How to exploit Cross Site Request Forgery attack on web applications where the request is posted in JSON format

What is Cross Site Request Forgery?

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

How does a CSRF Attack works?

CSRF is an attack that tricks the victim into submitting a malicious request. It inherits the identity and privileges of the victim to perform an undesired function on the victim's behalf. For most sites, browser requests automatically include any credentials associated with the site, such as the user's session cookie, IP address, Windows domain credentials, and so forth. Therefore, if the user is currently authenticated to the site, the site will have no way to distinguish between the forged request sent by the victim and a legitimate request sent by the victim.

Background:

A normal CSRF attack can be achieved where the target content type is text/html or text/plain by creating an HTML payload or a Crafted payload inside a link.

But for those scenarios where the target content is in JSON or some other content type this can be achieved using the below exploit.

In a CSRF attack if application is validating the Content-type and data format, the attack can be achieved using flash and 307 redirect as mentioned in the detailed steps below.

Prerequisites:

1. A compatible web browser supporting flash applications.
2. A web server supporting php applications like XAMPP.

Steps to exploit:

1. Install XAMPP with PHP 7.2.5 in your machine.
2. Create a php file to redirect the JSON data to the vulnerable website as below :

```
<?php
header("Location: http://csrfvulnerablesite.com/jsoncsrf.html", true, 307);
?>
```

3. Create a flash file to post the payload to the above php file with the expected content type header which will then redirect the payload to the vulnerable site.
4. Now host these two files in the XAMPP “htdocs” folder and start the Apache server.
5. Now assume the victim is logged in to the application, we just need to share him the link to the hosted flash file.

Note: A crossdomain file with the below content is required if the flash file & redirector page is not hosted on the same domain so that flash file can make request to attacker’s host.

```
<cross-domain-policy>
<allow-access-from domain="*" secure="false"/>
<allow-http-request-headers-from domain="*" headers="*" secure="false"/>
</cross-domain-policy>
```

6. Once the victim clicks on the link to the flash file it will request for the php file we created, this will make 307 redirect to mentioned application endpoint, and as 307 is special redirect which will post the JSON data as well which got received from the flash file to the target endpoint and CSRF will take place successfully as shown in below example.

7. In the below example the victim was tricked to post a request to change its password crafted in the attackers payload and ultimately his password got changed unwantedly. A sample post data is shown in the below Fig 1.

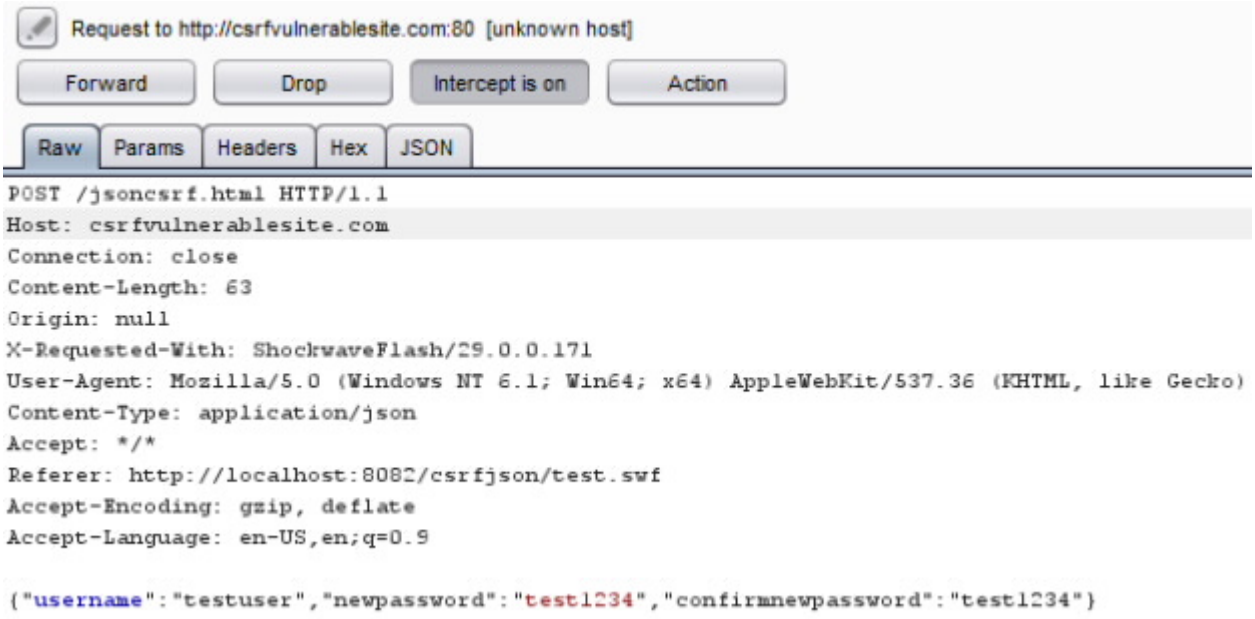

Fig 1: Executing CSRF Attack

Recommendations:

1. Verifying the source of the requests by validating the Same Origin with Standard Headers.
2. If the Origin header is not present, verify the hostname in the Referer header matches the target origin.
3. Once you have verified that the request appears to be a same origin the next recommendation is to implement custom defense mechanisms using CSRF specific tokens created and verified by your application.