

# DevSecOps

---

Application Security from start to finish



# Hackers in movies VS



## How people think they get hacked



## How they really get hacked

<p>341 KXO/B Feb 12 · 🌐</p> <p>12K</p> <p>100K Comments · 2.3K Shares</p> <p>Like Comment Share</p>	<p>Up 99.3 Feb 12 · 🌐</p> <p>12K</p> <p>26.4K Comments · 937 Shares</p> <p>Like Comment Share</p>	<p>Feb 12 · 🌐</p> <p>10.8K Comments · 2.8K Shares</p>
<p>Feb 12 · 🌐</p> <p>14.5K</p> <p>700K Comments · 6.9K Shares</p> <p>Like Comment Share</p>	<p>Feb 12 · 🌐</p> <p>14.5K</p> <p>700K Comments · 6.9K Shares</p> <p>Like Comment Share</p>	<p>Feb 12 · 🌐</p> <p>14.5K</p> <p>700K Comments · 6.9K Shares</p> <p>Like Comment Share</p>



# Michael Kaufmann

Managing Director, Xpirit Germany



Microsoft  
Regional Director



>20 Jahre Softwareentwickler

>15 Jahre ALM & DevOps

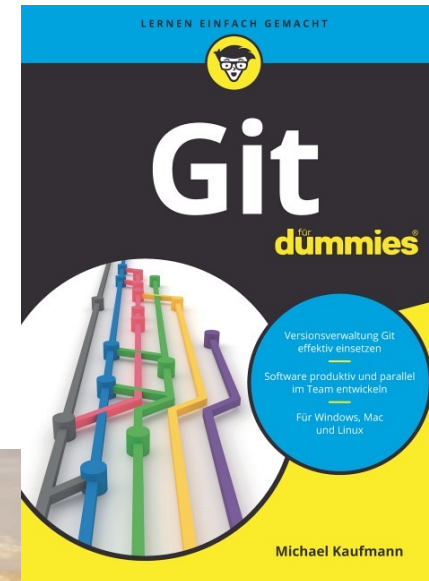
>10 Jahre Git

Microsoft Regional Director & MVP

 @mike\_kaufmann

 @wulfland

 <https://writeabout.net>



# The event-stream incident



Social engineering attack



Supply chain attack:  
event-stream@3.3.6 -> flatmap-stream@0.1.1



Code execution in build process  
targeting **copay**



Harvest the user's bitcoin and  
private keys

**Malicious Package in flatmap-stream**  
Critical severity | GitHub Reviewed | Published on 1 Sep 2020 · Updated on 1 Oct 2021

Vulnerability details | Dependabot alerts 0

Package: flatmap-stream (npm) | Affected versions: 0.1.1

**Description**

Version 0.1.1 of flatmap-stream is considered malicious.

This module runs an encrypted payload targeting a very specific application, copay and because they shared the same description it would have likely worked for copay-dash.

The injected code:

- Read in AES encrypted data from a file disguised as a test fixture
- Grabbed the npm package description of the module that imported it, using an automatically set environment variable
- Used the package description as a key to decrypt a chunk of data pulled in from the disguised file

The decrypted data was part of a module, which was then compiled in memory and executed.

This module performed the following actions:

- Decrypted another chunk of data from the disguised file
- Concatenated a small, commented prefix from the first decrypted chunk to the end of the second decrypted chunk
- Performed minor decoding tasks to transform the concatenated block of code from invalid JS to valid JS (we believe this was done to evade detection by dynamic analysis tools)
- Wrote this processed block of JS out to a file stored in a dependency that would be packaged by the build scripts:

The chunk of code that was written out was the actual malicious code, intended to be run on devices owned by the end users of Copay.

This code would do the following:

- Detect the current environment: Mobile/Cordova/Electron
- Check the Bitcoin and Bitcoin Cash balances on the victim's copay account
- If the current balance was greater than 100 Bitcoin, or 1000 Bitcoin Cash:
  - Harvest the victim's account data in full
  - Harvest the victim's copay private keys
  - Send the victim's account data/private keys off to a collection

**GHSA ID**  
GHSA-9x64-5r7x-2q53

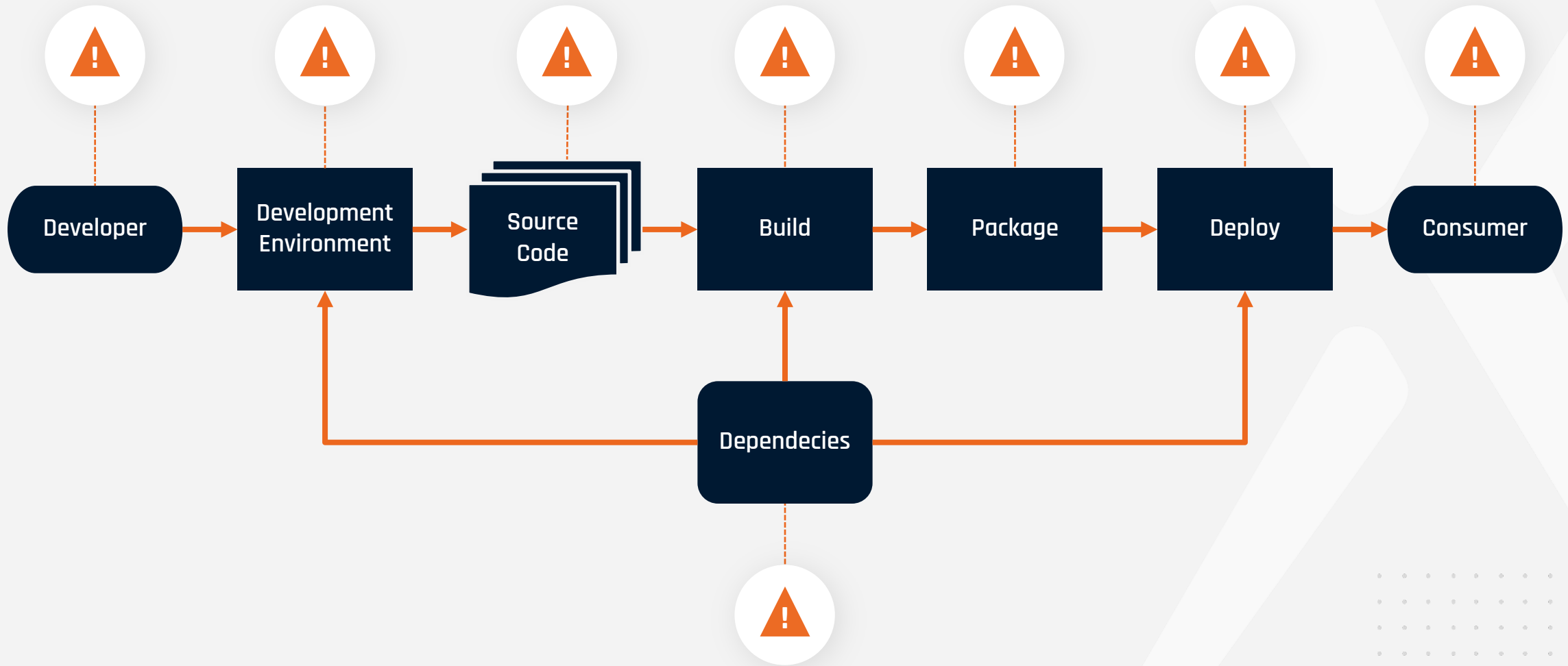
**CWEs**  
CWE-506

**CVSS Score**  
9.8 Critical  
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

This advisory has been edited. See History.

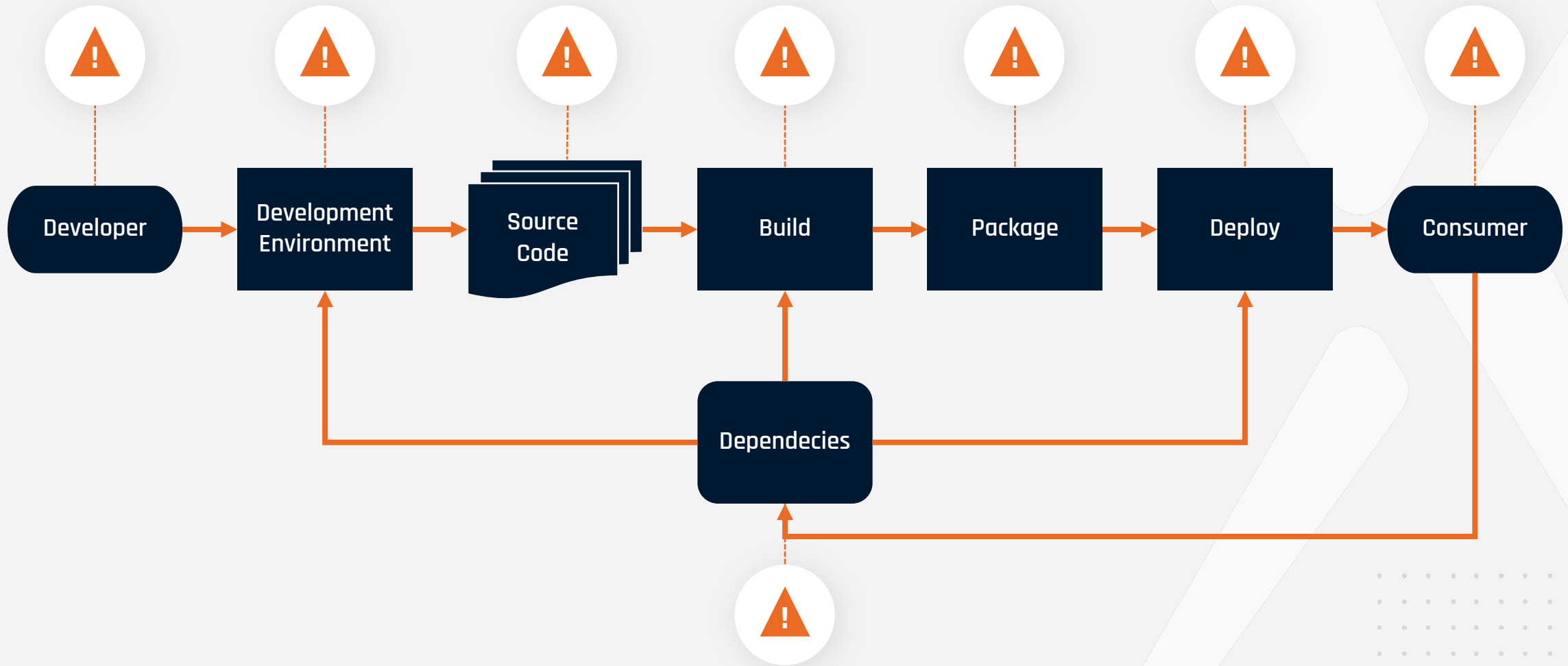
See something to contribute? Suggest improvements for this vulnerability

# Attack vectors



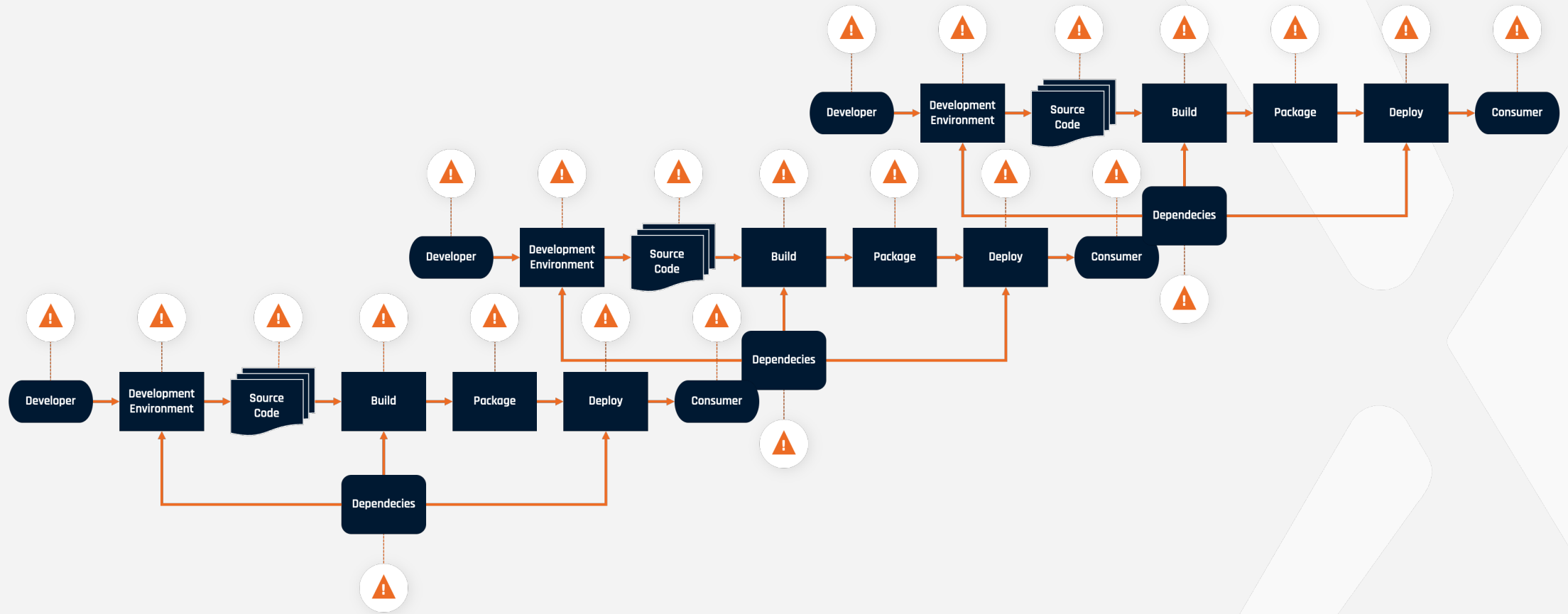


# Attack vectors





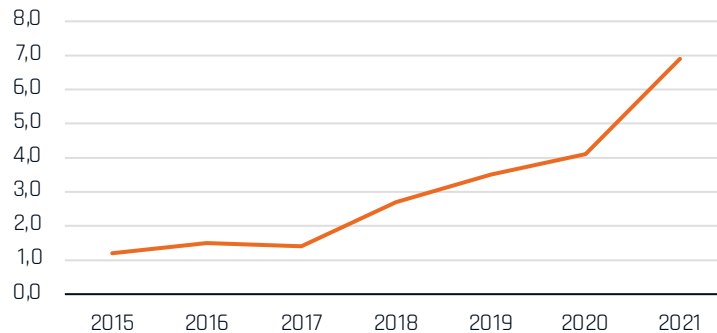
# Attack vectors



# Losses caused by **cyber attacks** reported to IC3

<b>2020</b>	<b>2021</b>
\$ 4,100,000,000	\$ 6,900,000,000

Loss caused by reported cyber crime  
(in billion USD)



## Top 5 crime types:

- > Phishing
- > Non-Payment / Delivery
- > Data Breach
- > Identity Theft
- > Extortion

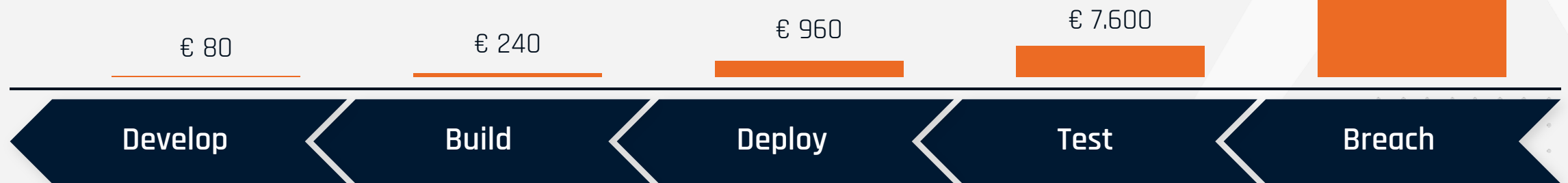
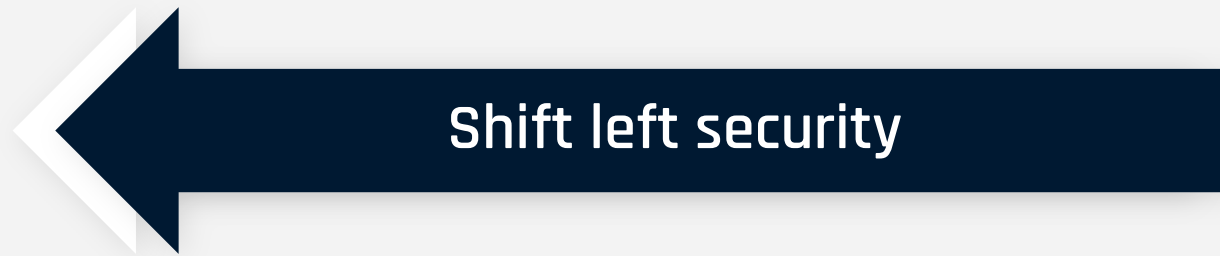


## Trends

- > Confidence fraud / Romance scams
- > Cryptocurrency
- > Ransomware
- > Tech support fraud

# Costs for fixing a security vulnerability

€ Millionen



Source: Ponemon Institute Cost of a Data Breach 2020



# Attack vector: developer



# Attacking developers

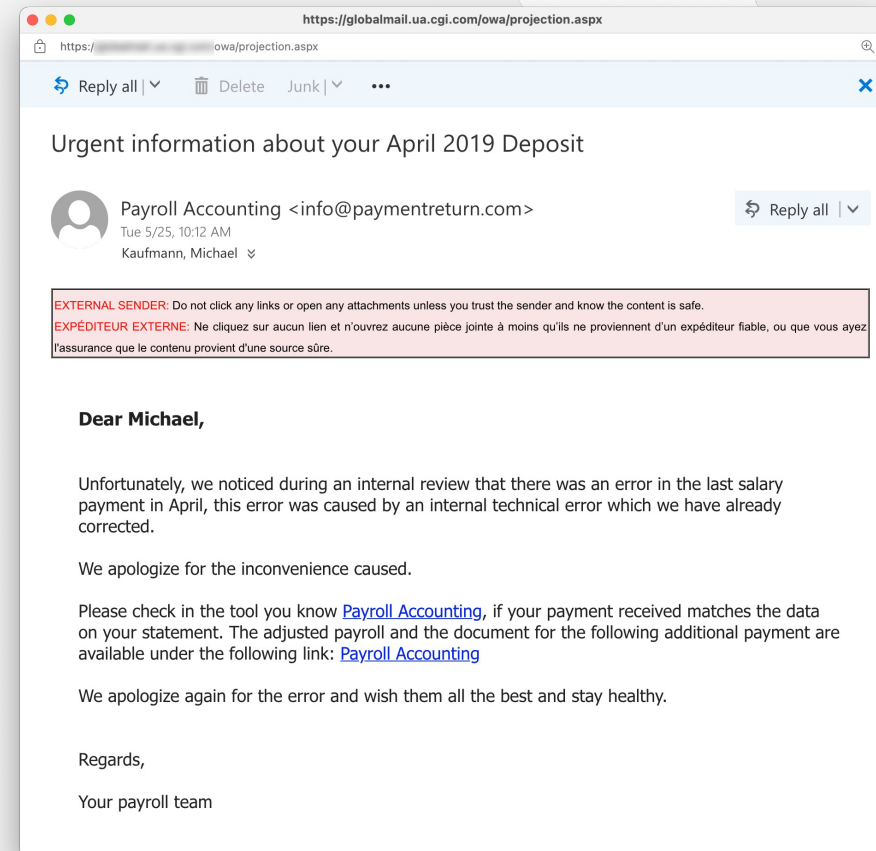
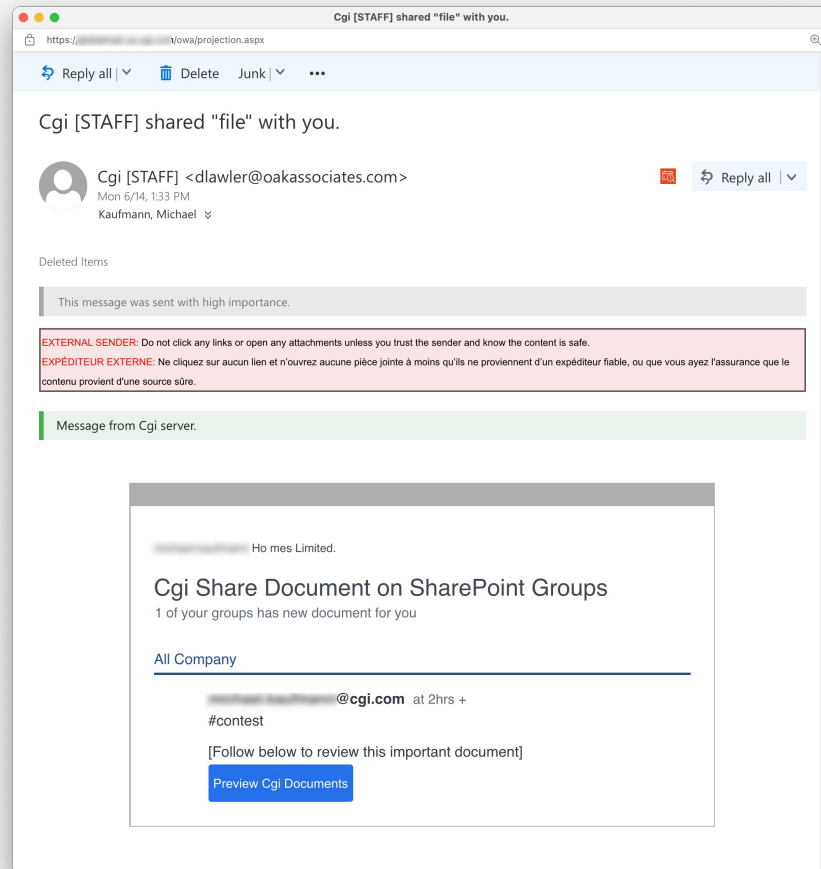
Phishing / Spear Phishing

Social engineering

Unsecured connections to test systems

“A developer is just a normal employee - that works as local admin, can push and execute code on various systems in minutes, and often runs unsecured web servers.”

# Phishing



# Credentials Developer

**E-Mail**



Spear phishing

**Access machines**



Log on, Mimikatz

**Source**



Inject code

**Pipeline**



Execute code / scripts

**Access test environment**



Test against prod?

**Access prod?**



# Attacking **developers**



Typo squatting

---



Namespace shadowing

---





# Typo squatting

```
$ npm install crossenv
```



Steals all  
your  
environment  
variables

```
$ npm install cross-env
```



Normal  
package



# Namespace shadowing

```
$ npm install @azure/core-tracing
```



Normal  
package

```
$ npm install core-tracing
```



Upload data to  
a control server



# Typo squatting

- ▶ Attack supply chain at build time (npm install)

---
- ▶ Attack consumer at run time by shadowing a function

---
- ▶ Version ranges in transient dependencies can delay attack

---



# What to do?



Security Awareness Trainings



Security Games



Red team | blue team simulations



Ephemeral, containerized environments



# Attack vector: dev environment



# Attack vector: dev environment

- ▶ Passwords in text files / memory
- ▶ Mimikatz
- ▶ Build tools
- ▶ Modify code
- ▶ Modify pipeline / execute code

# Credentials in text files

```
TestCredentials.xml — Bearbeitet
<TestAccounts>
  <TestAccount name="domain\testUser1" password="testP@ssw0rd1!" />
  <TestAccount name="domain\testUser2" password="testP@ssw0rd2!" />
  <TestAccount name="domain\testUser3" password="testP@ssw0rd3!" />
</TestAccounts>
```

# Credentials in text files

```
{  
  "/api/*": {  
    "changeOrigin": true,  
    "target": "https://api.project-demo.de",  
    "auth": "  
  }  
}
```

```
const id_token_string = '  
;  
  
app.use(  
  '/api/test',  
  createProxyMiddleware({  
    target: 'https://api.test.  
    changeOrigin: true,  
    pathRewrite: {  
      '/api/test': '/test',  
    },  
    headers: {  
      'Content-Type': 'application/json',  
      'Credentials': true,  
      'Cookie': `id_token=${id_token_string}`  
    }  
  })  
);
```

Schema: <https://json.schemastore.org/appsettings.json>

```
1  {  
2  "AppSettings": {  
3    "ScheduleExecutionIntervalInMinutes": 60  
4  },  
5  "MailSettings": {  
6    "Url": "Outlook.office365.com",  
7    "User": "  
8    "Password": "  
9    "ProcessedFolderName": "  
10   "ErrorFolderName": "  
11   "OutOfToleranceFolderName": "  
12  }  
13 }
```

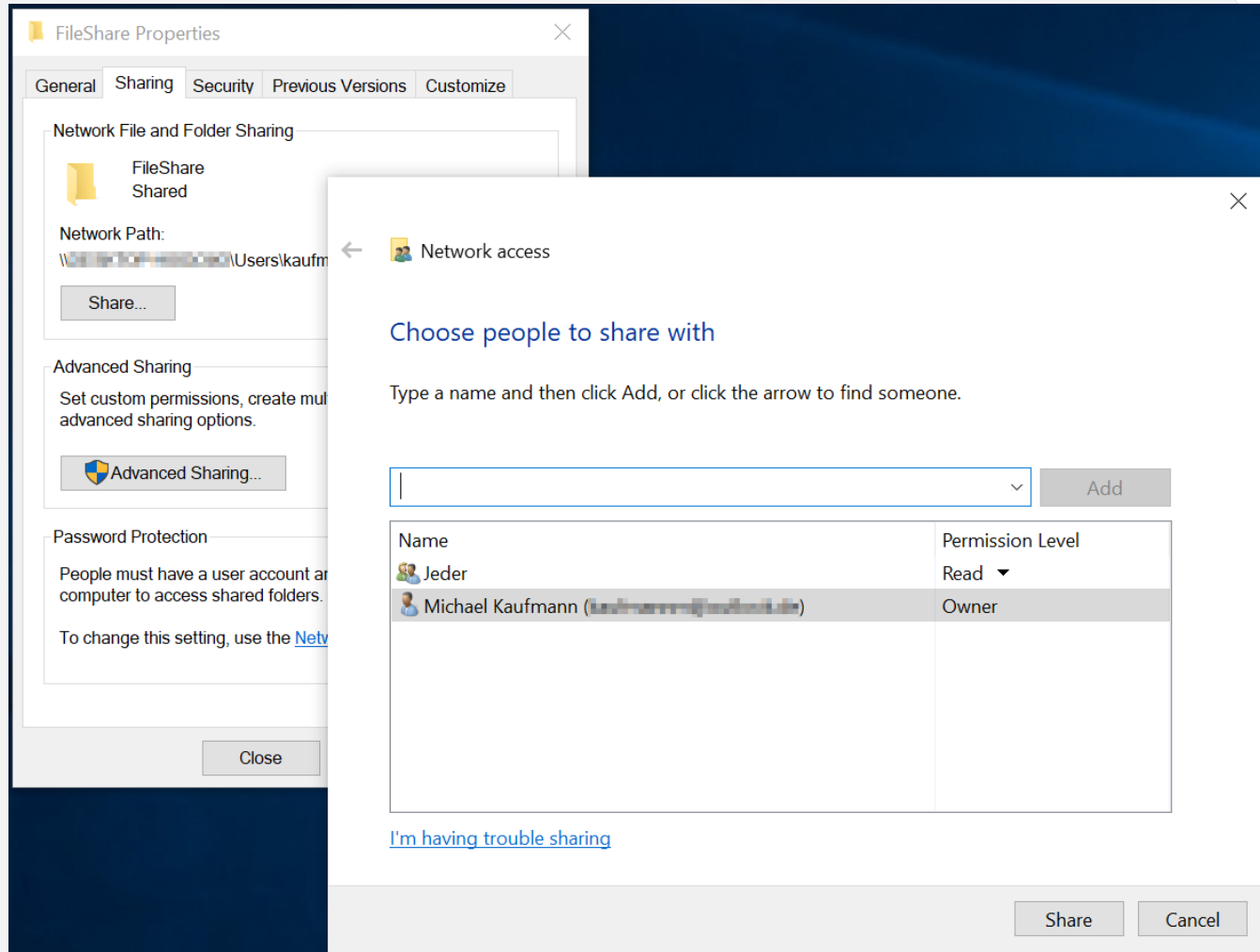


# Credentials in text files

27 lines (27 sloc) | 1.45 KB

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <configuration>
3   <configSections>
4     <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsG
5       <section name="SPOEmulators.Tests.Properties.Settings" type="System.Configuration.Cl
6     </sectionGroup>
7   </configSections>
8   <applicationSettings>
9     <SPOEmulators.Tests.Properties.Settings>
10      <setting name="OnPremUrl" serializeAs="String">
11        <!-- Enter thr url to a on prem site for integration testing. -->
12        <value>https://localhost/sites/dev</value>
13      </setting>
14      <!-- Enter thr url to a 0365 site for integration testing. -->
15      <setting name="0365Url" serializeAs="String">
16        <value>http://xxxx.sharepoint.com</value>
17      </setting>
18      <!-- Enter credentials to connect to the site (0365 or on prem if necessary) -->
19      <setting name="0365User" serializeAs="String">
20        <value>user@tenant.onmicrosoft.com</value>
21      </setting>
22      <setting name="0365Password" serializeAs="String">
23        <value>****</value>
24      </setting>
25    </SPOEmulators.Tests.Properties.Settings>
26  </applicationSettings>
27 </configuration>
```

# Unsecured file shares / visible repositories



⚡ kaufm@DESKTOP-HI0G080 ~\OneDrive\Downloads\mimikatz\_trunk\x64

[11:21]

> .\mimikatz.exe

```
.#####.   mimikatz 2.2.0 (x64) #19041 May 31 2021 00:08:47
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > https://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX           ( vincent.letoux@gmail.com )
'#####'   > https://pingcastle.com / https://mysmartlogon.com ***/
```

```
mimikatz # privilege::debug
Privilege '20' OK
```

```
mimikatz # sekurlsa::logonpasswords
```

```
Authentication Id : 0 ; 12361794 (00000000:00bca042)
Session           : Service from 0
User Name         : 77046A79-3B72-43E3-B7DD-EME189B6DC86
Domain           : NT VIRTUAL MACHINE
Logon Server      : (null)
Logon Time        : 22.06.2021 08:26:38
SID               : S-1-5-83-1-2010489593-1138965362-3789610473-2262611593
```

```
msv :
```

```
tspkg :
```

```
wdigest :
```

```
* Username : DESKTOP-HI0G080$
```

```
* Domain   : WORKGROUP
```

```
* Password : (null)
```

```
kerberos :
```

```
ssp :
```

\* Username : kaufmann@outlook.de  
\* Domain : MicrosoftAccount  
\* Password : (null)

ssp :

credman :

[0000000]

\* Username : mkgwritesabout.net  
\* Domain : outlook.office365.com  
\* Password : [REDACTED]

[0000001]

\* Username : (null)  
\* Domain : MicrosoftOffice16 Data:SSPI:mkgwritesabout.net  
\* Password : [REDACTED]

[0000002]

\* Username : github.auth  
\* Domain : vscodevscode.github-authentication/github.auth  
\* Password : [REDACTED]

44 4b 63 6c 43 46 37 4c 4f 72 4e 37 67 66 6a 44 55 36 4b 6a 51 36 65 6b 64 45 38 33 2f 8d 8a 35 61 78 57 42 4e 58 44 71  
42 43 4a 52 44 74 42 39 47 63 34 45 4a 59 5a 69 43 36 4b 4a 2f 53 49 68 39 6e 55 67 6f 4e 6e 53 57 67 71 4a 47 6d 33 58  
4e 38 34 34 75 4c 6e 4c 52 42 2f 58 32 43 51 8d 8a 45 41 45 61 2f 34 5a 4e 86 2b 47 54 6b 53 56 4d 65 75 52 79 49 74 4e  
56 76 5a 5a 6f 35 42 68 4e 75 75 58 61 38 2f 4f 6a 58 5a 38 62 7a 7a 53 71 58 74 61 62 43 78 74 31 6b 6f 36 35 59 74 6d  
72 8d 8a 53 38 2b 72 33 6c 69 47 46 34 42 55 32 48 6a 78 4f 5a 73 43 47 6a 6c 46 63 37 6d 72 46 6f 4c 78 64 78 6a 37 59  
64 43 2b 47 48 5a 4a 37 68 79 2b 78 48 6f 48 48 7a 32 51 76 6d 2f 78 5a 6d 38 75 8d 8a 38 55 49 6c 47 38 52 2b 71 37 47  
4d 55 74 44 6f 72 43 7a 69 4a 78 4a 47 31 59 67 49 51 34 51 45 79 61 7a 55 4e 49 67 4f 78 63 68 74 47 4f 57 44 41 6e 35  
5a 53 6d 2b 41 4c 51 37 38 6b 37 53 6a 8d 8a 64 5a 43 34 74 6c 65 6e 75 31 77 60 65 35 59 59 60 2f 68 33 56 32 78 39 5a

69 57 37 74 2f 54 6e 2f 64 6d 74 55 73 2f 42 6e 41 3d 3d 0d 0a 00 00

[00000003]

\* Username : wulfland  
\* Domain : GitHub - https://api.github.com/wulfland  
\* Password : 001d005700410070003

[00000004]

\* Username : wulfland@hotmail.com  
\* Domain : https://gitlab.com  
\* Password : 001d005700410070003

cloudap :

Cachedir : d470ef8c8f14149c4504f15101041052c5fe9c1a53cc2b3ea1fdaf444010495d  
Key GUID : {00000000-5ffc-30a9-0000-000000000000}  
PRT :   
DPAPI Key: 5800190075006f0019006a001d0057004100700034001b00300234005100670019005002310003006100790051006000

58001900c0063007000300050005500970eff4252dd33423c0f46db9ff01ec0901b60c4f5af (sha1: 1057570c0ea9fc74860051f3e1f85sec  
e041f575)

Authentication Id : 0 ; 195058 (00000000:0002f9f2)  
Session : Interactive from 1  
User Name : DWM-1  
Domain : Window Manager  
Logon Server : (null)  
Logon Time : 21.06.2021 22:24:46  
SID : S-1-5-00-0-1

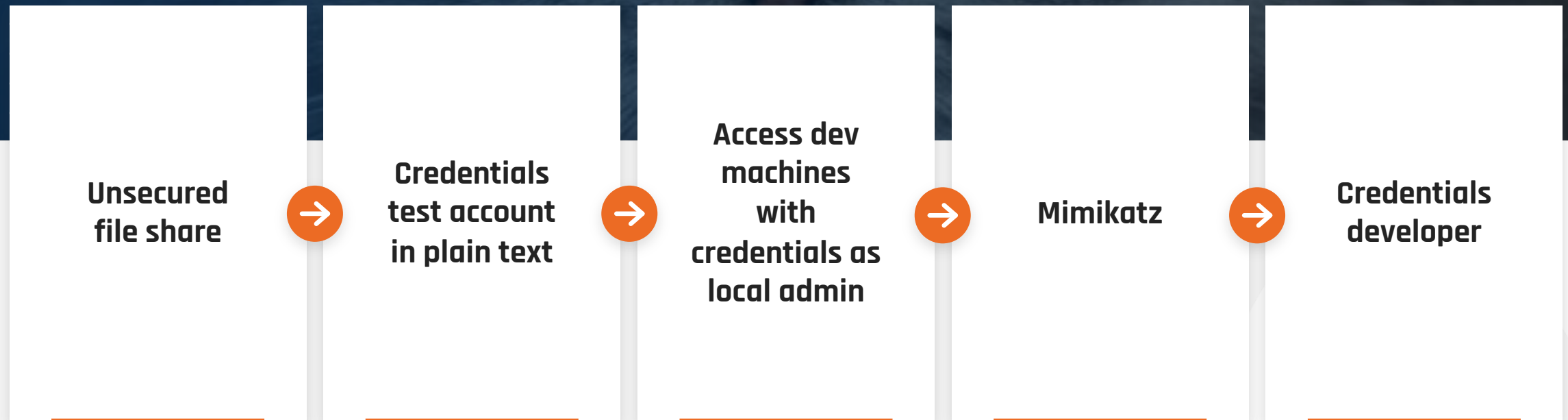
msv :

tspkg :

wdigest :

\* Username : DESKTOP-H10G090\$  
\* Domain : WORKGROUP  
\* Password : {null}

# Example



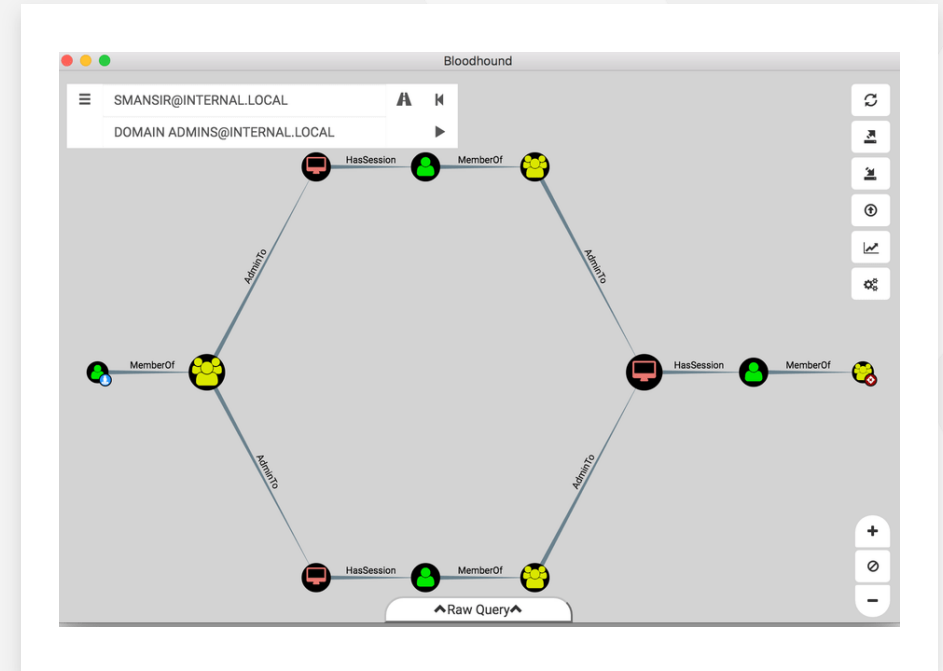
# From Dev to Prod

▶ **Bloodhound:** <https://github.com/adaptivethreat/Bloodhound>

▶ **grpMgr test01 Administrators /enum**

▶ **Other possible entry points:**

- > Phishing
- > Responder (<https://github.com/lqandx/Responder>)
- > Pineapple
- > ...
- > Weak passwords



TestAccounts.txt  
dom\test01



mkadev01  
dom\mka  
dom\test01



testsrv01  
dom\mka  
dom\admlisa



prod01  
dom\admlisa

A black and white photograph of Michael V. Hayden, a former General of the US Air Force and former Director of NSA and CIA. He is wearing a dark military uniform with "U.S." insignia on the lapels and a pilot's wings on his left chest. He is wearing glasses and has a serious expression. The background is a blurred American flag.

—  
Assume  
breach

“Fundamentally, if somebody wants to get in, they’re getting in. Alright, good. Accept that.”

- Michael V. Hayden

former General of the US Air Force and  
former Director of NSA and CIA



# Zero-Trust-Policy



All systems are protected like if they were connected to the internet

---



MFA, SSL, always patched



Least privilege principle

---



Separate accounts

# What to do?

- ▶ Virtual development environments
- ▶ Specific for project
- ▶ No local admin rights
- ▶ Codespaces
- ▶ Secret scanning

# Secret Scanning

## ▶ Code

- › GitHub Secret Scanning
- › gitLeaks
- › SpectralOps
- › Git-Secrets
- › Whispers
- › Gittyleaks
- › Git-all-secrets
- › ...

## ▶ Fileshare

- › Bash/PowerShell
- › Dumpster

<b>Adafruit IO</b> Adafruit IO Key	<b>Dropbox</b> Dropbox Access Token Dropbox Short Lived Access Token	<b>Plivo</b> Plivo Auth Token
<b>Adobe</b> Adobe Device Token Adobe JSON Web Token Adobe Service Token Adobe Short-Lived Access Token	<b>Dynatrace</b> Dynatrace Access Token Dynatrace Internal Token	<b>Postman</b> Postman API Key
<b>Alibaba Cloud</b> Alibaba Cloud Access Key ID and Access Key Secret pair	<b>Finicity</b> Finicity App Key	<b>Proctorio</b> Proctorio Consumer Key Proctorio Linkage Key Proctorio Registration Key Proctorio Secret Key
<b>Amazon Web Services (AWS)</b> Amazon AWS Access Key ID and Secret Access Key pair	<b>Frame.io</b> Frame.io Developer Token Frame.io JSON Web Token	<b>Pulum</b> Pulum Access Token
<b>Atlassian</b> Atlassian API Token Atlassian JSON Web Token	<b>GitHub</b> GitHub App Installation Access Token GitHub OAuth Access Token GitHub Personal Access Token GitHub Refresh Token GitHub SSH Private Key	<b>PyPI</b> PyPI API Token
<b>Azure</b> Azure Active Directory Application Secret Azure DevOps Personal Access Token Azure SAS Token Azure Service Management Certificate Azure SQL Connection String Azure Storage Account Key	<b>GoCardless</b> GoCardless Live Access Token GoCardless Sandbox Access Token	<b>RubyGems</b> RubyGems API Key
<b>Clojars</b> Clojars Deploy Token	<b>Google Cloud</b> Google API Key Google Cloud Private Key ID	<b>Samsara</b> Samsara API Token Samsara OAuth Access Token
<b>CloudBees CodeShip</b> CloudBees CodeShip Credential	<b>Hashicorp Terraform</b> Terraform Cloud / Enterprise API Token	<b>SendGrid</b> SendGrid API Key
<b>Databricks</b> Databricks Access Token	<b>Hubspot</b> Hubspot API Key	<b>Shopify</b> Shopify Access Token Shopify App Shared Secret Shopify Custom App Access Token Shopify Private App Password
<b>Datadog</b> Datadog API Key	<b>Mailchimp</b> Mailchimp API Key Mandrill API Key	<b>Slack</b> Slack API Token Slack Incoming Webhook URL Slack Workflow Webhook URL
<b>Discord</b> Discord Bot Token	<b>Mailgun</b> Mailgun API Key	<b>SSLMate</b> SSLMate API Key SSLMate Cluster Secret
<b>Doppler</b> Doppler CLI Token Doppler Personal Token Doppler SCIM Token Doppler Service Token	<b>MessageBird</b> MessageBird API Key	<b>Stripe</b> Stripe Live API Restricted Key Stripe Live API Secret Key Stripe Test API Restricted Key Stripe Test API Secret Key
	<b>npm</b> npm Access Token	<b>Tencent Cloud</b> Tencent Cloud Secret ID
	<b>NuGet</b> NuGet API Key	<b>Twilio</b> Twilio Account String Identifier Twilio API Key
	<b>OpenAI</b> OpenAI API Key	<b>Valour</b> Valour Access Token
	<b>Palantir</b> Palantir JSON Web Token	



# Attack vector: supply chain



# Supply Chain Attacks

## Libraries / Packages



### All libraries used in your applications:

- › Authentication
- › Encryption
- › Backend access
- › ...

## Software



### Software and tooling used in the process of building your application:

- › npm ci
- › dotnet build / msbuild
- › Terraform
- › Splunk
- › ...

# Know your dependencies!

▶ Naming conflict of npm package with Kick in 2016 ( <https://www.kick.com/> )

---

▶ Npm sides with kick

---

▶ Azer Koçulu retracted all packages  
- one of them **left-pad**

---

▶ 11 lines of code - broke the internet

---

```
1 module.exports = leftpad;
2 function leftpad (str, len, ch) {
3   str = String(str);
4   var i = -1;
5   if (!ch && ch !== 0) ch = ' ';
6   len = len - str.length;
7   while (++i < len) {
8     str = ch + str;
9   }
10  return str;
11 }
```

# Software Composition Analysis (SCA)

- > GitHub (Dependency-Graph/Dependabot)
- > anchore (<https://anchore.com/> )
- > Dependency-Track (<https://dependencytrack.org/> )

**Dependabot alerts** Dismiss all

4 Open ✓ 0 Closed Manifest Sort

<b>marsdb</b> 3 minutes ago by GitHub  package.json <span>critical severity</span>
<b>express-jwt</b> 3 minutes ago by GitHub  package.json <span>high severity</span>
<b>sanitize-html</b> 3 minutes ago by GitHub  package.json <span>moderate severity</span>
<b>jsonwebtoken</b> 3 minutes ago by GitHub  package.json <span>critical severity</span>

## Dependency graph

Dependencies Dependents Dependabot

We found potential security vulnerabilities in your dependencies.  
Dependencies defined in these manifest files have known security vulnerabilities and should be updated:  
**package.json** 7 vulnerabilities found  
[View Dependabot alerts](#)  
Only the owner of this repository can see this message.

These dependencies are defined in **workshop-2021-learning-journey's** manifest files, such as [package.json](#) and [frontend/package.json](#).

Dependencies defined in **package.json** 138

> <b>auth0 / express-jwt</b> <span>Known security vulnerability in <u>0.1.3</u></span>
> <b>auth0 / node-jsonwebtoken</b> jsonwebtoken <span>Known security vulnerability in <u>0.4.0</u></span>
> <b>c58 / marsdb</b> <span>Known security vulnerability in <u>0.6.11</u></span>
> <b>apostrophecms / sanitize-html</b> <span>Known security vulnerability in <u>1.4.2</u></span>
> <b>istanbuljs / istanbuljs</b> @istanbuljs/nyc-config-typescript <span>^ 1.0.1</span>
<b>Seally / types-chai</b> @types/chai <span>^ 4.2.14</span>
> <b>DefinitelyTyped / DefinitelyTyped</b> @types/chai-as-promised <span>^ 7.1.3</span>

# Frameworks



▶ **OWASP**  
Software Component Verification Standard

---

▶ v1 since 2020: <https://xpir.it/SCVS>

▶ **S**upply chain **L**evels for **S**oftware  
**A**rtifacts

---

▶ Currently in Alpha: <https://slsa.dev/>



# OWASP SCVS

	L1	L2	L3	L4
<b>V1</b> - Inventory				
<b>V2</b> - Software Bill of Materials (SBOM)				
<b>V3</b> - Build Environment				
<b>V4</b> - Package Management				
<b>V5</b> - Component Analysis				
<b>V6</b> - Pedigree and Provenance				



<https://owasp-scvs.gitbook.io>

# Software Bill of Materials (V2 OWASP SCVS)

## Multiple standards for SBoM formats:

### ▶ Software Package Data Exchange (SPDX)

- › Linux Foundation
- › Focusses on license information
- › ISO/IEC 5962:2021 - fulfills NTIA's minimum elements for a SBoM
- › Syft, Anchore  
( <https://github.com/marketplace/actions/anchore-sbom-action> )

### ▶ CycloneDX (CDX)

- › OWASP
- › Focusses on vulnerabilities and security
- › Used in OWASP Dependency Track
- › <https://cyclonedx.org/>

```
- name: Anchore SBOM Action
  uses: anchore/sbom-action@v0.6.0
  with:
    path: .
    image: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
    registry-username: ${{ github.actor }}
    registry-password: ${{ secrets.GITHUB_TOKEN }}
```

<https://github.com/wulfland/container-demo/actions/runs/2179243137>

### ▶ Software Identification Tags (SWID)

- › SWID is an ISO/IEC industry standard (ISO/IEC 19770-2)
- › Focus on inventory in Software Asset Management
- › Snow, System Center, ServiceNow ITOM

# What to do?

- ▶ Know your dependencies
- ▶ Keep your dependencies up to date
- ▶ Ephemeral build environments



# Attack vector: vulnerabilities



# OWASP TOP 10

( <https://owasp.org/www-project-top-ten/> )

01 A01:2021-Broken Access Control

02 A02:2021-Cryptographic Failures

03 A03:2021-Injection

04 A04:2021-Insecure Design

05 A05:2021-Security Misconfiguration

06 A06:2021-Vulnerable and Outdated Components

07 A07:2021-Identification and Authentication Failures

08 A08:2021-Software and Data Integrity Failures

09 A09:2021-Security Logging and Monitoring Failures

10 A10:2021-Server-Side Request Forgery

# A03:2021 - Injection

- › 94% of the applications were tested for some form of injection
- › max incidence rate of 19%, an average incidence rate of 3%, and 274k occurrences.
- › 33 CWEs mapped. For example:
  - › CWE-79: Cross-site Scripting (XSS)
  - › CWE-89: SQL Injection
  - › CWE-73: External Control of File Name or Path



# SQL Injection

- > txtUserId = getRequestString("UserId");  
txtSQL = "SELECT \* FROM Users WHERE UserId = " + txtUserId;
- > 105; DROP TABLE Suppliers

```
11 module.exports = function searchProducts () {
12   return (req, res, next) => {
13     let criteria = req.query.q === 'undefined' ? '' : req.query.q || ''
14     criteria = (criteria.length <= 200) ? criteria : criteria.substring(0, 200)
15     models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%${criteria}%' OR description LIKE '%${criteria}%') AND deletedAt IS NULL) ORDER BY name`)
16     .then([products]) => {
17       const dataString = JSON.stringify(products)
18       if (utils.notSolved(challenges.unionSqlInjectionChallenge)) { // vuln-code-snippet hide-start
19         let solved = true
20         models.User.findAll().then(data => {
21           const users = utils.queryResultToJson(data)
22           if (users.data?.length) {
23             for (let i = 0; i < users.data.length; i++) {
24               solved = solved && utils.containsOrEscaped(dataString, users.data[i].email) && utils.contains(dataString, users.data[i].password)
25               if (!solved) {
26                 break
27               }
28             }
29             if (solved) {
30               utils.solve(challenges.unionSqlInjectionChallenge)
31             }

```

# XSS (Cross-Site-Scripting)

```
143 // vuln-code-snippet start localXssChallenge xssBonusChallenge
144 filterTable () {
145   let queryParams: string = this.route.snapshot.queryParams.q
146   if (queryParams) {
147     queryParams = queryParams.trim()
148     this.ngZone.runOutsideAngular(() => { // vuln-code-snippet hide-start
149       this.io.socket().emit('verifyLocalXssChallenge', queryParams)
150     }) // vuln-code-snippet hide-end
151     this.dataSource.filter = queryParams.toLowerCase()
152     this.searchValue = this.sanitizer.bypassSecurityTrustHtml(queryParams)
153     this.gridDataSource.subscribe((result: any) => {
154       if (result.length === 0) {
155         this.emptyState = true
156       } else {
157         this.emptyState = false
158       }
159     })
160   } else {
161     this.dataSource.filter = ''
162     this.searchValue = undefined
163     this.emptyState = false
164   }
165 }
```

```
var Affix = function (element, options) {
  this.options = $.extend({}, Affix.DEFAULTS, options)

  this.$target = $(this.options.target)
  .on('scroll.bs.affix.data-api', $.proxy(this.checkPosition, this))
  .on('click.bs.affix.data-api', $.proxy(this.checkPositionWithEventLoop, this))

  this.$element = $(element)
  this.affixed = null
  this.unpin = null
  this.pinnedOffset = null

  this.checkPosition()
}
```



# Shift left security

# Static Application Security Testing (SAST)

## Whitebox-Testing

- › GitHub Code Analysis
- › SonarQube

- › Semgrep ( <https://semgrep.dev/> )
- › Mobile-Security-Framework (MobSF) (<https://github.com/MobSF/Mobile-Security-Framework-MobSF>)

Client-side cross-site scripting

Writing user input directly to the DOM allows for a cross-site scripting vulnerability.

Open Error CWE-79 CWE-116 security

Branch: main Dismiss

```
frontend/src/app/search-result/search-result.component.ts
149   this.io.socket().emit('verifyLocalXssChallenge', queryParam)
150   }) // vuln-code-snippet hide-end
151   this.dataSource.filter = queryParam.toLowerCase()
152   this.searchValue = this.sanitizer.bypassSecurityTrustHtml(queryParam) // vuln-code-snippet vuln-line localXssChallenge
```

Cross-site scripting vulnerability due to user-provided value.

CodeQL Show paths

```
153   this.gridDataSource.subscribe((result: any) => {
154     if (result.length === 0) {
155       this.emptyState = true
```

Tool	Rule ID	Query
CodeQL	js/xss	View source

Directly writing user input (for example, a URL query parameter) to a webpage without properly sanitizing the input first, allows for a cross-site scripting vulnerability.

Show more

First detected in commit 9474e68 3 days ago

Create codeql-analysis.yml

Verified 9474e68

frontend/src/app/search-result/search-result.component.ts#L152 on branch main



Tool	Rule ID	Query
CodeQL	js/xss	View source

Directly writing user input (for example, a URL query parameter) to a webpage without properly sanitizing the input first, allows for a cross-site scripting vulnerability.

This kind of vulnerability is also called *DOM-based* cross-site scripting, to distinguish it from other types of cross-site scripting.

### Recommendation

To guard against cross-site scripting, consider using contextual output encoding/escaping before writing user input to the page, or one of the other solutions that are mentioned in the references.

### Example

The following example shows part of the page URL being written directly to the document, leaving the website vulnerable to cross-site scripting.

```
function setLanguageOptions() {
  var href = document.location.href,
      deflt = href.substring(href.indexOf("default=")+8);
  document.write("<OPTION value=1">deflt"</OPTION>");
  document.write("<OPTION value=2">English</OPTION>");
}
```

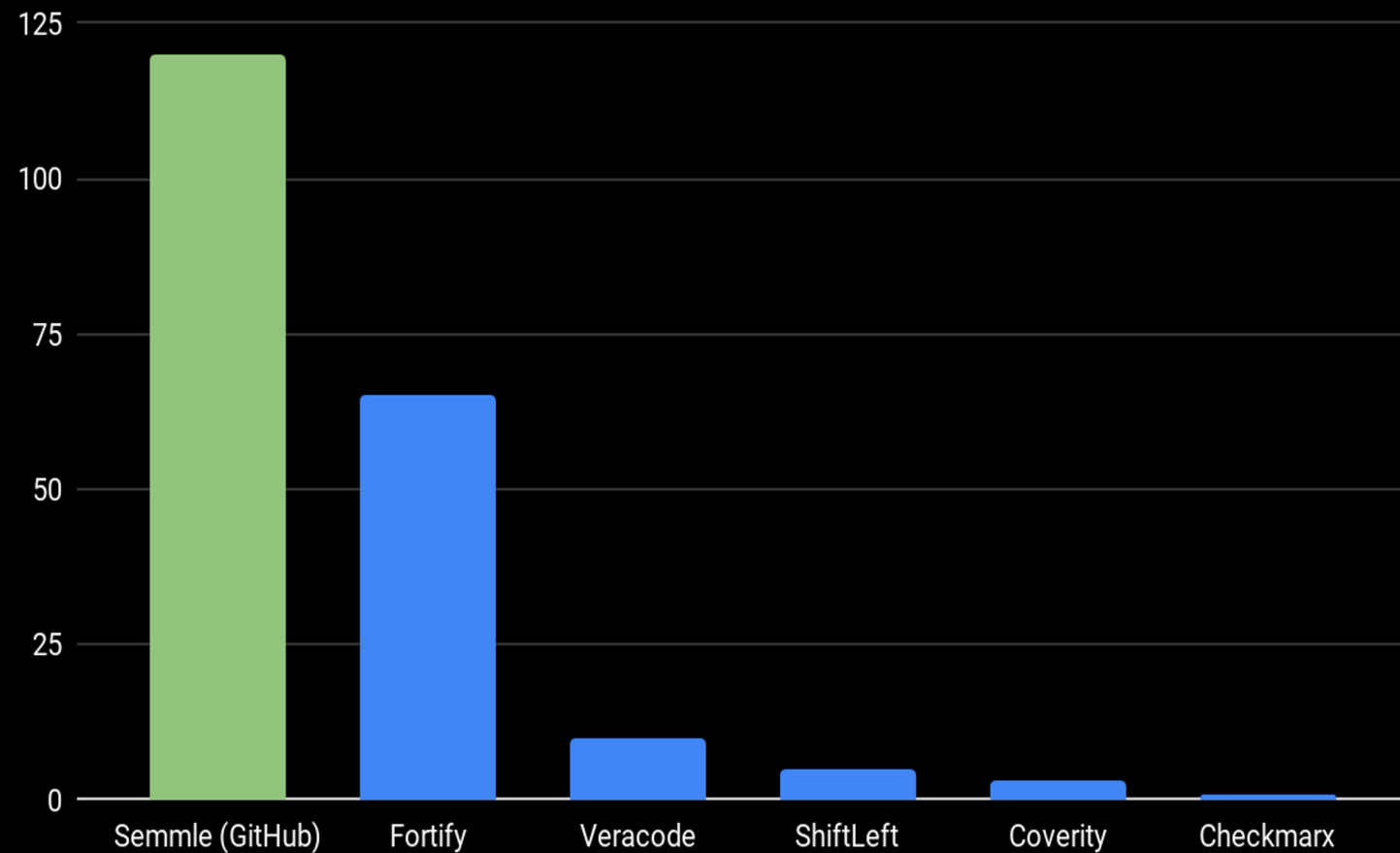
### References

- OWASP: [DOM based XSS Prevention Cheat Sheet](#).
- OWASP: [XSS \(Cross Site Scripting\) Prevention Cheat Sheet](#).
- OWASP [DOM Based XSS](#).
- OWASP [Types of Cross-Site Scripting](#).
- Wikipedia: [Cross-site scripting](#).
- Common Weakness Enumeration: [CWE-79](#).
- Common Weakness Enumeration: [CWE-116](#).

Show less

# Security researchers find more vulnerabilities with CodeQL

CVEs discovered, by vendor, 2018 - 2019



- More CVEs than any other SAST vendor team
- 50 CVEs in the last 3 months
- Testimonials from top security teams, including Microsoft and Uber

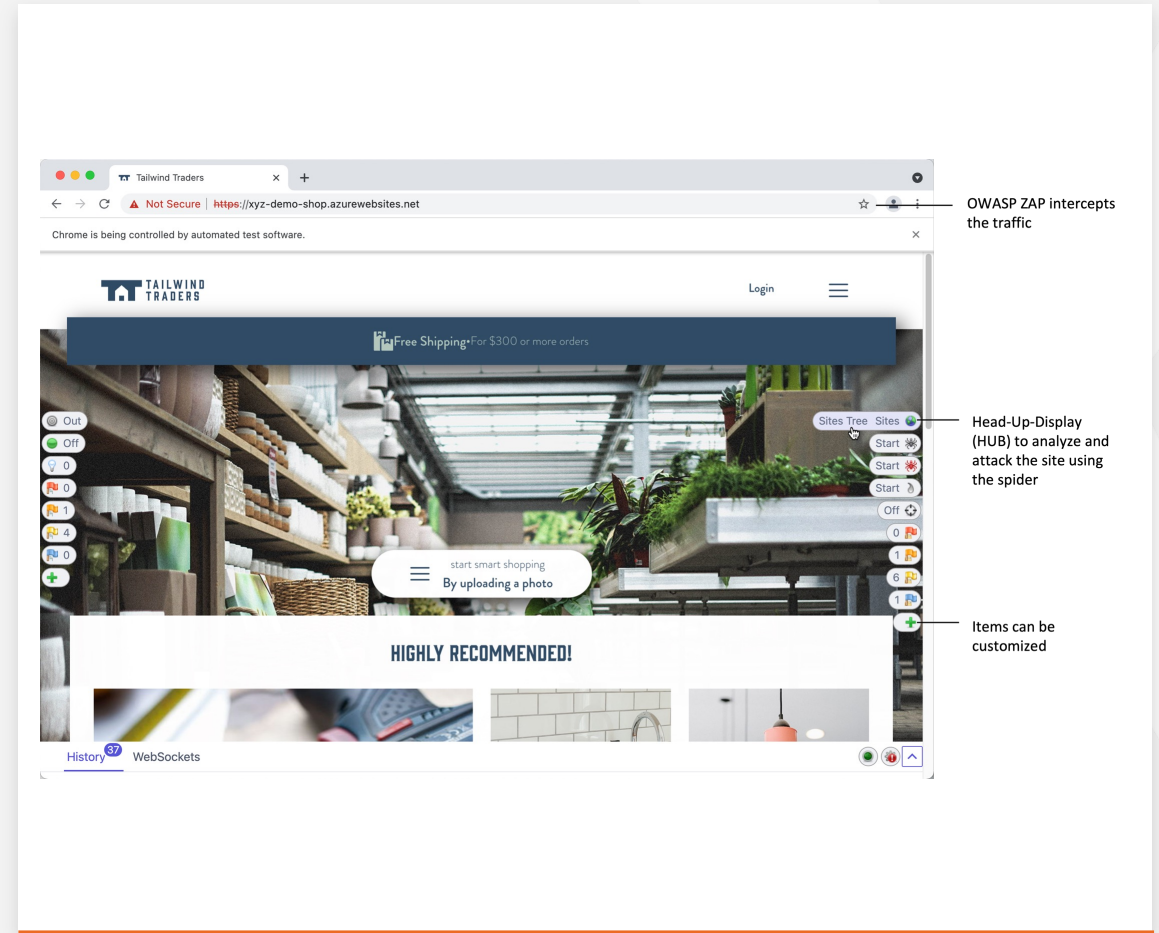
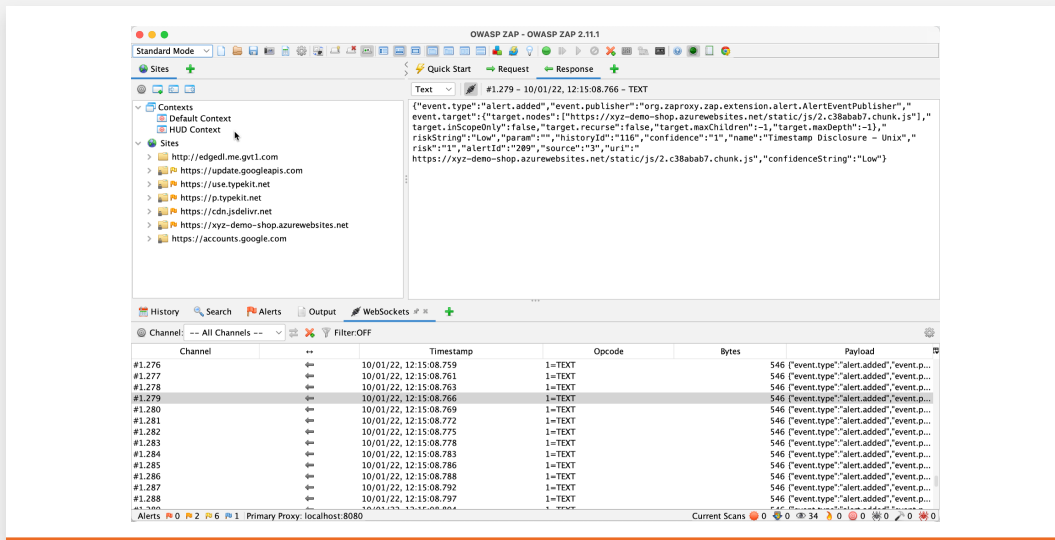
## Examples:

- Zip Slip at Microsoft
- Bug bounty at Uber

# Dynamic Application Security Testing (DAST)

## ▶ Blackbox-Testing

- › OWASP ZAP ( Zed Attack Proxy, <https://owasp.org/www-project-zap> )
- › Burp Suite von PortSwigger ( <https://portswigger.net/burp> )



# Dynamic Application Security Testing (DAST)

## ▶ Blackbox-Testing

- › OWASP ZAP ( Zed Attack Proxy, <https://owasp.org/www-project-zap> )
- › Burp Suite von PortSwigger ( <https://portswigger.net/burp> )

Search results for "OWASP ZAP" showing 3 results. The top result is "OWASP ZAP Full Scan" by zaproxy, which scans the web application with the OWASP ZAP Full Scan. Below the search results is a workflow snippet:

```
jobs:  
  owasp:  
    name: OWASP Full Scan  
    runs-on: ubuntu-latest  
    steps:  
      - name: OWASP ZAP Full Scan  
        uses: zaproxy/action-full-scan@v0.2.0  
        with:  
          # GitHub Token to create issues in the repository  
          token: ${{ github.token }}  
          target: https://target
```

**ZAP Scanning Report**  
Sites: <http://xyz-demo-shop.azurewebsites.net> <https://xyz-demo-shop.azurewebsites.net>  
Generated on Sun, 9 Jan 2022 19:16:45

**Summary of Alerts**

Risk Level	Number of Alerts
High	0
Medium	3
Low	8
Informational	3
False Positives:	0

**Alerts**

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	3
Missing Anti-clickjacking Header	Medium	3
Proxy Disclosure	Medium	18
Cookie with SameSite Attribute None	Low	2
Cookie without SameSite Attribute	Low	2
HTTPS Content Available via HTTP	Low	11
Incomplete or No Cache-control Header Set	Low	5
Private IP Disclosure	Low	1
Strict-Transport-Security Header Not Set	Low	11
Timestamp Disclosure - Unix	Low	20
X-Content-Type-Options Header Missing	Low	11
Cookie Stack Detector	Informational	18
Information Disclosure - Suspicious Comments	Informational	2
Modern Web Application	Informational	4

**Alert Detail**

Medium	Content Security Policy (CSP) Header Not Set
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate attacks such as Cross Site Scripting (XSS) and data injection attacks. These attacks are used for site defacement or distribution of malware. CSP provides a set of standard HTTP headers to site owners to declare approved sources of content that browsers should be allowed to load. Types are JavaScript, CSS, HTML, frames, fonts, images and embeddable objects such as audio and video files.
URL	<a href="https://xyz-demo-shop.azurewebsites.net/">https://xyz-demo-shop.azurewebsites.net/</a>

A report is added as a build artifact in HTML, JSON, and Markdown

ZAP Full Scan Report #173

github-actions bot · opened this issue 21 hours ago · 0 comments

github-actions bot · commented 21 hours ago

New Alerts

- Content Security Policy (CSP) Header Not Set [10038] total: 3:
  - <https://xyz-demo-shop.azurewebsites.net/>
  - <https://xyz-demo-shop.azurewebsites.net/robots.txt>
  - <https://xyz-demo-shop.azurewebsites.net/sitemap.xml>
- Missing Anti-clickjacking Header [10020] total: 3:
  - <https://xyz-demo-shop.azurewebsites.net/>
  - <https://xyz-demo-shop.azurewebsites.net/robots.txt>
  - <https://xyz-demo-shop.azurewebsites.net/sitemap.xml>
- Proxy Disclosure [40025] total: 18:
  - <https://xyz-demo-shop.azurewebsites.net/>
  - <https://xyz-demo-shop.azurewebsites.net/apple-touch-icon.png>
  - <https://xyz-demo-shop.azurewebsites.net/favicon-16x16.png>
  - <https://xyz-demo-shop.azurewebsites.net/favicon-32x32.png>

# Infrastructure Scanning

## ▶ Container Vulnerability Analysis (CVA) / Container Security Analysis (CSA)

### ▶ Open source:

- ▶ Anchore gryp  
<https://github.com/anchore/grype/>
- ▶ Clair  
<https://quay.github.io/clair/>

### ▶ Commercial:

- ▶ WhiteSource  
<https://www.whitesourcesoftware.com/solution-for-containers/>
- ▶ Aqua  
<https://www.aquasec.com/products/container-security/>

```
- name: Anchore Container Scan
  uses: anchore/scan-action@v3.2.0
  with:
    image: ${ env.REGISTRY }/${ env.IMAGE_NAME }
    debug: true
```

<https://github.com/wulfland/container-demo/actions/runs/2179243137>

The screenshot shows a GitHub Actions workflow named 'build-and-push-image' that failed 3 minutes ago. The workflow consists of several steps, most of which are successful. The 'Anchore Container Scan' step has failed with an error.

```
build-and-push-image
failed 3 minutes ago in 22s

> [✓] Set up job 3s
> [✓] Checkout repository 1s
> [✓] Log in to the Container registry 0s
> [✓] Extract metadata (tags, labels) for Docker 0s
> [✓] Build and push Docker image 4s
> [✓] Anchore SBOM Action 3s
▼ [✗] Anchore Container Scan 8s
  1 ▶ Run anchore/scan-action@v3.2.0
  12 /usr/bin/chmod +x /home/runner/work/_temp/ca1e5e61-229a-43b7-b70f-c317932ac1c0
  13 /home/runner/work/_temp/ca1e5e61-229a-43b7-b70f-c317932ac1c0 -b
  /home/runner/work/_temp/ca1e5e61-229a-43b7-b70f-c317932ac1c0_grype v0.27.3
  14 [info] checking github for release tag='v0.27.3'
  15 [info] fetching release script for tag='v0.27.3'
  16 anchore/grype info checking GitHub for tag 'v0.27.3'
  17 anchore/grype info found version: 0.27.3 for v0.27.3/linux/amd64
  18 anchore/grype info installed /home/runner/work/_temp/ca1e5e61-229a-43b7-b70f-
c317932ac1c0_grype/grype
  19
  20 Analyzing: ghcr.io/wulfland/container-demo
  21 Executing: grype -vv -o json --fail-on medium ghcr.io/wulfland/container-demo
  22 ▶ grype output...
  163 Error: Failed minimum severity level. Found vulnerabilities with level medium or
higher

> [✓] Post Build and push Docker image 0s
> [✓] Post Log in to the Container registry 0s
> [✓] Post Checkout repository 0s
> [✓] Complete job 0s
```

# Infrastructure Scanning

## ▶ Infrastructure policies

## ▶ Open source:

- › Checkov  
<https://www.aquasec.com/products/container-security/>
- › OpenVAS

## ▶ Commercial:

- › Defender for Cloud  
<https://azure.microsoft.com/en-us/services/defender-for-cloud>
- › Azure Policy  
<https://docs.microsoft.com/de-de/azure/governance/policy/>

```
- name: Checkov GitHub Action
uses: bridgecrewio/checkov-action@master
with:
  directory: ch15_sec/
  output_format: sarif

- name: Upload SARIF file
uses: github/codeql-action/upload-sarif@v1
with:
  sarif_file: results.sarif
if: always()
```

## Code scanning

Add more scanning tools

Latest scan	Branch	Workflow	Lines scanned	Duration	Result
10 minutes ago	main	CodeQL	1.45k / 1.39k ⓘ	5m 26s	21 alerts

Filters ▾ 🔍 tool:checkov is:open branch:main

✕ Clear current search, filters and sorts

<input type="checkbox"/>	✓ 2 Open	✕ 0 Closed	Tool ▾	Rule ▾	Branch ▾	Severity ▾	Sort ▾
<input type="checkbox"/>	Ensure that S3 bucket has a Public Access block			aws.tf:1	main	Error	
				Detected 15 minutes ago by checkov			
<input type="checkbox"/>	Ensure that S3 bucket has cross-region replication enabled			aws.tf:1	main	Error	
				Detected 15 minutes ago by checkov			

# What to do?

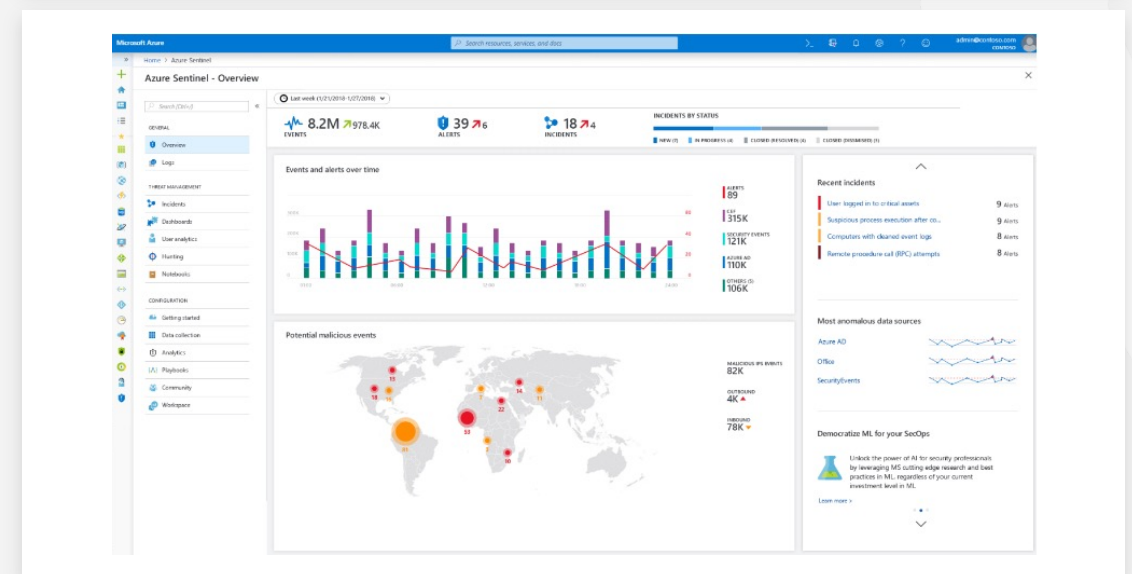
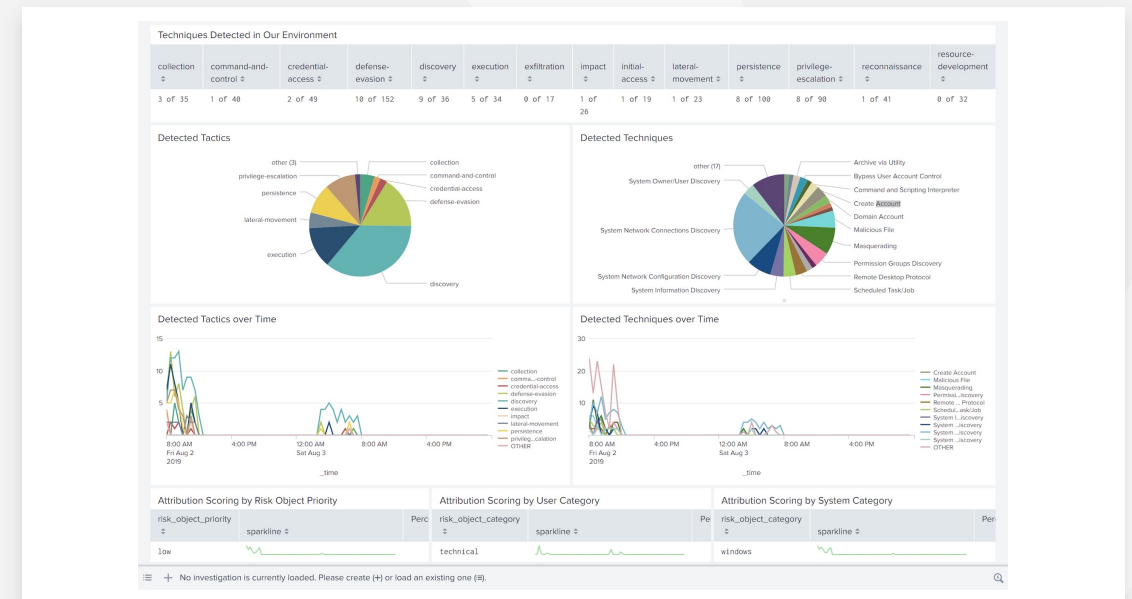
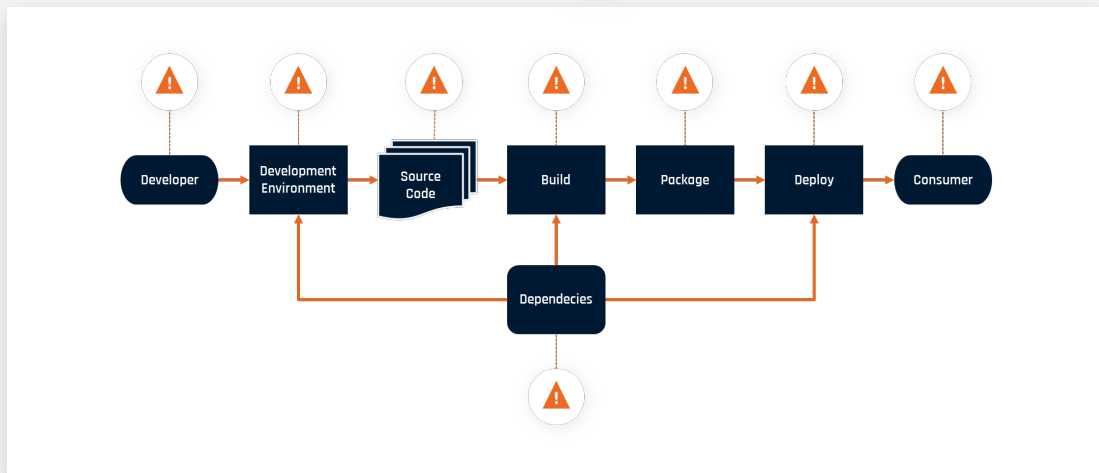
- ▶ SAST and DAST
- ▶ Infrastructure Scanning
- ▶ Shift left security
- ▶ Codespaces
- ▶ Secret scanning





# Security Information & Event Management (SIEM)

- ▶ Azure Sentinel
- ▶ Splunk
- ▶ Central logging
- ▶ Multi cloud/hybrid
- ▶ Detect anomalies (ML)
- ▶ Realtime warnings



# 6 tips to integrate security into your DevOps practices

1

**Build a security-first culture across the business**

2

**Integrate security in the early stages of the development lifecycle**

3

**Monitor and observe continuously with purpose**

4

**Embrace everything-as-code**

5

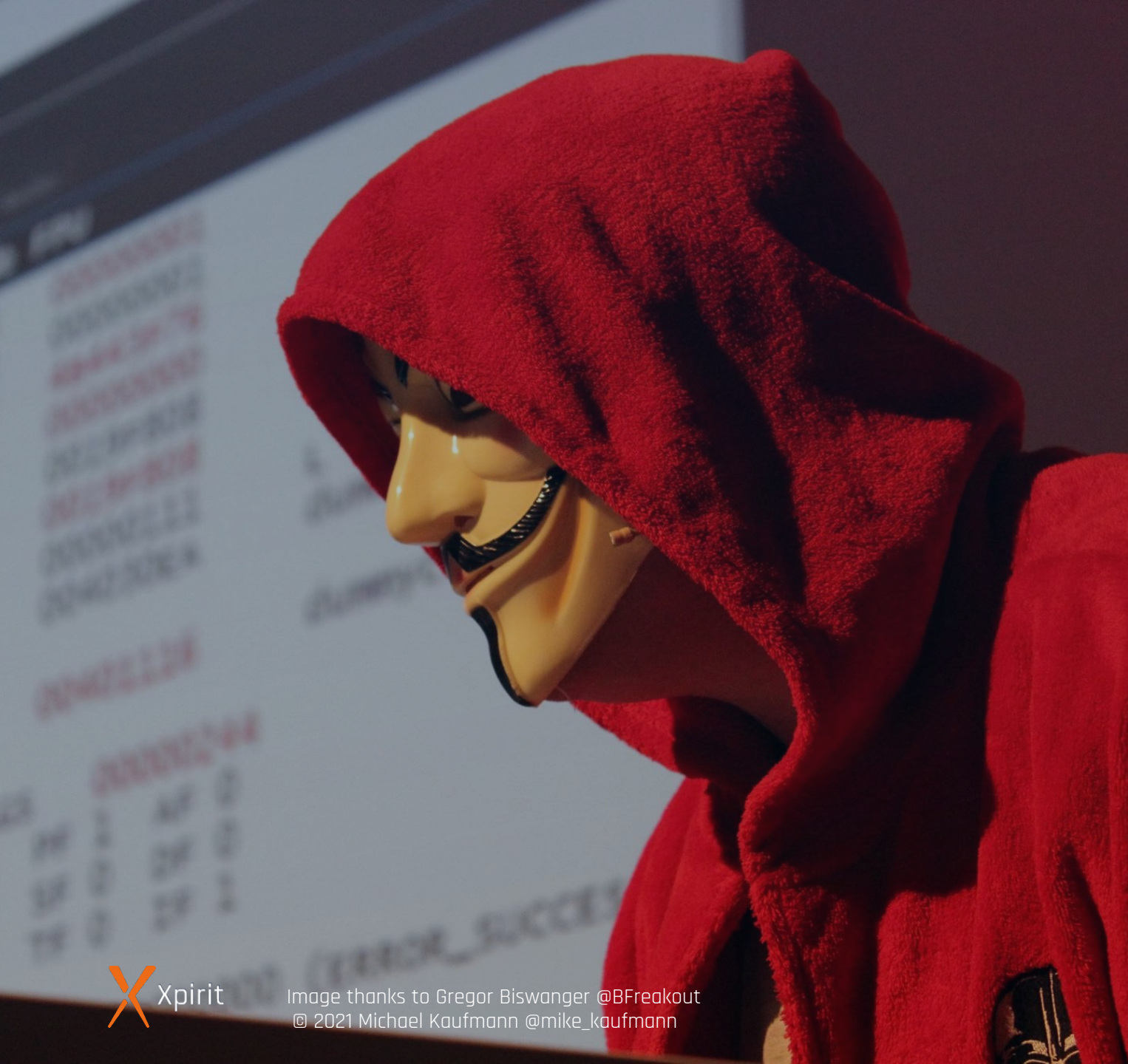
**Realize compliancy with policy automation**

6

**Secure and visualize your software supply chain**



<https://azure.microsoft.com/en-us/resources/6-tips-to-integrate-security-into-your-devops-practices/>



# Hackers in movies VS

# Thank you



Blog : <https://writeabout.net>



Twitter : @mike\_kaufmann



GitHub : @wulfland



LinkedIn : <https://www.linkedin.com/in/mikaufmann/>