RSA Conference2019
Asia Pacific & Japan
Singapore | 16–18 July | Marina Bay Sands

BETTER.

SESSION ID: SPO-R01A

API Security: Learning from the 20 Years of Appsec Failures

Matthieu Estrade

Chief Technology Officer
42Crunch
@mestrade

RSAConference2019
Asia Pacific & Japan

20 years... really ?

A bit of history of web application security

Web application security for early adopters

- 1999: Protect a website
- Mostly static content
- Thousands of URLs
- No parameters
- PHFvulnerability
- Cgi-bin !

Security Teams enter the game

- 2003: Pass the buck
- Secure something that you don’t know
- Web application scanner to understand what to secure
- Server misconfiguration
- PHP and dynamic Web Applications

Number of web applications increase

- 2006: More than 100 web apps in large organizations
- Dynamic web apps everywhere
- SQL injection
- Database + web application to secure
- Static code analysis
- Authentication & authorizations

Web services – machine to machine

- 2009: SOAP + XML
- Web applications are now complex
- Javascript
- WAF + SAST + DAST + educating developers
- XML and SOAP are complex and specifications are huge
- Too many standards

The client is javascript

- 2012: > 1000 web applications to secure in large organizations
- The client is javaScript or native mobile Apps
- Client-side attacks / XSS / XSRF etc.
- Speaking to REST + JSON backend
- Quick and dirty backend

There are two kind of people ("…")

- 2015: a clear separation between client and server
- The one developing the frontend and the one developing the backend
- Frontend is JavaScript
- React/Angular/frameworks…
- IoT/browser/APIs

Everything is an API

- 2018: 83% of traffic at the edge is APIs (Akamai)
- Poor implementation of REST+JSON
- Parser attacks
- TLS is not enough to secure an API
- No data validation

After 20 years…

- Most of the companies still work in siloed organizations
- Developers are not talking to security teams that are not talking to operations team
- Vendors are adding AI to understand how a web application is supposed to work
- All of these tools are not designed or developed to interact together

But we have a chance to improve that a lot!

We can change the game with API security

APIs everywhere

- REST + JSON is the way backend and frontend are communicating
- For web applications
- For mobile applications
- For IoT

Microservices

- APIs are split in microservices
- Clarify the processing
- Help in troubleshooting
- Increase scalability
- Enable splitting the development effort across teams

Kubernetes and Infrastructure as Code

- Docker is a game changer
- APIs are now delivered quickly
- Deployment is simple
- CI/CD automates tasks
- Infrastructure can be defined as code
- Developers and operations people can work in the same team

RSAConference2019
Asia Pacific & Japan

Infrastructure as Code enable DevOps

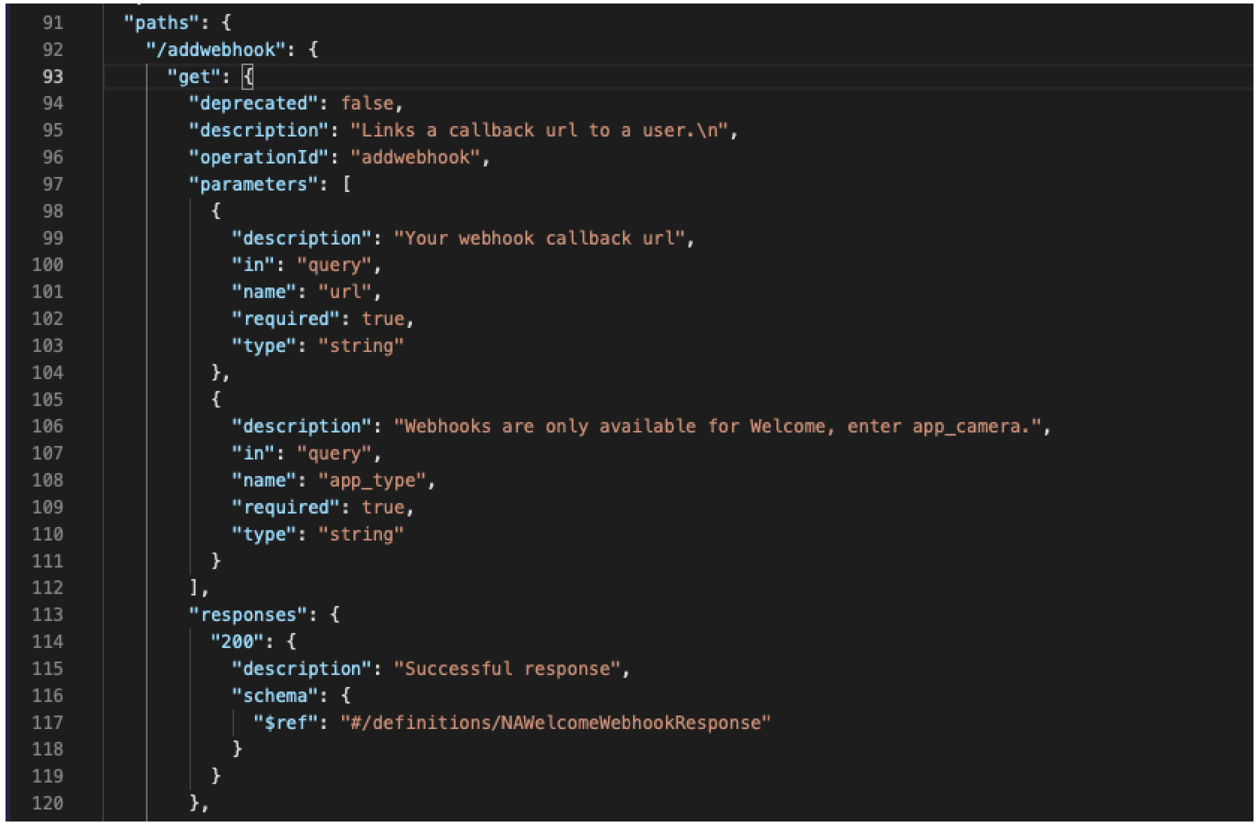
But what about security in the team?

Define your API with OpenAPI definitions

- Describe how your API is working
- What paths do you allow
- What operations do you allow
- What parameters do you allow
- What authentication does the API need
- What objects are sent back by the API

Define your API with OpenAPI definitions cont.

- Define valid formats for parameters
- Define min/max for integers
- Define maxLength/pattern for string
- Provide example values for each parameters
- Define mandatory objects



```
"paths": {
  "/addwebhook": {
    "get": {
      "deprecated": false,
      "description": "Links a callback url to a user.\n",
      "operationId": "addwebhook",
      "parameters": [
        {
          "description": "Your webhook callback url",
          "in": "query",
          "name": "url",
          "required": true,
          "type": "string"
        },
        {
          "description": "Webhooks are only available for Welcome, enter app_camera.",
          "in": "query",
          "name": "app_type",
          "required": true,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "Successful response",
          "schema": {
            "$ref": "#/definitions/NAWelcomeWebhookResponse"
          }
        }
      },
```

How does it help?

- Your security team now knows what they have to secure
- They know what is the normal behavior
  - They know how to test it
  - They know how to protect it

The developer is the one who knows how the application is developed!

If we know what an API is doing, it's more easy to secure!

Dynamic testing from OpenAPI definition

- No more Burp integration to intercept URLs to test (we have a list of valid calls)
- No more guessing parameters to understand values (we have examples)
- Better testing with real values
- Conformance testing versus vulnerabilities testing

Protect your APIs from OpenAPI definition

- Only allow what is defined
- Conformance reports
- Conformance check is cheap to execute (CPU)
- Easy to troubleshoot (vs complex detection engines)

RSA Conference2019 Asia Pacific & Japan

DevSecOps unifies developers, security engineers, operations in the same team.

OpenAPI definition or “API contract” makes them speak the same language!

The OpenAPI advantage

- More and more tools to create and play with OpenAPI definitions
- More and more development frameworks generate OpenAPI definitions from the code
- More and more development frameworks generate code from OpenAPI definitions
- Definitions can be created manually for an existing APIs

RSAConference2019
Asia Pacific & Japan

Use OpenAPI specifications in your organization

Make OpenAPI mandatory for your teams!

- Easier to understand what your APIs are doing
- Audit your APIs to find the ones dealing with critical data
- Checks which APIs use authentications and which are open to everybody
- Simplify your entire API security lifecycle

Identify and document your existing APIs

- Find each REST+JSON endpoint
- Ask your developers to document using OpenAPI
- Check if it's correctly implemented using an API security scanner that support OpenAPI
- Once you're happy with the documentation, deploy an API firewall that support OpenAPI

Design first!

- A design-first approach for all your new APIs
- Perform threat modelling to understand risk before engaging with development effort
- Generate code from the OpenAPI definition
- Use the free audit tool at APIsecurity.io to verify how your design handles security

Automate your security

- On each new API contract version, you can run a conformance scan to check that the implementation stays correct
- Choose a DAST that supports OpenAPI
- If the new API contract is well defined, you can automate the deployment of your new API firewall to detect API contract violation
- Choose a Firewall that support OpenAPI

In case of emergency

- Review your API contract with your developers
- Fix the issue in the OpenAPI definition
- Deploy an API Firewall that support OpenAPI and that will enforce your traffic regarding the API contract
- Fix the issue in the code and verify implementation with an API testing tool that support OpenAPI