Pragmatic Web Security

Security training for developers

Angular and the OWASP top 10

Philippe De Ryck

@PhilippeDeRyck – philippe@PragmaticWebSecurity.com

Dr. Philippe De Ryck

- Deep understanding of the web security landscape
- Google Developer Expert (not employed by Google)
- Course curator of the SecAppDev course
  (https://secappdev.org)

Pragmatic Web Security

High-quality security training for developers and managers

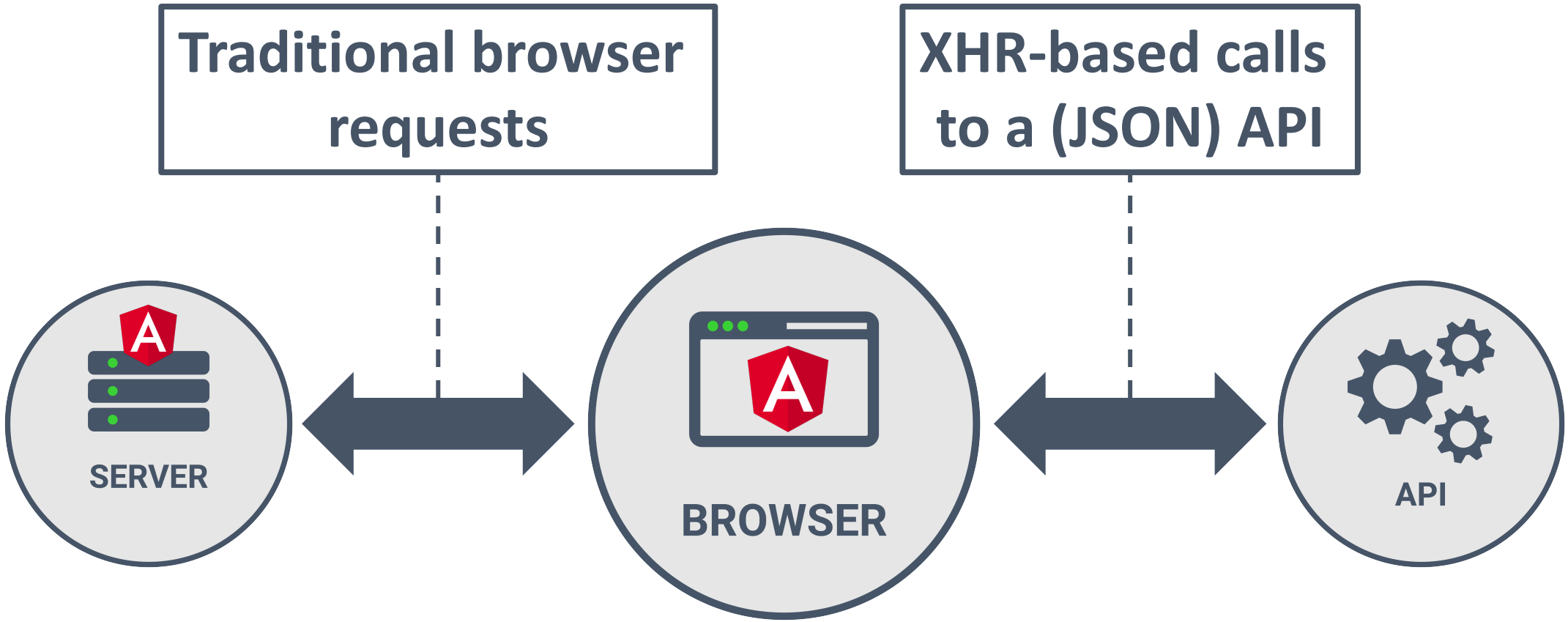
Custom courses covering web security, API security, Angular security, …

Consulting services on security, Oauth 2.0, OpenID Connect, …

@PhilippeDeRyck

https://PragmaticWebSecurity.com

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]


Traditional browser requests
XHR-based calls to a (JSON) API
SERVER
BROWSER
API

1 Injection

2 Broken authentication

3 Sensitive data exposure

4 XML External Entities (XXE)

5 Broken access control

6 Security misconfiguration

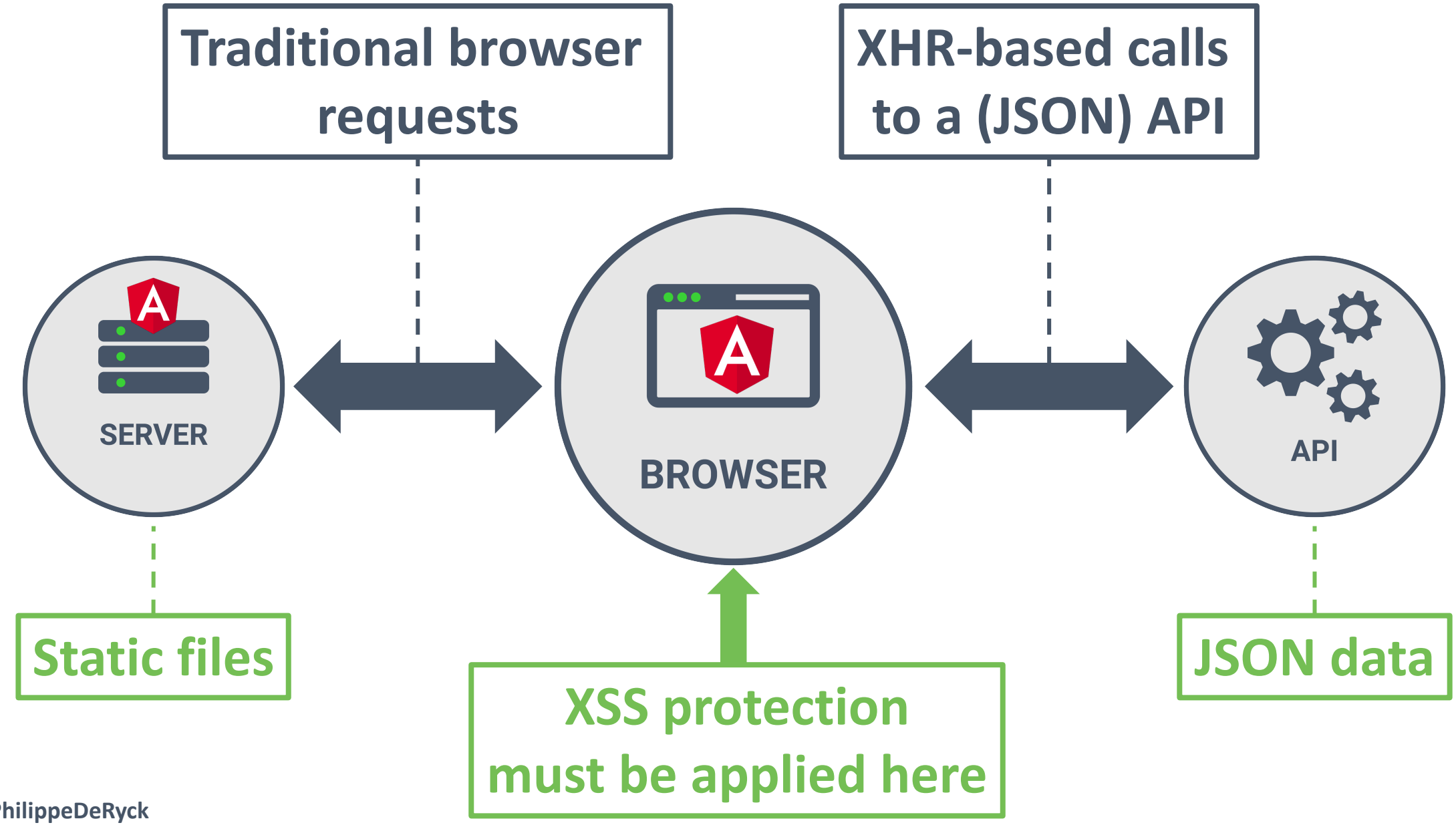
7 Cross-Site Scripting (XSS)

8 Insecure deserialization

9 Using components with known vulnerabilities

10 Insufficient logging & monitoring

9	Using components with known vulnerabilities
2	Broken authentication
7	Cross-Site Scripting (XSS)
5	Broken access control
3	Sensitive data exposure
6	Security misconfiguration
1	Injection
10	Insufficient logging & monitoring
8	Insecure deserialization
4	XML External Entities (XXE)


Traditional browser requests
XHR-based calls to a (JSON) API
SERVER
BROWSER
API
Static files
XSS protection must be applied here
JSON data

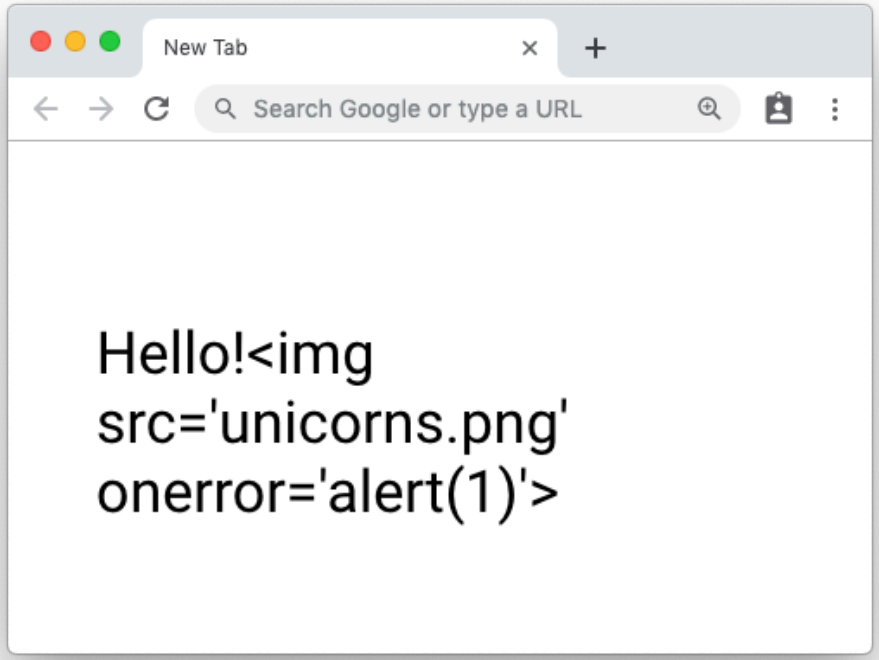
ANGULAR CODE

```
{{untrustedData}}
```

UNTRUSTED DATA

```
Hello!<img src='unicorns.png' onerror='alert(1)'>
```

RENDERED PAGE



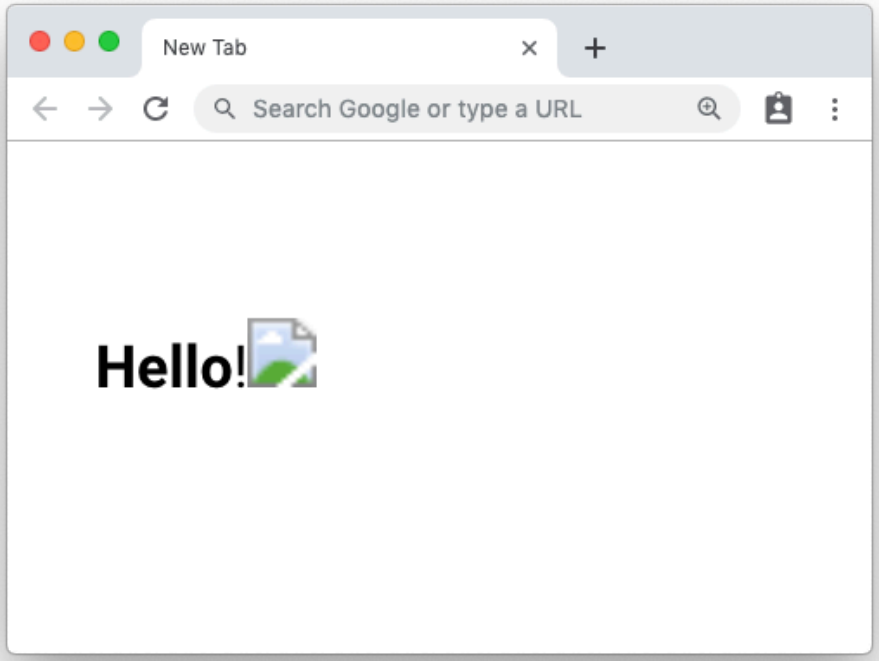
ANGULAR CODE

```
<div [innerHTML]="untrustedData"></div>
```

UNTRUSTED DATA

```
<b>Hello</b>!<img src='unicorns.png' onerror='alert(1)'>
```

RENDERED PAGE



Avoiding XSS in Angular

Get out of the way and follow the "Angular way"

```
<a [href]="'javascript:alert(1)'">
  A link with an untrusted HREF value
</a>
```

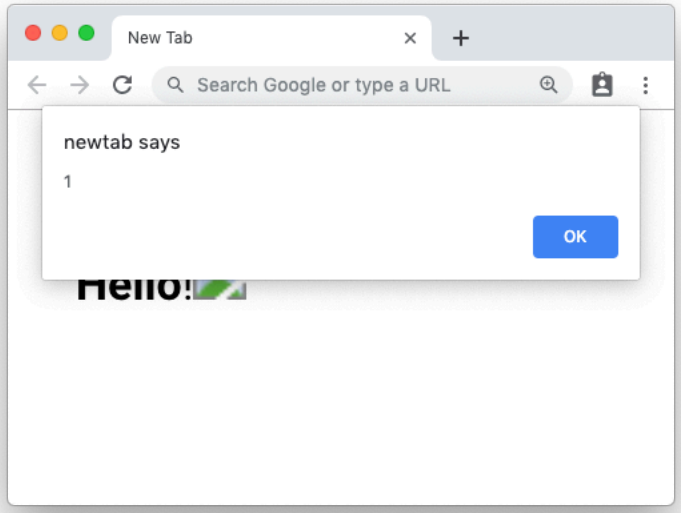
```
⚠ ▶WARNING: sanitizing unsafe URL value javascript:alert(1)      core.js:11210
(see http://g.co/ng/security#xss)
```

Avoiding XSS in Angular

Get out of the way and follow the "Angular way"

Do not use *bypassSecurityTrust** for untrusted data

```
<div [innerHTML]="sanitizer.bypassSecurityTrustHtml(untrustedData)">
</div>
```



Avoiding XSS in Angular

Get out of the way and follow the "Angular way"

Do not use *bypassSecurityTrust for untrusted data**

Do not use raw DOM APIs or third-party libraries

Avoid giving the user control over resource URLs

ANGULAR CODE

```
<iframe [src]="myFrameUrl"></div>
```

LEGITIMATE DATA

```
https://youtu.be/wlxAXIXXOYw
```

MALICIOUS DATA

```
javascript:alert('Moar XSS!')
```

RENDERED PAGE

```
▶ ERROR Error: unsafe value used in a resource URL context (see http://g.co/ng/security#xss)                AppComponent.html:1
    at DomSanitizerImpl.push../node_modules/@angular/platform-browser/fesm5/platform-browser.js.DomSanitizerImpl.sanitize (platform-browser.js:1824)
    at setElementProperty (core.js:21109)
    at checkAndUpdateElementValue (core.js:21061)
    at checkAndUpdateElementInline (core.js:21008)
    at checkAndUpdateNodeInline (core.js:23359)
    at checkAndUpdateNode (core.js:23325)
    at debugCheckAndUpdateNode (core.js:23959)
    at debugCheckRenderNodeFn (core.js:23945)
    at Object.eval [as updateRenderer] (AppComponent.html:3)
    at Object.debugUpdateRenderer [as updateRenderer] (core.js:23937)
▶ ERROR CONTEXT ▶ DebugContext_ {view: {…}, nodeIndex: 0, nodeDef: {…}, elDef: {…}, elView: {…}}   AppComponent.html:1
```

Angular code

```
private getYoutubeVideo(input : string) : SafeResourceUrl {
  // Always define scheme, host and path separator
  const host = "https://www.youtube.com/embed/";

  // Additional check to ensure the URL is safe to use
  let url = this.sanitizer.sanitize(SecurityContext.URL, host + input);

  // Mark the URL safe to use in a resource URL context
  return this.sanitizer.bypassSecurityTrustResourceUrl(url);
}
```

Untrusted data

```
wlxAXIXXOYw
```

--aot

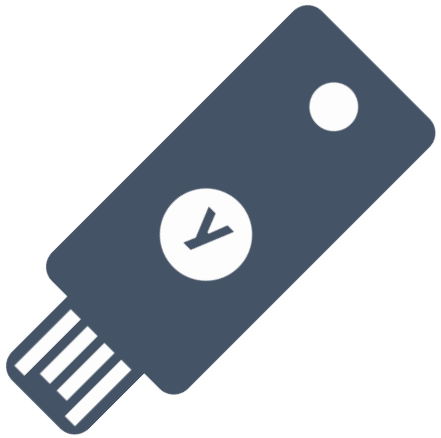
CROSS-SITE SCRIPTING (XSS)

Angular automatically applies XSS defenses

Use AOT and do everything "the Angular way"

Avoid using bypassSecurityTrust functions with untrusted input*

****



angular-oauth2-oidc

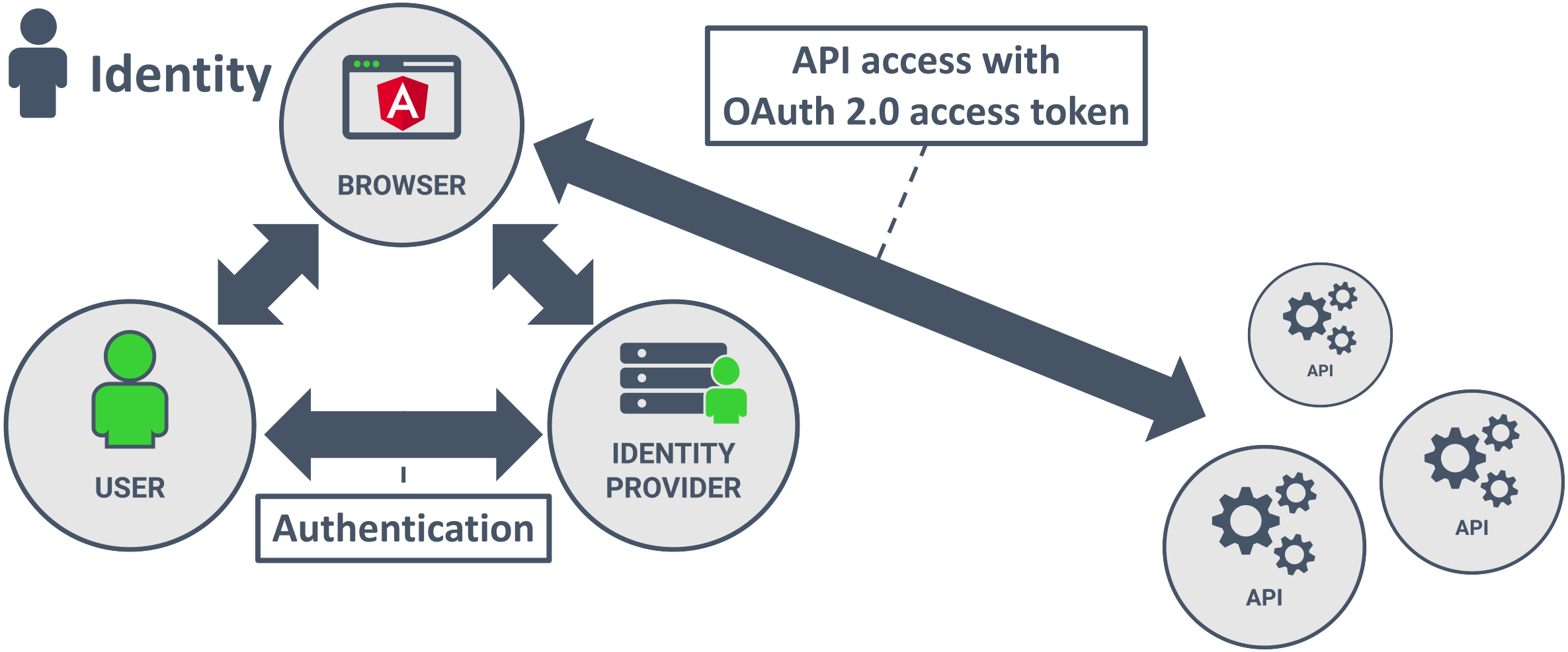
8.0.4 • Public • Published 2 months ago

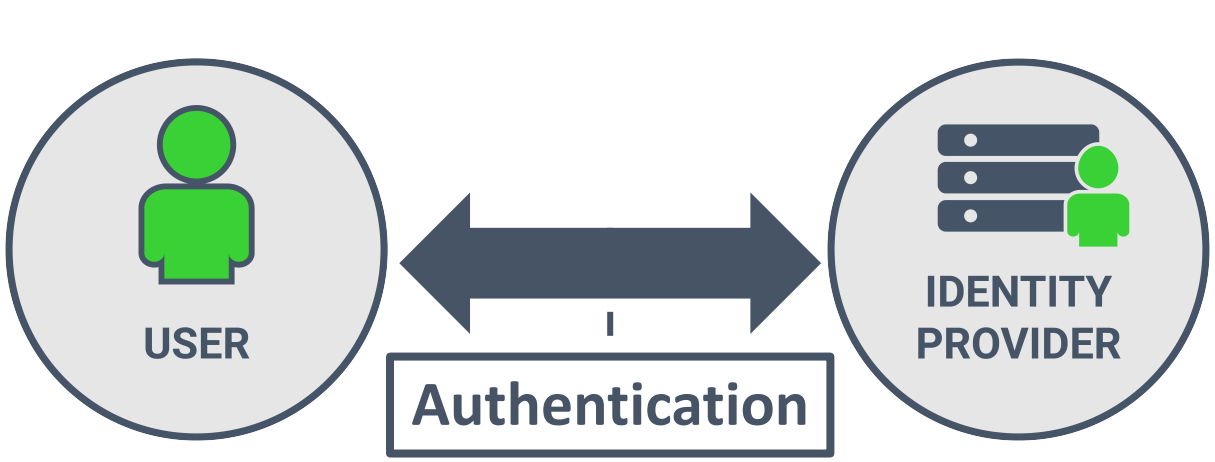
Readme | 2 Dependencies

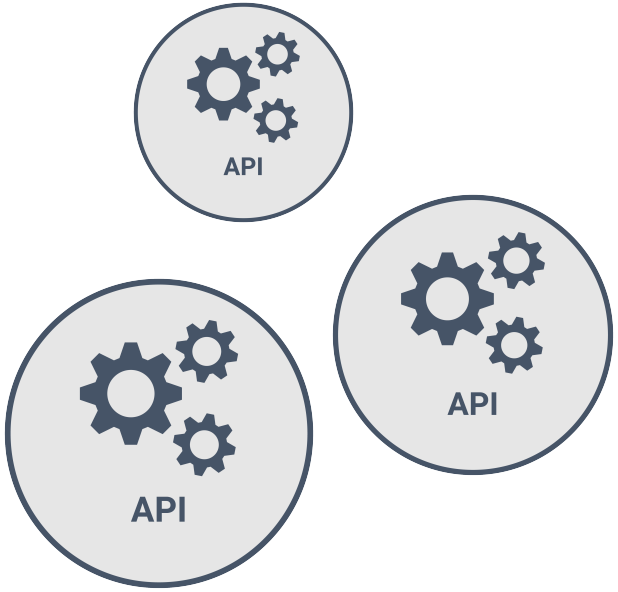
angular-oauth2-oidc

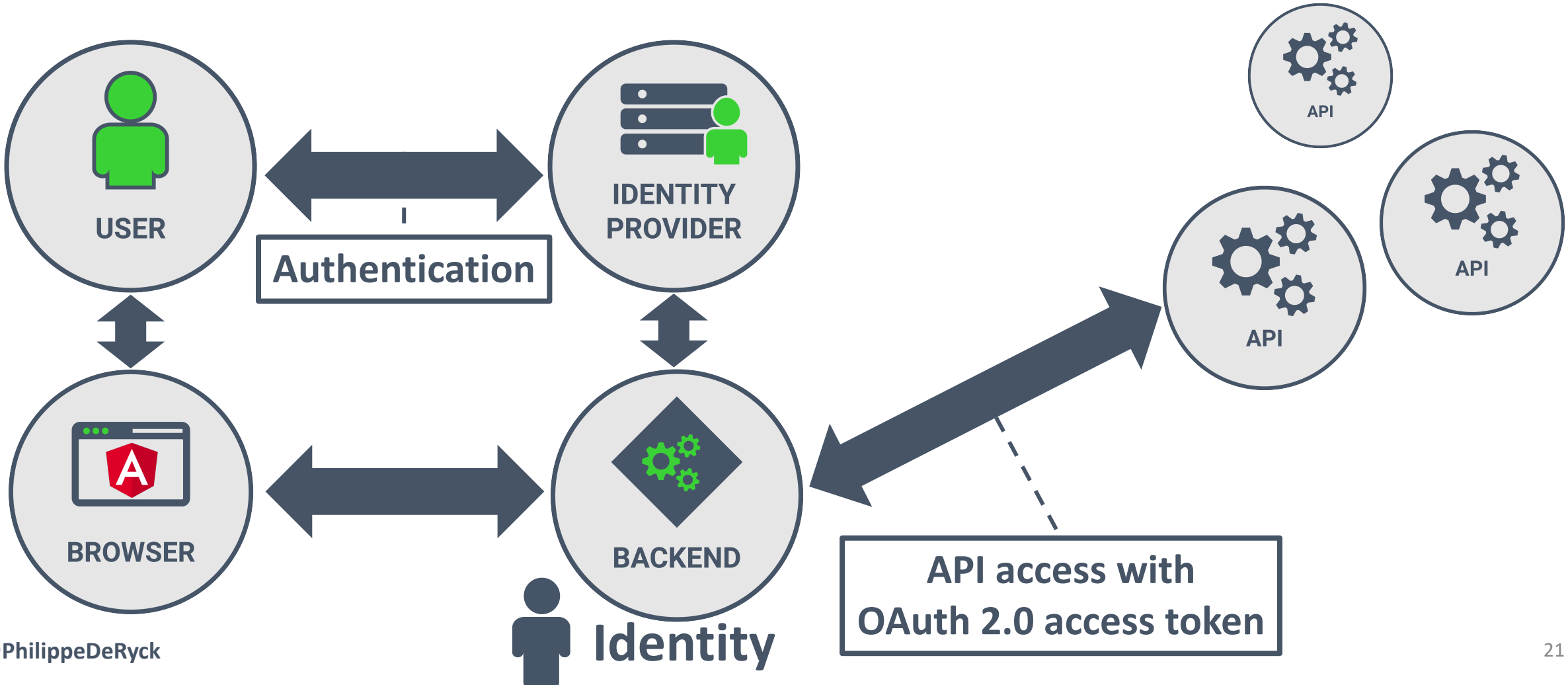
Support for OAuth 2 and OpenId Connect (OIDC) in Angular.

OpenID® CERTIFIED


Identity
BROWSER
API access with
OAuth 2.0 access token
USER
IDENTITY
PROVIDER
Authentication
API
API
API


USER
Authentication
IDENTITY
PROVIDER


API
API
API


USER
IDENTITY PROVIDER
Authentication
API
API
API
BROWSER
BACKEND
API access with
OAuth 2.0 access token
Identity

Broken authentication

OpenID Connect is the best way to offload authentication

OAuth 2.0 enables authorization for accessing APIs

Rely on high-quality libraries for OAuth 2.0 and OIDC

> 97%

of code in a modern web app are dependencies

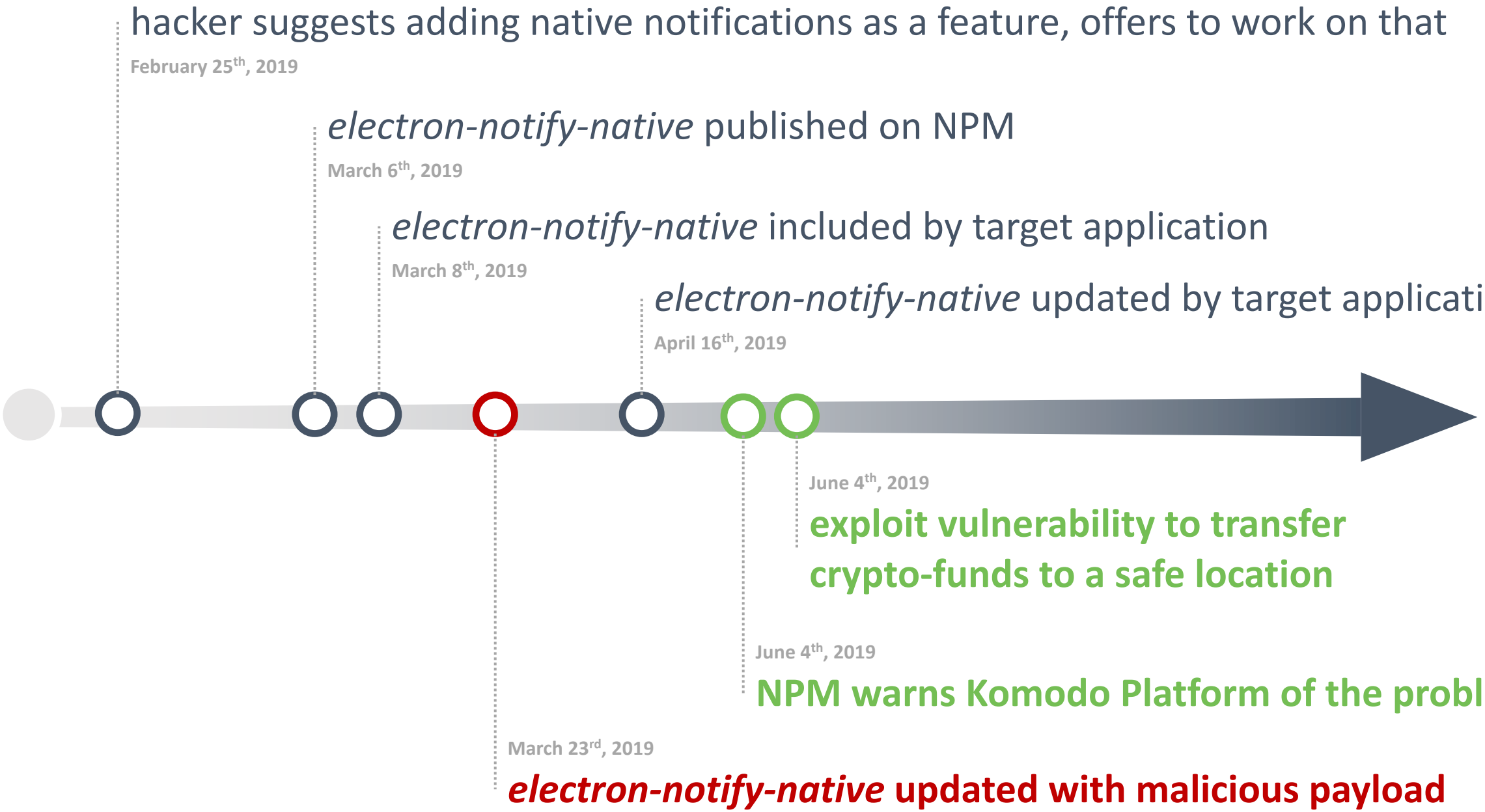
```
$ ng new clean-app

? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? Sass


added 1169 packages from 1030 contributors and audited
42445 packages in 28.75s
```

```
$ cloc node_modules/
```

Language	files	blank	comment	code
JavaScript	12683	145344	525680	1773037
JSON	1555	104	0	161571
Markdown	1385	65564	4	157446
TypeScript	2892	9625	90588	104376
HTML	274	1656	218	33724
CSS	148	299	2301	22382
C++	75	3784	3501	22332
Python	51	4205	7606	18695
C/C++ Header	101	2758	1858	15114
LESS	482	1611	410	11321
XML	20	3237	1300	7617
YAML	163	140	112	2416
Bourne Shell	18	292	333	1500
SVG	8	2	2	776
make	30	236	39	715
Windows Module Definition	7	115	0	641
DTD	1	179	177	514
...				
SUM:	19983	239598	634354	2336228


hacker suggests adding native notifications as a feature, offers to work on that
February 25th, 2019
electron-notify-native published on NPM
March 6th, 2019
electron-notify-native included by target application
March 8th, 2019
electron-notify-native updated by target application
April 16th, 2019
June 4th, 2019
exploit vulnerability to transfer crypto-funds to a safe location
June 4th, 2019
NPM warns Komodo Platform of the problem
March 23rd, 2019
electron-notify-native updated with malicious payload

- Pulse
- Contributors
- Traffic
- Commits
- Code frequency
- **Dependency graph**
- Alerts
- Network
- Forks

Dependency graph

Dependencies | Dependents

We found potential security vulnerabilities in your dependencies.

Dependencies defined in these manifest files have known security vulnerabilities and should be updated:

restograde-angular/package-lock.json *5 vulnerabilities found*
reviewer-angular/package-lock.json *5 vulnerabilities found*

See security alerts

Only the owner of this repository can see this message.
Learn more about vulnerability alerts

These dependencies are defined in **pws-restograde**'s manifest files, such as **reviewer-angular/package-lock.json**, **reviewer-angular/package.json**, and **restograde-angular/package-lock.json**.

DEPENDENCY-CHECK

How to read the report | **Suppressing false positives** | **Getting Help: google group** | **github issues**

Project: Restograde

Scan Information (show all):

- *dependency-check version*: 4.0.0
- *Report Generated On*: Dec 5, 2018 at 09:43:02 +01:00
- *Dependencies Scanned*: 1217 (1048 unique)
- *Vulnerable Dependencies*: 10
- *Vulnerabilities Found*: 10
- *Vulnerabilities Suppressed*: 0
- ...

Snyk for Developers & DevOps

Snyk continuously monitors your application's dependencies and lets you quickly respond when new vulnerabilities are disclosed.

Fix your vulnerabilities

- **Single click fix - generate a fix PR from UI, CLI wizard**
- Upgrade - Automatically calculates the minimal direct dependency version upgrade needed
- Precision patch - Use patches backported by Snyk security team to fix when direct upgrade is not available or it'll take time to have upgrade implemented
- Automatic fix for new vulnerabilities - Automatically generate fix pull requests for newly discovered vulnerabilities

USING COMPONENTS WITH (KNOWN) VULNERABILITIES

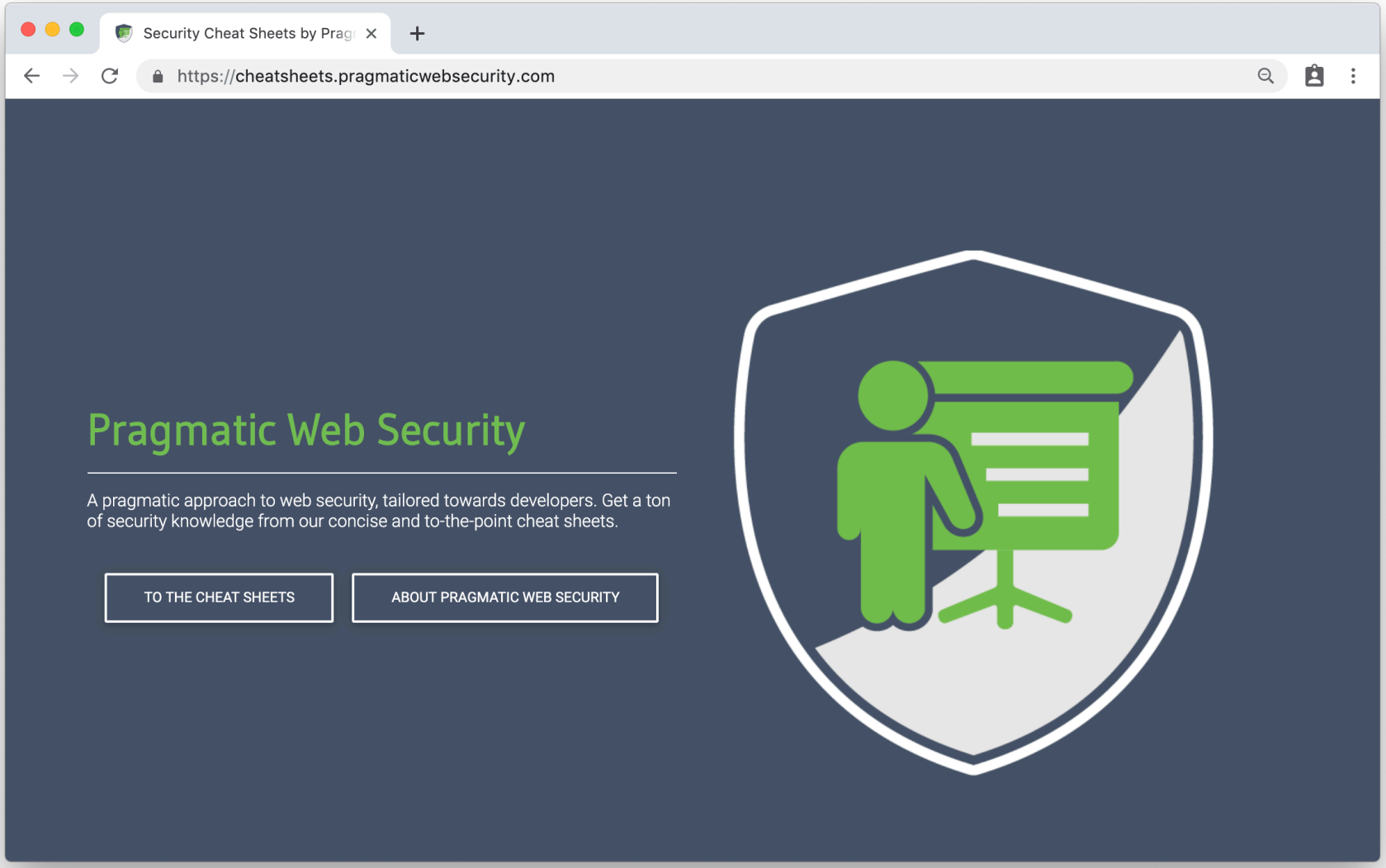
Setup dependency checking for your applications

Include regular maintenance intervals in your processes

Triage potential vulnerabilities for the urgency to patch

9	Using components with known vulnerabilities
2	Broken authentication
7	Cross-Site Scripting (XSS)
5	Broken access control
3	Sensitive data exposure
6	Security misconfiguration
1	Injection
10	Insufficient logging & monitoring
8	Insecure deserialization
4	XML External Entities (XXE)

FREE SECURITY CHEAT SHEETS FOR MODERN APPLICATIONS



https://cheatsheets.pragmaticwebsecurity.com/

Pragmatic Web Security

Security training for developers

/in/PhilippeDeRyck

@PhilippeDeRyck

philippe@pragmaticwebsecurity.com