More Web

Software Security

XSS

Cross Site Scripting (XSS)

- Attackers can insert custom code/scripts into webpages

- Vulnerable sites
  - Allow user input
  - Place that user input on the webpage
  - User input isn’t sanitized

- Vuln presents in the HTML (or other source) of a webpage on a client

Cross Site Scripting (XSS)

Rank	ID	Name	Score
[1]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	75.56
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.69
[3]	CWE-20	Improper Input Validation	43.61
[4]	CWE-200	Information Exposure	32.12
[5]	CWE-125	Out-of-bounds Read	26.53
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	24.54
[7]	CWE-416	Use After Free	17.94
[8]	CWE-190	Integer Overflow or Wraparound	17.35
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	15.54
[10]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.10

Types of XSS

- DOM-based, local XSS
  - Code injection takes place entirely within a page or client-side application
- Reflected, Nonpersistent XSS
  - User input is echoed into a webpage
  - Not validated
- Stored, Persistent XSS
  - Like reflected, but attacker input is stored for later
  - More dangerous
    - No phishing
    - User could be going to a normal site for them

Reflected XSS Example

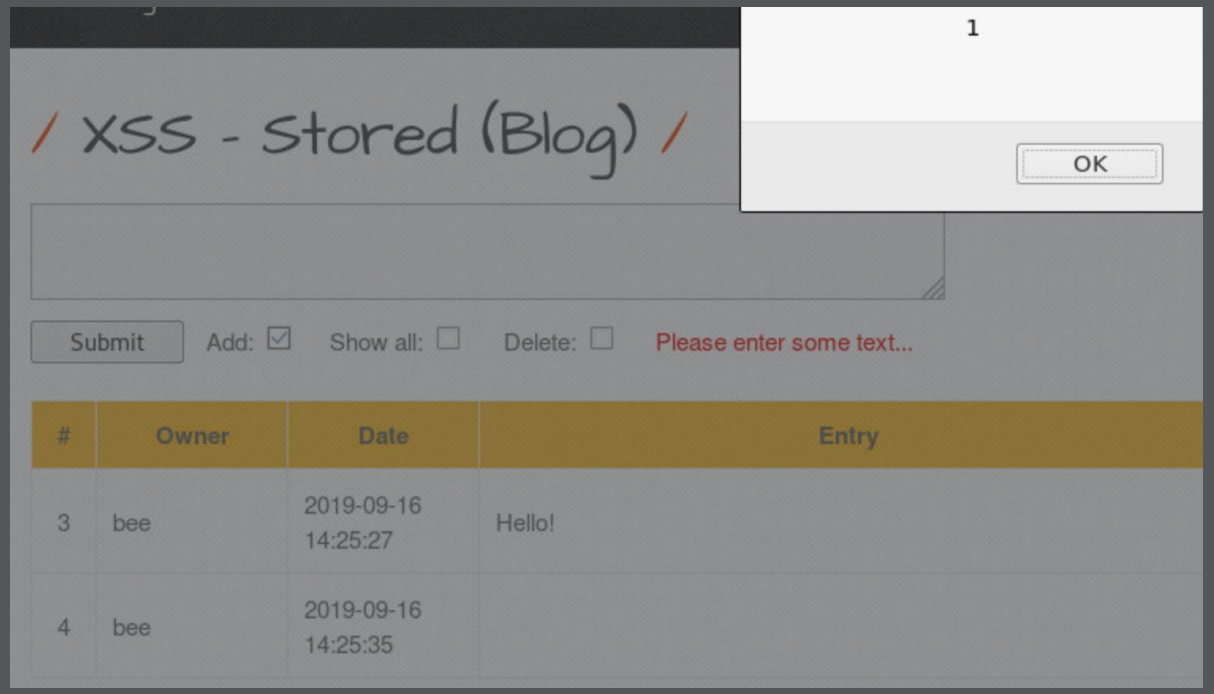
- Seems harmless, right?

```php
<?php
$name = $_GET['name'];
echo "Welcome $name!<br>";
?>
```

- What if we put some script in place of our name?
  - `<script>alert(‘oops!’);</script>`
  - `<script>window.location.replace("http://www.dsu.edu");</script>`

Stored XSS Example

- Classic example – guestbook
- Affects everyone who visits



XSS - Remediation

- Validate and escape input

- Easiest (but impractical) solution
  - Never put untrusted data into your web page

Escaping HTML Entities

- Makes user input safe to be returned in certain areas
- Doesn’t necessarily make user input safe in all areas though

- Notable places entity escaping should NOT be used
  - Within script tags
  - Inside HTML comments
  - In attribute names
  - In tag names
  - Directly in CSS

```
& --> &amp;
< --> &lt;
> --> &gt;
" --> &quot;
' --> &#x27;
/ --> &#x2F;
```

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html#rule-0-never-insert-untrusted-data-except-in-allowed-locations

Special Cases

- Within attributes – escape all ASCII characters < 256
- Within JavaScript – escape data values generated based on user input
  - Do not allow user control of anything within setInterval, setTimeout, etc
- When returning user controlled data in JSON, use the correct content-type (application/json)
- Within CSS only allow user data in property values but not areas where expressions can be evaluated
  - Escape all ASCII non-alphanumeric ASCII characters < 256

More Special Cases

- When user input is used as a URL argument escape all non-alphanumeric ASCII characters < 256
  - Not effective for data URLs
  - Don’t encode entire URLs
- If user’s are allowed to enter HTML, sanitize with an existing tool – don’t roll your own

Other Mitigations

- Cookie Security
  - SameSite – Controls where cookies are sent
  - HTTPOnly – Disallows JavaScript access to cookies
  - Secure – Allows sending of cookie only over HTTPS connections
- Content Security Policy
  - Describes where resources can be loaded from
- Modern templating libraries handle escaping automatically contextually

CSRF / XSRF

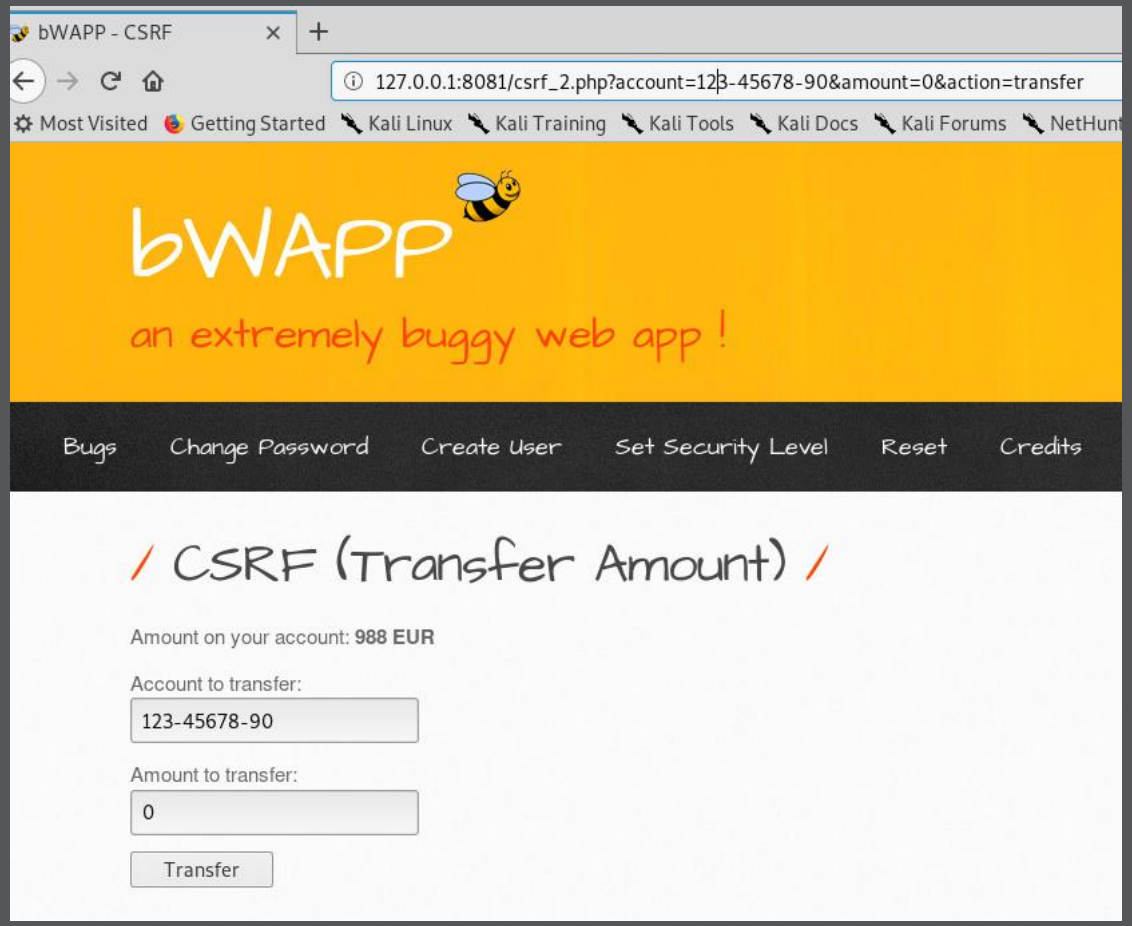
Cross Site Request Forgery (CSRF)

- a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the web application trusts

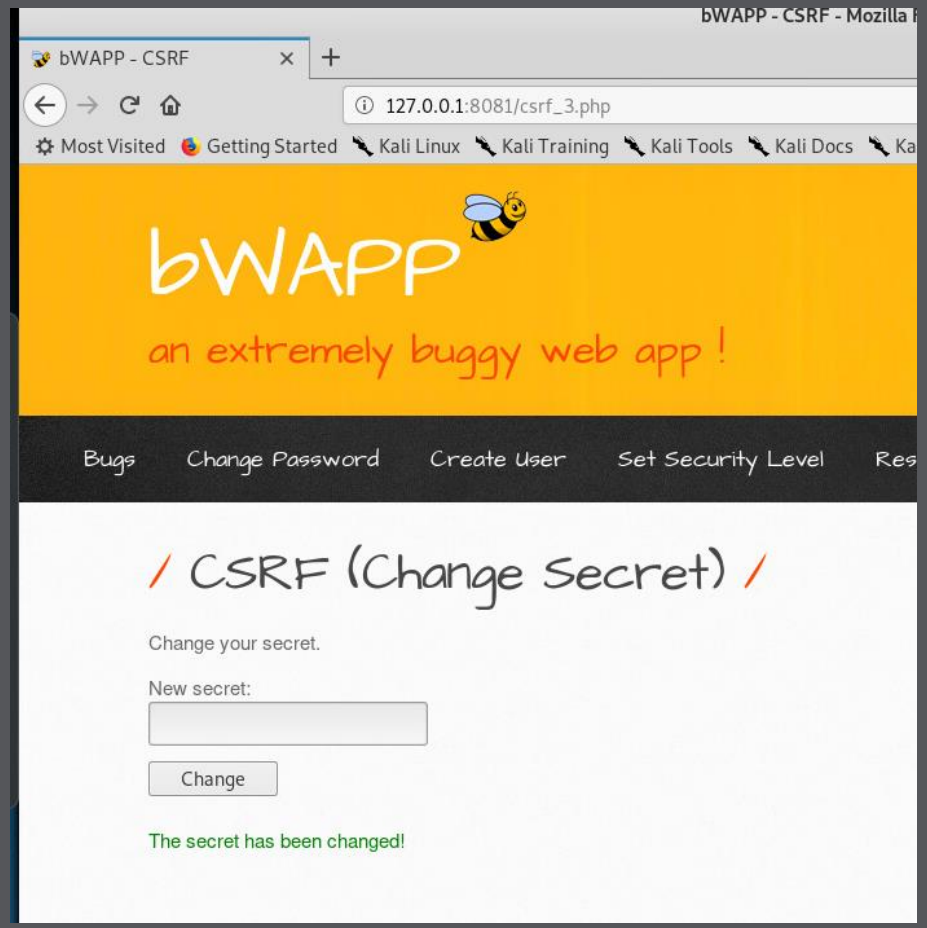
Cross Site Request Forgery (CSRF)

Rank	ID	Name	Score
[1]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	75.56
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.69
[3]	CWE-20	Improper Input Validation	43.61
[4]	CWE-200	Information Exposure	32.12
[5]	CWE-125	Out-of-bounds Read	26.53
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	24.54
[7]	CWE-416	Use After Free	17.94
[8]	CWE-190	Integer Overflow or Wraparound	17.35
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	15.54
[10]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.10

Example - GET


bWAPP - CSRF
127.0.0.1:8081/csrf_2.php?account=123-45678-90&amount=0&action=transfer
Most Visited
Getting Started
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
bWAPP
an extremely buggy web app !
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
/ CSRF (Transfer Amount) /
Amount on your account: 988 EUR
Account to transfer:
123-45678-90
Amount to transfer:
0
Transfer

Example - POST



Examples

XSRF Remediation

- Add a secret to the client and server that isn't included in a cookie
  - Usually called a CSRF token

- Additionally you can:
  - Set SameSite cookie attribute for session cookie
  - Verify the request origin
  - Add a timeout to the session
  - Re-authenticate users for critical changes
  - (weak) use POST rather than GET

HTTP Response Splitting

HTTP Response Splitting

- All that separates headers in a response is a CRLF (\n)
- Untrusted input ending up here results in chaos

- https://www.owasp.org/index.php/HTTP_Response_Splitting

Remediation

- Remove all new lines by URL encoding or other methods when included in headers

Summary

- Trust nobody
- Like ever
- If you must use untrusted data in a response – encode as necessary

References

- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html