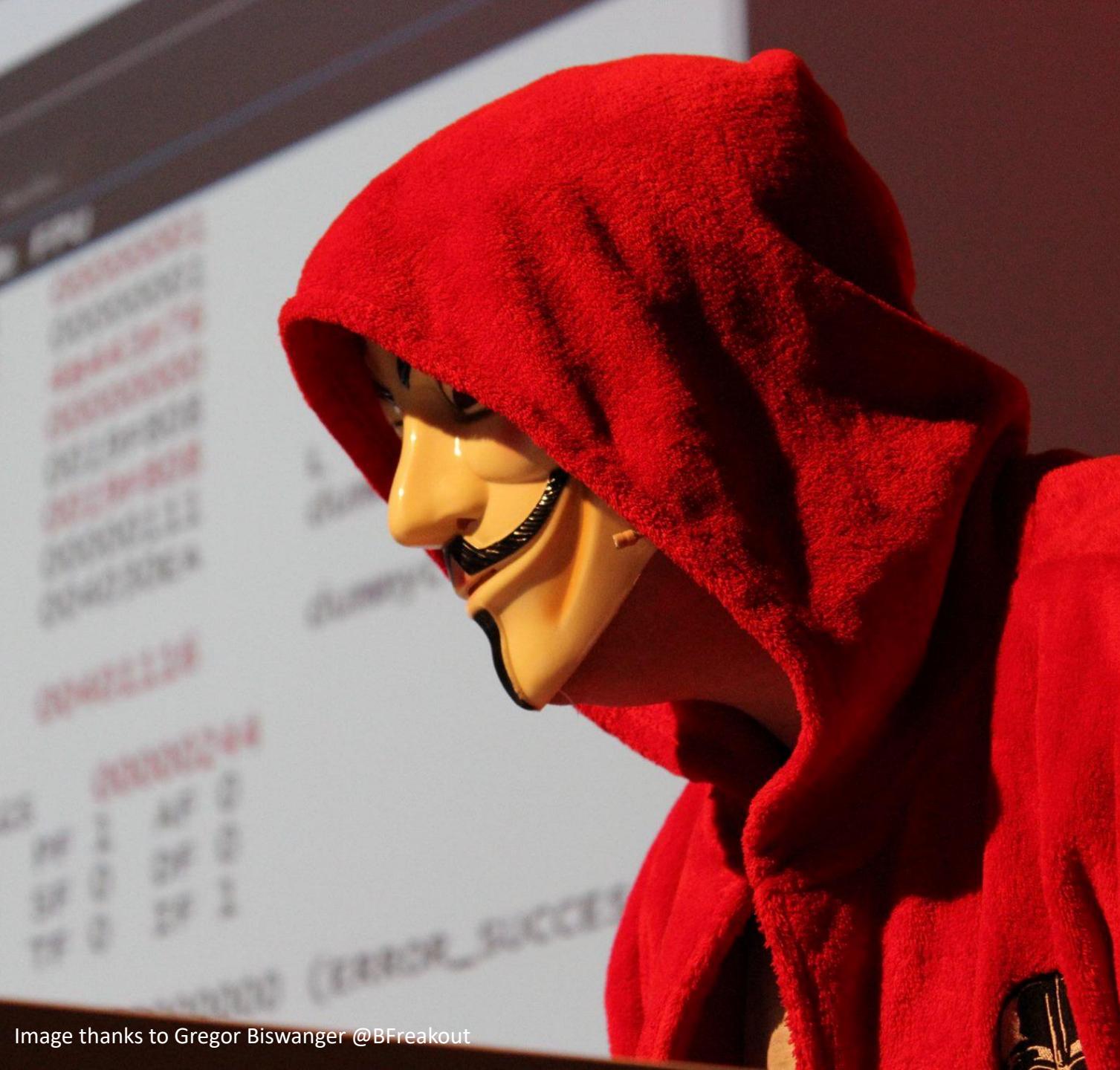


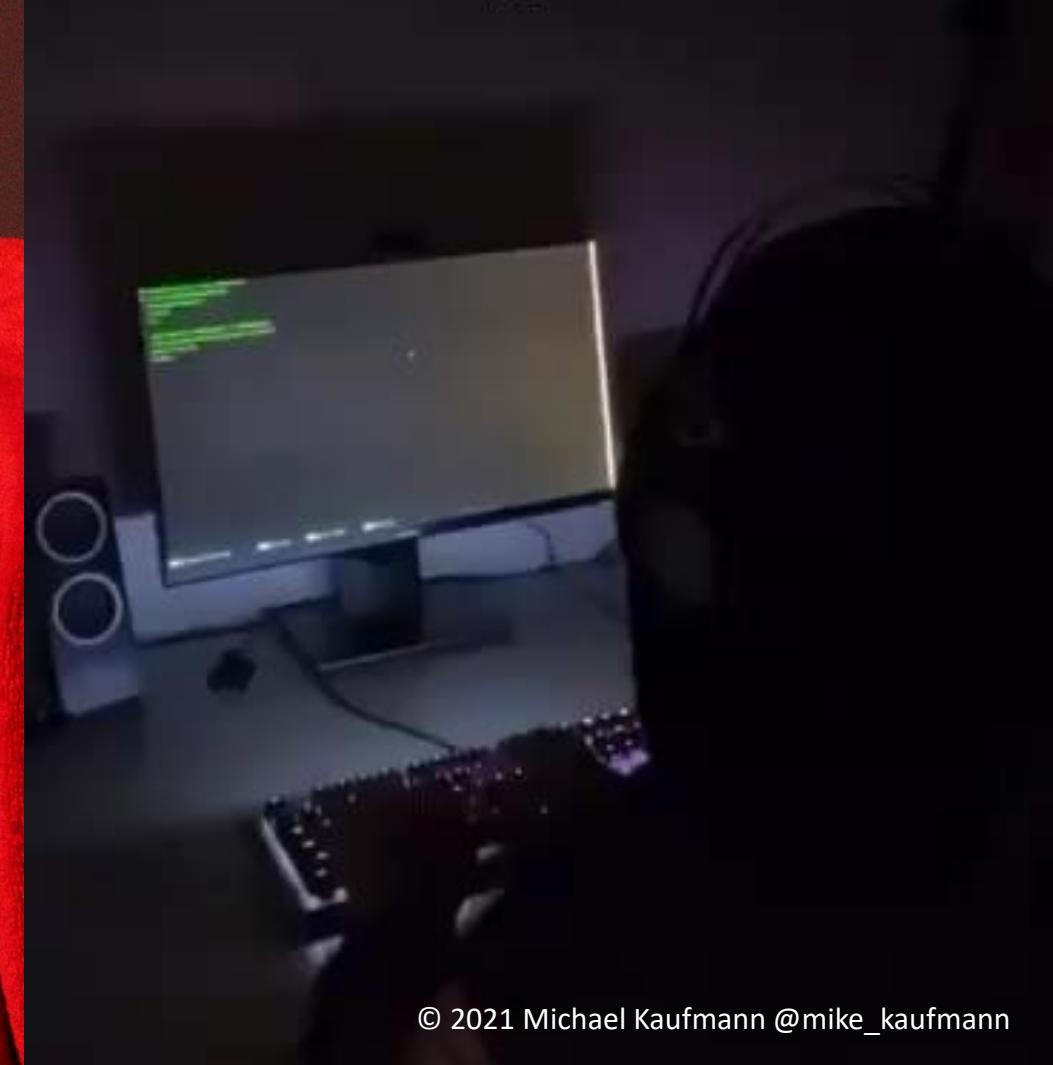
Secure DevOps

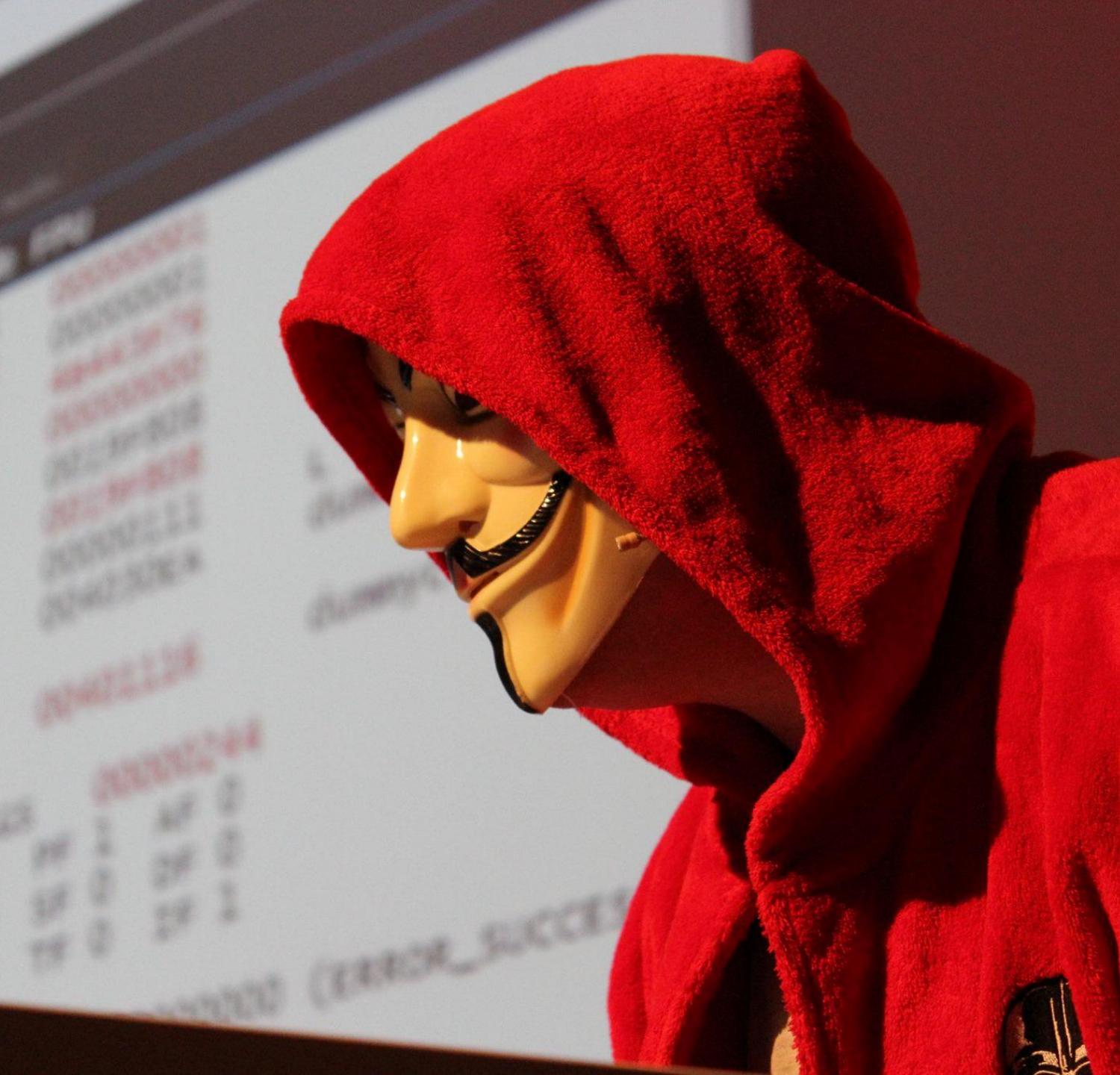
#DevSecOps

Seguridad en tiempos de despliegues diarios



Hackers in movies
vs





How people think they get hacked



How they really get hacked

The collage includes the following elements:

- A Facebook post from "94.1 KXQJ-B" asking, "If you had to marry your spouse where you met them, where would your wedding have been?" with 100K comments and 2.3M shares.
- A Facebook post from "Up 99.3" asking, "Your porn name , is your middle name , and the first car you had." with 26.4K comments and 1.87M shares.
- A Facebook post from "The Farmer's Wife" asking, "The car you passed your drivers test in was a _____" with 14.5K comments and 6.3M shares.
- A text overlay: "Asking where you'd get married again is like asking where you met."
- A text overlay: "A 'porn name' that exposes your middle name AND a street you grew up on are TWO pieces of info that people need."
- A text overlay: "STOP. THINK. DO NOT SHARE INFO."
- A Facebook post from "NAME A SONG" asking, "How far away do you live from the place you were born?" with 21.8K comments and 2.8M shares.
- A Facebook post from "THAT TAKES YOU BACK TO HIGH SCHOOL" showing a group of people from high school.
- A small note at the bottom right: "Don't forget to like, comment, and share! See what city or town was your first job?"

Cuando hablamos de seguridad...

- “Hasta ahora nunca hemos tenido un incidente con la seguridad”
- “Nuestros desarrolladores son expertos”
- “Nuestra empresa no es tan importante como para ser atacada”
- “Estamos bien protegidos”
- “Nuestros cortafuegos son seguros”

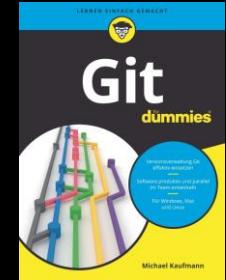


Michael Kaufmann

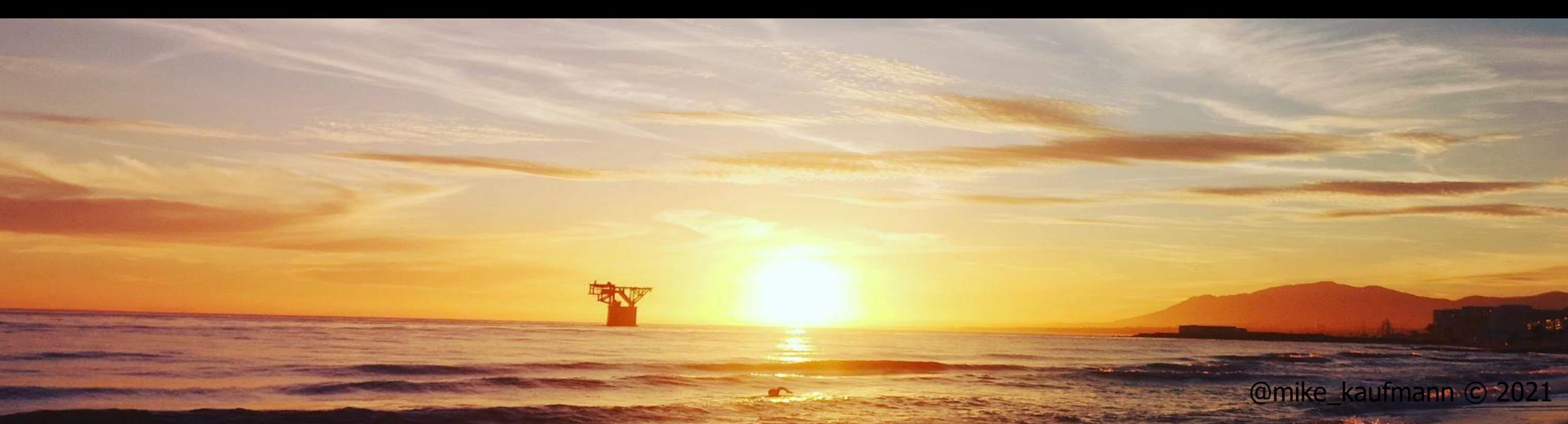
DevOps Consultant & Trainer

 @mike_kaufmann
 @wulfland
<https://writeabout.net>

CEO Xpirit Germany GmbH
20 años Desarrollador / Informatico
15 años ALM & DevOps
Microsoft Regional Director & MVP



Git für Dummies, Michael Kaufmann, Wiley-VCH, 2021



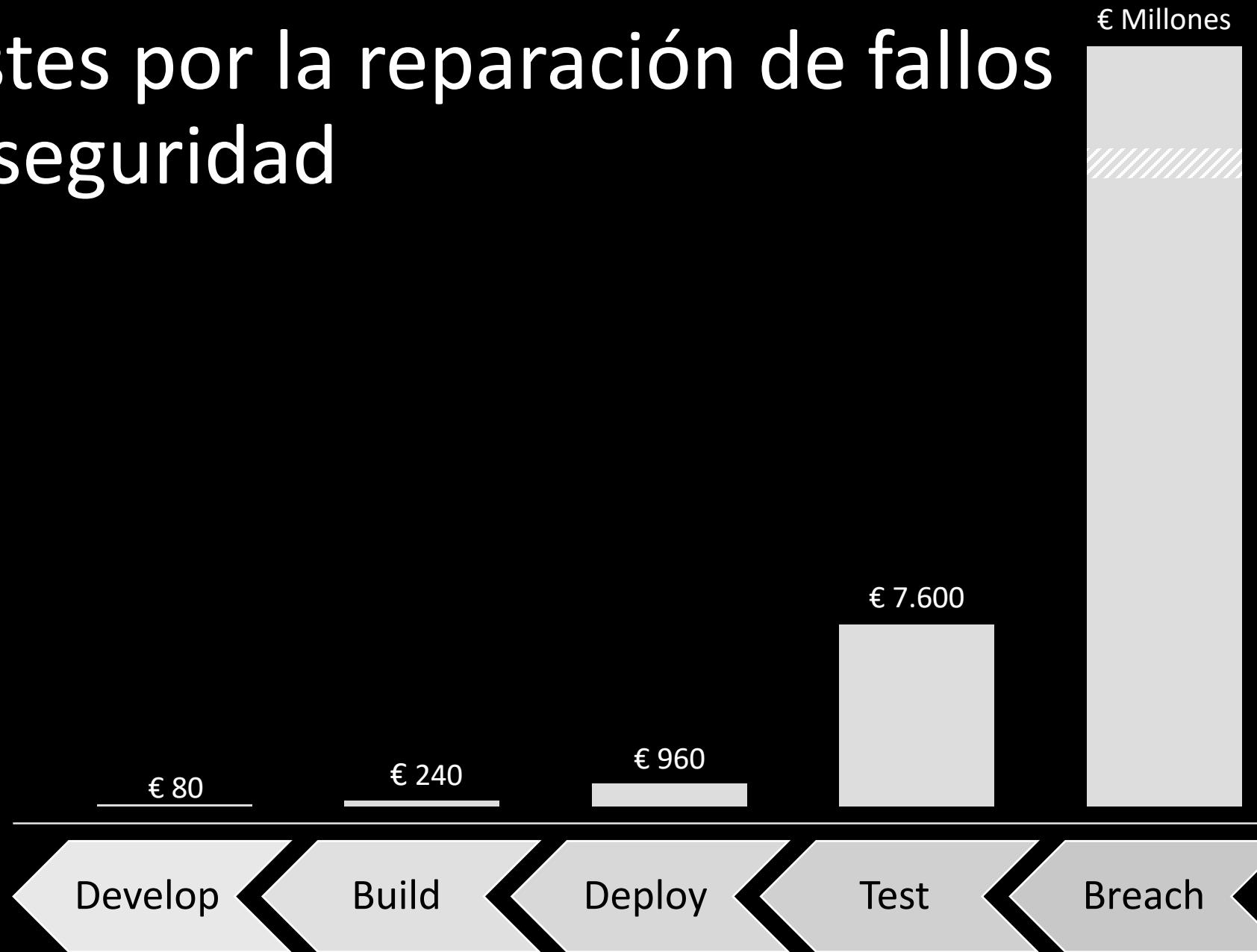
Daños causados por los ataques ciberneticos denunciados al IC3:



SolarWinds

March to June 2020: access to government and other systems through a compromised update to SolarWinds' **Orion** software

Costes por la reparación de fallos de seguridad



Assume-Breach-Paradigma

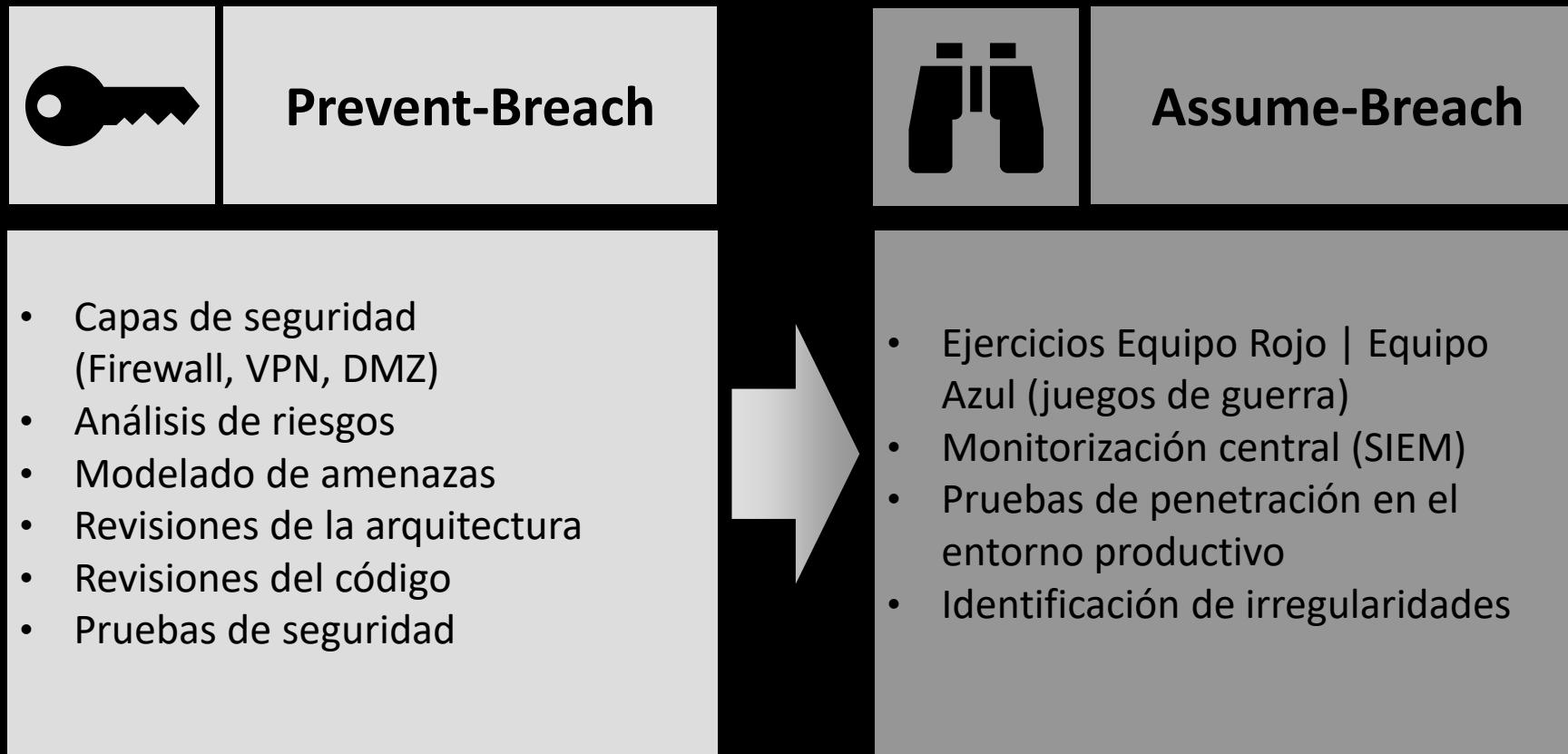
“Básicamente es así, si alguien quiere entrar, entra. Está bien. Acéptenlo.”

- Michael V. Hayden

Exgeneral de la Fuerza Aérea de EE.UU. y exdirector de la NSA y de la CIA)



De la prevención de fallos (Prevent-Breach) a la asunción de fallos (Assume-Breach)



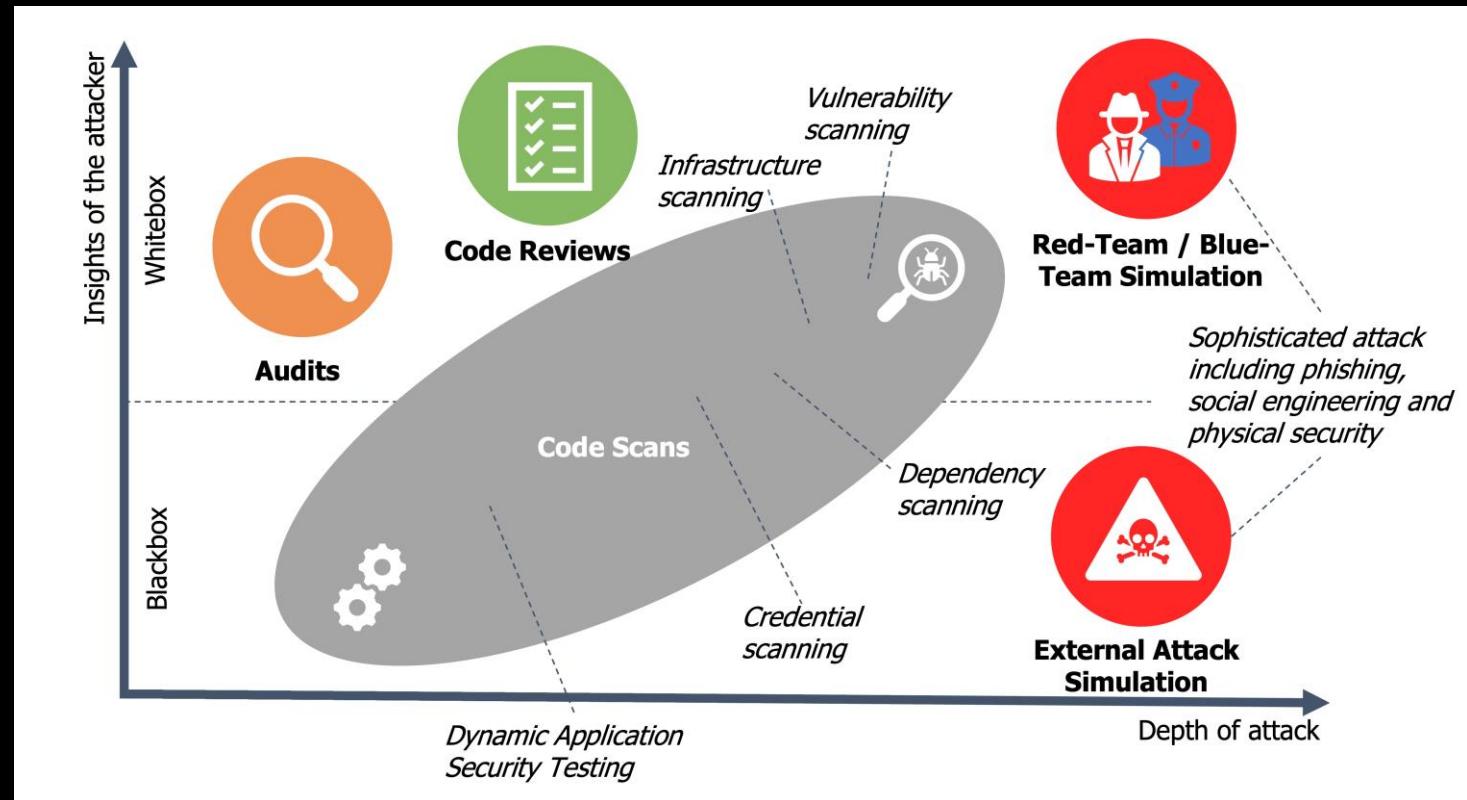
Zero-Trust- Policy

- Todos los sistemas, también los internos, deben asegurarse de tal modo como si fueran accesibles a través de Internet: MFA activado, encriptado HTTPS, sistema siempre parcheado, etc.
- Los empleados solo deben contar con los accesos absolutamente necesarios mientras sean necesarios. Deberán evitarse los derechos de administrador locales.
- Nadie debe tener acceso a los sistemas de producción con sus datos de accesos normales. Ni siquiera las cuentas de prueba que se utilizan en los entornos de desarrollo y de prueba deben tener acceso a la producción.



Equipo Rojo | Equipo Azul

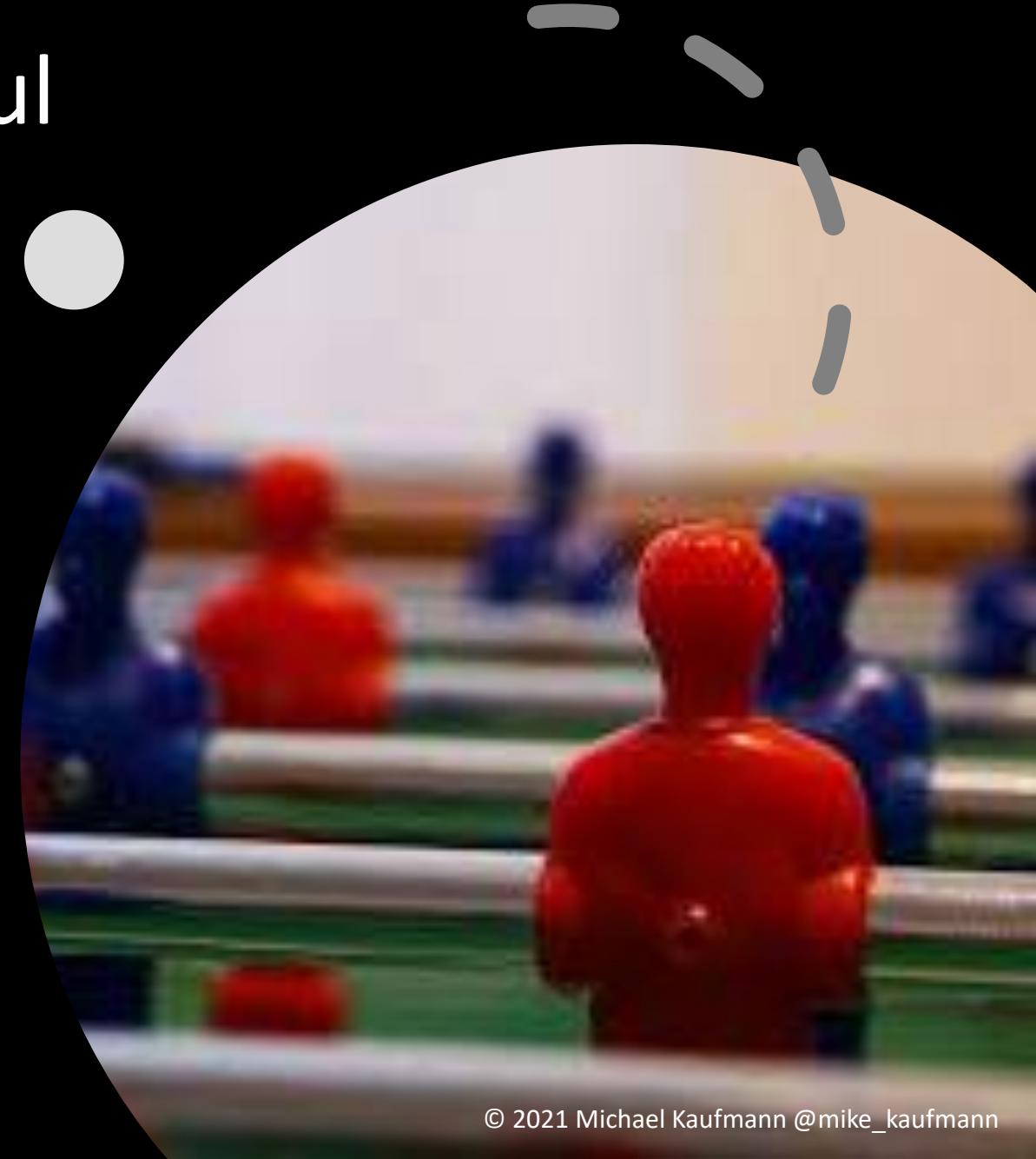
- Objetivo del juego
- Alineación del equipo
- Duración del juego



Equipo Rojo | Equipo Azul

Código de comportamiento:

- No causar ningún daño real
- No hacer más de lo necesario para alcanzar el objetivo
- Ataques físicos dentro de lo normal (no amenazar a nadie, no robar o imprimir tarjetas de acceso, etc. “Tener sentido común”)
- No exponer nombres de personas que hayan sido comprometidas

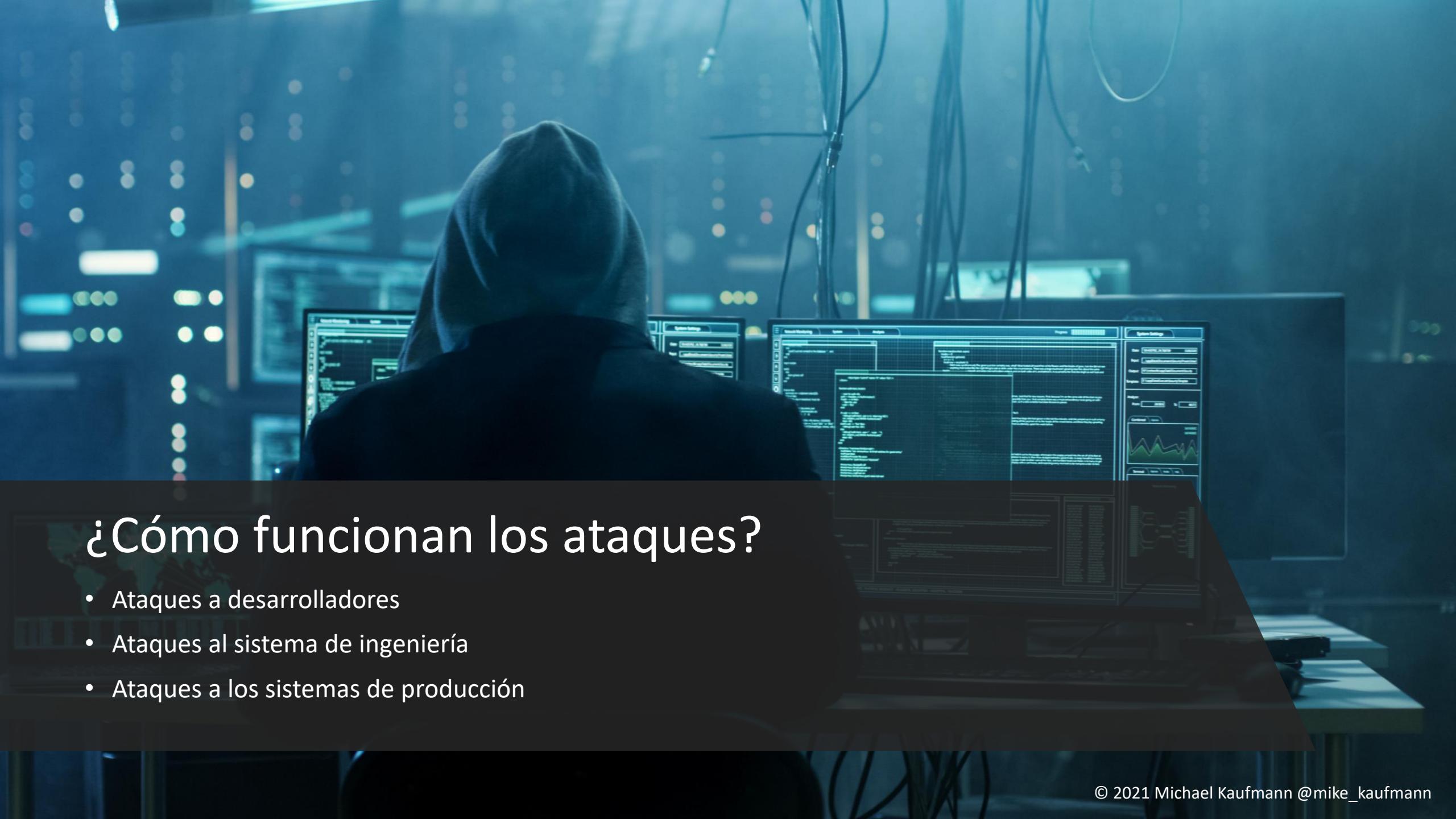


Equipo Rojo | Equipo Azul

Reglas de juego:

- No debe perjudicarse la disponibilidad del sistema de producción → el equipo rojo ha ganado
- No deben comprometerse los datos reales de los clientes (pero sí los datos de clientes internos)
- Los datos comprometidos (datos de acceso, datos personales, etc.) deben depositarse de manera segura
- La seguridad del sistema de producción no debe debilitarse excesivamente
- No deben ser ocasionados daños innecesarios
- ¡Prueba!





¿Cómo funcionan los ataques?

- Ataques a desarrolladores
- Ataques al sistema de ingeniería
- Ataques a los sistemas de producción

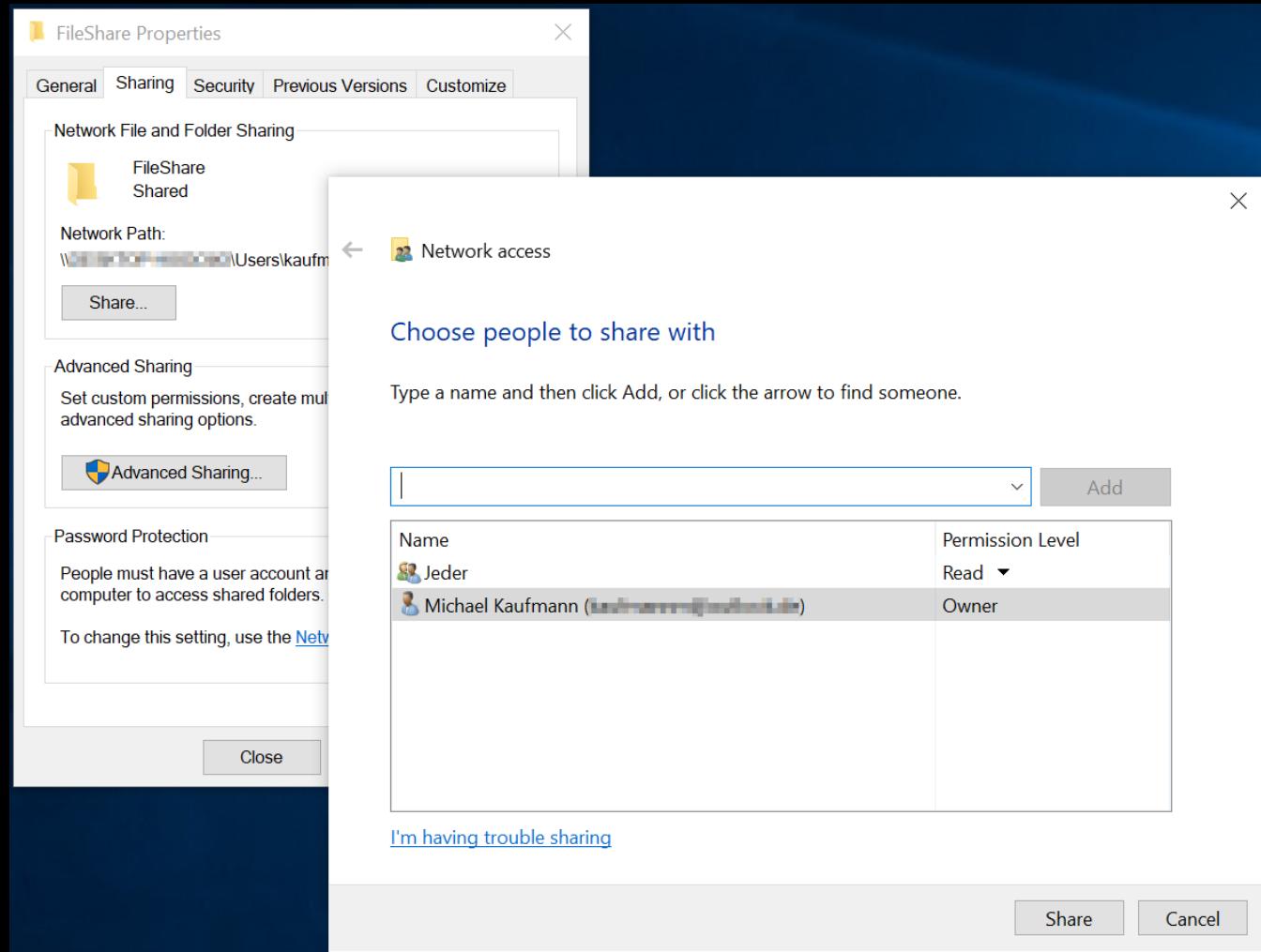
Ataques a desarrolladores

- Intercambios de archivos no protegidos
- Datos de acceso en archivos de texto o en el código
- Mimikatz
- Phishing

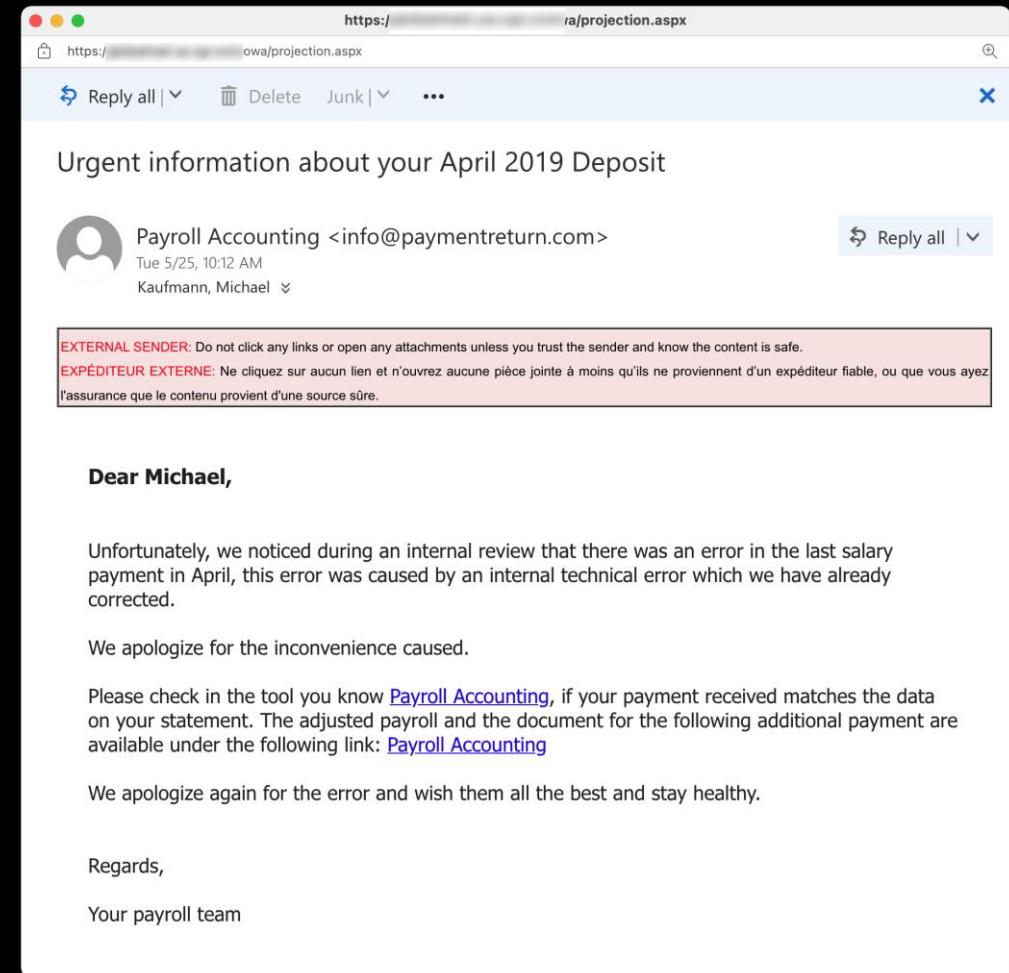
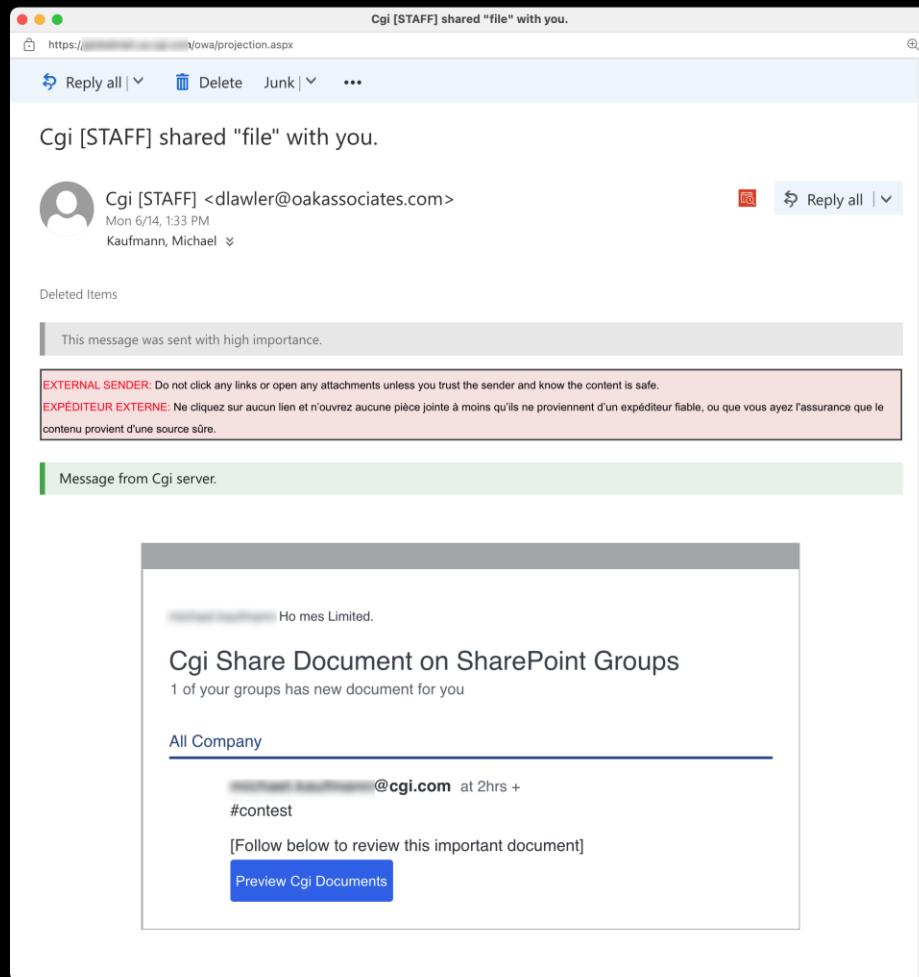
Ataques al sistema de ingeniería

- Entornos de desarrollo
- Entornos de prueba / cuentas de prueba
- Ataques a las cadenas de suministro de software
- Cambios de código (basta con una línea)
- Acceso a la pipeline (ejecución de código/scripts)

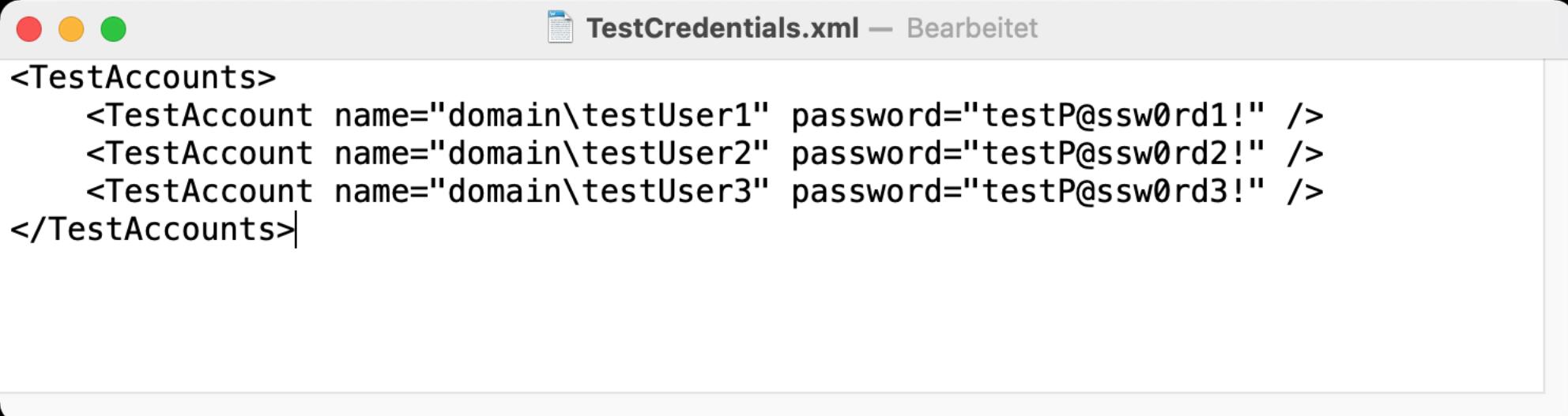
Intercambios de archivos no protegidos



Phishing



Datos de acceso en archivos de texto



A screenshot of a text editor window titled "TestCredentials.xml — Bearbeitet". The window contains the following XML code:

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

kaufm@DESKTOP-HI0G080 ~\OneDrive\Downloads\mimikatz_trunk\x64
.\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        : 77046479-3B72-43E3-8700-E0E189B60C86
Domain           : NT VIRTUAL MACHINE
Logon Server     : (null)
Logon Time       : 22.06.2021 08:26:38
SID              : S-1-5-83-1-3810489593-1138965362-3789618423-3260611593
```

```
msv :
tspkg :
wdigest :
* Username : DESKTOP-HI0G080$  
* Domain   : WORKGROUP
* Password  : (null)
kerberos :
ssp :
```

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

```
ssp :
```

```
credman :
```

```
[00000000]
```

```
* Username : mkgwriteteabout.net
* Domain   : outlook.office365.com
```

```
* Password : 0000-d0057000-4120-07B0-0000
```

```
[00000001]
```

```
* Username : (null)
```

```
* Domain   : MicrosoftOffice16 Data:SSPI:mkgwriteteabout.net
```

```
* Password : 0000-d0057000-4120-07B0-0000
```

```
[00000002]
```

```
* Username : github.auth
```

```
* Domain   : vscodevscode.github-authentication/github.auth
```

```
* Password : 0000-d0057000-4120-07B0-0000 0000-d0057000-4120-07B0-0000
```

```
64 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 30 2f 8d 4a 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 30 34 34 75 4c 6e 4c 52 42 2f 38 32 43 51 8d 6a 45 41 43 61 2f 34 5a 4e 66 2b 47 54 6b 53 56 4d 69 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 61 36 35 59 74 6d
```

```
55 a4 a4 a4 54 75 64 4e 66 47 64 54 42 45 41 45 61 2f 34 5a 4e 66 2b 47 54 6b 53 56 4d 65 75 52 79 49 74 4e
```

```
64 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 30 2f 8d 4a 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 30 34 34 75 4c 6e 4c 52 42 2f 38 32 43 51 8d 6a 45 41 43 61 2f 34 5a 4e 66 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 61 36 35 59 74 6d
```

```
72 8d 8a 53 38 2b 72 53 6e 69 47 46 34 42 55 33 48 6a 78 4f 5a 73 43 47 6a 6e 46 63 37 6d 73 46 6f 4e 78 64 78 6a 37 59
```

```
64 43 2b 47 48 5a 4a 37 68 79 2b 78 48 6f 46 48 7a 32 51 76 6d 2f 78 5a 6a 36 75 8d 8a 38 55 49 6c 47 38 52 2b 71 37 47
```

```
4d 55 34 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 34 47 4f 53 44 41 6e 35
```

```
5a 53 6d 2b 41 4c 51 37 38 6b 37 53 6a 8d 8a 84 5a 40 34 74 6c 45 6e 73 31 77 68 63 35 59 58 80 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 4a 00 00

[00000003]

* Username : wulfland

* Domain : GitHub - https://api.github.com/wulfland

* Password : 096d805700410070000

[00000004]

* Username : wulfland@hotmail.com

* Domain : https://gitlab.com

* Password : 096d805700410070000

cloudap :

Cachedir : d470e18c8f14149c4504f151e1041892c5fe9c1a50cc2b0ea1fdaf444010495d

Key GUID : {00000000-5f1c-30a9-0000-000000000000}

PRT : []

DPAPI Key: 540039007500610039006a006d0057004100700034000b00300034005100070009005000310003006100790051006000

5a0001000c00003007B0000005800059005590078eff4292dd33423ca0146db9f101ec09a1b69c4faaf (sha1: 1457570c0ea9fc74668051f3e1185acc
c4411575)

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-1-5-20-0-1

msv :

tspkg :

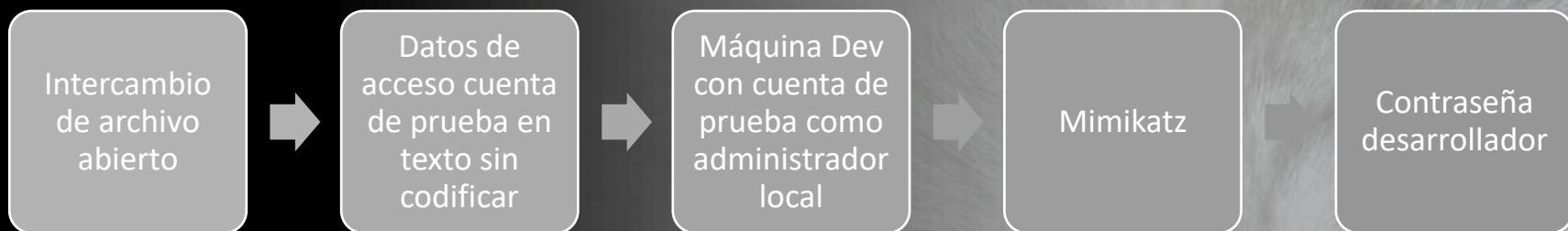
wdigest :

* Username : DESKTOP-HJ8G09S

* Domain : WORKGROUP

* Password : (null)

Ejemplo



Datos de acceso del desarrollador

- E-Mail → Phishing de compañeros
- Acceso a máquinas → Logon, Mimikatz
- Código fuente → Insertar/modificar código
- Acceso Pipeline → Ejecutar scripts
- Acceso entorno de prueba → Prueba Prod?
- Acceso Prod → Game Over



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

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

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

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

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</pre>
```

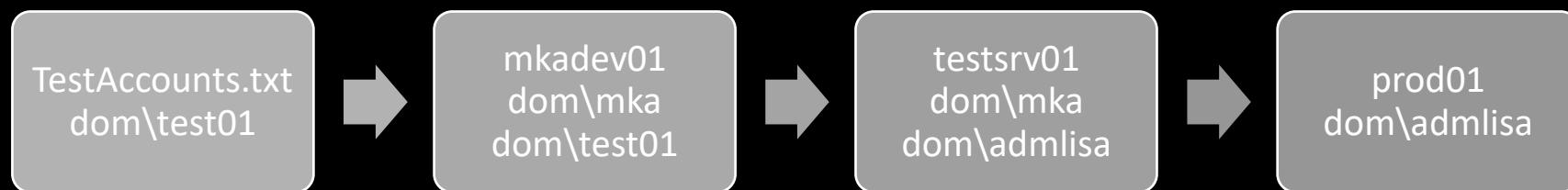
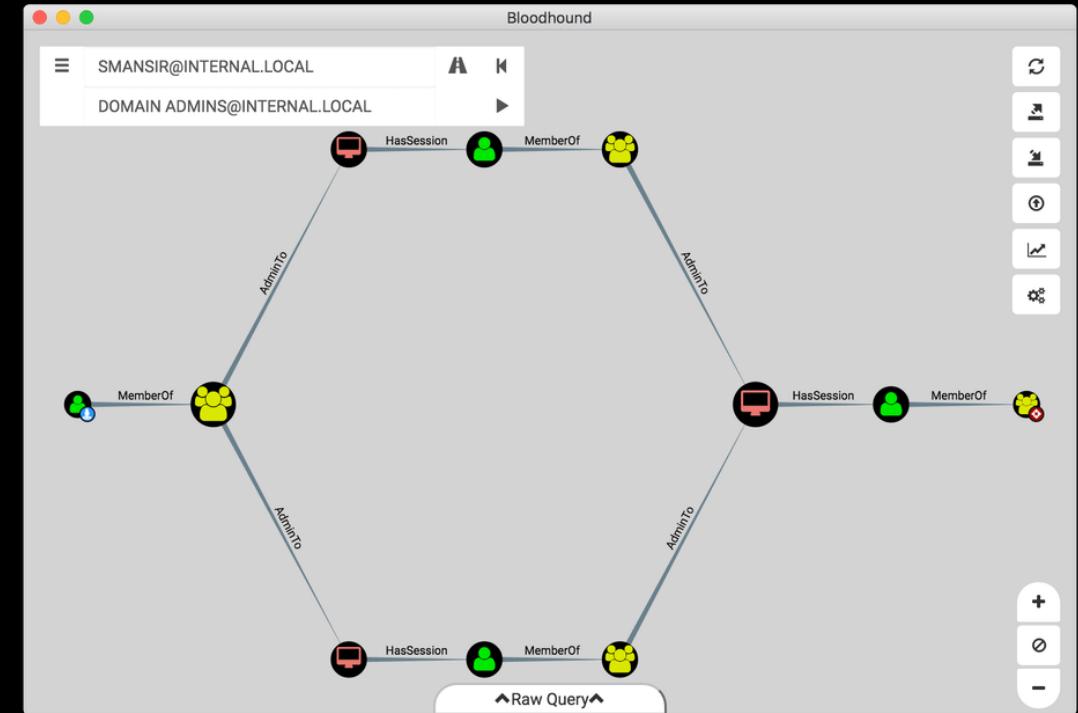
De Dev a Prod

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

grpmgr test01 Administrators /enum

Otras posibilidades de conseguir datos de acceso:

- Phishing
- Responder (<https://github.com/Igandx/Responder>)
- Pineapple
- ...
- Contraseñas no seguras



Ataques a los sistemas de producción

- SQL-Injection
- XSS (Cross-Site-Scripting)



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                             }
32                         }
33                     })
34                 }
35             })
36         }
37     }
38 }
```

XSS (Cross-Site-Scripting)

```
143 // vuln-code-snippet start localXssChallenge xssBonusChallenge
144 filterTable () {
145   let queryParam: string = this.route.snapshot.queryParams.q
146   if (queryParam) {
147     queryParam = queryParam.trim()
148     this.ngZone.runOutsideAngular(() => { // vuln-code-snippet hide-start
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)
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()
}
```

¿Qué se puede hacer (Equipo Azul)?

- MFA para todos los sistemas
- Retirar todos los roles de administrador local
- Estricta separación Test/Prod
- Key-Vault para todos los Secrets
 - Separación tiempo de ejecución/despliegue
 - Test/Prod
 - Rotación automática
- Suscripciones separadas
- Codespaces / VDI para Development
- SIEM (Azure Security Center, Splunk)
- Credential Scanning (Code/Fileshares)



How to shift left security?



Secret Scanning

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

Code

- GitHub (private!)
- gitLeaks.
- SpectralOps
- Git-Secrets
- Whispers
- Gittyleaks
- Git-all-secrets
- ...

Fileshare

- Bash/PowerShell
- Dumpster

Software Composition Analysis (SCA)

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

Dependabot alerts

Dismiss all ▾

⚠ 4 Open ✓ 0 Closed

Manifest ▾ Sort ▾

Severity	Count
critical severity	2
high severity	1
moderate severity	1
critical severity	1

- ⚠ **marsdb**
3 minutes ago by GitHub [package.json](#)
- ⚠ **express-jwt**
3 minutes ago by GitHub [package.json](#)
- ⚠ **sanitize-html**
3 minutes ago by GitHub [package.json](#)
- ⚠ **jsonwebtoken**
3 minutes ago by GitHub [package.json](#)

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](#).

Dependency	Vulnerability	Version
auth0 / express-jwt	Known security vulnerability	0.1.3
auth0 / node-jsonwebtoken	Known security vulnerability	0.4.0
c58 / marsdb	Known security vulnerability	^ 0.6.11
apostrophecms / sanitize-html	Known security vulnerability	1.4.2
istanbuljs / istanbuljs	@istanbuljs/nyc-config-typescript	^ 1.0.1
Seally / types-chai	@types/chai	^ 4.2.14
DefinitelyTyped / DefinitelyTyped	@types/chai-as-promised	^ 7.1.3

Software Composition Analysis (SCA)

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

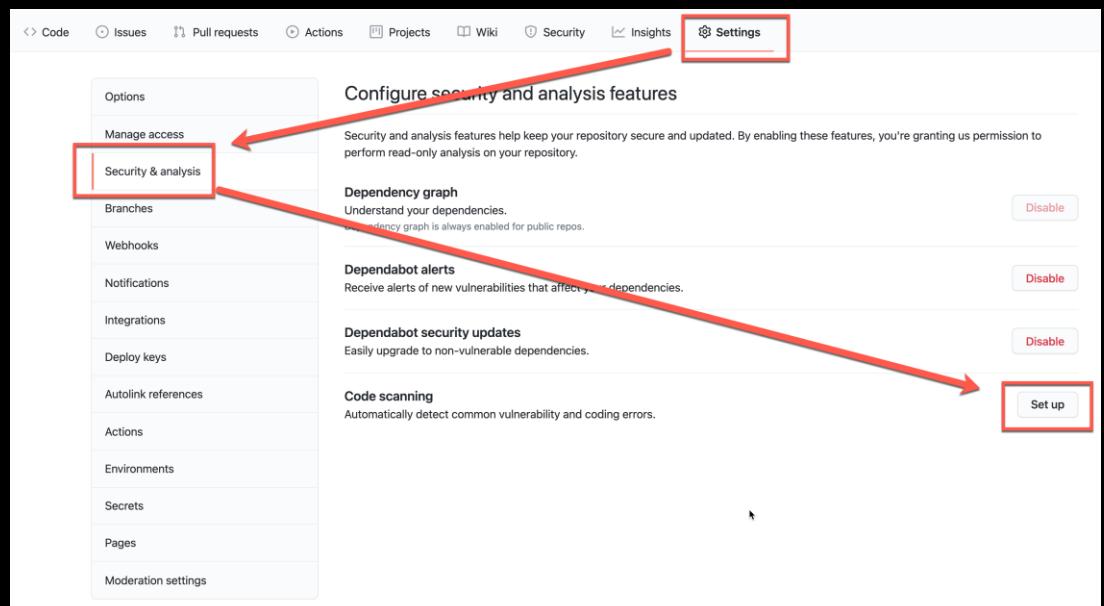
The screenshot shows a GitHub security alert digest for the week of Jun 23 - Jun 30. It highlights a known security vulnerability for the dependency 'bootstrap'. The current version is 3.0.0, and it is recommended to upgrade to 3.4.1. The vulnerability is identified as CVE-2019-8331 with a moderate severity. A blue button at the bottom right encourages users to 'Review all vulnerable dependencies'.

The screenshot shows a GitHub pull request titled 'Bump bootstrap from 3.0.0 to 3.4.1 in /src/CupCakeHeaven/CupCakeHeaven #1'. The pull request is from the dependabot bot and is intended to fix a security vulnerability. The message states: 'This automated pull request fixes a security vulnerability'. The PR has 0 commits, 0 checks, and 1 file changed. The status bar indicates 'moderate severity'. The pull request details show the command used: 'Bump bootstrap from 3.0.0 to 3.4.1'. The PR is labeled with 'dependencies'. The review section notes: 'This branch has no conflicts with the base branch'. There is a green 'Merge pull request' button. The right sidebar provides project and repository metadata.

Static Application Security Testing (SAST)

Whitebox-Testing

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



A screenshot of the GitHub Marketplace page for code scanning. The top navigation bar includes 'Overview', 'Security policy', 'Security advisories', 'Dependabot alerts', and 'Code scanning alerts'. The main content area is titled 'Get started with code scanning' and 'Automatically detect common vulnerabilities and coding errors'. It features a grid of cards for different tools:

- CodeQL Analysis** by GitHub: Security analysis from GitHub for C, C++, C#, Java, JavaScript, TypeScript, Python, and Go developers. Includes a 'Set up this workflow' button.
- 42Crunch API Security Audit** by 42crunch: Use the 42Crunch API Security Audit REST API to perform static application security testing (SAST) on OpenAPI/Swagger files. Includes a 'Set up this workflow' button.
- Codacy Security Scan** by Codacy: Free, out-of-the-box, security analysis provided by multiple open source static analysis tools. Includes a 'Set up this workflow' button.
- CxSAST** by Checkmarx: Scan your code with Checkmarx CxSAST and see your results in the GitHub security tab. Includes a 'Set up this workflow' button.
- DefenseCode ThunderScan** by DefenseCode: Scan your code with ThunderScan® SAST to detect security vulnerabilities in more than 30 programming languages. Includes a 'View in marketplace' link.
- DevSkim** by Microsoft CST-E: DevSkim is a security linter that highlights common security issues in source code. Includes a 'Set up this workflow' button.
- Fortify on Demand Scan** by Micro Focus: Integrate Fortify's comprehensive static code analysis(SAST) for 27+ languages into your DevSecOps workflows to build secure software faster. Includes a 'Set up this workflow' button.
- Kubescan** by Controlplane: Security risk analysis for Kubernetes resources. Submit pod-types (such as deployment, cronjob) to receive an itemised security risk score. Includes a 'Set up this workflow' button.
- Mayhem for API** by ForAllSecure: Automatically test your REST APIs with your OpenAPI specs and Postman collections. Includes a 'Set up this workflow' button.

Code Scanning (CodeQL)

Fork: <https://github.com/advanced-security/workshop-2021-learning-journey>

```
name: "CodeQL"

on:
  push:
    branches: [ master ]
  pull_request:
    # The branches below must be a subset of the branches above
    branches: [ master ]

  jobs:
    analyze:
      name: Analyze
      runs-on: ubuntu-latest
      permissions:
        actions: read
        contents: read
        security-events: write

    strategy:
      fail-fast: false
      matrix:
        language: [ 'javascript', 'java' ]
        # CodeQL supports [ 'cpp', 'csharp', 'go', 'java', 'javascript', 'python' ]

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      # Initializes the CodeQL tools for scanning.
      - name: Initialize CodeQL
        uses: github/codeql-action/init@v1
        with:
          languages: ${matrix.language}
          # If you wish to specify custom queries, you can do so here or in a config file.
          # By default, queries listed here will override any specified in a config file.
          # Prefix the list here with "+" to use these queries and those in the config file.
          # queries: ./path/to/local/query, your-org/your-repo/queries@main

      # Autobuild attempts to build any compiled languages (C/C++, C#, or Java).
      # If this step fails, then you should remove it and run the build manually (see below)
      - name: Autobuild
        uses: github/codeql-action/autobuild@v1
```

The screenshot shows the GitHub Security tab with the 'Code scanning' section selected. It displays a summary of the latest scan (3 days ago, main branch, CodeQL tool, 50.5k / 49.7k lines scanned, 3m 33s duration, 50 alerts) and a list of open alerts categorized by severity (main). The alerts include:

- Client-side cross-site scripting (frontend/src/app/search-result/search-result.component.ts#L152)
- Missing CSRF middleware (server.ts#L236)
- Uncontrolled data used in path expression (routes/quarantineServer.ts#L13, routes/profileImageUrlUpload.ts#L28, routes/logfileServer.ts#L13, routes/keyServer.ts#L13, routes/fileUpload.ts#L43, routes/fileUpload.ts#L38, routes/fileServer.ts#L30)
- Arbitrary file write during zip extraction ("Zip Slip") (routes/fileUpload.ts#L50)

Code Scanning (CodeQL)

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
156       }
157     })
158   }
159 }
```

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

-o Create codeql-analysis.yml

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

Verified ✅ 9474e68

Client-side cross-site scripting

7 steps in search-result.component.ts ▾ X

Step 1 this.ro ... yParams

frontend/src/app/search-result/search-result.component.ts

```
142
143 // vuln-code-snippet start localXssChallenge xssBonusChallenge
144 filterTable () {
145   let queryParam: string = this.route.snapshot.queryParams.q
146   if (queryParam) {
147     queryParam = queryParam.trim()
148     this.ngZone.runOutsideAngular(() => { // vuln-code-snippet hide-start
```

Step 2 this.ro ... arams.q

frontend/src/app/search-result/search-result.component.ts

```
142
143 // vuln-code-snippet start localXssChallenge xssBonusChallenge
144 filterTable () {
145   let queryParam: string = this.route.snapshot.queryParams.q
146   if (queryParam) {
147     queryParam = queryParam.trim()
148     this.ngZone.runOutsideAngular(() => { // vuln-code-snippet hide-start
```

Step 3 queryParam

frontend/src/app/search-result/search-result.component.ts

```
142
143 // vuln-code-snippet start localXssChallenge xssBonusChallenge
144 filterTable () {
145   let queryParam: string = this.route.snapshot.queryParams.q
146   if (queryParam) {
147     queryParam = queryParam.trim()
148     this.ngZone.runOutsideAngular(() => { // vuln-code-snippet hide-start
```

Step 4 queryParam

frontend/src/app/search-result/search-result.component.ts

```
144 filterTable () {
145   let queryParam: string = this.route.snapshot.queryParams.q
146   if (queryParam) {
147     queryParam = queryParam.trim()
148     this.ngZone.runOutsideAngular(() => { // vuln-code-snippet hide-start
149       this.io.socket().emit('verifyLocalXssChallenge', queryParam)
150     }) // vuln-code-snippet hide-end
```

Step 5 queryParam.trim()

frontend/src/app/search-result/search-result.component.ts

Code Scanning (CodeQL)

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
156       }
157     })
158   }
159 }
```

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

-o Create codeql-analysis.yml

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

Verified ✅ 9474e68

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 ^

Code Scanning (CodeQL)

The image shows two screenshots of a code scanning tool interface. On the left, a pull request detail page displays a vulnerability titled "Client-side cross-site scripting". It shows the code snippet:

```
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(queryParam) // vuln-code-snippet vuln-line localXssChallenge
```

A red arrow points from the "View source" link in the tool table to the right-hand side panel, which displays the generated CodeQL query:

```
1  /**
2  * @name Client-side cross-site scripting
3  * @description Writing user input directly to the DOM allows for
4  *             a cross-site scripting vulnerability.
5  * @kind path-problem
6  * @problem.severity error
7  * @security-severity 6.1
8  * @precision high
9  * @id js/xss
10 * @tags security
11 *      external/cwe/cwe-079
12 *      external/cwe/cwe-116
13 */
14
15 import javascript
16 import semmle.javascript.security.dataflow.DomBasedXss::DomBasedXss
17 import DataFlow::PathGraph
18
19 from DataFlow::Configuration cfg, DataFlow::PathNode source, DataFlow::PathNode sink
20 where
21   (
22     cfg instanceof HtmlInjectionConfiguration or
23     cfg instanceof JQueryHtmlOrSelectorInjectionConfiguration
24   ) and
25     cfg.hasFlowPath(source, sink)
26 select sink.getNode(), source, sink,
27       sink.getNode().(Sink).getVulnerabilityKind() + " vulnerability due to $@.", source.getNode(),
28       "user-provided value"
```

Code Scanning (CodeQL)

Database query built from user-controlled sources

Building a database query from user-controlled sources is vulnerable to insertion of malicious code by the user.

[Open](#) [Error](#) [CWE-89](#) [security](#)

Branch: main ▾

Dismiss ▾

routes/search.ts

```
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 ...')
16   .then(([products]) => {
17     const dataString = JSON.stringify(products)
18     if (utils.notSolved(challenges.unionSqlInjectionChallenge)) { // vuln-code-snippet hide-start
```

This query depends on a user-provided value.

CodeQL [Show paths](#)

Tool	Rule ID	Query
CodeQL	js/sql-injection	View source

If a database query (such as a SQL or NoSQL query) is built from user-provided data without sufficient sanitization, a malicious user may be able to run malicious database queries.

Show more ▾

First detected in commit 9474e68 3 days ago

27 lines (25 sloc) | 883 Bytes

```
1 /**
2  * @name Database query built from user-controlled sources
3  * @description Building a database query from user-controlled sources is vulnerable to insertion of
4  *             malicious code by the user.
5  * @kind path-problem
6  * @problem.severity error
7  * @security-severity 8.8
8  * @precision high
9  * @id js/sql-injection
10 * @tags security
11 *         external/cwe/cwe-089
12 */
13
14 import javascript
15 import semmle.javascript.security.dataflow.SqlInjection
16 import semmle.javascript.security.dataflow.NosqlInjection
17 import DataFlow::PathGraph
18
19 from DataFlow::Configuration cfg, DataFlow::PathNode source, DataFlow::PathNode sink
20 where
21   (
22     cfg instanceof SqlInjection::Configuration or
23     cfg instanceof NosqlInjection::Configuration
24   ) and
25     cfg.hasFlowPath(source, sink)
26   select sink.getNode(), source, sink, "This query depends on @{$.}, source.getNode(),
27   "a user-provided value"
```

CodeQL: Pull-Request Integration

Update server.ts #1

Open wulfand wants to merge 1 commit into main from wulfand-patch-search

Conversation 0 Commits 1 Checks 3 Files changed 1

+10 -0

Changes from all commits ▾ File filter ▾ Conversations ▾ Jump to ▾ Dismiss ▾

server.ts

```
@@ -371,6 +371,16 @@
371   371     app.get('/api/Deliveries/:id', delivery.getDeliveryMethod())
372   372     // vuln-code-snippet end changeProductChallenge
373   373
374 + 374     + app.get('/api/search', function (req, res, context) {
375 + 375       var search_name = req.query.search;
376 + 376       if (search_name !== null) {
377 + 377         models.sequelize.query(`SELECT * FROM Products WHERE name LIKE '%${search_name}%' ORDER BY name`)
378 + 378       .then(([ products ]) => {
379 + 379         res.json(utils.queryResultToJson(products))
380 + 380       })
381 + 381     })
382 + 382   }
383 + 383 });

374 384   /* Verify the 2FA Token */
375 385   app.post('/rest/2fa/verify',
376 386     new RateLimit({ windowMs: 5 * 60 * 1000, max: 100 }),
```

Check failure on line 374 in server.ts

Code scanning

Missing rate limiting

This route handler performs a database access, but is not rate-limited.

Show more details

Dismiss ▾

375 + var search_name = req.query.search;
376 +
377 + if (search_name !== null) {
378 + models.sequelize.query(`SELECT * FROM Products WHERE name LIKE '%\${search_name}%' ORDER BY name`)

Check failure on line 378 in server.ts

Code scanning

Database query built from user-controlled sources

This query depends on a user-provided value.

Show more details

Show paths

Dismiss ▾

379 + .then(([products]) => {
380 + res.json(utils.queryResultToJson(products))
381 + })
382 + }
383 + });

374 384 /* Verify the 2FA Token */
375 385 app.post('/rest/2fa/verify',
376 386 new RateLimit({ windowMs: 5 * 60 * 1000, max: 100 }),

Database query built from user-controlled sources

Building a database query from user-controlled sources is vulnerable to insertion of malicious code by the user.

Open Error CWE-89 security

Branch: refs/pull/1/merge

Dismiss ▾

server.ts

```
375 var search_name = req.query.search;
376
377 if (search_name !== null) {
378   models.sequelize.query(`SELECT * FROM Products WHERE name LIKE '%${search_name}%' ORDER BY name`)
```

This query depends on a user-provided value.

CodeQL Show paths

379 .then(([products]) => {
380 res.json(utils.queryResultToJson(products))
381 })

Tool Rule ID Query

CodeQL js/sql-injection View source

If a database query (such as a SQL or NoSQL query) is built from user-provided data without sufficient sanitization, a malicious user may be able to run malicious database queries.

Show more ▾

First detected in commit 95fe43a 44 seconds ago

Merge 906306f into d45d3f0

server.ts#L378 on branch refs/pull/1/merge

Verified 95fe43a

CodeQL: Config

```
# Initializes the CodeQL tools for scanning.
- name: Initialize CodeQL
  uses: github/codeql-action/init@v1
  with:
    languages: ${{ matrix.language }}
    # If you wish to specify custom queries, you can do so here or in a config file.
    # By default, queries listed here will override any specified in a config file.
    # Prefix the list here with "+" to use these queries and those in the config file.
    # queries: ./path/to/local/query, your-org/your-repo/queries@main
    queries: security-extended
```

CodeQL: Config

```
# Initializes the CodeQL tools for scanning.  
- name: Initialize CodeQL  
  uses: github/codeql-action/init@v1  
  with:  
    languages: ${{ matrix.language }}  
  # If you wish to specify custom queries, you can do so here or in a config file.  
  # By default, queries listed here will override any specified in a config file.  
  # Prefix the list here with "+" to use these queries and those in the config file.  
  # queries: ./path/to/local/query, your-org/your-repo/queries@main  
  # queries: security-extended  
  config-file: ./github/codeql-config.yml
```

[Raw](#)

```
codeql-config.yml
```

```
1  name: "Custom CodeQL Configuration"  
2  
3  # Disabling the default queries is critical part of this.  
4  # Nothing will be disabled if the default queries still run.  
5  # - https://docs.github.com/en/code-security/secure-coding/configuring-code-scanning#disabling-the-default-queries  
6  disable-default-queries: true  
7  
8  queries:  
9    # Point to the custom query suite that needs to be used  
10   - uses: ./github/codeql/custom-javascript.qls  
11  
12 # Ignore some of the paths as these are some open source components.  
13 # This only works for Interpreted Languages (JavaScript / TypeScript / Python)  
14 # - https://docs.github.com/en/code-security/secure-coding/automatically-scanning-your-code-for-vulnerabilities-and-errors/conf  
15 paths-ignore:  
16   - '**/node_modules'  
17   # Ignoring our test folders  
18   - '**/test'
```

CodeQL: Config

```
custom-javascript.qls
Raw

1 # Documentation
2 # - https://codeql.github.com/docs/codeql-cli/creating-codeql-query-suites/
3 # - https://github.com/github/codeql/actions/workflows/query-list.yml
4 # - https://github.com/github/codeql/blob/main/misc/suite-helpers/security-extended-selectors.yml
5 # - https://codeql.github.com/docs/codeql-cli/creating-codeql-query-suites/#examples
6
7 - description: "Custom JavaScript Suite"
8
9 # Import/Extend from an existing CodeQL suite
10 - import: codeql-suites/javascript-security-extended.qls
11   from: codeql-javascript
12
13
14 # Exclude queries that are not relevant for my organisation or have a
15 # higher False Positive rate than we are confident with.
16 - exclude:
17   id:
18     # https://github.com/github/codeql/blob/main/javascript/ql/src/Security/CWE-770/MissingRateLimiting.ql
19     - js/missing-rate-limiting
```

CodeQL Queries propios

<https://github.com/wulfland/CodeQLDemo>

The screenshot shows the Visual Studio Code extension marketplace. A search bar at the top has "CodeQL" typed into it. Below the search bar, the "CodeQL" extension by GitHub is listed. It has a rating of 18.262 and 5 stars. The extension version is v1.5.1. The "Install" button is visible. The main content area displays the "CodeQL extension for Visual Studio Code" page, which includes a description of the extension's purpose, a list of features, and sections for "Quick start overview", "Quick start: Installing and configuring the extension", "Quick start: Using CodeQL", and "Quick start: Installing and configuring the extension".

The screenshot shows the VS Code interface with the "CodeQL" extension installed. The "EXPLORER" sidebar shows a file tree with several CodeQL query files (Query1.ql, Query2.ql, etc.) and a "README.md" file, with "README.md" highlighted. The "QUERIES" tab in the Explorer is also visible. In the center, a code editor window shows a query named "Query2.ql" with the following content:

```
import javascript
// find all calls to jquery $
from CallExpr dollarCall, Expr dollarArg
where
    dollarArg = dollarCall.getArgument(0) and
    dollarCall.getCalleeName() = "$"
select dollarCall, dollarArg
```

To the right of the editor, the "CodeQL Query Results" panel shows the results of the query, with 737 results found. The results table includes columns for "#", "dollarCall", and "dollarArg". Some rows from the table are:

#	dollarCall	dollarArg
1	\$('funct ... \n \n)')	function ... \n \n)
2	\$('sele ... ntrol')	'select ... ontrol'
3	\$('#myButton')	'#myButton'
4	\$(this)	this
5	\$('#myStateButton')	'#myStateButton'
6	\$this	this
7	\$('funct ...)\n\n })	function ...)\n\n }
8	\$window	window
9	\$document.body	document.body
10	'bs-do ... f="#"')	'bs-do ... f="#"')
11	\$('bs- ... debar')	'bs-docs-sidebar'
12	\$('bs-docs-nav')	'bs-docs-nav'
13	\$('bs-docs-footer')	'bs-docs-footer'
14	\$('bs-top')	'bs-top'
15	\$('#bs- ... sheet')	'bs-th ... esheet'
16	\$('bs- ... oggle')	'bs-do ... toggle'
17	\$('tooltip-demo')	'tooltip-demo'
18	\$('popover-demo')	'popover-demo'
19	\$('tooltip-test')	'tooltip-test'
20	\$('popover-test')	'popover-test'
21	\$('bs- ... pover')	'bs-docs-popover'
22	\$('#loa ... e-btn')	'#load ... e-btn'
23	\$this	this
24	\$('#exampleModal')	'#exampleModal'
25	\$event ... Target)	event.relatedTarget
26	\$this	this

At the bottom of the interface, the status bar shows "Ln 9, Col 29 Spaces: 4 UTF-8 LF QL CodeQL CLI v2.5.6".

[https://github.com/githubsatellit/eworkshops/codeql/releases/do wnload/v1.0/esbena_bootstrap-pre-27047_javascript.zip](https://github.com/githubsatellit/eworkshops/codeql/releases/download/v1.0/esbena_bootstrap-pre-27047_javascript.zip)

Dynamic Application Security Testing (DAST)

Blackbox-Testing

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

```
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
```

The screenshot shows the GitHub Action page for 'OWASP ZAP Full Scan'. At the top, there's a blue circular icon with a white lightning bolt, the text 'GitHub Action', the name 'OWASP ZAP Full Scan', a green button for 'Use latest version', and a note that it's 'v0.2.0 (Latest version)'. Below this is a section titled 'ZAP Action Full Scan' with a brief description: 'A GitHub Action for running the OWASP ZAP Full Scan to perform Dynamic Application Security Testing (DAST).'. It explains that the action runs the ZAP spider against the specified target, followed by an optional ajax spider scan and a full active scan. Alerts are reported as GitHub issues. A warning notes that the action performs attacks on the target website and should only be used on targets with permission. The 'Inputs' section includes fields for 'target' (with a description: 'Required The URL of the web application to be scanned. This can be either a publicly available web application or a locally accessible URL.'), 'docker_name' (with a description: 'Optional The name of the docker file to be executed. By default the action runs the stable version of ZAP. But you can configure the parameter to use the weekly builds.'), and 'rules_file_name' (with a description: 'Optional You can also specify a relative path to the rules file to ignore any alerts from the ZAP scan. Make sure to create the rules file inside the relevant repository. The following shows a sample rules file configuration. Make sure to checkout the repository (actions/checkout@v2) to provide the ZAP rules to the scan action.'). At the bottom, there's a code block showing two sample rules:

```
10011 IGNORE (Cookie Without Secure Flag)  
10015 IGNORE (Incomplete or No Cache-control and Pragma HTTP Header Set)
```

Infrastructure Scanning

Container Vulnerability Analysis (CVA)

Container Security Analysis (CSA)

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

Otra infraestructura:

- Azure Policy
<https://docs.microsoft.com/de-de/azure/governance/policy/>
- Checkov
<https://www.aquasec.com/products/container-security/>
- OpenVAS
<https://www.openvas.org/>

```
name: build
on:
  push:
    branches:
      - master
  pull_request:
jobs:
  build:
    name: Build
    runs-on: ubuntu-18.04
    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Run Trivy vulnerability scanner in repo mode
        uses: aquasecurity/trivy-action@master
        with:
          scan-type: 'fs'
          ignore-unfixed: true
          format: 'template'
          template: '@/contrib/sarif.tpl'
          output: 'trivy-results.sarif'
          severity: 'CRITICAL'

      - name: Upload Trivy scan results to GitHub Security tab
        uses: github/codeql-action/upload-sarif@v1
        with:
          sarif_file: 'trivy-results.sarif'
```

DefectDojo

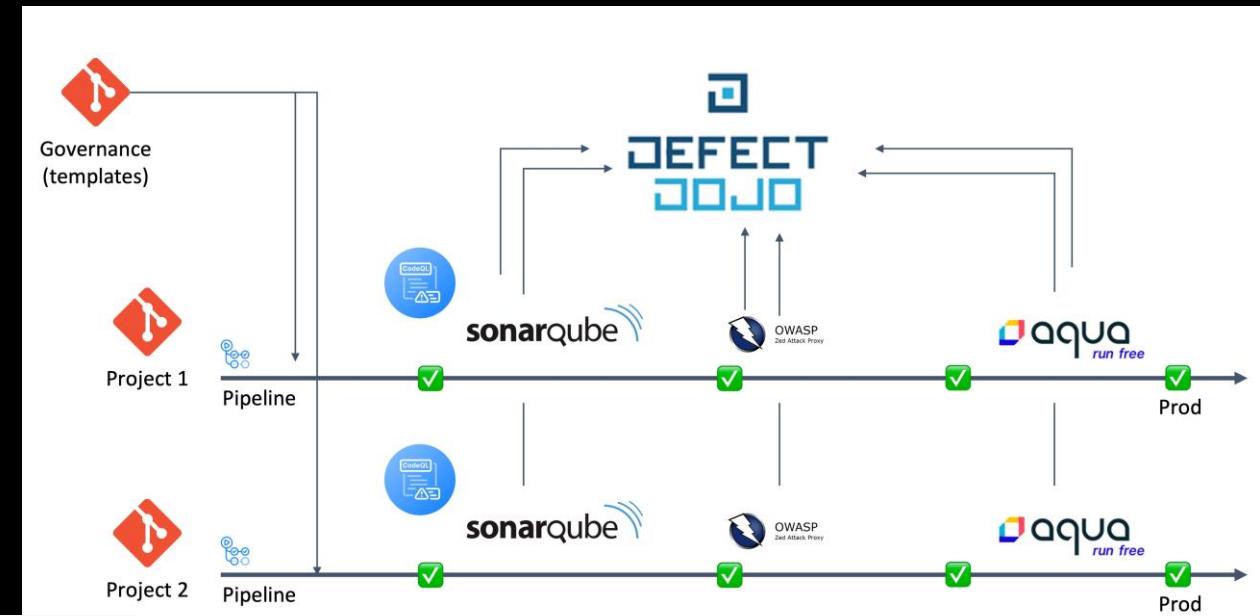
<https://github.com/DefectDojo/django-DefectDojo>

Demo: demo.defectdojo.org

Log in: admin / defectdojo@demo#appsec

The screenshot shows the DefectDojo application interface. At the top, there's a navigation bar with links for Overview, Metrics, Engagements, Findings, Endpoints, Benchmarks, and Settings. A search bar and a notification count of 227 are also present. The main content area is divided into several sections:

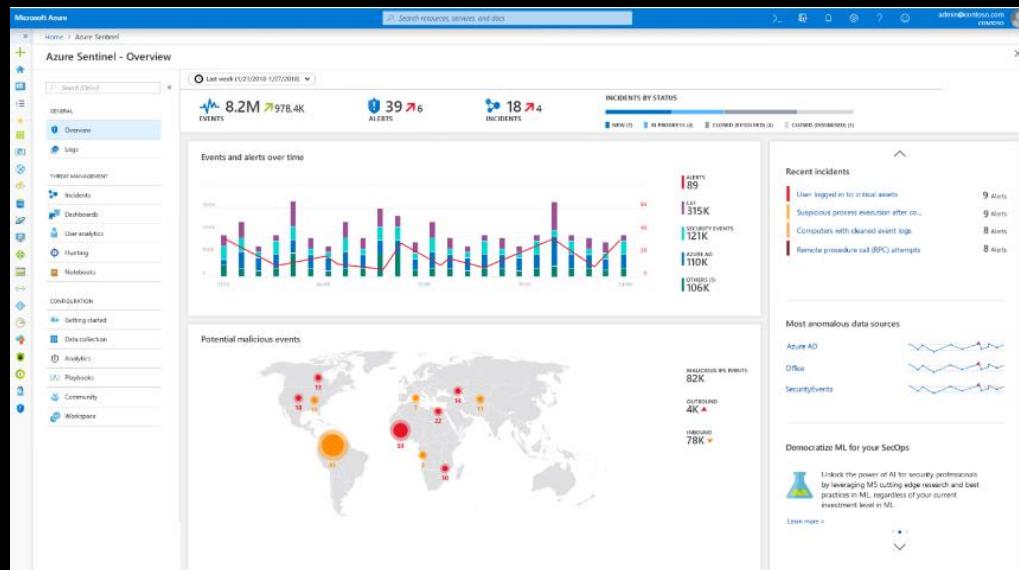
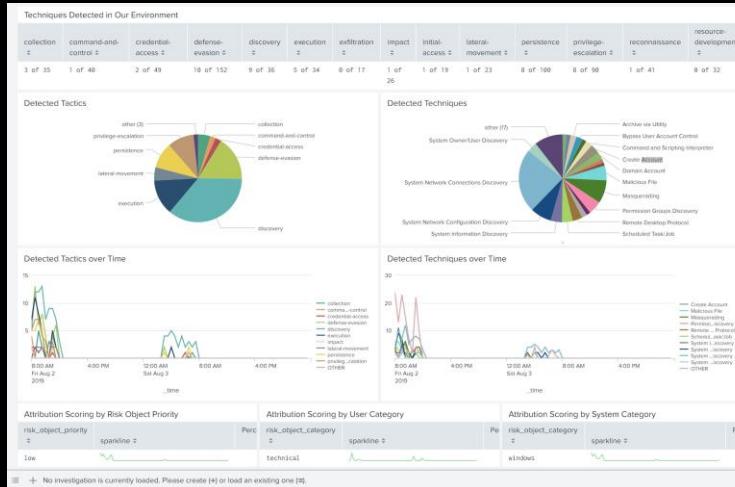
- Description:** A detailed description of the project, mentioning Bodgelit and its features.
- Metrics:** A summary of findings by severity: CRITICAL (0), HIGH (33), MEDIUM (80), LOW (85), INFORMATIONAL (6), and TOTAL (204).
- Technologies:** Apache 2.0 version v.1.
- Regulations:** GDPR EU & EU Data Extra-Territorial Applicability.
- Metadata:** Project details including Business Criticality (High), Product Type (Research and Development), Platform (Web), Lifecycle (Construction), Origin (Third Party Library), User Records (1,000), and Revenue (50,000.00).
- Languages:** A pie chart showing the distribution of code files by language: XML (red), JavaScript (green), and Java (light green). The chart indicates Total Files: 55 and Lines of Code: 10,400.



Security Information & Event Management (SIEM)

Ejemplo:

- Azure Sentinel
- Splunk
- Un log central
- Multicloud/Hybrid
- Detección de anomalías (ML)
- Avisos en tiempo real



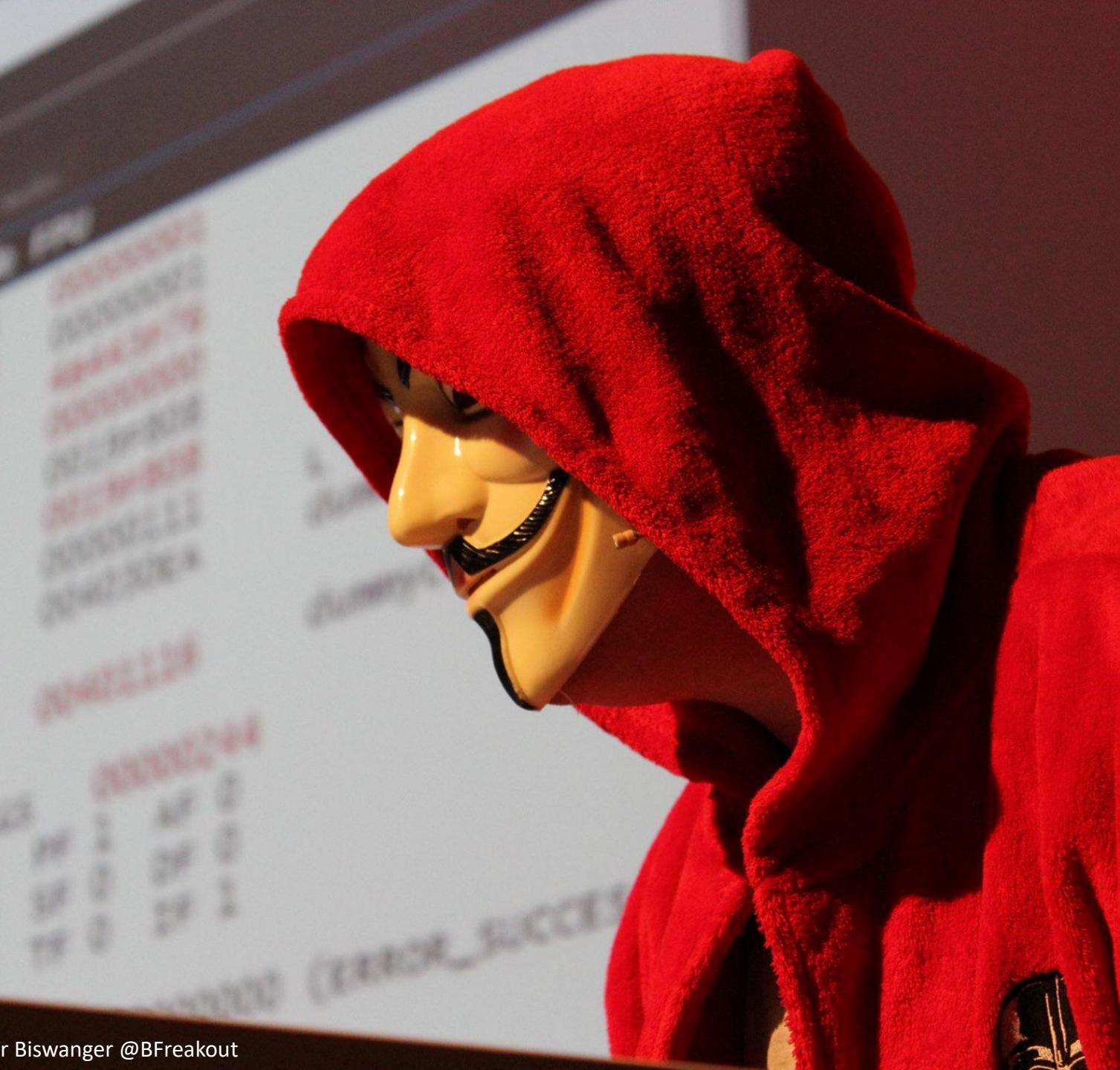
6 tips to integrate security into your DevOps practices

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**

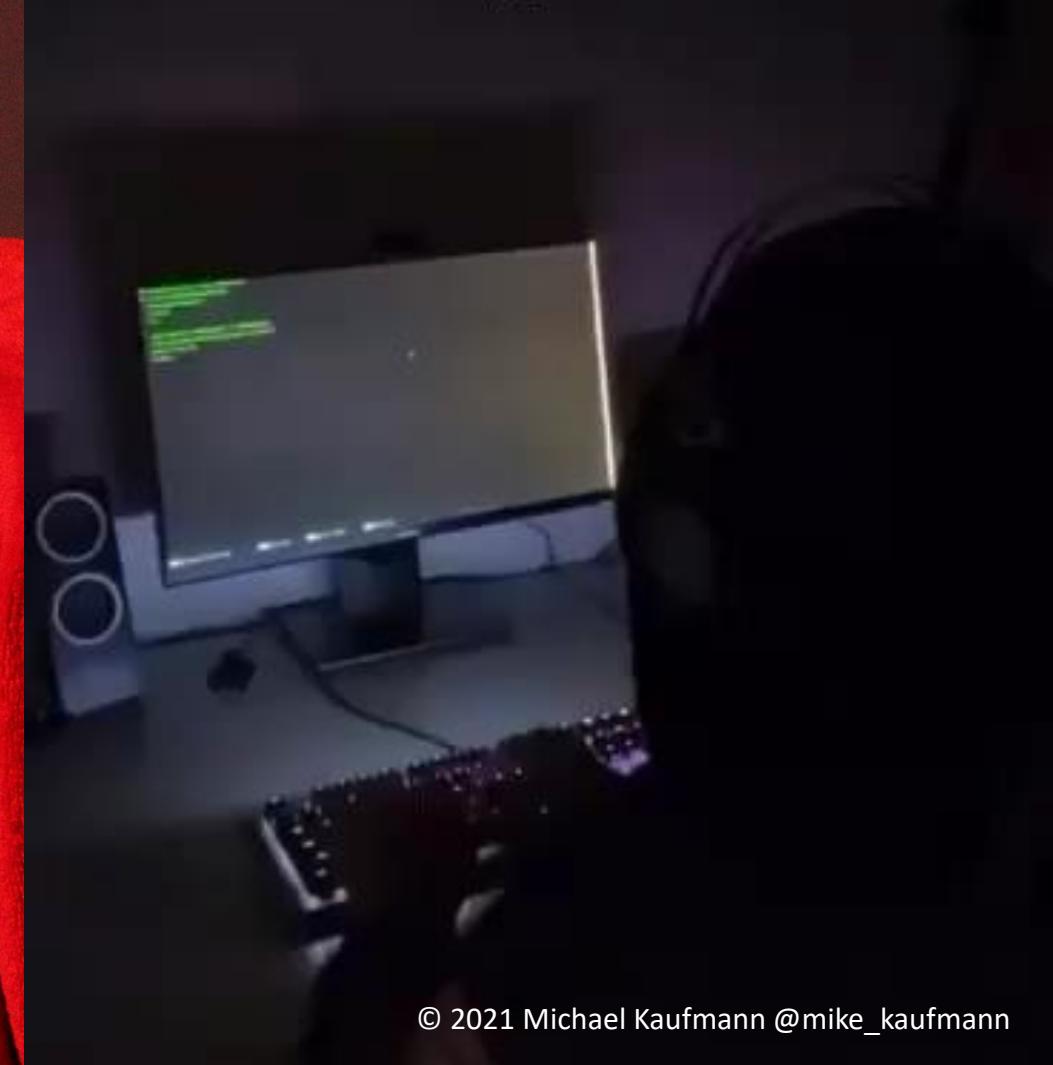
<https://azure.microsoft.com/en-us/resources/6-tips-to-integrate-security-into-your-devops-practices/>

Cuando hablamos de seguridad...

- “Hasta ahora nunca hemos tenido un incidente con la seguridad”
- “Nuestros desarrolladores son expertos”
- “Nuestra empresa no es tan importante como para ser atacada”
- “Estamos bien protegidos”
- “Nuestros cortafuegos son seguros”
- “¡Tenemos que hacer algo!”
- “¡La seguridad tiene máxima prioridad!”
- “¡Si hemos tenido un incidente con la seguridad, es que es demasiado tarde!”



Hackers in movies
vs





Muchas gracias

Blog: <https://writeabout.net>

Twitter: @mike_kaufmann

GitHub: @wulfland

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