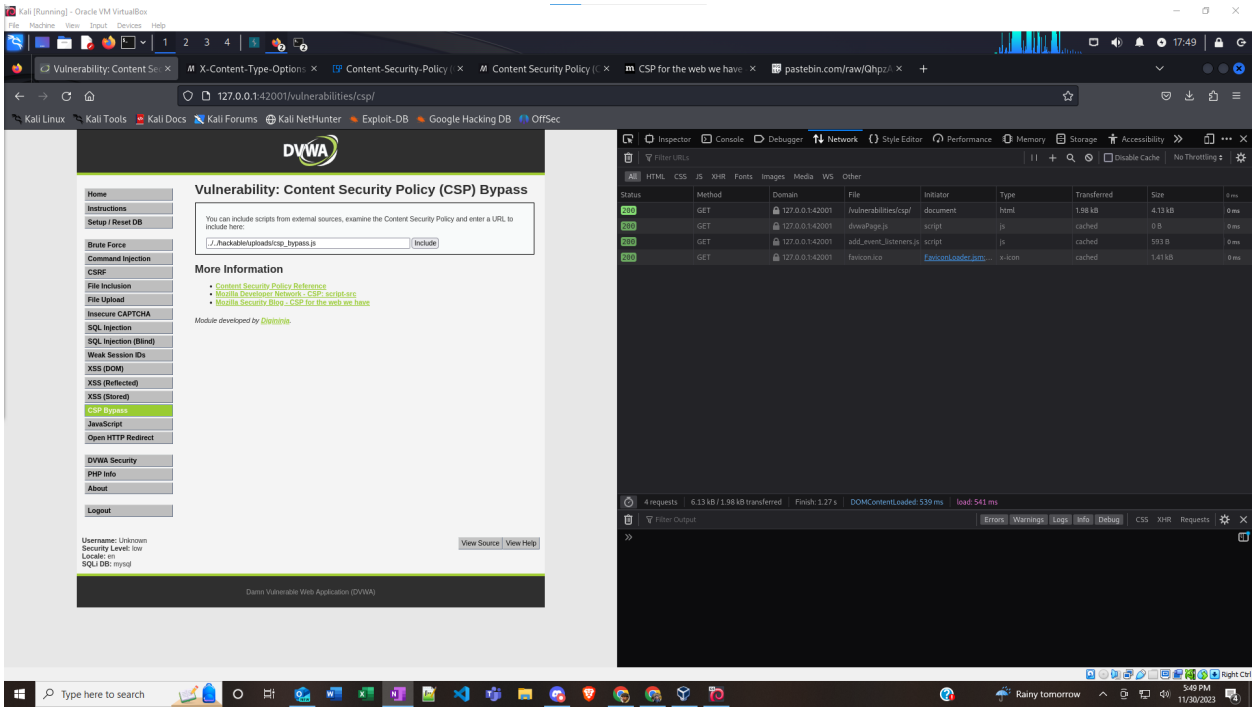

Figure 67: Include the JavaScript URL.

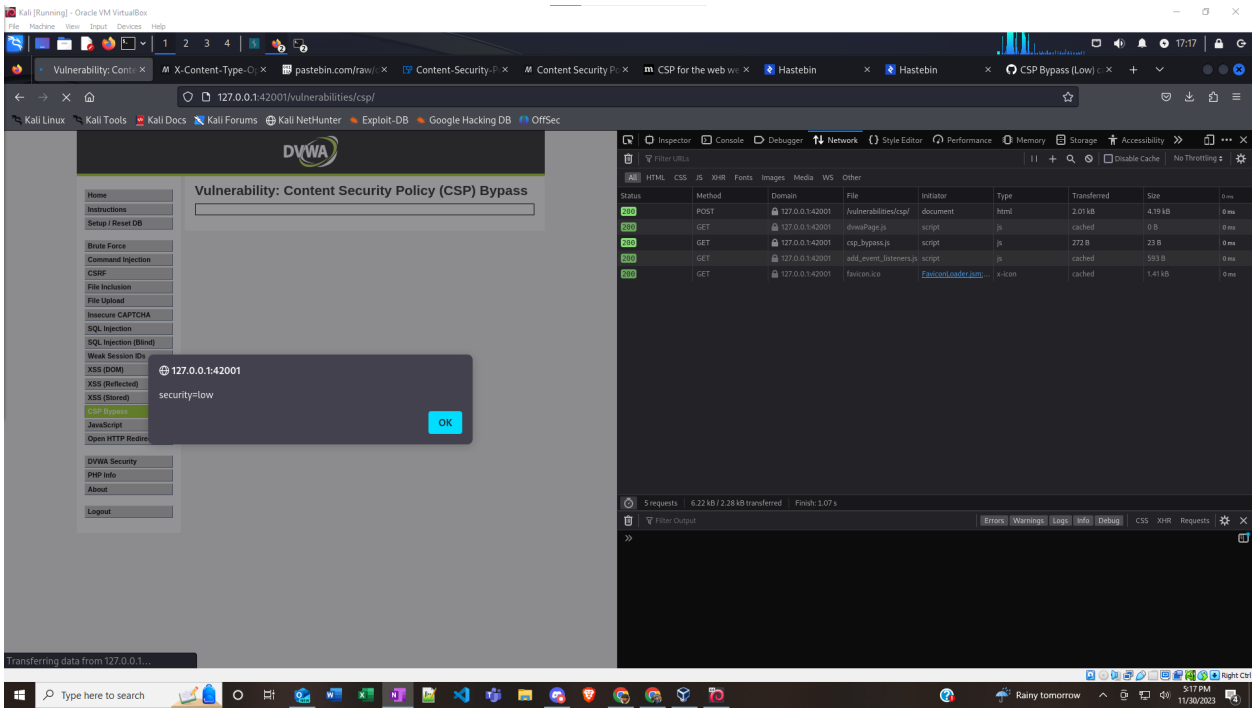

Figure 68: JavaScript file executes after submitting.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection
- A04:2021 – Insecure Design
- A05:2021 – Security Misconfiguration

Resources

- Mark R. (2020, January 12). *Challenge 13: CSP Bypass.md*.GitHub. https://github.com/keewenaw/dvwa-guide-2019/blob/master/low/Challenge%2013%3A%20CSP%20Bypass.md
- CryptoCat. (2021, February 27). *13 - CSP Bypass (low/med/high) - Damn Vulnerable Web Application (DVWA)* [Video]. YouTube. https://www.youtube.com/watch?v=ERksJHl0DC0&list=PLHUKi1UlEgOJLPSFZaFKMoexpM6qhOb4Q&index=15
- Wilbamanto, K. (2020, September 9). *CSP Bypass (Low) can't be solved with pastebin anymore #382*. GitHub. https://github.com/digininja/DVWA/issues/382
- Cheeso. (2010, January 14). *What is JSONP, and why was it created?*. Stackoverflow. https://stackoverflow.com/questions/2067472/what-is-jsonp-and-why-was-it-created

Vulnerability 12 - JavaScript

How does this feature normally work?

This feature has a text entry box and a Submit button. The user is prompted to enter the text “success” into the entry field and submit. When the user does this, the message “Invalid token.” appears. If anything other than “success” is entered, the message “You got the phrase wrong.” appears.

What does it take to exercise the vulnerability?

Viewing the source code reveals the JavaScript functions that take the entered phrase and generate a token by using a rot13 (a variant of the Caesar Cipher) and an MD5 hash. The console can be used to invoke these methods and set the value of the token. After the token has been set with the correct value, the phrase “success” can be entered and the congratulatory message appears on the response page.

A simpler method to exercise the vulnerability after viewing and understanding the source code is to enter “success” in the text entry field without hitting the Submit button. The method `generate_token()` can then be called in the console. Because JavaScript is running completely client-side in the browser, it will use the current text entry as its input to the method.

The token can then be generated correctly and when the Submit button is pressed the confirmation message appears.

How did the feature work differently than normal use?

Rather than displaying “Incorrect token.”, the response page displays the phrase “Well done!” in red text.

Why did this work differently?

The “Well done!” message is returned because the token has been set correctly. JavaScript can make modifications in the browser and does not need to communicate with the server.

Why we should care about this vulnerability and potential loss

We should care about this vulnerability because it is possible to steal session cookies or other sensitive data in this way. It is also possible to distribute malware.

Briefly describe how to fix the vulnerability

According to the View Help button on the application page, steps can be taken to make Client Side JavaScript attacks more difficult, but there is no way to fully prevent it. Any code that is sent to the client can be manipulated or bypassed. Steps include breaking JavaScript into its own file to make it more difficult to access and obfuscating the code to make it extremely difficult to understand.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality.

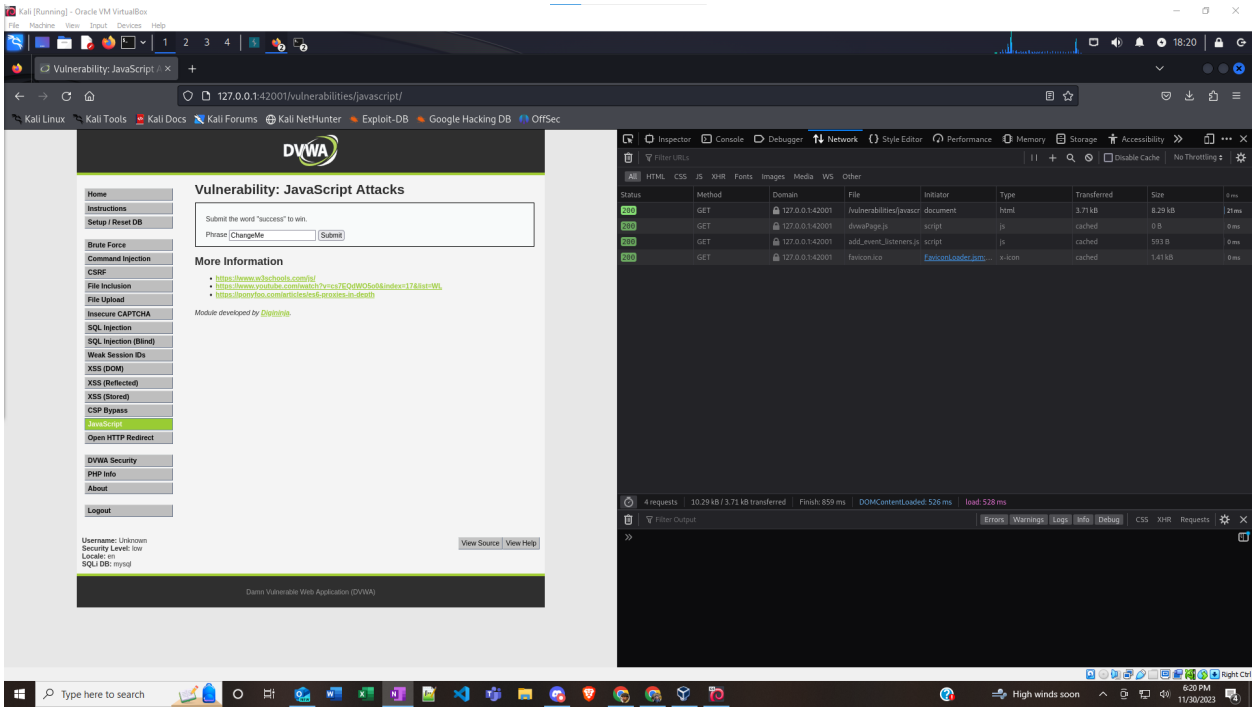

Figure 69: Initial application page.

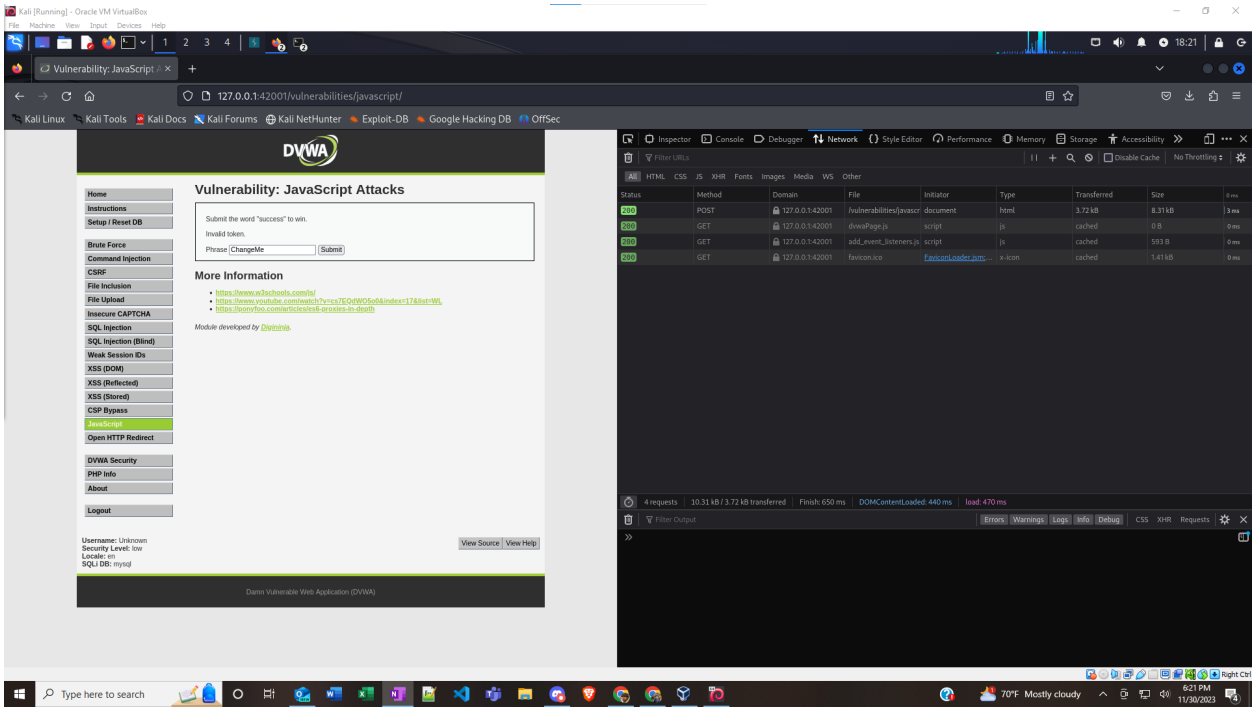

Figure 70: Result of entering the phrase “success”.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

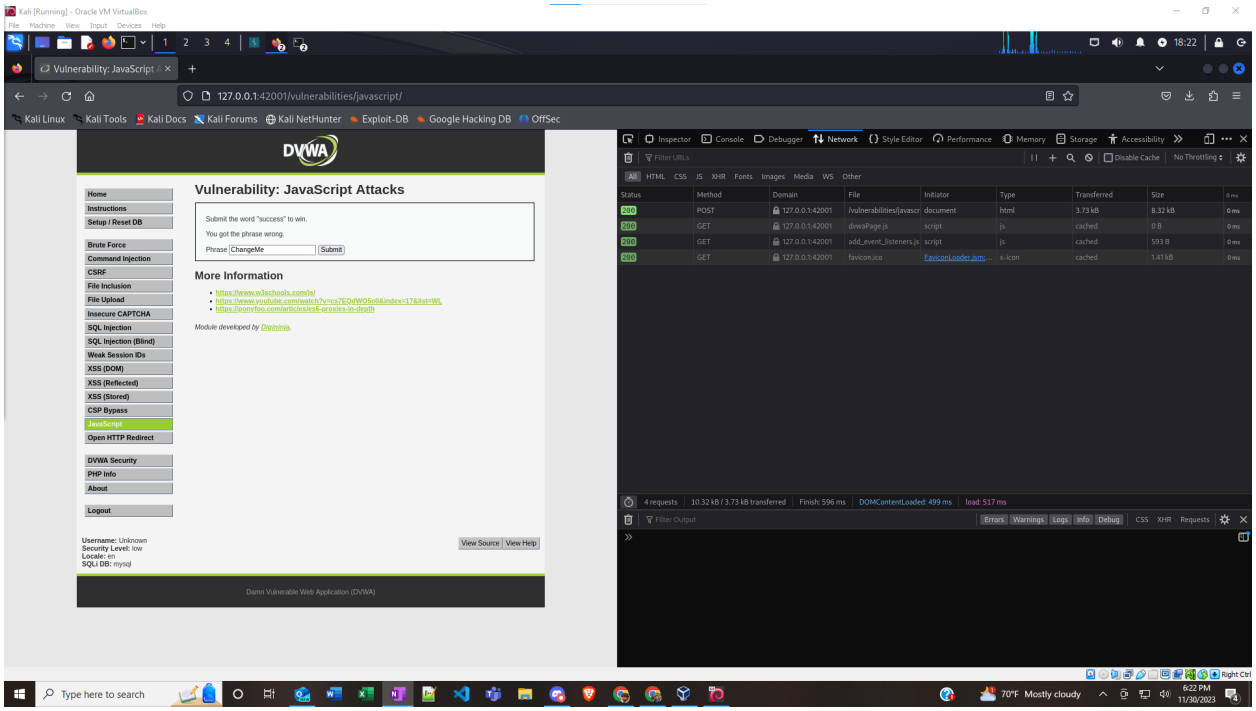

Figure 71: Result of entering an alternate phrase.

Step 2: View the source code.

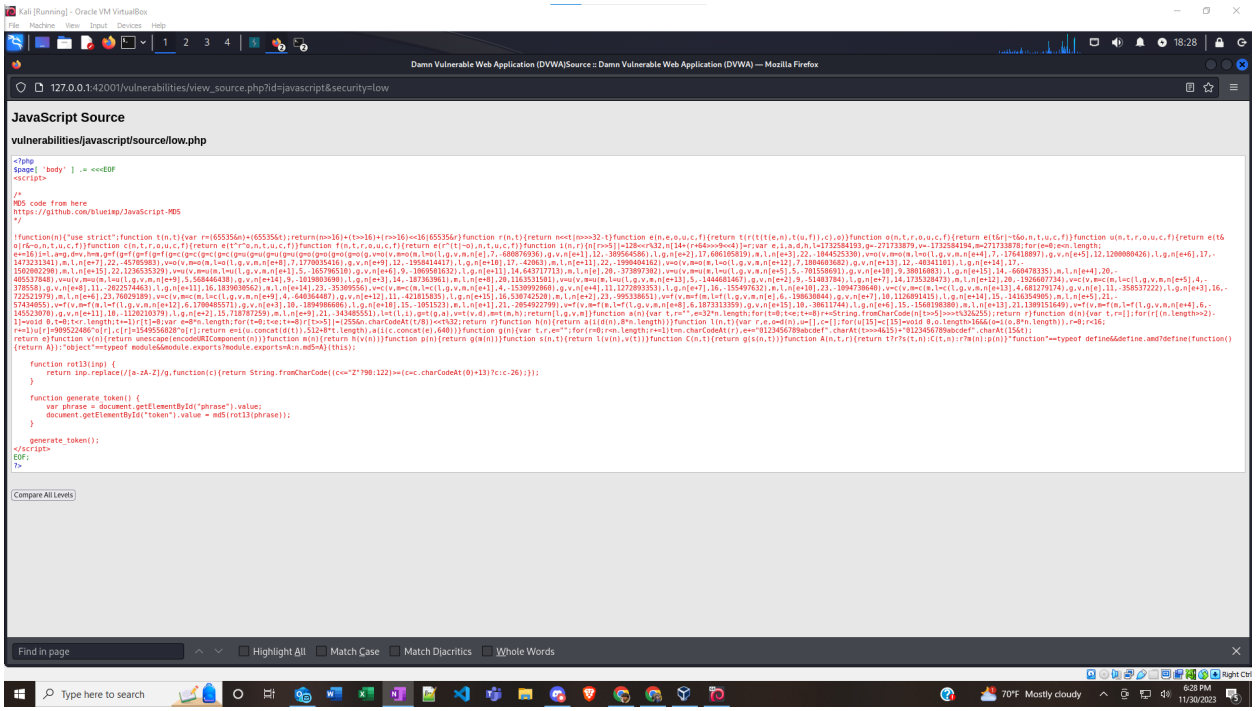

Figure 72: Source code displays the JavaScript functions used to generate the token.

Step 3: Reproduce the correct token.

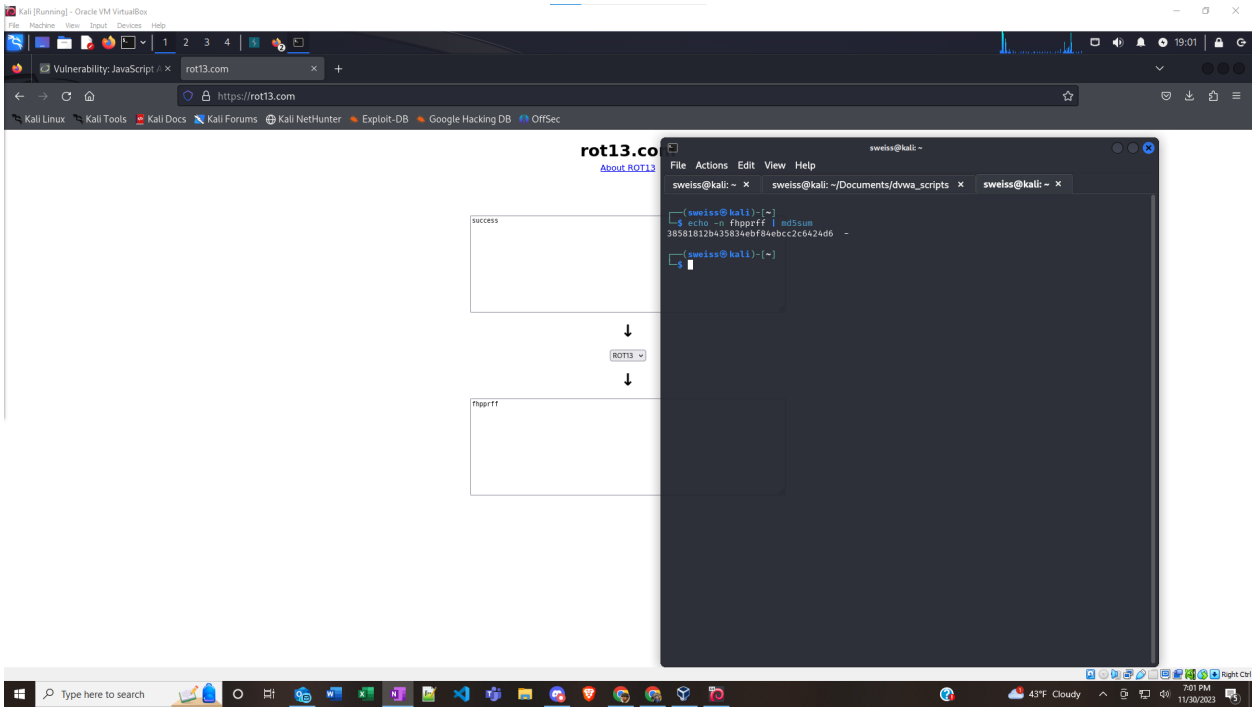

Figure 73: The correct token can be reproduced by online and/or command line tools.

Step 4: Use JavaScript in the console to set the “token” value.

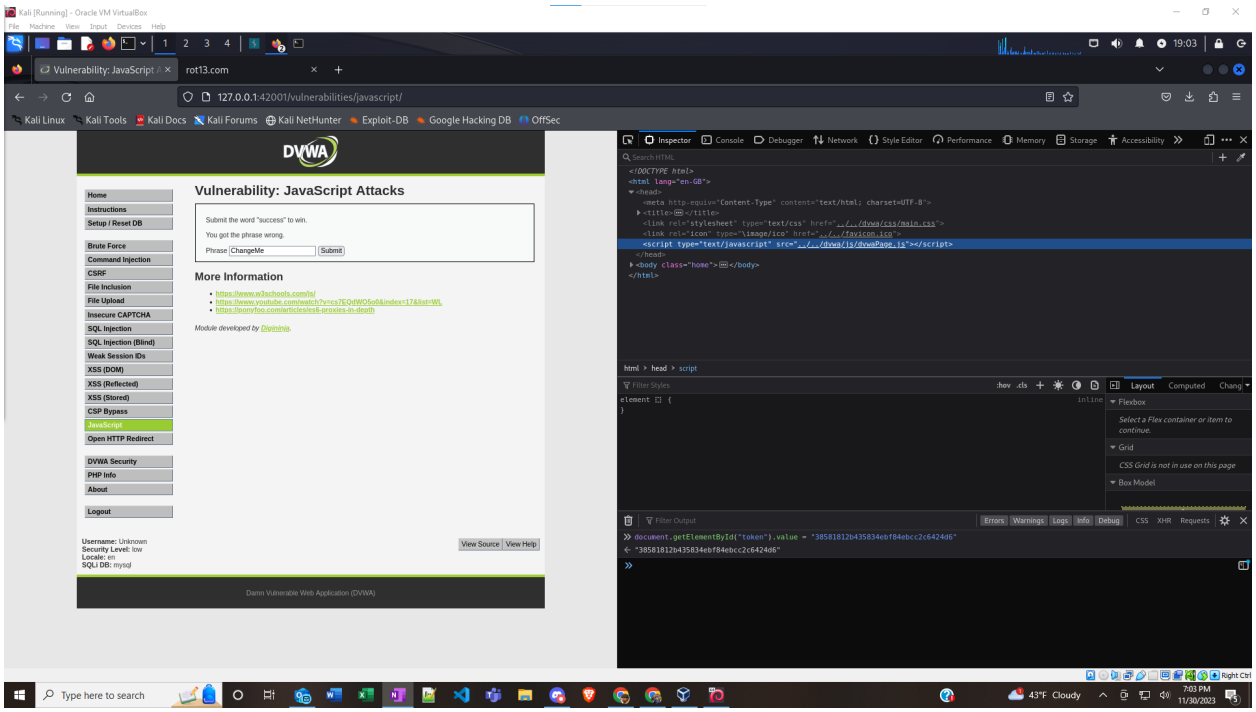

Figure 74: JavaScript is used to set the token with the correct value.

Step 5: Click the Submit button.

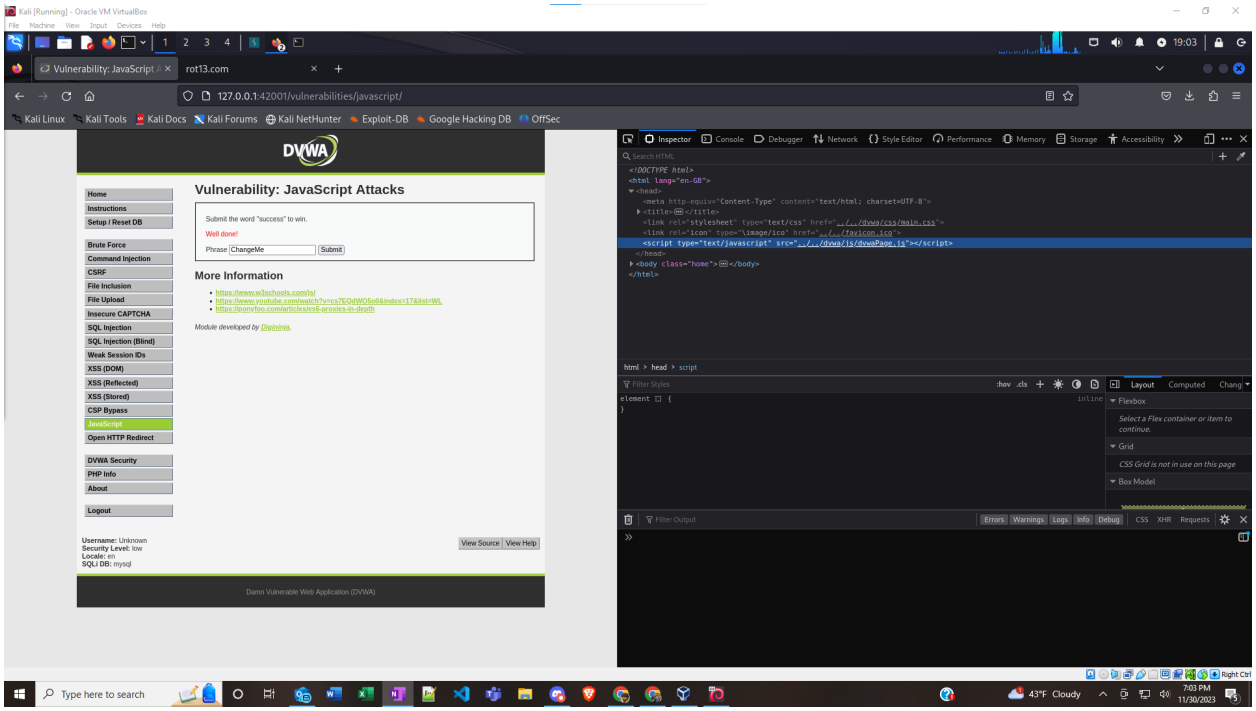

Figure 75: After the token is set with the correct value, the validation message is displayed when the Submit button is clicked.

Step 6: Refresh the app and try an alternate method. Confirming that the token is no longer correctly set.

Step 7: Enter “success” in the text field but do not hit Submit.

Step 8: Call the `generate_token()` method in the console.

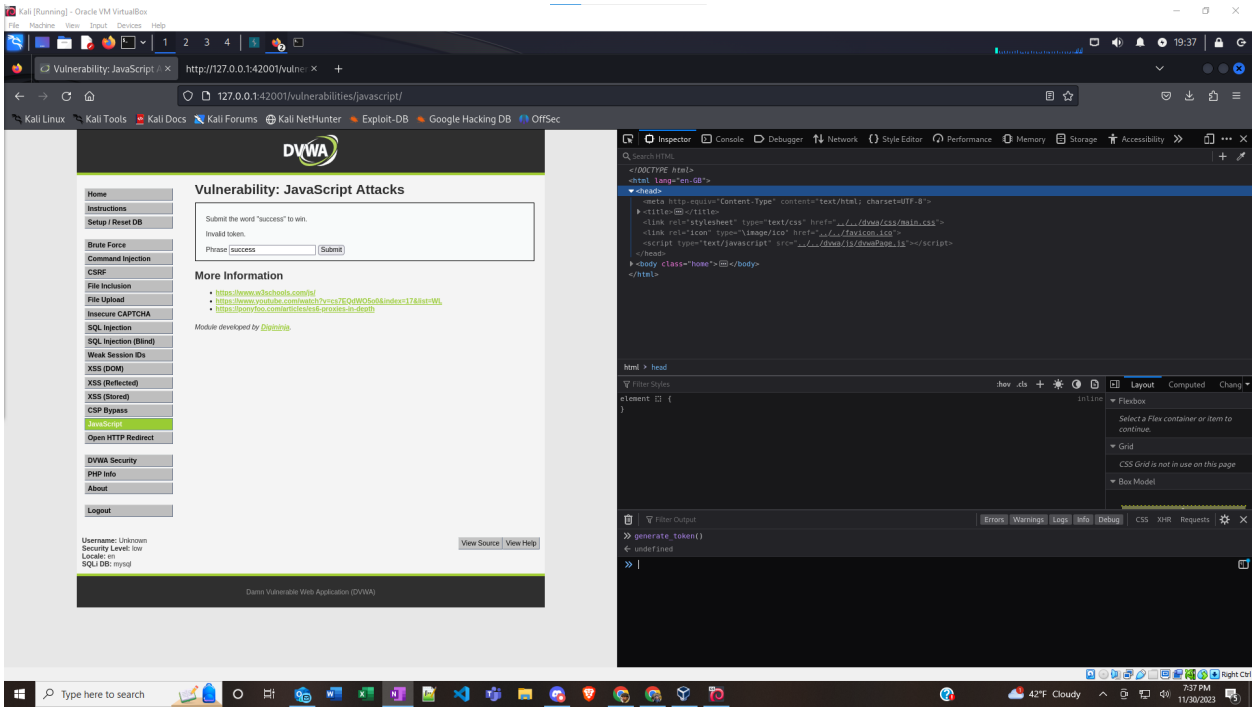

Figure 76: The `generate_token()` method is called from the console with “success” in the text entry box.

Step 9: Click the submit button and view the validation message.

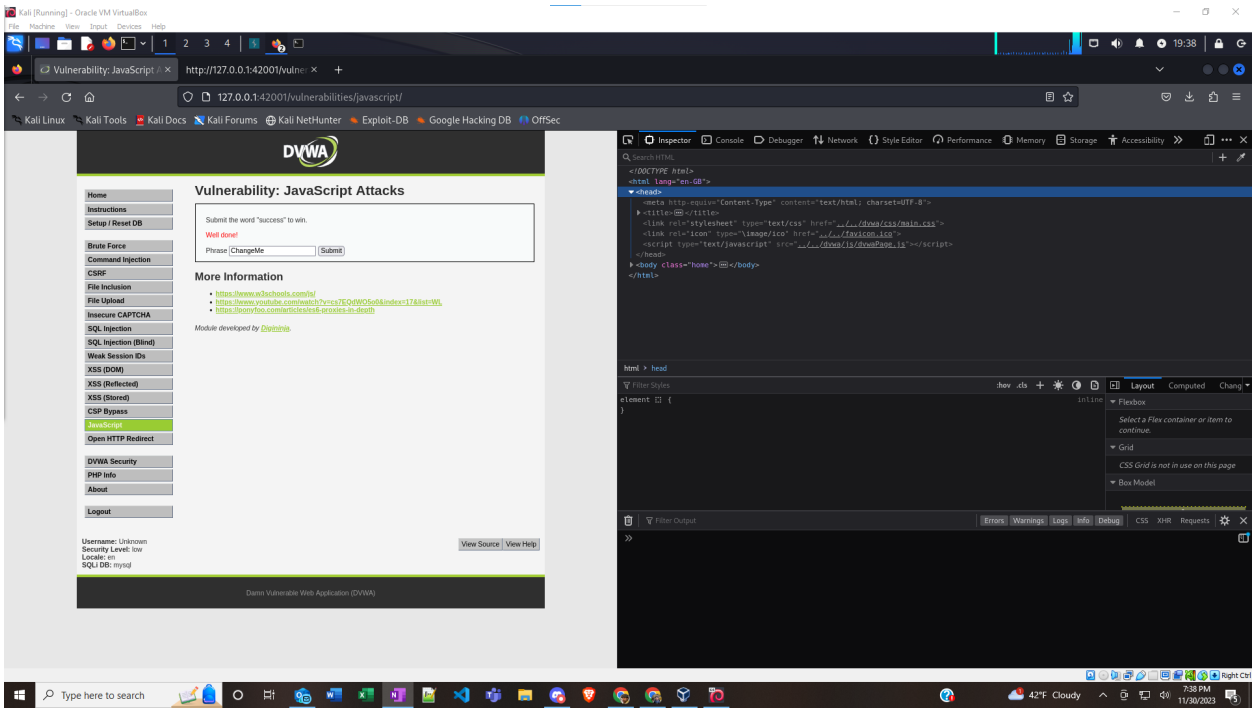

Figure 77: Validation message displayed.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection

Resources

- Mark R. (2020, Jan 12). *Challenge 14: Javascript.md*. GitHub. https://github.com/keewenaw/dvwa-guide-2019/blob/master/low/Challenge%2014%3A%20Javascript.md
- CryptoCat. (2021, February 27). *14 - JavaScript (low/med/high) - Damn Vulnerable Web Application (DVWA)* [Video]. YouTube. https://www.youtube.com/watch?v=3IfHy97pog0&list=PLHUKi1UlEgOJLPSFZaFKMoexpM6qhOb4Q&index=16

Vulnerability 13 - Open HTTP Redirect

How does this feature normally work?

This feature has two links, each to a page displaying a quote. The quote id is displayed in the URL query string when the link is clicked.

What does it take to exercise the vulnerability?

This vulnerability is easily exercised by altering the end of this URL by adding a redirect field in the query string.

How did the feature work differently than normal use?

Instead of bringing up one of the quote pages, the URL now directs the user to the redirect site. The GET request shows a 302 Found code, which signifies a successful redirect.

Why did this work differently?

This works because the application does not have any limitations on redirecting. When the server sees the redirect field in the request, it completes this request by redirecting to the requested link.

Why we should care about this vulnerability and potential loss

This can be used to make a more convincing phishing link. A user inspecting the link might see the “trusted” domain at the front of the URL without noticing the redirect. If they do notice the redirect, the address can be crafted to look similar to a legitimate domain name, or it can be obfuscated by using URL encoding. The redirection can take the user to an imposter site where they may enter their username and password which is then stored in the attacker's website for later exploitation, or the site may distribute malware.

Briefly describe how to fix the vulnerability

The View Help button on the application page solves this issue on the Impossible security level by parameterizing and hard coding the queries. This prevents the redirect field from being abused.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality.

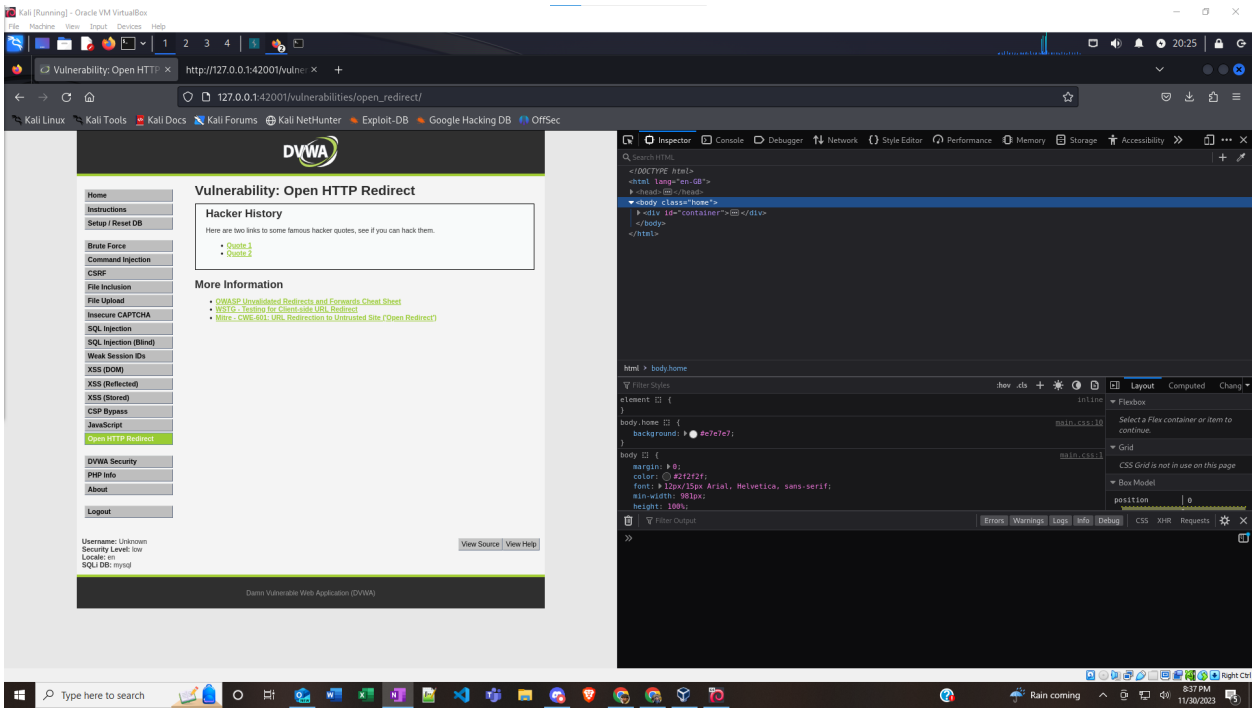

Figure 78: Initial application page has two Quote links.

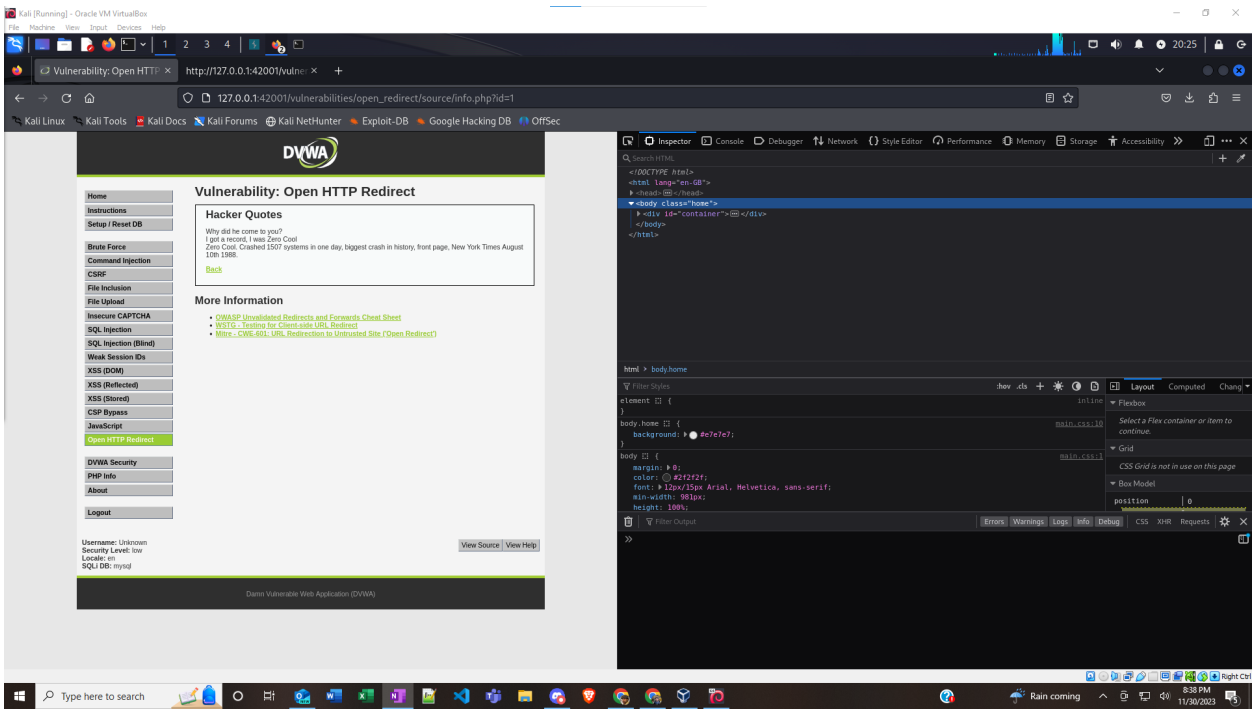

Figure 79: Results of clicking a Quote link.

Step 2: Modify the URL to contain a redirect element in the query string.

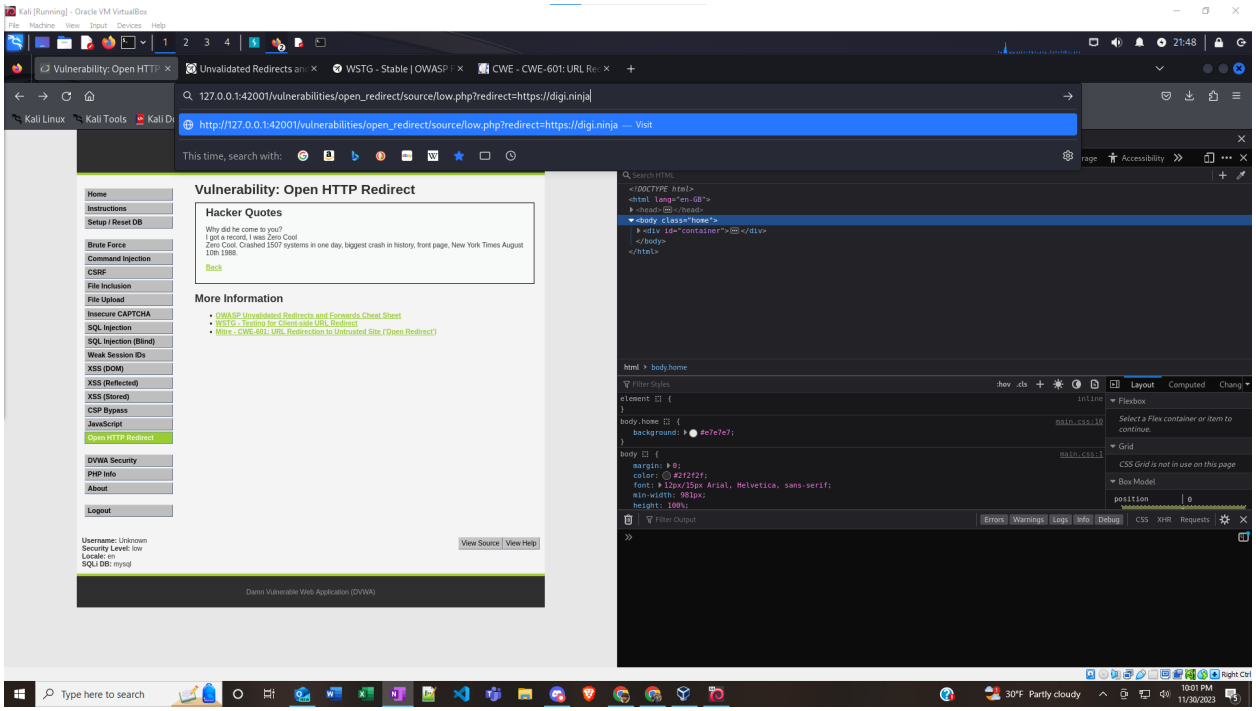

Figure 80: A modified URL containing a redirect element. This specific redirect was suggested by the View Help button on the application.

Step 3: Enter the link and view confirm the 302 Found code in the Inspector window.

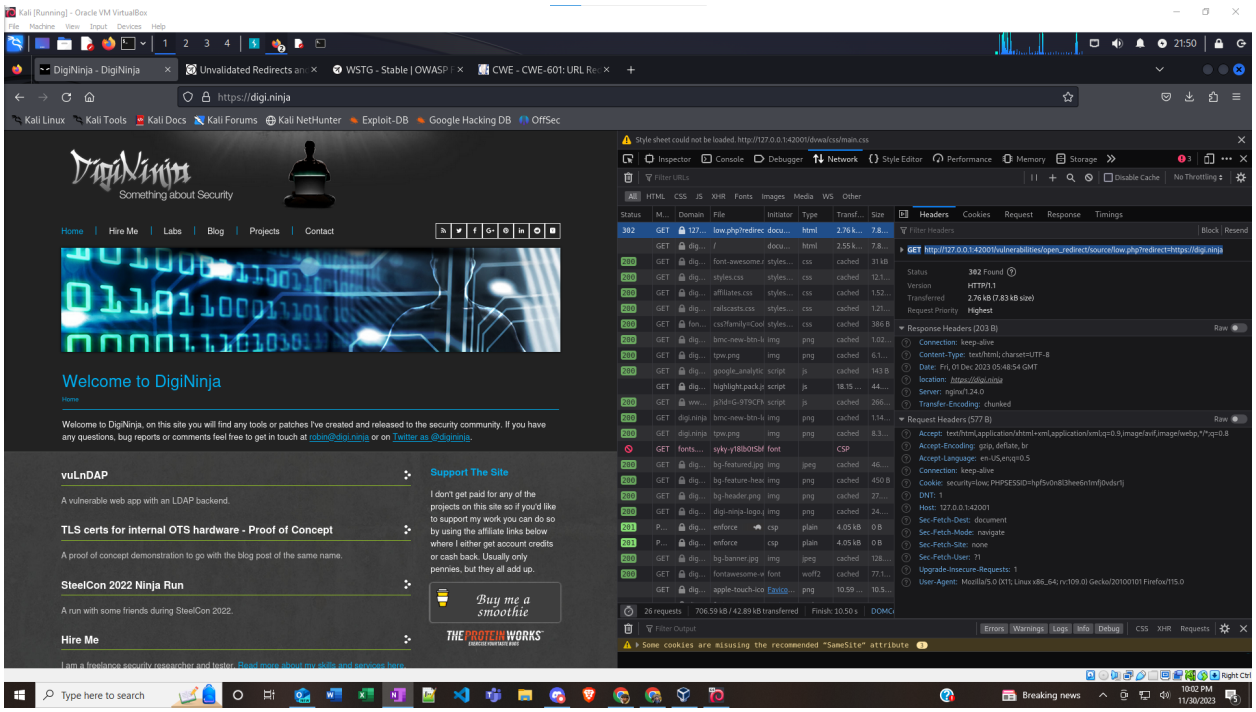

Figure 81: The URL in the address bar displays the redirected site. The GET request in the Inspector window shows the full redirect URL and teh 302 Found confirmation code.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A01:2021 – Broken Access Control

Resources

- "View Help" button on the application
- (2023). *Unvalidated Redirects and Forwards Cheat Sheet*. OWASP. https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html
- (n.d.). *Testing for Client Side URL Redirect*. OWASP. https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/11-Client_Side_Testing/04-Testing_for_Client_Side_URL_Redirect

Seth Weiss
weissse@oregonstate.edu

Vulnerability 14 - SQL Injection (Blind)

How does this feature normally work?

The feature contains a text entry box where a user is prompted to enter a User ID and click the Submit button. If that User ID exists in the application database, then a confirmation message that reads “User ID exists in the database.” appears in red text in the response page. If the ID does not exist, then the message “User ID is MISSING from the database.”

What does it take to exercise the vulnerability?

Using a Boolean check or a sleep timer will reveal if the page is susceptible to SQL Injection. Both involve starting the entry with a known User ID followed by a single quotation mark (such as `1'`). Following this quotation mark, a sleep timer can be entered, followed by a pound symbol (`#`) to negate the rest of the SQL query on the server. The response time should match the time set in the injection attack.

How did the feature work differently than normal use?

When the sleep timer method is used to confirm the potential for SQL injection, the response time from the server matches the injected time. When a Boolean false value is added to the query, a known User ID will return a “MISSING” response instead of the expected “exists” confirmation. When a Boolean true is included, the known User ID will return with the expected “exists” text.

Why did this work differently?

This works because even though the response will not display the direct information of the users, it will still return true or false, depending on the query results. This is used to gather information about the database.

Why we should care about this vulnerability and potential loss

Despite not divulging explicit information, this type of vulnerability allows an attacker to learn about the features of a database through a series of Boolean checks. Instead of asking the database for the passwords directly, the attacker asks a series of yes or no questions to build up an idea of what exists behind the scenes. Once enough information is gathered, the attacker can then target an attack on sensitive information allowing them to steal login credentials.

Briefly describe how to fix the vulnerability

The fixes to Blind SQL Injection are the same as those of the standard form of SQL Injection. As described in the View Help button on the application, parameterizing the queries so that no dynamic user data is used for a query is the ultimate fix.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore the intended feature functionality.

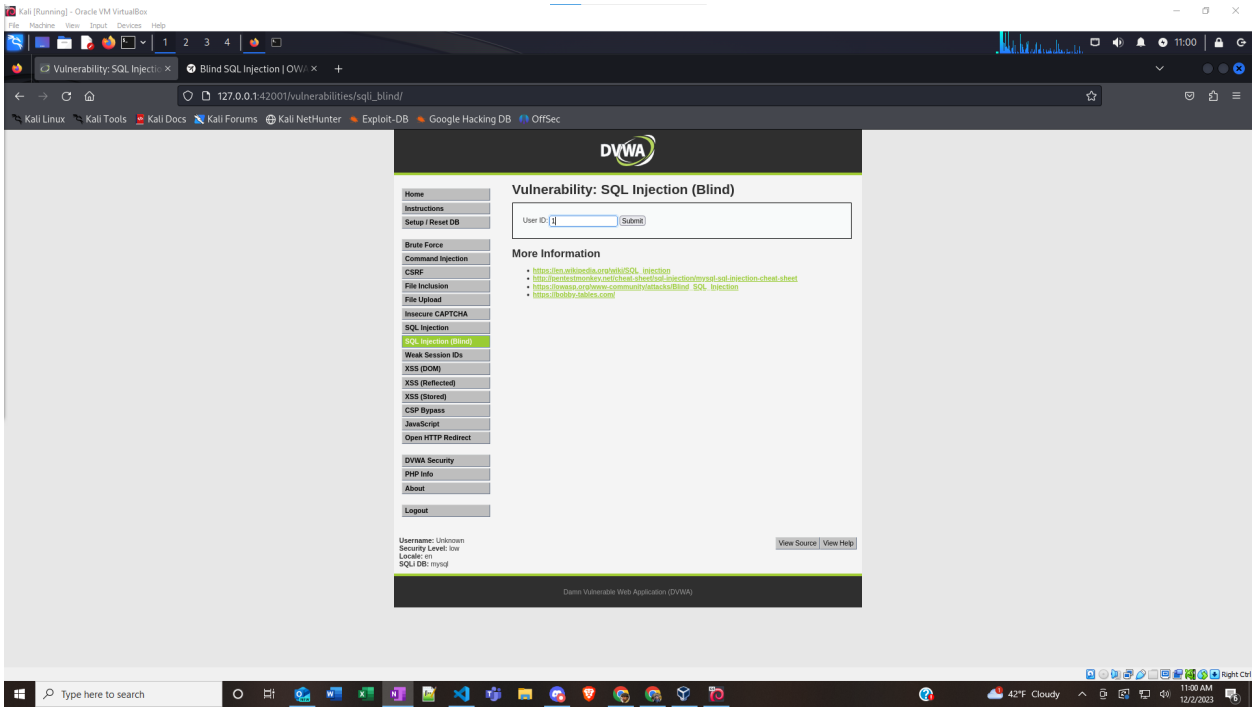

Figure 82: Initial application feature allows for entry of a User ID.

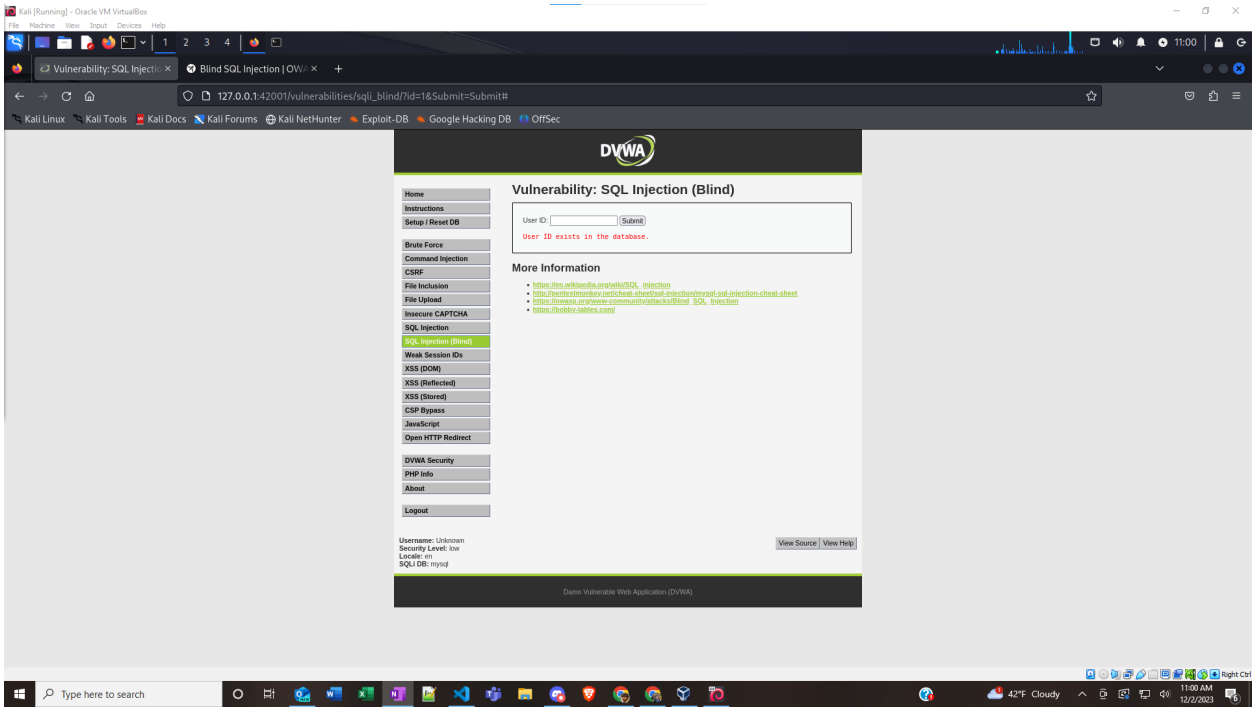

Figure 83: Results of entering a User ID that exists in the database.

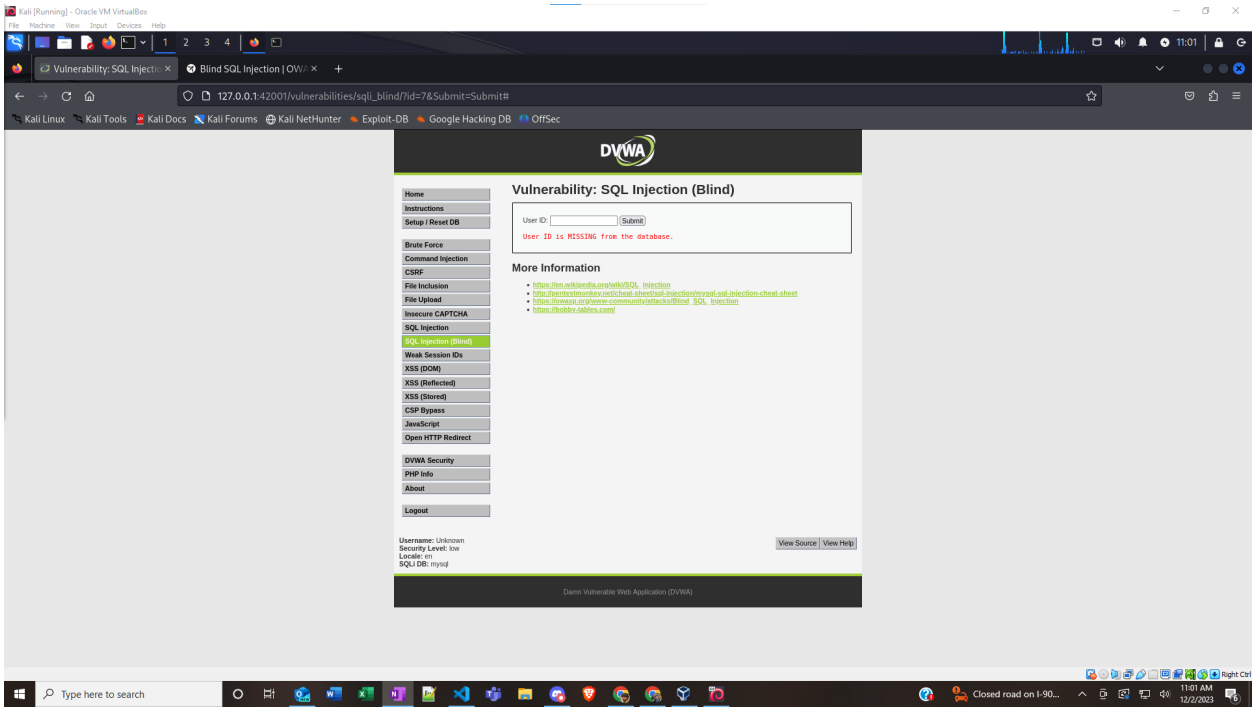

Figure 84: Results of entering a User ID that does *not* exist in the database.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Step 2: Attempt a sleep timer injection to confirm that SQL Injection is possible.

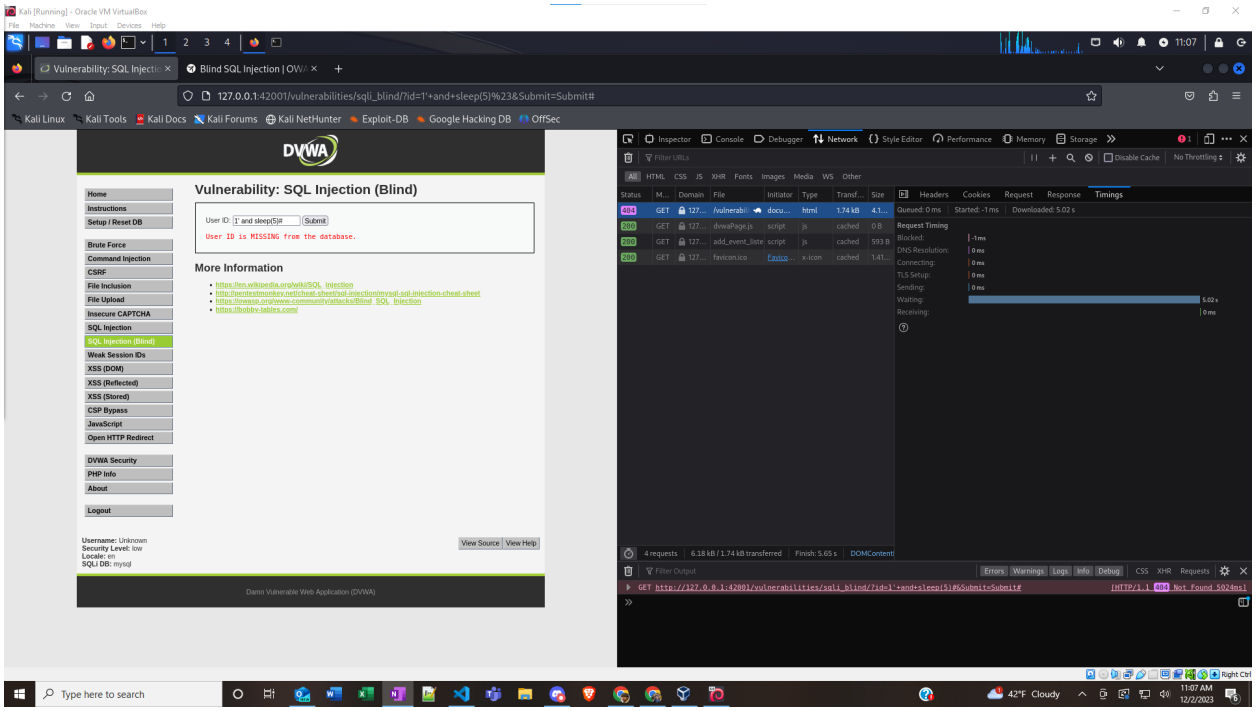

Figure 85: Results of a sleep timer injection. The Network tab in the Inspector window reveals the response time took the same amount of time as the sleep timer.

Step 3: Use Boolean checks to determine the number of columns in the database table.

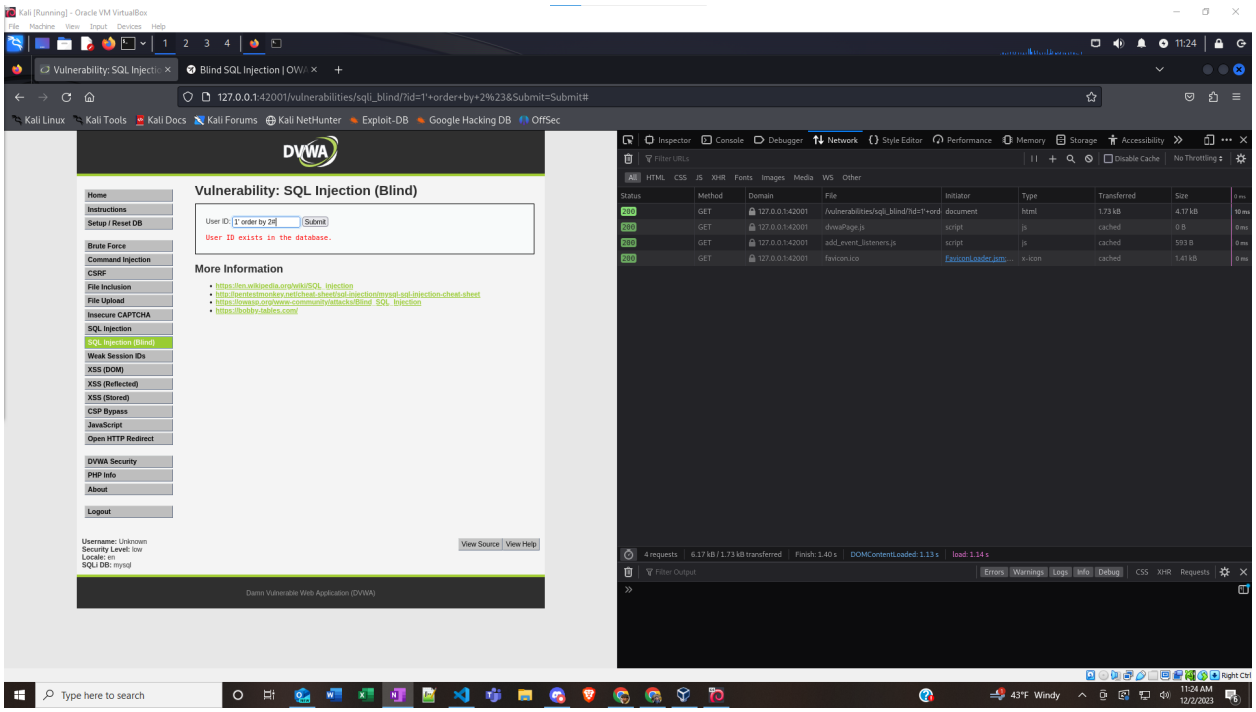

Figure 86: Checking for two columns results in a Boolean true.

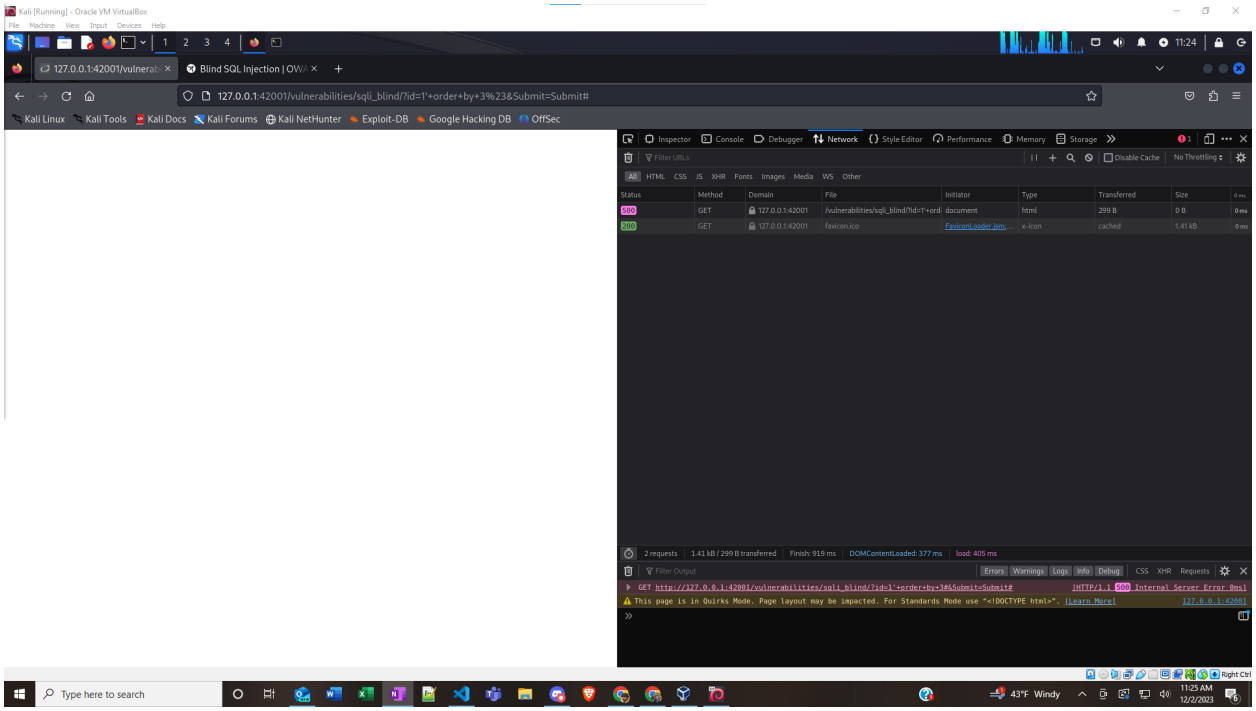

Figure 87: Checking for 3 columns results in an error, further suggesting that there are two columns in the database table.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

Step 4: Use Boolean statements to check the length of the database table.

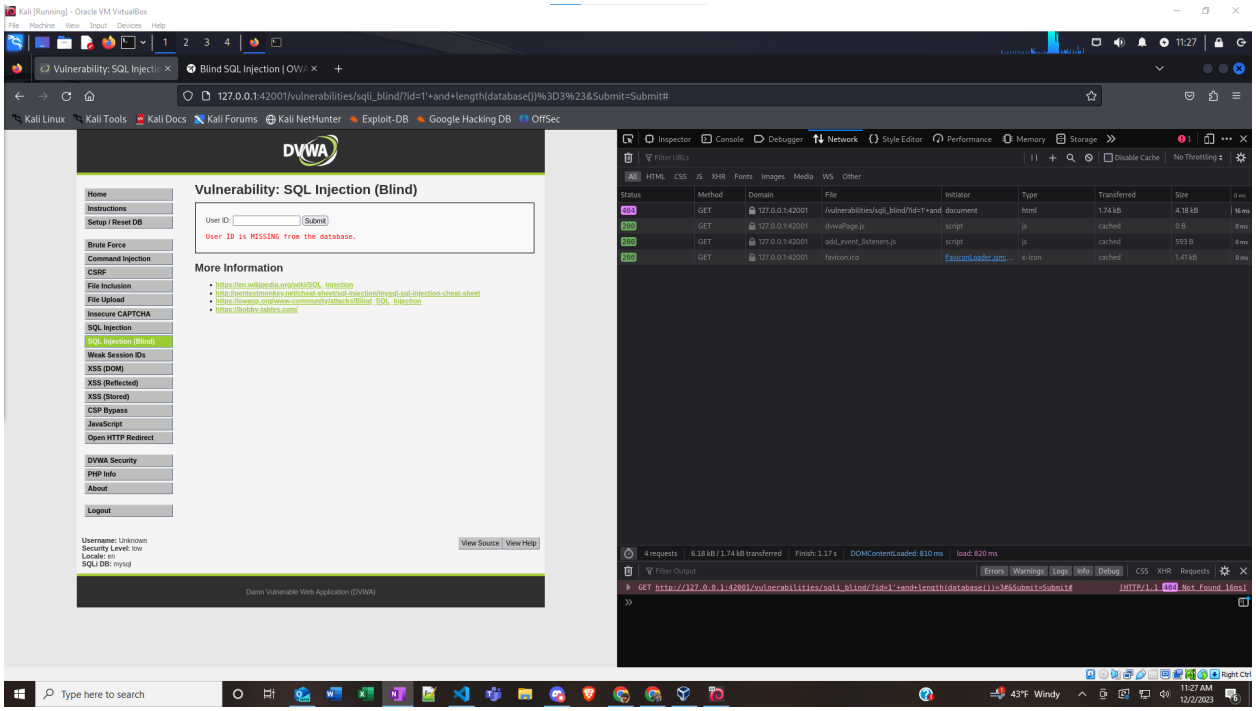

Figure 88: Checking for a database length of 3 results in a Boolean false.

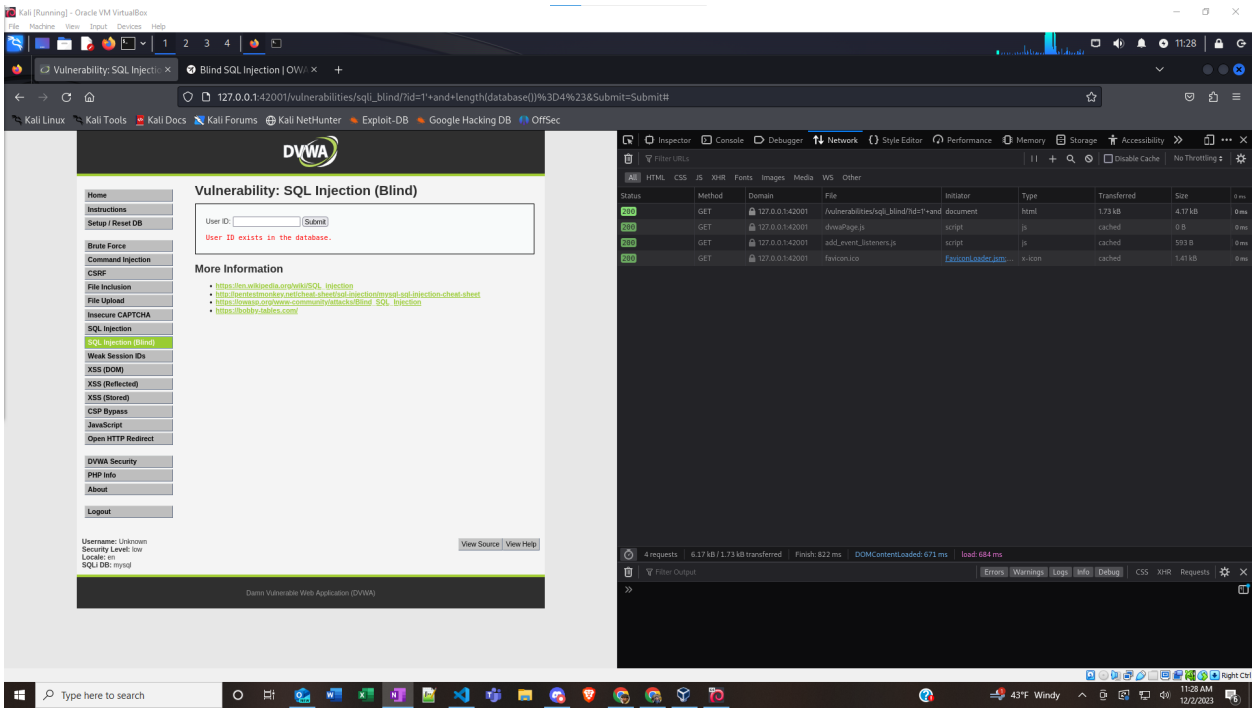

Figure 89: Checking for a database length of 4 results in a Boolean true.

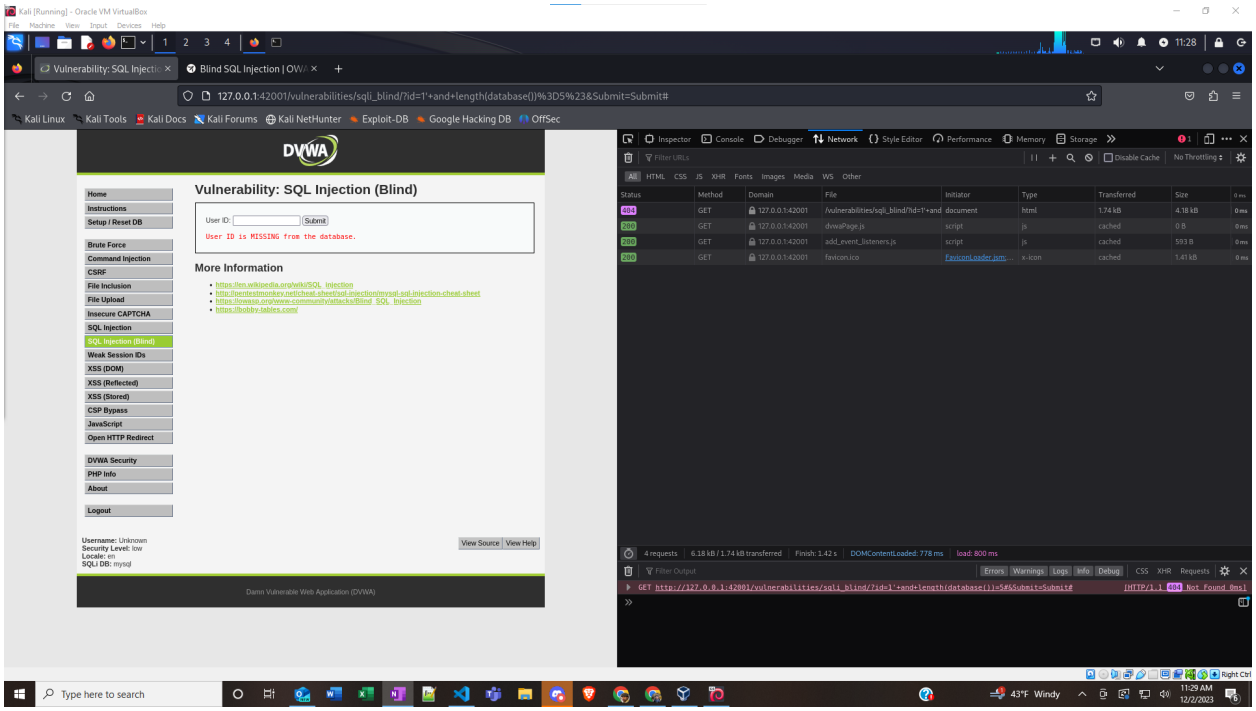

Figure 90: Checking for database length of 5 results in a Boolean false. This confirms there are 4 entries in the database.

Step 5: Enter a known User ID to obtain a GET request. This will be used in the sqlmap tool.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

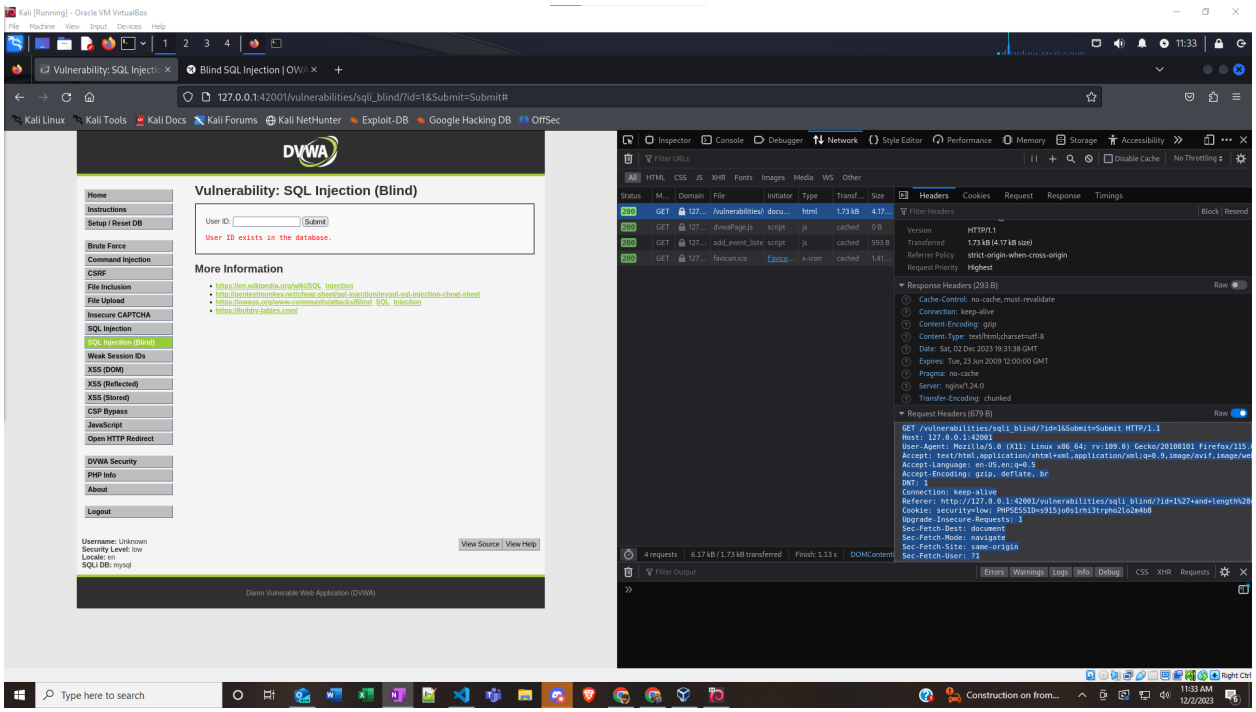

Figure 91: Copy the GET request for a known User ID.

Step 6: Save the GET request in a text file and run sqlmap with it. The sqlmap tool automates Blind SQL Injection queries and reports its findings. It then offers options such as trying the found vulnerabilities to dump the database contents and cracking obtained passwords.

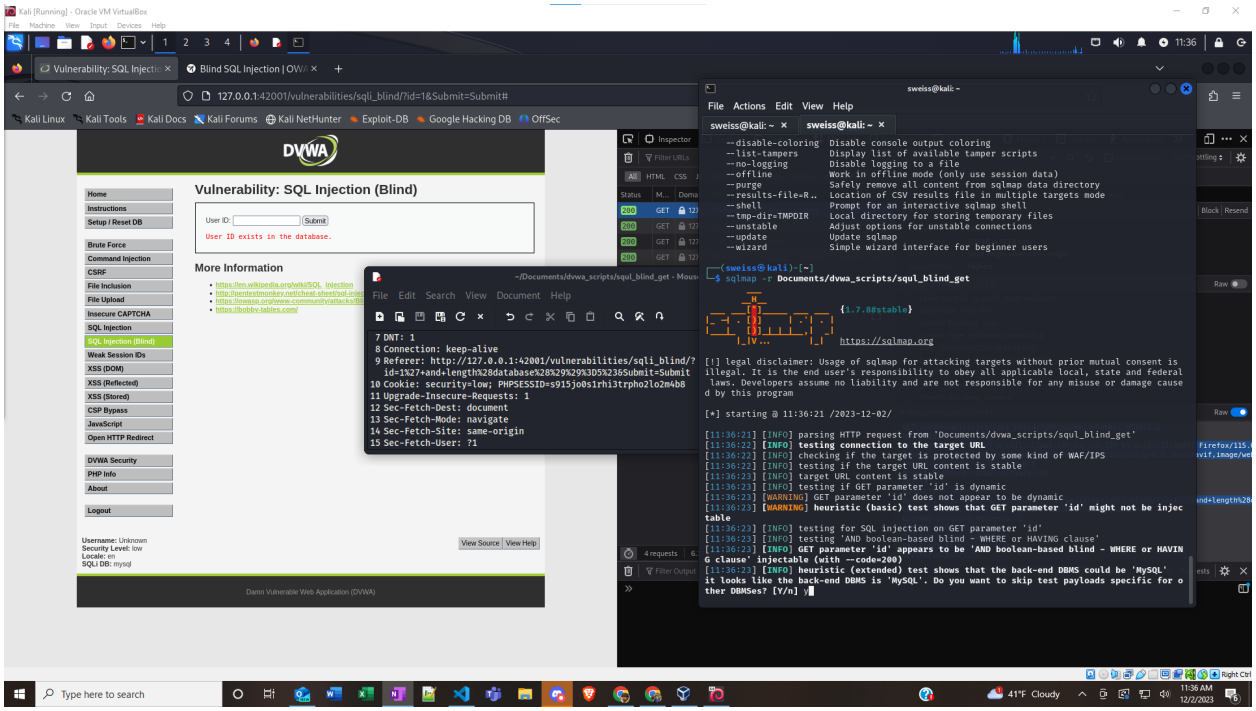

Figure 92: Run sqlmap with the saved GET request.

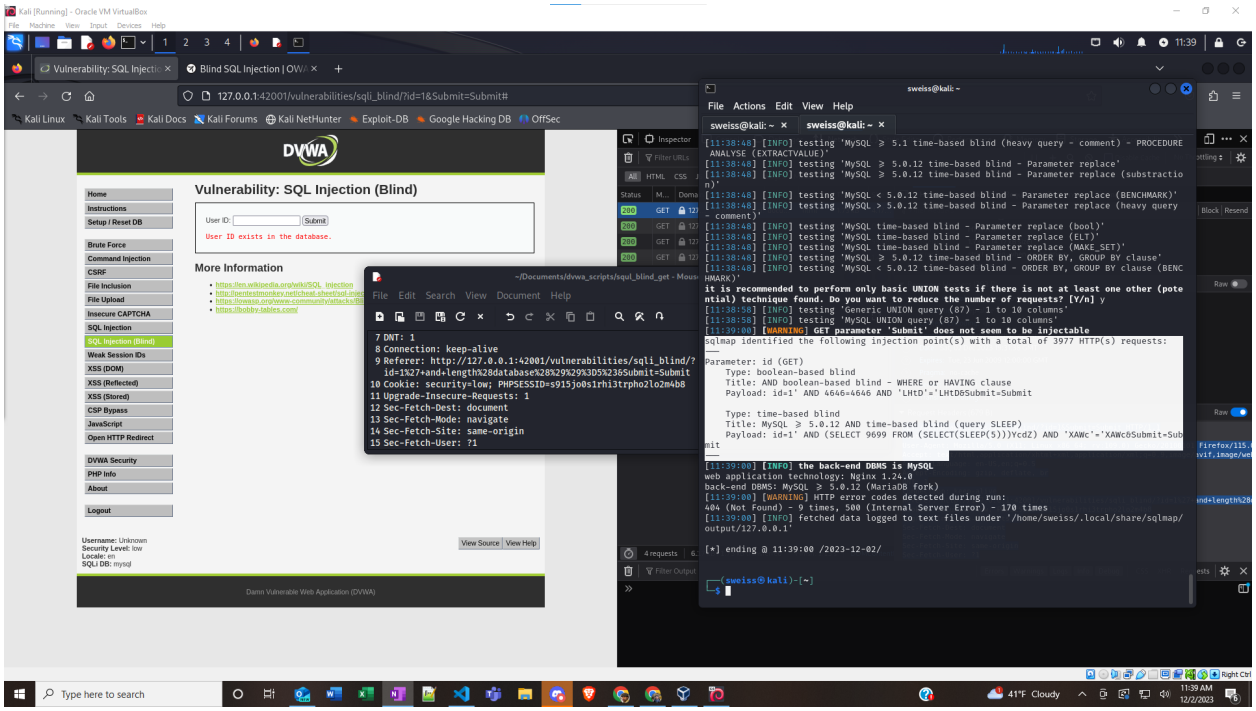

Figure 93: Resulting GET request from sqlmap analysis.

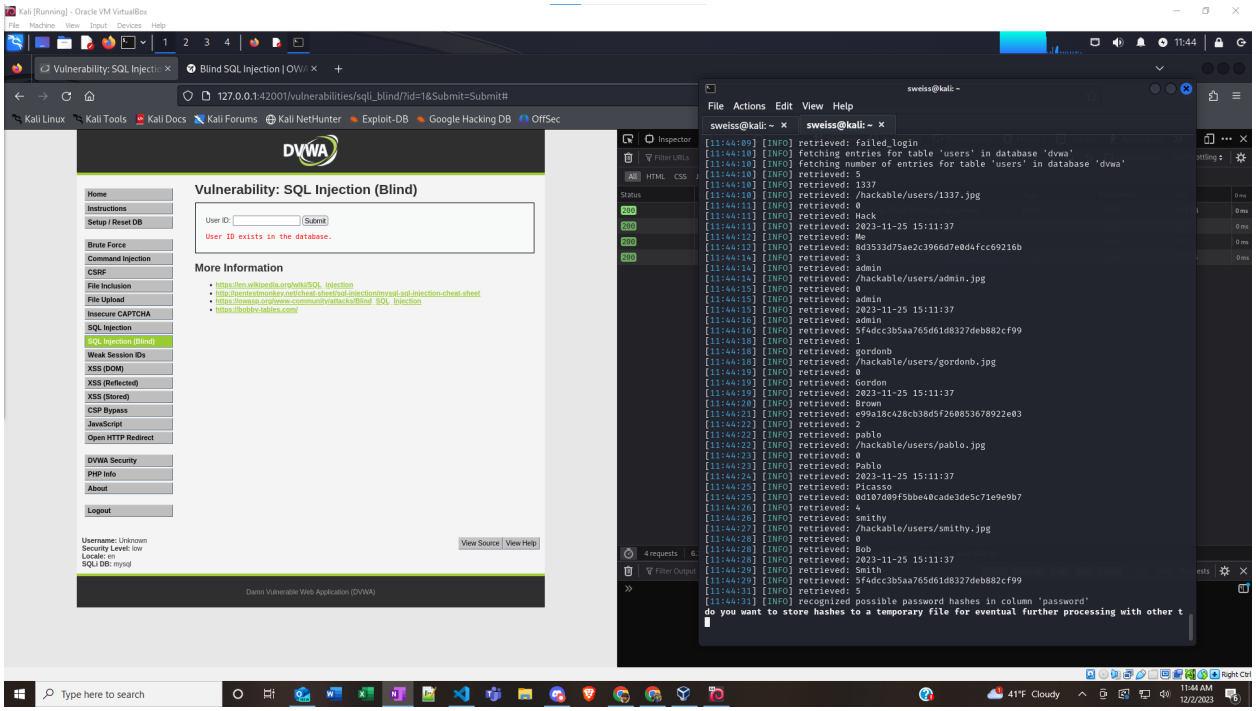

Figure 94: Found username and password information from sqlmap `–dump` option.

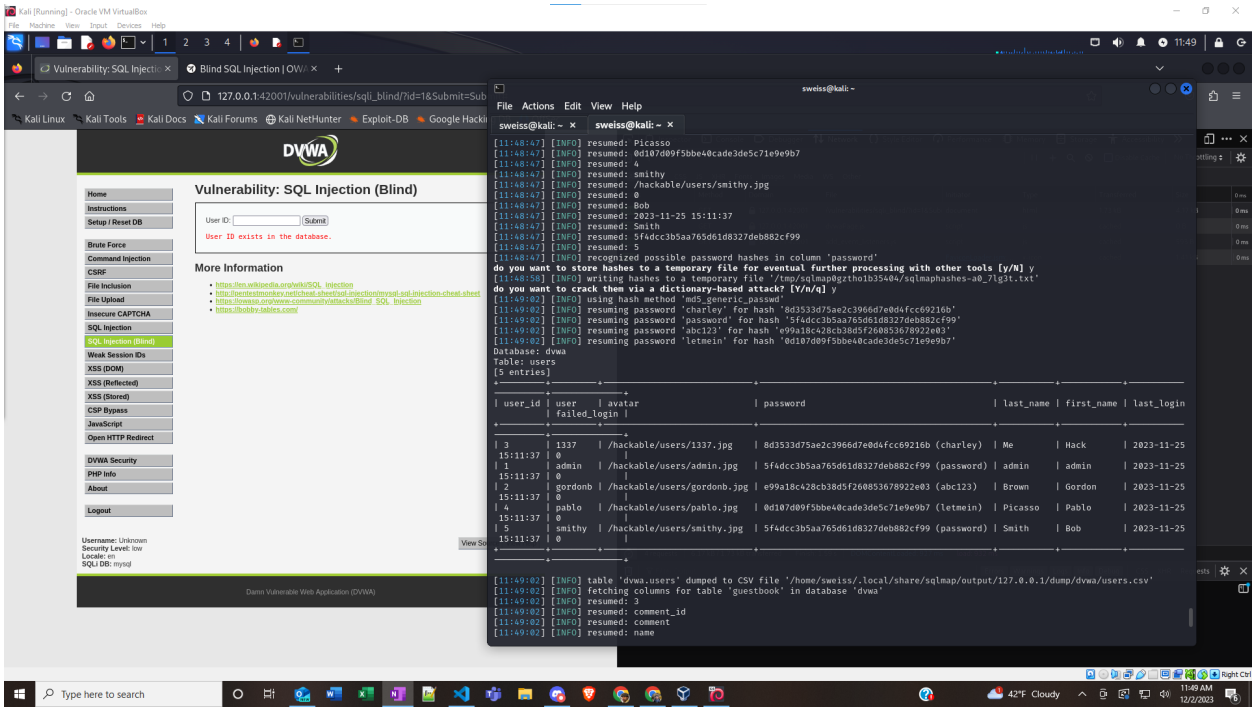

Figure 95: Passwords cracked by sqlmap.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A03:2021 – Injection

Resources

- Kingthorin. (n.d.). *Blind SQL Injection*. OWASP. https://owasp.org/www-community/attacks/Blind_SQL_Injection
- CryptoCat. (2021, February 27). *8 - Blind SQL Injection (low/med/high) - Damn Vulnerable Web Application (DVWA)* [Video]. YouTube. https://www.youtube.com/watch?v=uN8Tv1exPMk&list=PLHUKi1UlEgOJLPSFZaFKMoexpM6qhOb4Q&index=10

Vulnerability 15 - Insecure CAPTCHA

How does this feature normally work?

After registering a reCAPTCHA and including the keys in the config file, this feature has two text entry boxes, one for a password to change to, and the second to confirm the password. Beneath this is a reCAPTCHA check box to ensure that the user is not a robot. Upon checking the box, the user is presented a photo challenge where they are to select all photos containing the requested item. Upon completion, the user can then click the Change button, which brings up a response page with the text “You passed the CAPTCHA! Click the button to confirm your change.” There is another Change button and when the user clicks this one, a response page displaying the message “Password Changed.” appears. The user can then confirm this change by logging out of the application and logging back in. If the user attempts to change the password without clicking the reCAPTCHA checkbox, the text “The CAPTCHA was incorrect. Please try again.” is displayed.

What does it take to exercise the vulnerability?

Using the Burp Suite Browser allows the Burp Proxy to keep track of all requests. When the user is presented with the password change confirmation page, the POST request for this page can be found in the Burp Proxy. In this request the information that this is step 2 is next to the plaintext password and confirmation password. Sending this request to the Burp Repeater allows the attacker to change the value of the password and resend the request. Because the step 2 is included, it bypasses the reCAPTCHA and directly sends a request to the password change page.

How did the feature work differently than normal use?

This feature works differently in that the reCAPTCHA can be bypassed and only now requires the password and the confirmation password.

Why did this work differently?

This works differently because of the two step process to confirming the password change. Because the reCAPTCHA confirmation happens in a separate step before the password change step, it can be bypassed if the attacker knows the correct request to send directly to the second step.

Why we should care about this vulnerability and potential loss

This is an important vulnerability to be aware of because even if an attacker is only able to change their own password in this way, they are now able to mount a brute force attack against other users. Once the other user accounts have been enumerated, the typical risks of account compromise apply (credit card theft, identity theft, account takeover, vandalism, stolen trade secrets, etc.).

Briefly describe how to fix the vulnerability

The Impossible mode of the application combines the reCAPTCHA and password reset in a single step. In this way, the reCAPTCHA cannot be bypassed.

Browser screenshot of each step in the sequence of exercising the vulnerability

Step 1: Explore intended functionality. (After setting up reCAPTCHA)

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

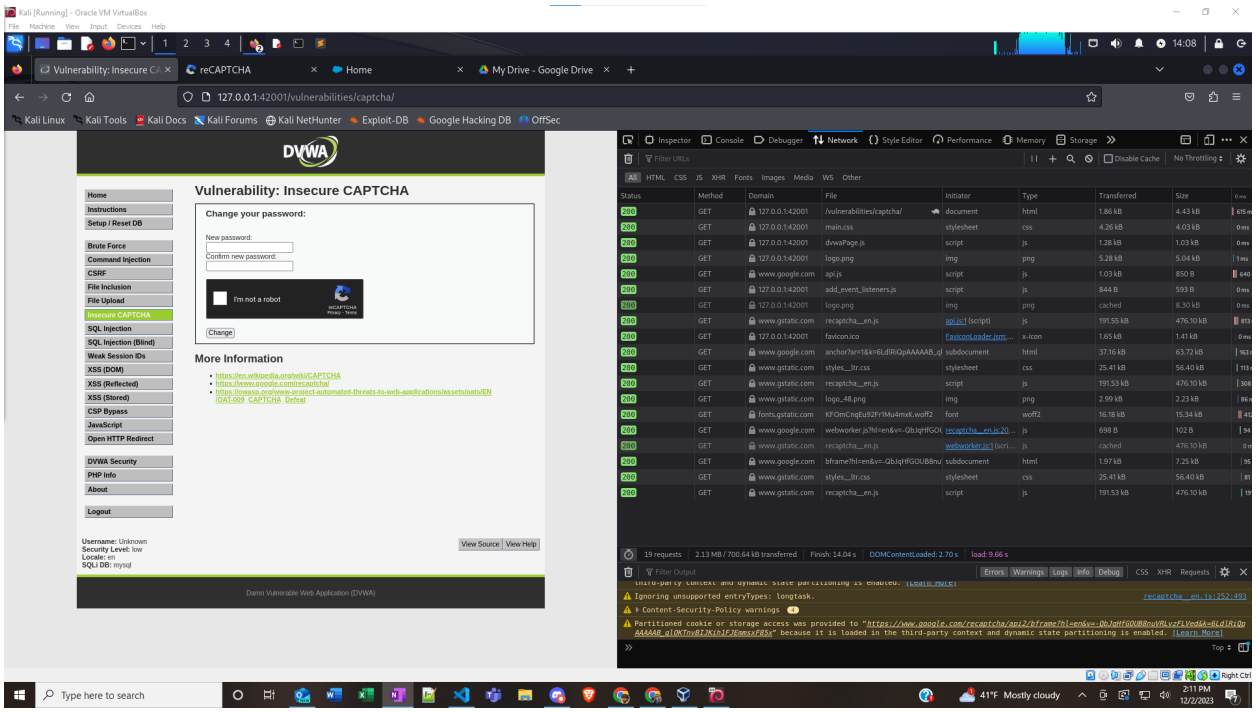

Figure 96: Password change screen with reCAPTCHA.

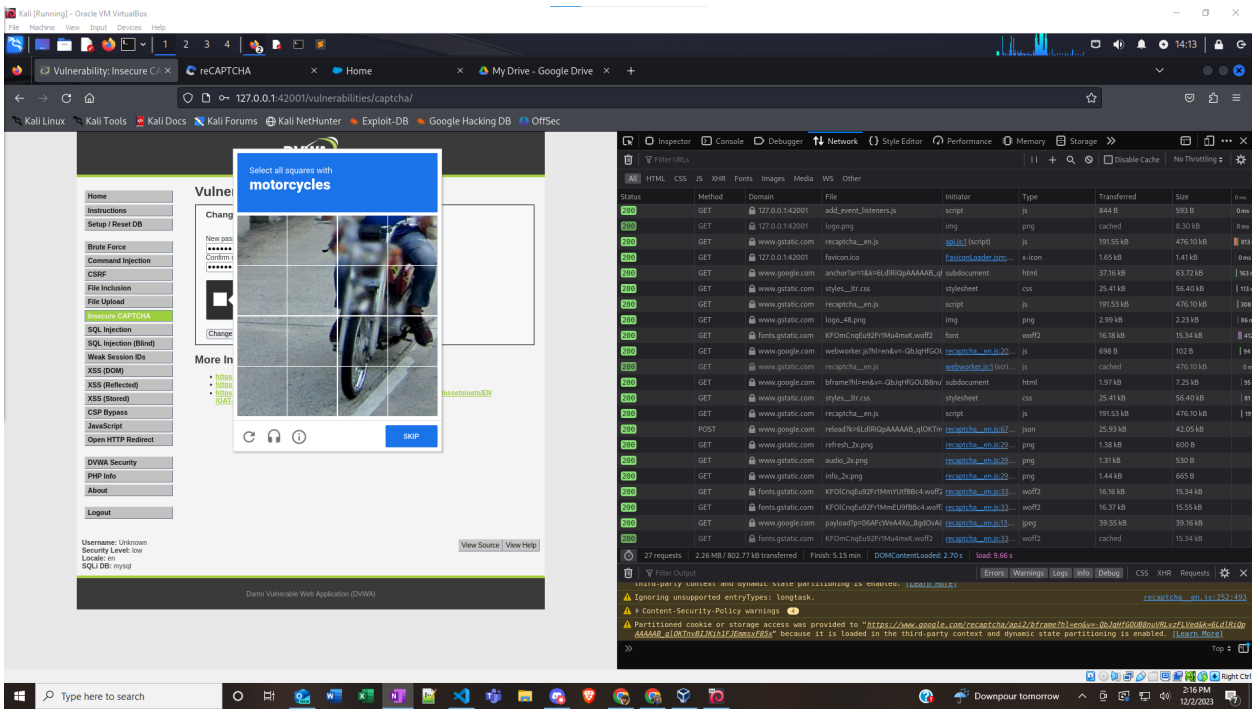

Figure 97: reCAPTCHA photo challenge.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

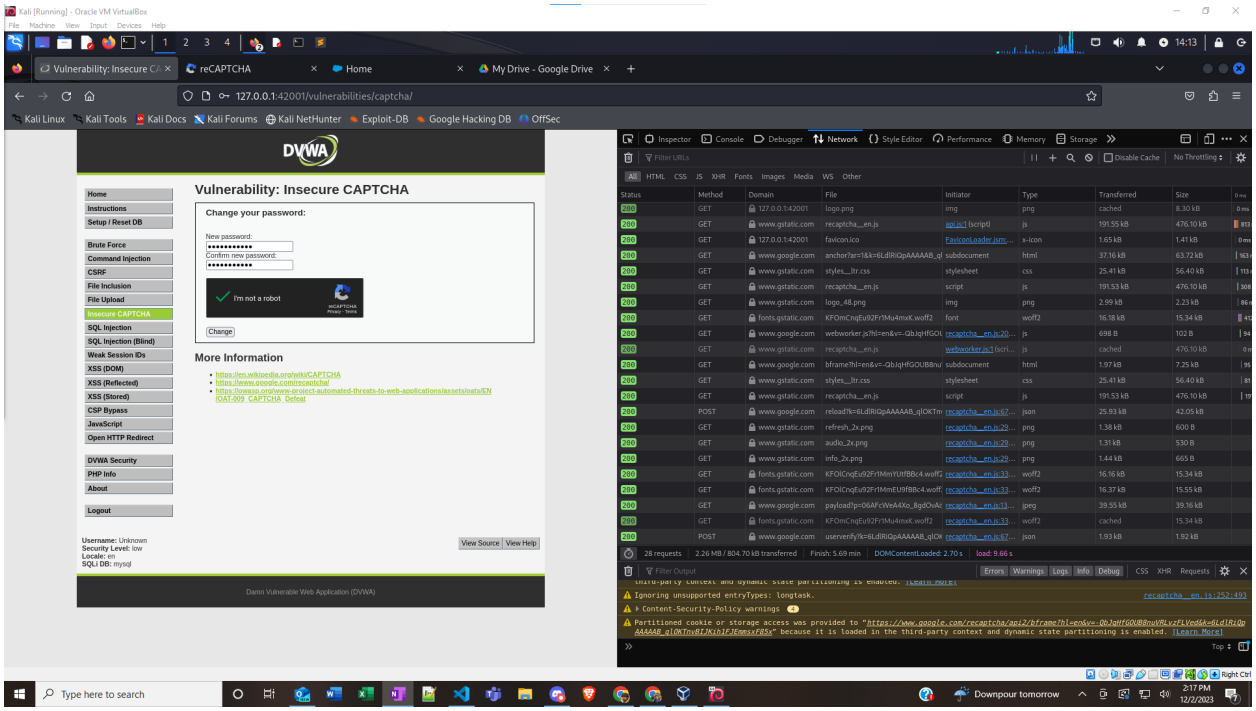

Figure 98: reCAPTCHA challenge complete, password change queued in entry fields.

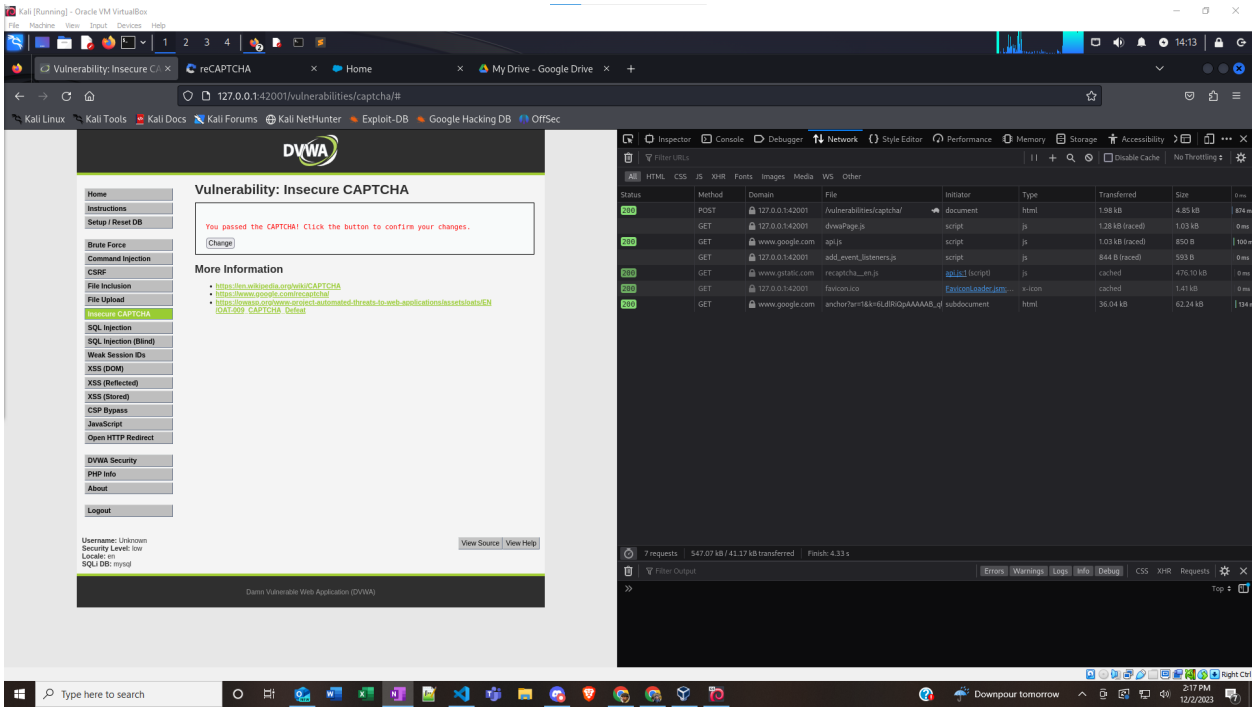

Figure 99: Successful reCAPTCHA confirmation page.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

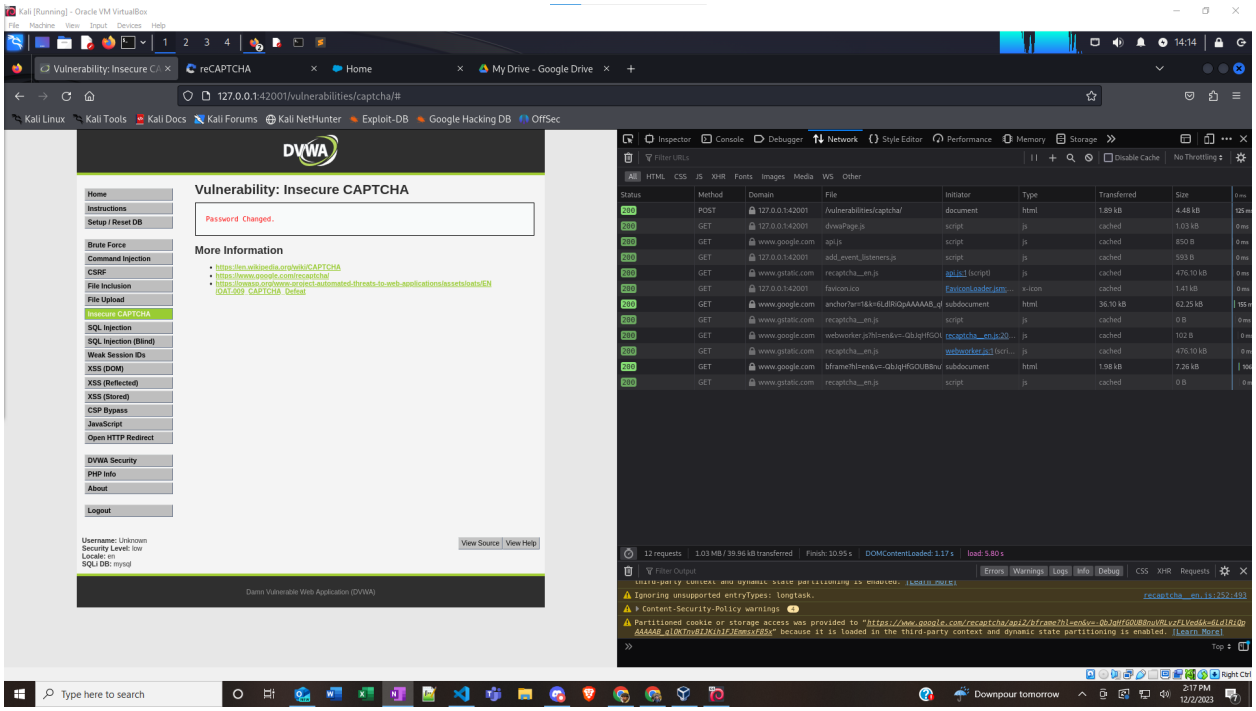

Figure 100: Successful password change page.

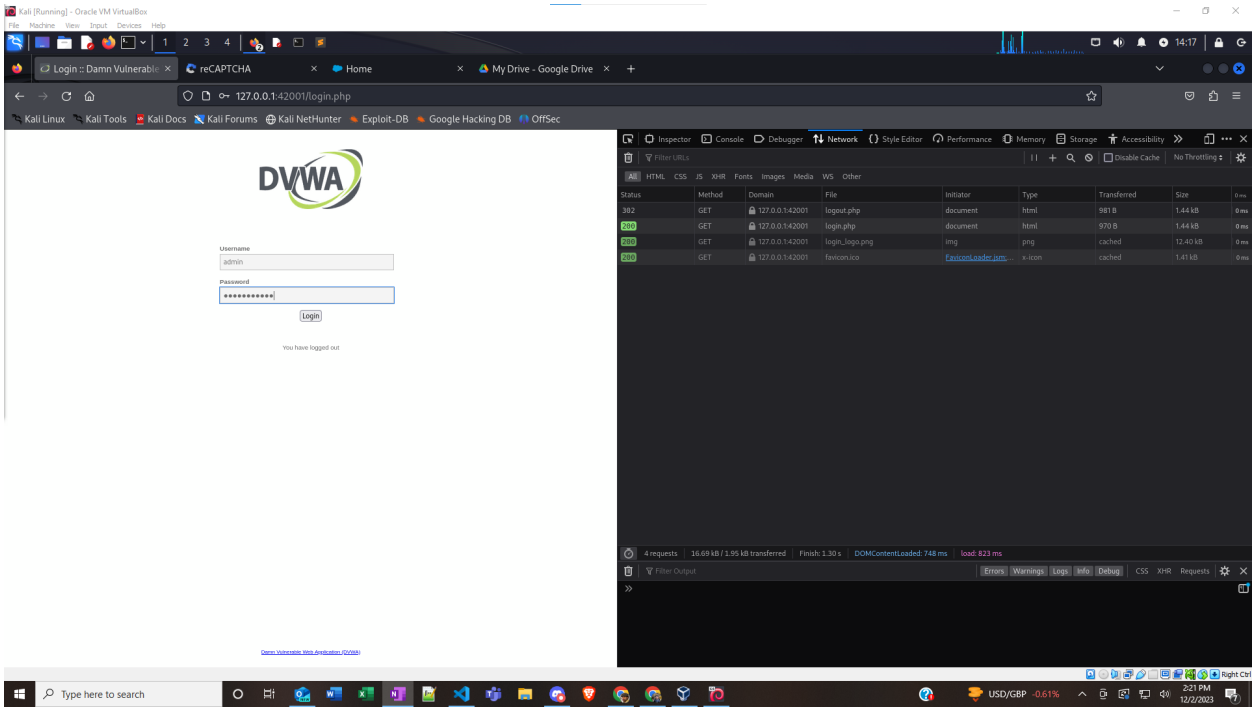

Figure 101: Log out and log back in to confirm password change.

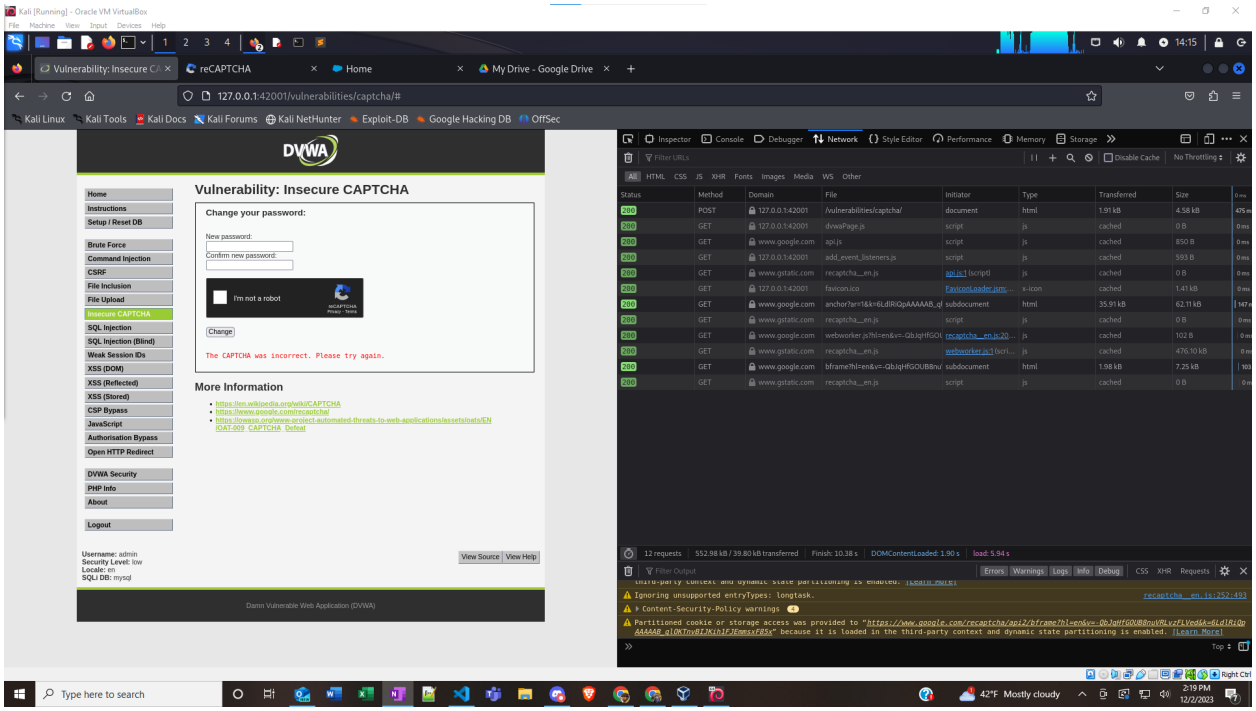

Figure 102: Result of attempting to change password without checking the reCAPTCHA box.

Step 2: Open application in Burp Browser and change the password again.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

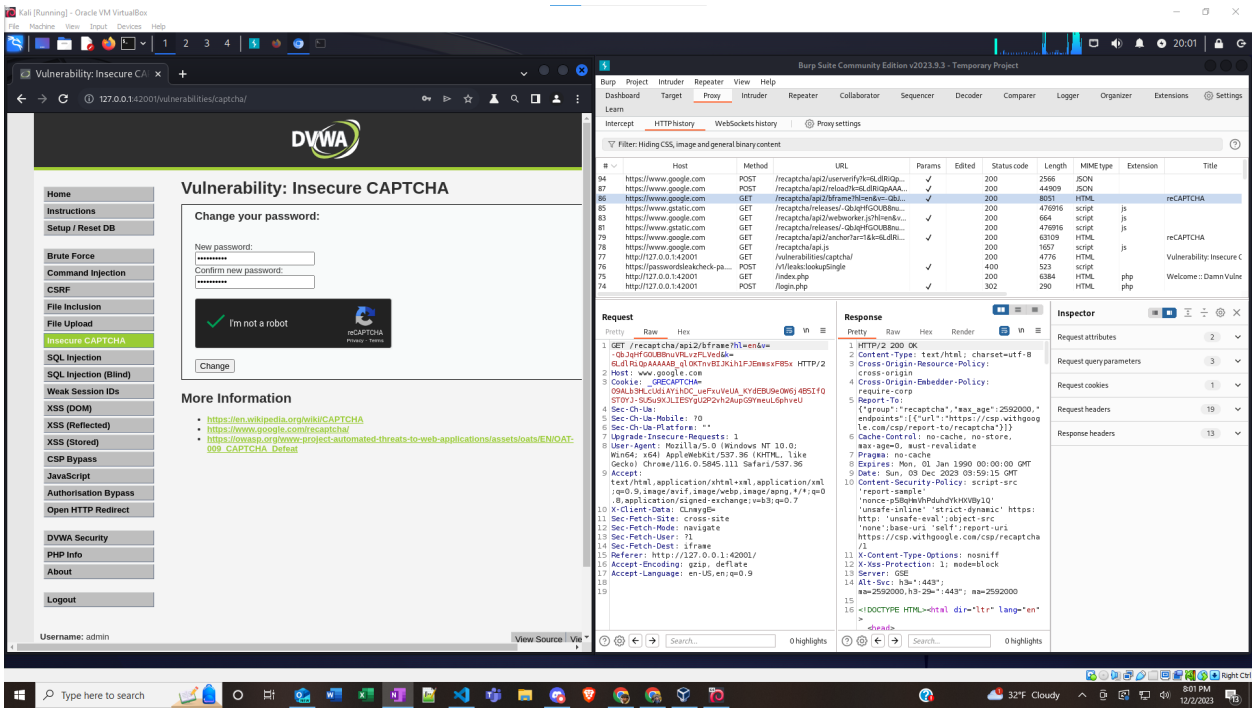

Figure 103: Burp Browser is used to change the password, while Burp Proxy captures and records all traffic.

Step 3: Find the password changed confirmation POST request in the Burp Proxy HTTP history tab and send it to the Repeater.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

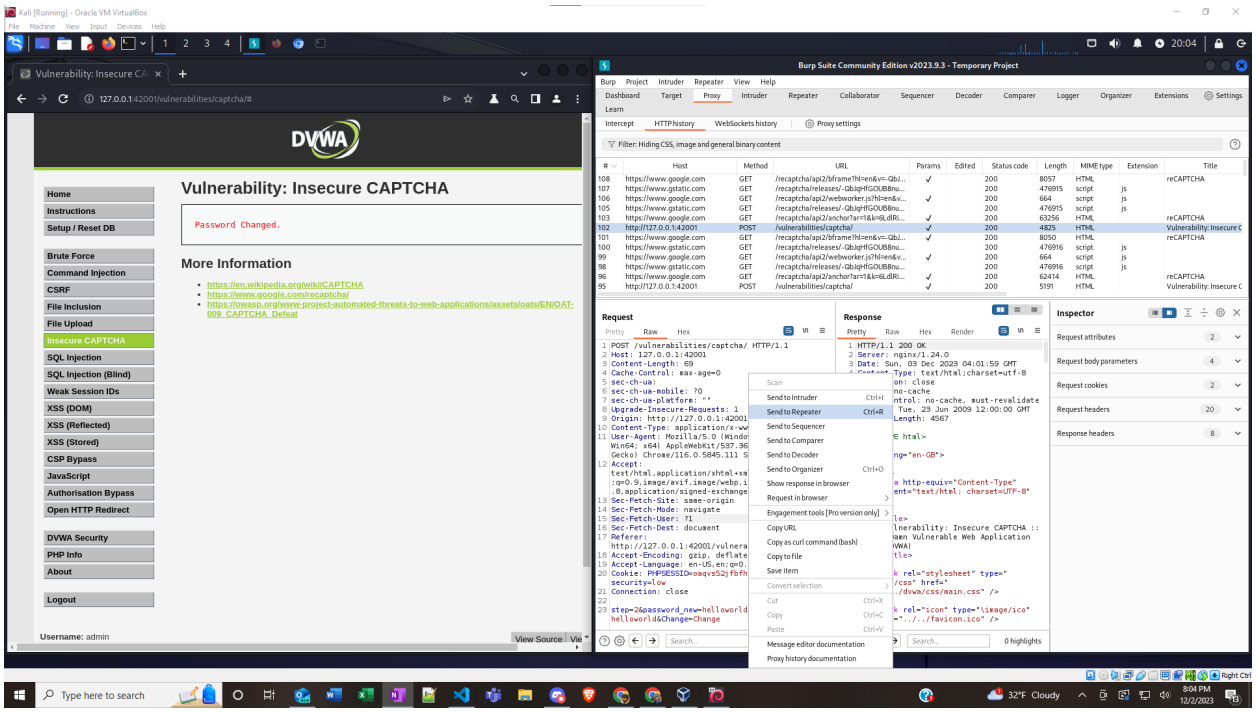

Figure 104: Send the POST request containing the new password at step 2 to the Repeater.

Step 4: Use the Repeater to modify the password in the POST request and send a new request. Use the Response window to confirm the password has changed.

Seth Weiss
weissse@oregonstate.edu
CS 370 - Intro to Security
Programming Project 3 - Damn Vulnerable Web App
Fall 2023

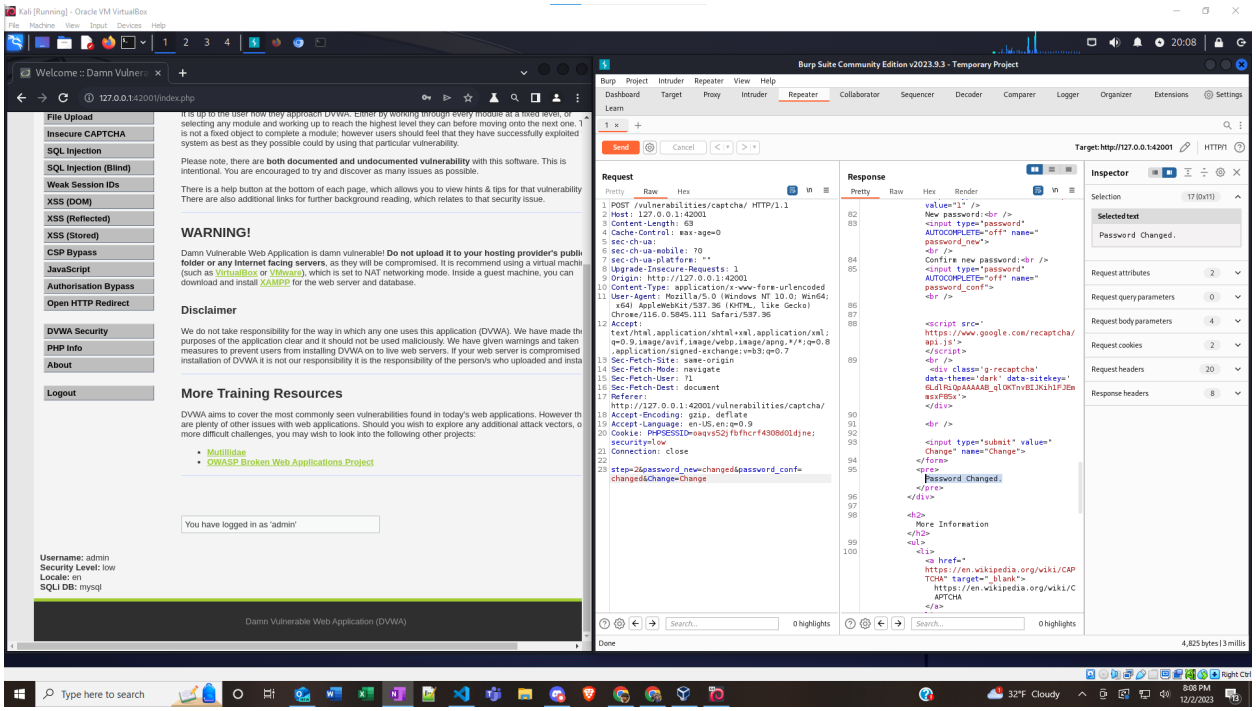

Figure 105: Modify the password and send a new request to step 2 using the Repeater.

Step 5: Log out and log back into the application to confirm the password change.

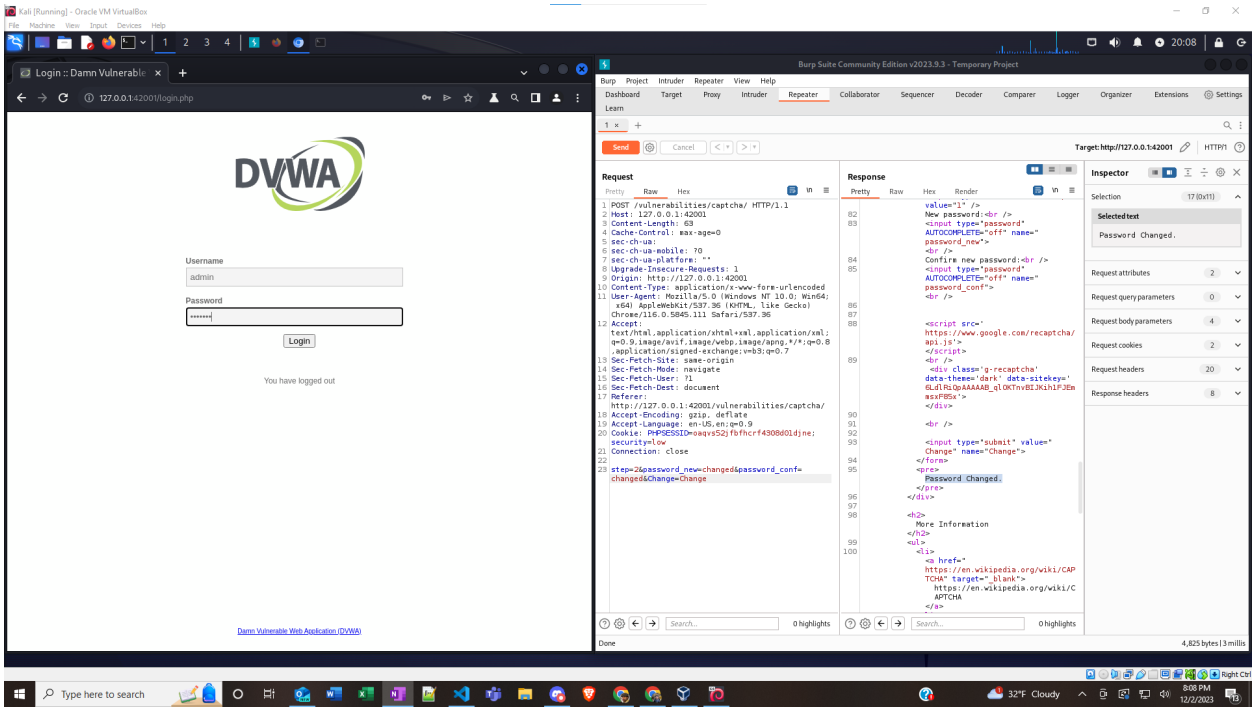

Figure 106: Log out and enter the new password.

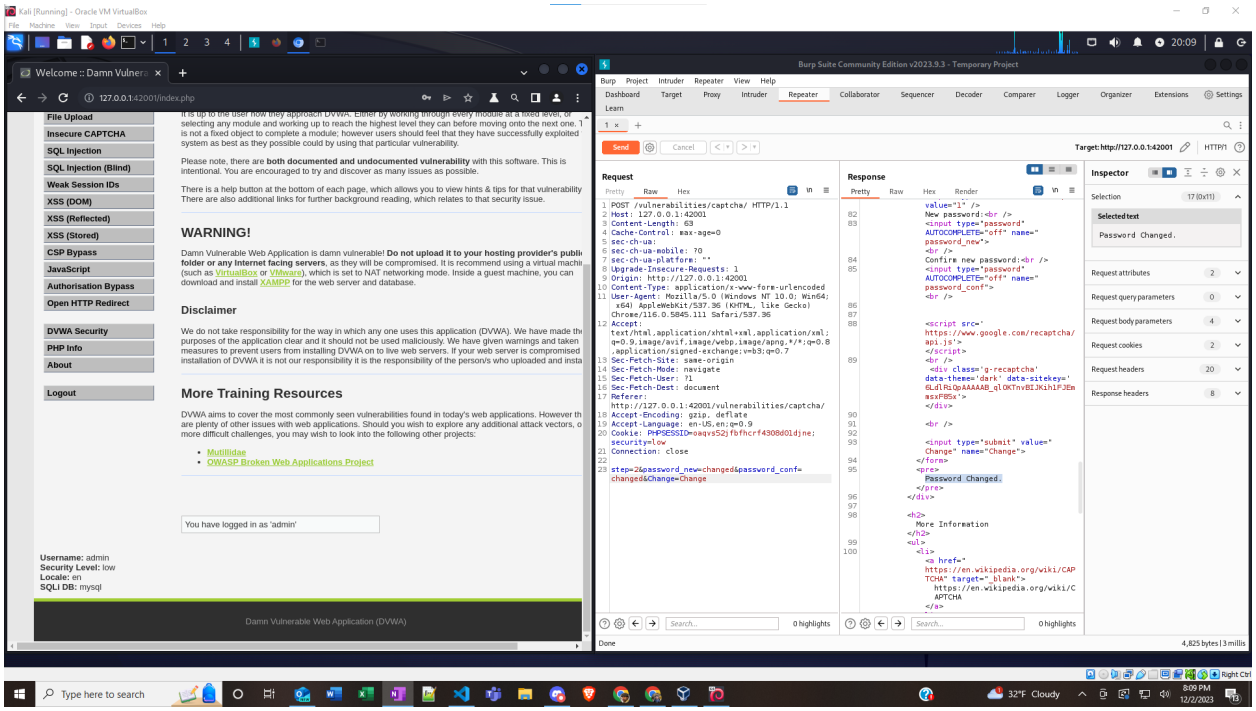

Figure 107: Confirm login with new password.

Identify which OWASP Top 10 vulnerability this relates to (note: some may be related to more than one).

- A01:2021 – Broken Access Control
- A07:2021 – Identification and Authentication Failures

Resources

- CryptoCat. (2021, February 27). *6 - Insecure Captcha (low/med/high) - Damn Vulnerable Web Application (DVWA)* [Video]. YouTube. https://www.youtube.com/watch?v=But-uBPdsKk&list=PLHUKi1UlEgOJLPSFZaFKMoexpM6qhOb4Q&index=9&ab_channel=CryptoCat
- (2023, October 10). *Intercept HTTP traffic with Burp Proxy*. PorSwigger. https://portswigger.net/burp/documentation/desktop/getting-started/intercepting-http-traffic