



GETTING API SECURITY RIGHT

DR. PHILIPPE DE RYCK

<https://PragmaticWebSecurity.com>

Business

Mobile Health Apps Systematically Expose PII and PHI Through APIs, New Findings from Knight Ink and Approov Show

9 February 2021, 12:00 CET

<https://www.bloomberg.com/press-releases/2021-02-09/mobile-health-apps-systematically-expose-pii-and-phi-through-apis-new-findings-from-knight-ink-and-approov-show>

“Of the 30 popular apps Knight Ink tested, 77 percent contained hardcoded API keys, some which don’t expire, and seven percent contained hardcoded usernames and passwords.”

Business

Mobile Health Apps Systematically Expose PII and PHI Through APIs, New Findings from Knight Ink and Approov Show

9 February 2021, 12:00 CET

<https://www.bloomberg.com/press-releases/2021-02-09/mobile-health-apps-systematically-expose-pii-and-phi-through-apis-new-findings-from-knight-ink-and-approov-show>

“100 percent of API endpoints tested were vulnerable to BOLA attacks that allowed the researcher to view the PII and PHI for patients that were not assigned to the researcher’s clinician account.”

START TAKING SECURITY SERIOUSLY



The cowboy years are over. Security is a crucial requirement for every application from day 1, and not an afterthought for a quiet period.



I am *Dr. Philippe De Ryck*



Founder of Pragmatic Web Security



Google Developer Expert



Auth0 Ambassador



SecAppDev organizer

I help developers with security



Hands-on in-depth security training



Advanced online security courses



Security advisory services



<https://pragmaticwebsecurity.com>

Reverse Engineering Bumble's API

When you have too much time on your hands and want to dump out Bumble's entire user base and bypass paying for premium Bumble Boost features.



Sanjana Sarda

Follow

Nov 14 · 8 min read



“
Our accounts eventually got locked and hidden for more verification requirements. We tested retrieving user data while our account was locked, and it still worked.
”

<https://blog.securityevaluators.com/reverse-engineering-bumbles-api-a2a0d39b3a87>

“

It's possible to view deleted fleets via Twitter's API endpoint, to view existing fleets without giving the poster a read notification and you can do both without being logged into Twitter.

”



cathode gay tube
@donk_enby

full disclosure: scraping fleets from public accounts without triggering the read notification

the endpoint is: api.twitter.com/fleets/v1/user...

12:51 AM · Nov 21, 2020 · Twitter Web App

528 Retweets 226 Quote Tweets 1.4K Likes



cathode gay tube @donk_enby · Nov 21

Replying to @donk_enby

for auth you just use the same leaked consumer keys from official twitter app that lets you use firehose for free: gist.github.com/shobotch/51600...

ddg api.twitter.com/auth/1/xauth_p... for how to get a token



Twitter (un)official Consumer Key

Twitter (un)official Consumer Key. GitHub Gist: instantly share code, notes, and snippets.

https://twitter.com/donk_enby/status/1329935540049817600

THE CLIENT IS IRRELEVANT FOR SECURITY



*The attack surface of an API
consists of all accessible endpoints,
regardless of how and if they are used by the client*



A security flaw in Grindr let anyone easily hijack user accounts

Zack Whittaker @zackwhittaker / 10:22 PM GMT+2 • October 2, 2020

 Comment



 Image Credits: SOPA Images / Getty Images

Grindr, one of the world's largest dating and social networking apps for gay, bi, trans, and queer people, has fixed a security vulnerability that allowed anyone to hijack and take control of any user's account using only their email address.

<https://techcrunch.com/2020/10/02/grindr-account-hijack-flaw/>

“

To reset a password, Grindr sends the user an email with a clickable link containing an account password reset token.

Grindr's password reset page was leaking password reset tokens to the browser.

”

The API response to retrieve online users



```
1  [  
2    {  
3      "id": 3,  
4      "name": "John",  
5      "address": "5 George's Dock, ...",  
6    },  
7    {  
8      "id": 6,  
9      "name": "Jakob",  
10     "address": "71-75 Shelton Street, ...",  
11   },  
12   {  
13     "id": 17,  
14     "name": "Philippe",  
15     "address": "Holsbeeksesteenweg 143, ...",  
16   }  
17 ]
```





If an API automatically exposes data, does it also automatically accept data?

The body of a legitimate request to update the user's name

```
1 {  
2   "name": "Dr. Phil"  
3 }
```

The API uses a framework that automatically transforms JSON data into domain objects, which are then used to update the persisted data

Without filtering the input properties, the API becomes vulnerable to mass assignment

The Java class of the User object

```
1 class User {  
2   String name;  
3   String email;  
4   String role;  
5 }
```

The body of a malicious request to update the user's name

```
1 {  
2   "name": "Philippe becomes admin",  
3   "role": "admin"  
4 }
```

TEST YOUR APIs IN THEIR NATURAL HABITAT



Make sure your API behaves the way you think it does. Code analysis is only one aspect. Runtime testing is necessary to get the full picture.





Is testing really the best strategy?

```
1  paths:
2    /online/users:
3      get:
4        responses:
5          '200':
6            description: A list of online users
7            content:
8              application/json:
9                schema:
10                  type: array
11                  items:
12                    type: object
13                    properties:
14                      id:
15                        type: integer
16                        description: The user ID
17                      name:
18                        type: string
19                        description: The display name of the user
```



Automated IDOR Discovery through Stateful Swagger Fuzzing



Aaron Loo, Engineering Manager
Jan 16, 2020

Scaling security coverage in a growing codebase to empower front-line developers to be able to ship code they make it to production servers.

Today, we're excited to announce that we've developed to identify [Insecure Direct Object References](#) through stateful [Swagger fuzzing](#), tailored to support our CI pipeline. It integrates with our Continuous Integration tool to provide coverage as web applications evolve.

Microsoft Research Blog

RESTler finds security and reliability bugs through automated fuzzing

Published November 16, 2020



Research Area

 [Security, privacy, and cryptography](#)



@PhilippeDeRyck

<https://engineeringblog.yelp.com/2020/01/automated-idor-discovery-through-stateful-swagger-fuzzing.html>
<https://www.microsoft.com/en-us/research/blog/restler-finds-security-and-reliability-bugs-through-automated-fuzzing/>

USE SWAGGER/OPENAPI DEFINITIONS FOR SECURITY



Write Swagger/OpenAPI definitions to specify the behavior of your API. Security tools consume such definitions for automatic detection and protection.



T-Mobile Website Allowed Hackers to Access Your Account Data With Just Your Phone Number

“ he could query for someone else's phone number and the API would simply send back a response containing the other person's data. ”



Olatunde Michael Garuba

FOLLOW

Full stack Javascript Developer

Build Node.js RESTful APIs in 10 Minutes

Published Jan 12, 2017 Last updated Aug 18, 2017



@PhilippeDeRyck

<https://www.codementor.io/@olatundegaruba/nodejs-restful-apis-in-10-minutes-q0sgsfhbd>

A REST API endpoint without any authorization

```
1 app.delete('/tasks/:taskId', function(req, res) {
2   Task.remove({
3     _id: req.params.taskId
4   }, function(err, task) {
5     if (err)
6       res.send(err);
7     res.json({ message: 'Task successfully deleted' });
8   });
9 };
```



Permissions on an endpoint do not suffice to stop *broken object-level authorization*

Checking permissions helps prevent *broken function-level authorization*

A REST API endpoint restricted to users with the specific "deleteTask" permission

```
1 app.delete('/tasks/:taskId', auth.hasPermission('deleteTask'), function(req, res) {
2   Task.remove({
3     _id: req.params.taskId
4   }, function(err, task) {
5     if (err)
6       res.send(err);
7     res.json({ message: 'Task successfully deleted' });
8   });
9 });
```



ENFORCE AUTHORIZATION AT THE FUNCTION LEVEL



By applying a sensible permission/role check to every endpoint, unauthorized requests can be rejected before they reach the application logic



A permission check only allows authorized users to access this endpoint

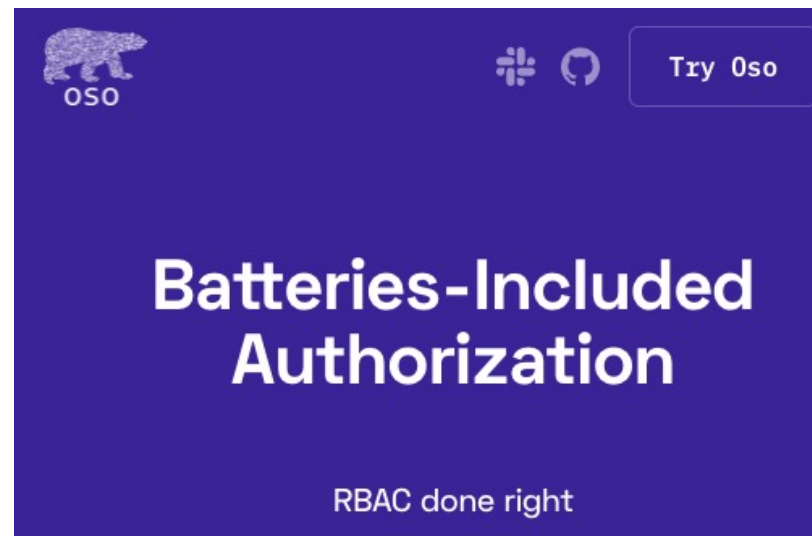
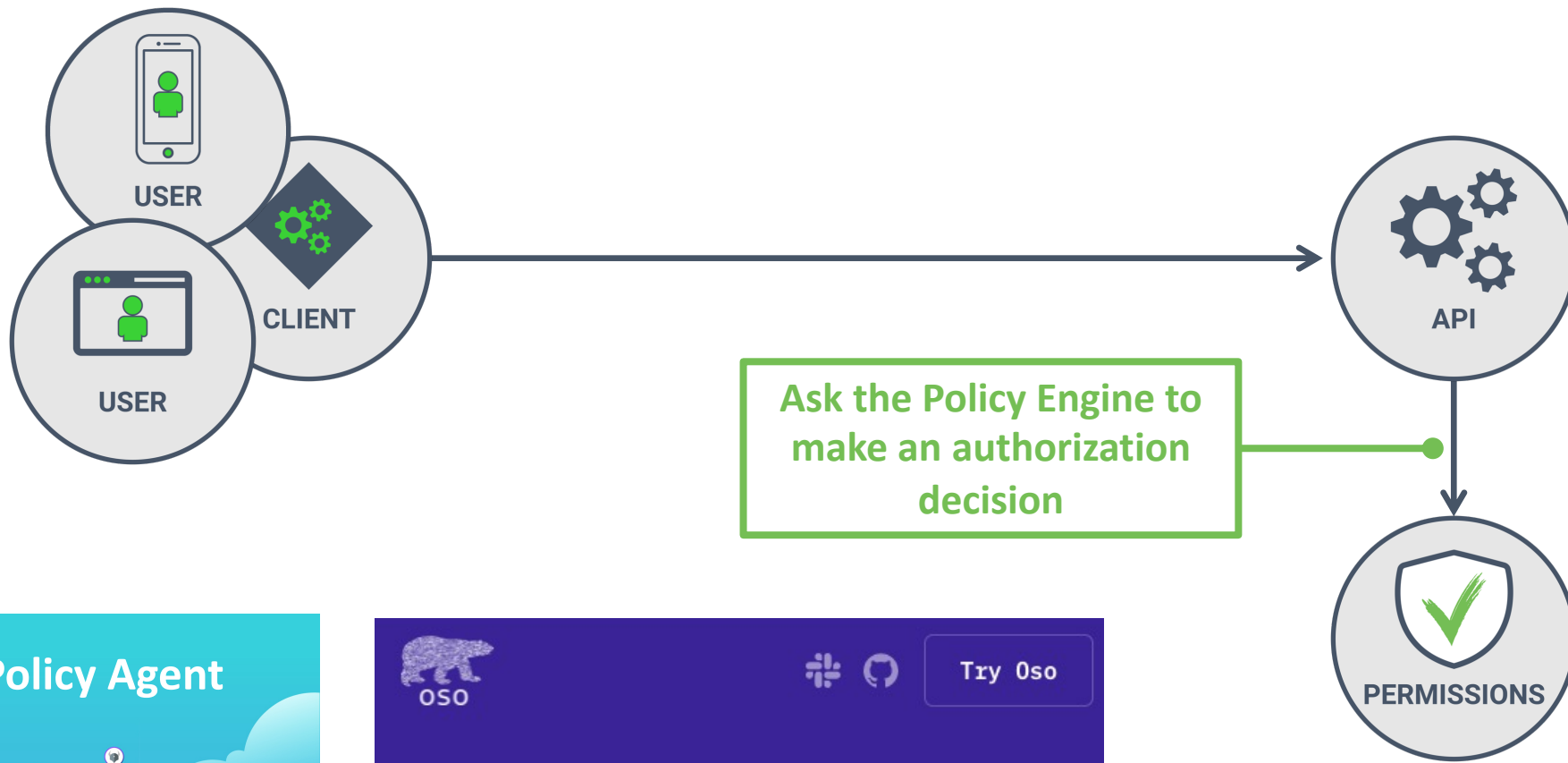
Object-level access control is often challenging to implement

```
1 app.delete('/tasks/:taskId', auth.hasPermission('deleteTask'), async (req, res) => {
2   let origTask = await Task.findById(req.params.taskId)
3
4   if(auth.hasRole('employee') && !origTask.owner.id == auth.currentUser.id)
5     res.status(403).send(
6       { message: 'You are not a manager. You can only delete your own tasks.'});
7
8   // Delete task
9 });
```

Policies scattered throughout the code like this are impossible to audit for security

Certain roles require additional restrictions, such as task ownership





ENCAPSULATE COMPLEX AUTHORIZATION LOGIC



Complex authorization logic should not be scattered throughout the code, but is best defined in a clear and understandable authorization policy



What happens when



goes wrong?



COMPARTMENTALIZE YOUR APIs



Many APIs combine sensitive features and mundane application logic into a single service.

Compartmentalization helps limit the impact of a vulnerability.



Now it is up to you ...

Audit your API authorization policy

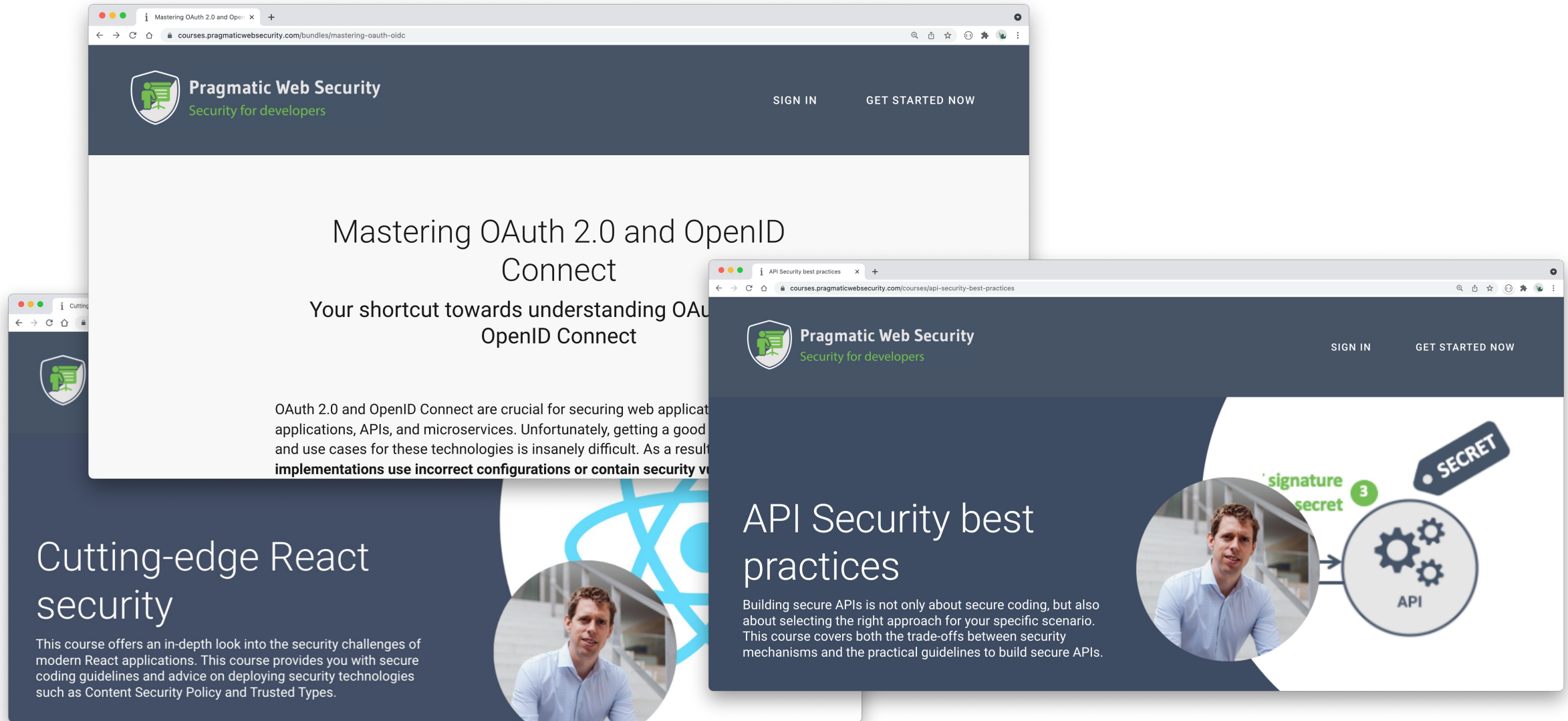
Write security tests to verify authorization and JWT handling

Build OpenAPI definitions to specify the exact API behavior

...



Keep learning with these in-depth security courses!



The image displays three overlapping browser windows showcasing Pragmatic Web Security courses. The top window is titled 'Mastering OAuth 2.0 and OpenID Connect' and features the text 'Your shortcut towards understanding OAuth 2.0 and OpenID Connect'. Below this, it states: 'OAuth 2.0 and OpenID Connect are crucial for securing web applications, APIs, and microservices. Unfortunately, getting a good understanding of the various use cases for these technologies is insanely difficult. As a result, many implementations use incorrect configurations or contain security vulnerabilities.' The bottom-left window is titled 'Cutting-edge React security' and includes the text: 'This course offers an in-depth look into the security challenges of modern React applications. This course provides you with secure coding guidelines and advice on deploying security technologies such as Content Security Policy and Trusted Types.' The bottom-right window is titled 'API Security best practices' and states: 'Building secure APIs is not only about secure coding, but also about selecting the right approach for your specific scenario. This course covers both the trade-offs between security mechanisms and the practical guidelines to build secure APIs.' Each window features the Pragmatic Web Security logo, navigation links for 'SIGN IN' and 'GET STARTED NOW', and a circular portrait of a man in a light blue shirt. The 'API Security best practices' window also includes a diagram of an API with a 'SECRET' tag and a 'signature secret' label.

Pragmatic Web Security
Security for developers

SIGN IN GET STARTED NOW

Mastering OAuth 2.0 and OpenID Connect

Your shortcut towards understanding OAuth 2.0 and OpenID Connect

OAuth 2.0 and OpenID Connect are crucial for securing web applications, APIs, and microservices. Unfortunately, getting a good understanding of the various use cases for these technologies is insanely difficult. As a result, many implementations use incorrect configurations or contain security vulnerabilities.

Cutting-edge React security

This course offers an in-depth look into the security challenges of modern React applications. This course provides you with secure coding guidelines and advice on deploying security technologies such as Content Security Policy and Trusted Types.

API Security best practices

Building secure APIs is not only about secure coding, but also about selecting the right approach for your specific scenario. This course covers both the trade-offs between security mechanisms and the practical guidelines to build secure APIs.

signature secret 3

SECRET

API

[HTTPS://COURSES.PRAGMATICWEBSECURITY.COM](https://courses.pragmaticwebsecurity.com)



Thank you!

Connect on social media
to stay in touch



@PhilippeDeRyck



/in/PhilippeDeRyck

