Assisted Lab: Performing and Detecting CSRF

Scenario

In this lab, you will perform a CSRF (Cross-site request forgery) attack, then investigate the result.

As a cybersecurity analyst, you are working to discover weaknesses and vulnerabilities that your organization, Structureality Inc., needs to mitigate throughout its internal network. In this lab, you will perform a cross-site request forgery (CSRF) attack against the public-facing website of your company hosted in their screened subnet. Finally, you will investigate the website's logs for evidence and IoCs (Indications of Compromise) related to CSRF activities.

Understand your environment

You will be working from a virtual machine named KALI, hosting Kali Linux, and a virtual machine named LAMP, hosting Ubuntu Server. You will initially use KALI to perform the attack targeting the DVWA website hosted on the LAMP VM, and then you will work from LAMP to perform the investigation.

Objectives

This activity is designed to test your understanding of and ability to apply content examples in the following CompTIA CySA+ objectives:

1.1 Explain the importance of system and network architecture concepts in security operations.

1.2 Given a scenario, analyze indicators of potentially malicious activity.

1.3 Given a scenario, use appropriate tools or techniques to determine malicious activity.

1.4 Compare and contrast threat-intelligence and threat-hunting concepts.

2.4 Given a scenario, recommend controls to mitigate attacks and software vulnerabilities.

3.2 Given a scenario, perform incident response activities.

3.5 Explain concepts related to attack methodology frameworks.

Perform CSRF

Cross-site request forgery (CSRF) is a web exploit that enables an attacker to trick users into performing actions that may harm their accounts, sessions, or targeted web servers. CSRF allows an attacker to potentially circumvent the security mechanism known as the same origin policy. The same origin policy is intended to prevent websites/servers from interfering with each other.

In this exercise, you will be acting like an attacker. First, you will discover a CSRF vulnerability. Then, you will exploit this weakness to take over an account.

This exercise uses the DVWA as the target of several database exploitations. However, you must first log into DVWA to access the various challenges. The DVWA has four difficulty levels (Low, Medium, High, and Impossible) and is set to the Low level by default.

DVWA or Damn Vulnerable Web Application is a safe and legal security playground that security professionals can use to improve their skills and learn tools and techniques related to web attacks and exploitations. DVWA is designed to be installed into a private (i.e., non-Internet) lab environment for internal use. Do NOT install DVWA on a production or an Internet-accessible system.

Connect to the KALI and sign in as root using Pa$$w0rd as the password.

Open Firefox by selecting its icon from the Kali Linux toolbar.

Maximize the Firefox window.

In the address field of Firefox, enter dvwa.structureality.com.

The initial page of DVWA is displayed along with the application's login fields. Type admin and

password into the Username and Password fields, respectively, then select Login.

In a real-world situation, you would attempt to exploit any input field you discover. However, with DVWA, you must first log in to the application itself to access the attack target elements.

The Welcome to Dann Vulnerable Web Application! page should be displayed.

If you scroll to the bottom of any DVWA page, you will see a footer that indicates several values, including the security level. To change the security level, select DVWA Security from the left-side navigation menu bar, make a selection from the pull-down list, then select Submit. This lab assumes the default security level of Low.

In the left-side navigation menu bar, select CSRF.

The Vulnerability: Cross Site Request Forgery (CSRF) page should be displayed.

This page is intended to allow a logged-on user to change their password.

What would make the password change page more secure? (Select two)

Use HTTPS

Ask to define security questions

Require the current password to be provided

Send a link to the password page via email

Have a timer on the password change page

Congratulations, you have answered the question correctly.

Select the Test Credentials button.

On the Test Credentials pop-up window, type admin in the Username field and password in the Password field, then select Login.

You should see the message Valid password for "admin".

Close the Test Credentials pop-up window.

What other means is there to test if a password change was effective?

Sending a URL via email

Sending a code via text message

The primary login page

Calling the user on the phone

Congratulations, you have answered the question correctly.

Consider the scenario of a user changing their password. You are currently logged into the DVWA site as admin. On the Change your admin password: page, type password1 into both the New password and Confirm new password: fields, then select Change.

In this instance, you are changing the password as the valid user.

You are changing the password to log into the DVWA web application, so be sure to type carefully and remember what you type. Otherwise, you may need to exit and re-launch the

entire lab to reset the password.

You should see the message Password Changed..

Select the Test Credentials button.

On the Test Credentials pop-up window, type admin in the Username field and password1 in the Password field, then select Login.

You should see the message Valid password for "admin".

This confirms the password change was effective.

Close the Test Credentials pop-up window.

Look at the URL in the Firefox address bar. It should be:

http://dvwa.structureality.com/vulnerabilities/csrf/?password_new=password1&password_conf=password1&Change=Change#

This URL can be modified to cause a password change when triggered by the victim clicking on a link that is disguised as something else.

As an attacker, you could craft a social engineering message to the victim with a link that includes the commands to alter their password. For example, the following message could be sent:

Dear admin.

We have determined that you are one of our most valued customers, and we would like to reward you with an additional year of service for free. Click the link below to apply your special coupon.

Your free year of service coupon!

Be sure to log out immediately after clicking the link to apply your new subscription end date to your account.

Congratulations,

The Support Staff

The email message shows a hyperlink of Your free year of service coupon!. But since it is rendered by the browser (or an HTML-interpreting email client), the actual URL is not displayed. The actual code behind this link is:

<A HREF="http://dvwa.structureality.com/vulnerabilities/csrf?password_new=abc123&password_conf=abc123&Change=Change">Your free year of service coupon!

The above message, if rendered in HTML by the victim's email client, will show a click link of Your free year of service coupon!, but the actual URL will be encoded behind the click link. So if the victim is fooled by the message, they would click on the link. Notice the URL is a modified version of the URL from the Change your admin password: page.

Obviously, including passwords in a URL is poor and insecure web design. However, even when a website does not leak sensitive information through a URL, it may still be possible to perform CSRF attacks. A CSRF attack is when an attacker is able to forge a request to a website that

seems to originate from the authenticated client system. This can be accomplished through a parameter-polluted link (as demonstrated in this exercise), through an adversary-in-the-middle (AitM) attack, or through automated malware infecting a victim's system. In all of the methods of implementing CSRF, it depends upon the victim having a current and active session with the target website. CSRF intends to abuse a website's trust in an authenticated user. The result is the targeted website performs tasks or executes commands that are forged to seem to be from the authenticated user.

Try using this URL yourself by playing the victim; however, rather than clicking on a link, you will need to enter the following into the address field of Firefox:

http://dvwa.structureality.com/vulnerabilities/csrf?password_new=abc123&password_conf=abc123&Change=Change#

By visiting this CSRF attack link, you are simulating a victim being fooled by the phishing email.

You can avoid typing in this entire URL by editing the existing URL to change the password1 values to abc123.

You are changing the password to log into the DVWA web application, so be sure to type carefully and remember what you type. Otherwise, you may need to exit and re-launch the entire lab to reset the password.

The Change your admin password: page will update and show the message Password Changed..

This is not necessarily a subtle attack. If the victim is paying attention, they should notice the message that their password has just been changed. This should be recognized as a very odd occurrence taking place immediately after clicking on a link from an email. A spry user might realize what has happened and be able to immediately change their password back to something they know.

The core of this CSRF attack is tricking a victim into clicking on a link that will run commands against their account on a website where they have a pre-established session. (True/False)

True

False

Congratulations, you have answered the question correctly.

Select Logout from the bottom of the left-side navigation menu bar.

While this attack is not elegant, it does get the job done. The admin account's password has been changed through a simulated click of a link delivered to a victim via an email message. In other words, this is a CSRF exploit.

Technically, the victimized user could notice the password change message while still logged on. And they could immediately change their password again to something they know. But they would have to recognize the attack for what it is and not log out as instructed by the scam message.

Attempt to log back into DVWA by typing in admin and password1 into the appropriate fields, then selecting Login.

Will will see a Login failed message below the Login button. This confirms that the CSRF exploit changed the admin user account's password.

Now that the victim cannot log into their account, you, as the attacker, can log in using the credentials you defined through the CSRF exploit. Type in admin and abc123 into the appropriate fields, then selecting Login.

You should be logged into the DVWA website.

Check your work

Confirm that you discovered a webpage vulnerable to CSRF abuse.

Confirm that you formed a CSRF URL to trick a victim into causing a password change.

Investigate CSRF

You have received a report that several users are no longer able to log into their accounts on the company website. As a cybersecurity analyst, you are tasked with investigating the issue. You perform a brief investigation of the website itself and notice that password changes are exposed through a URL (rather than containing the password change in the body of a POST message within an encrypted session). You suspect that the website was the target of a CSRF attack. In this exercise, you will investigate the log of the company's website to see if you can find evidence or IoCs of CSRF.

Connect to the LAMP virtual machine and sign in as lamp using Pa$$w0rd as the password.

Elevate to use root privileges by entering: sudo su and then entering Pa$$w0rd as the password.

Enter cd /var/log/apache2 to change into the apache2 log directory.

Enter ls -l to view the log filenames, sizes, and timestamps.

Enter less access.log to view the website's access.log. Look over the log for anything interesting.

When using the less file viewing utility, press the spacebar to view the next page. You can return to a previous page using b or scroll one line up or down utilizing the arrow keys.

You need to 'ignore' the directory path element of "/vulnerabilities/csrf/" as this is the obvious name of the HTML document on the DVWA (Damn Vulnerable Web Application) that is designed to demonstrate CSRF. In a real-world situation, you will not see the term "CSRF" in the logs. CSRF attacks are usually more subtle than that.

The Apache web server access log has two default log formats. The Common Log Format includes the following seven default fields:

IP address of the client

The identity of the client, but typically presented as only a hyphen (i.e., -)

User ID of requesting user, but will be a hyphen when there is no established user context

Date and time of the request (in square brackets)

The HTTP request type (i.e., GET, POST, etc.) and the resource being requested

The HTTP response status code

The size of the object returned to the client

The Combined Log Format includes the following two additional fields:

The HTTP referrer (i.e., the address from which the request for the resource originated.)

The User Agent of the client, which identifies information about the browser that the client is using to access the resource.

It is also possible to customize the fields of the Apache logs.

In this exercise, Apache is configured to use the Combined Log Format. Note: The User ID is a hyphen in the access.log for this exercise because when using the DVWA as the target, while you must log in as admin to access the vulnerable applications of the demo service, you are not using a user account or active login on most of the demonstration sub-pages.

...less

Starting from the top of the access.log file (i.e., the oldest entry in the log), look down through the entries to find the one with the following as its HTTP request:

"GET /vulnerabilities/csrf/ HTTP/1.1"

This is the URL stub representing when a visitor accesses the change password page on the website. On its own, this is a record that could be benign or an element of malicious activity. In this instance, this was you visiting the main Vulnerability: Cross Site Request Forgery (CSRF) page.

From this log record, you should be able to follow the progress of the user. The next record to find should have the following HTTP request:

"GET /vulnerabilities/csrf/test_credentials.php HTTP/1.1"

This is the URL stub for the page to verify that a user's credentials are correct.

The next interesting log record should have an HTTP request of:

"POST /vulnerabilities/csrf/test_credentials.php HTTP/1.1"

While the URL stub remains the same, the HTTP method has switched from GET to POST. The Test Credentials page is actually submitting credentials to the web server correctly (i.e., not in the URL itself, but in the body of a POST message). The presence of this log record simply indicates that a query to test credentials was performed. This is not evidence of benign or malicious activity on its own.

This is evidence of a malicious event.

True.

False

Congratulations, you have answered the question correctly.

The next log record to look for has the HTTP request of:

"GET /vulnerabilities/csrf/password_new=password1&password_conf=password1
&Change=Change HTTP/1.1"

This is the log record of the user changing their password to "password1". This is further evidence that the website is designed insecurely. There should be NO recordings of passwords into log files. While reviewing the website, you discovered that the password changes took place via URL. This log record solidifies that finding as its existence in the log is proof it was sent by the client's browser. This confirms the CSRF vulnerability but not that an attack is or has occurred.

Notice that there is a log record of a visit to test_credentials.php likely used to verify that the password change to "password1" was successful.

The next log record to look for has the HTTP request of:

"GET /vulnerabilities/csrf/password_new=abc123&password_conf=abc123&Change=Change
HTTP/1.1"

This log record is another change of password operation. It is not clearly an attack, but it is evidence of the attack. However, it is still indistinguishable from a valid user changing their own password. This is a frustrating issue with CSRF (since it is impersonating valid user activity); there is no obvious record of malicious events. Instead, you may be simply discovering vulnerability to CSRF more than uncovering direct proof that a CRSF attack took place.

Is this an IoC of CSRF without context?

Yes

No

Congratulations, you have answered the question correctly.

There should then be a pattern of log records with the following HTTP requests and referrers:

"GET /login.php HTTP/1.1" <referrer doesn't matter>

"POST /login.php HTTP/1.1" "http://dvwa.structureality.com/login.php"

"GET /login.php HTTP/1.1" "http://dvwa.structureality.com/login.php"

This pattern occurs whenever a failed login takes place. The first GET of login.php is the loading of the login page. The POST HTTP request is the submission of credentials. The second GET of login.php with a referrer login.php (i.e., the same login page) is proof that the submitted credentials were incorrect and the login page was presented again.

You can also see another pattern of log records with the following HTTP requests and referrers:

"GET /login.php HTTP/1.1" <referrer doesn't matter>

"POST /login.php HTTP/1.1" "http://dvwa.structureality.com/login.php"

"GET /index.php HTTP/1.1" "http://dvwa.structureality.com/login.php"

This pattern occurs whenever a successful login takes place.

At this point, you have viewed all the evidence available from a website's acecss.log related to a CSRF event. There is no obvious entry that is direct evidence of CSRF exploitation. This is because it simulates legitimate user activity and does not include any obvious injected code, there are no obfuscated commands, there are no percent-encoded symbols, nor any references to external URLs.

It is important to understand that not all intrusions, attacks, and exploits have easy-to-recognize evidence or clearly defined IoCs. Sometimes an investigation depends completely on context (i.e., information from victims and logs of other systems and services). You might not always be able to determine a root cause for one-off events. However, you may still discover vulnerabilities that can be mitigated to prevent future repeat occurrences.

In a real-world investigation of a CSRF event, you will need to find other corroborating sources of evidence to detect the IoC of CSRF. This would need to include interviewing the victims to see when the last time they remember logging in successfully and when they first discovered their "valid" credentials stopped working. You also need to ask when was the last time they purposefully changed their password and did they notice any strange events (such as clicking on a link from an email which resulted in a web page claiming their password was changed). Even with this, there is still only thin evidence of a CSRF. If numerous workers report the same issue, you may see a pattern in the timing of the events. You may also get lucky and find that the attacker set all of the compromised accounts to the same password.

Check your work

Confirm that you investigated a CSRF event.