

RSA[®]Conference2019

San Francisco | March 4–8 | Moscone Center



BETTER.

SESSION ID: SPO2-W12

Designing for API Doomsday

Josh Shaul

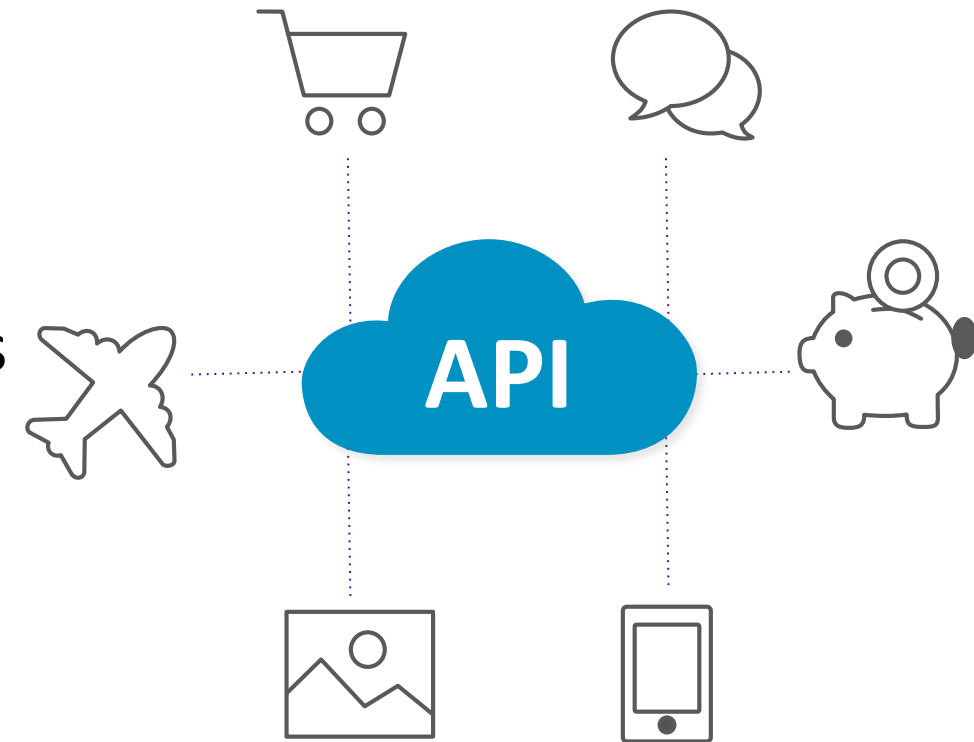
VP Web Security
Akamai Technologies



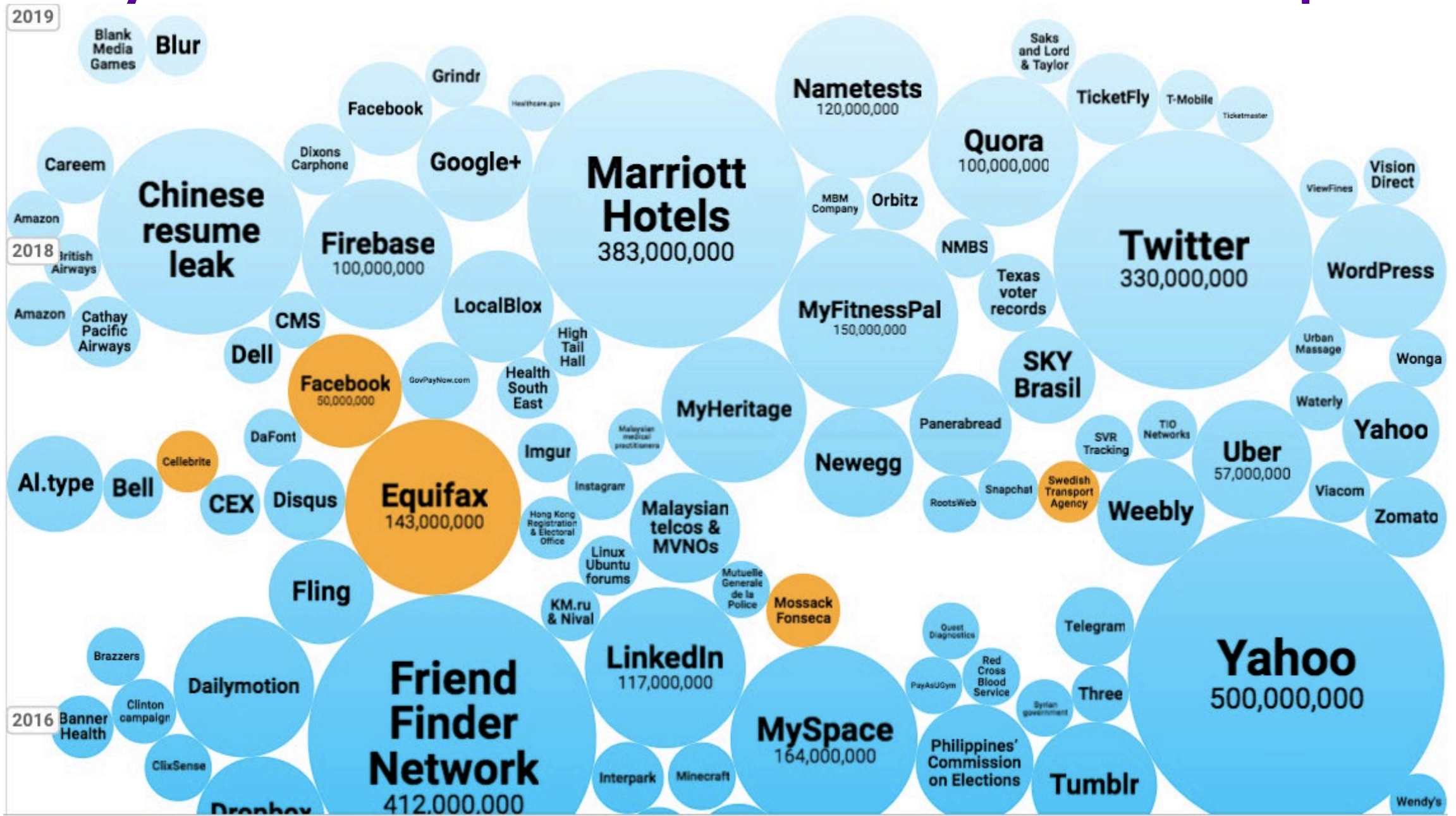
#RSAC

Why Are We Here? Because APIs Are Everywhere.

- Web APIs are becoming the center of every digital experience
 - Mobile apps run on web APIs
 - Web sites and applications rely on APIs for core functions (ex. login)
 - Modern, microservices-based architectures rely on APIs for communications
 - APIs power multiple user experiences
 - Web APIs can be required by regulators (ex. PSD2 / Open Banking in the UK)
 - Web APIs can be very attacker friendly....



Why Are We Here? Because Data Theft Is Still Rampant.



Web APIs: High Expectations and Broad Challenges



Business

- ✓ Speed up Innovation
- ✓ Improve Service Stability
- ✓ Drive Mobile Adoption
- ✓ Unlock New UX's
- ✓ Improve Customer Sat.



Attacker

- ✓ Back Door Access
- ✓ Data Theft / Modification
- ✓ Denial of Service
- ✓ Lower Cost to Attack



Legitimate user

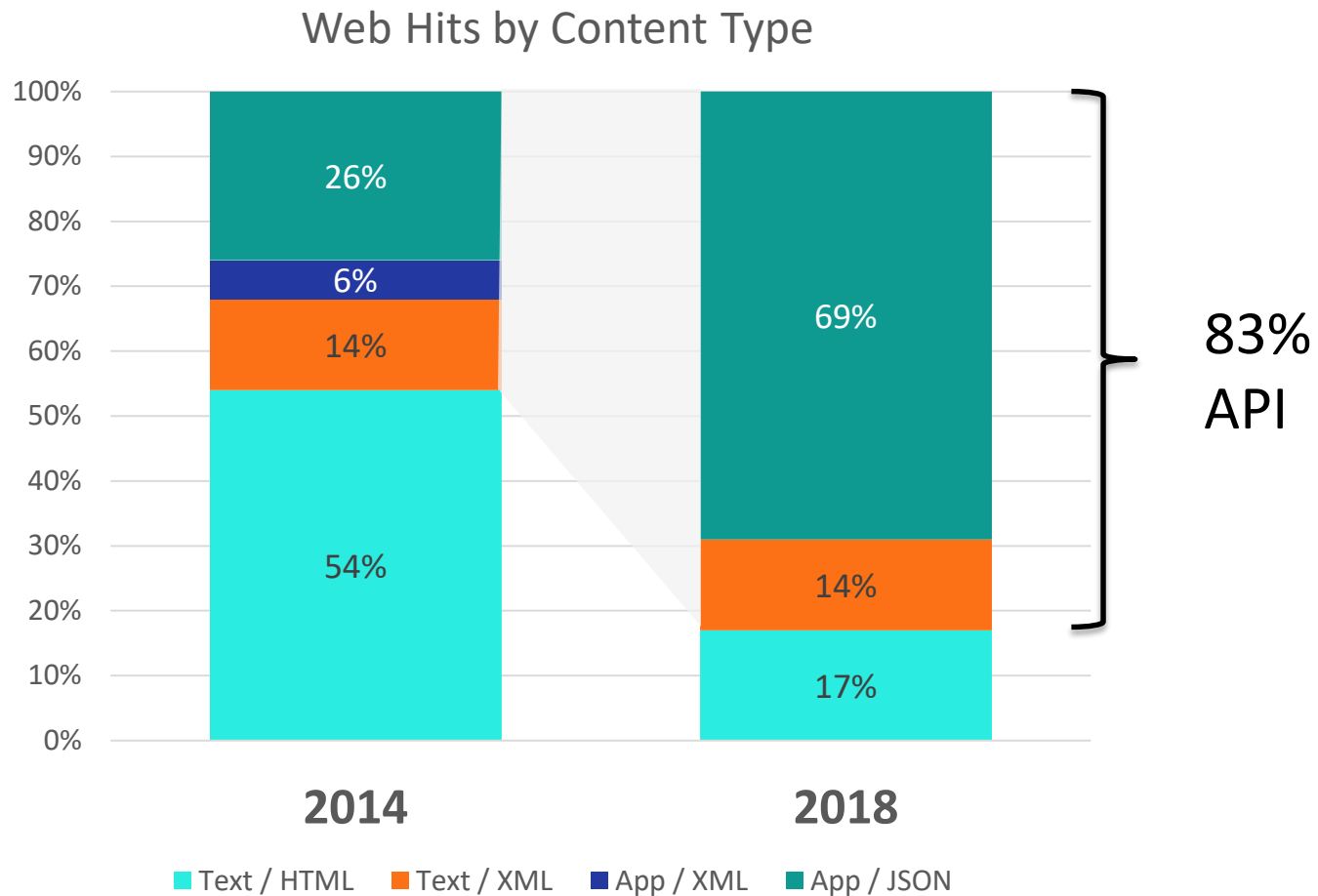
- Quota Enforcement
- AuthN / AuthZ
- Unpredictable load

RSA®Conference2019

**Let's Have a Look at the Internet and
See what we can Observe**



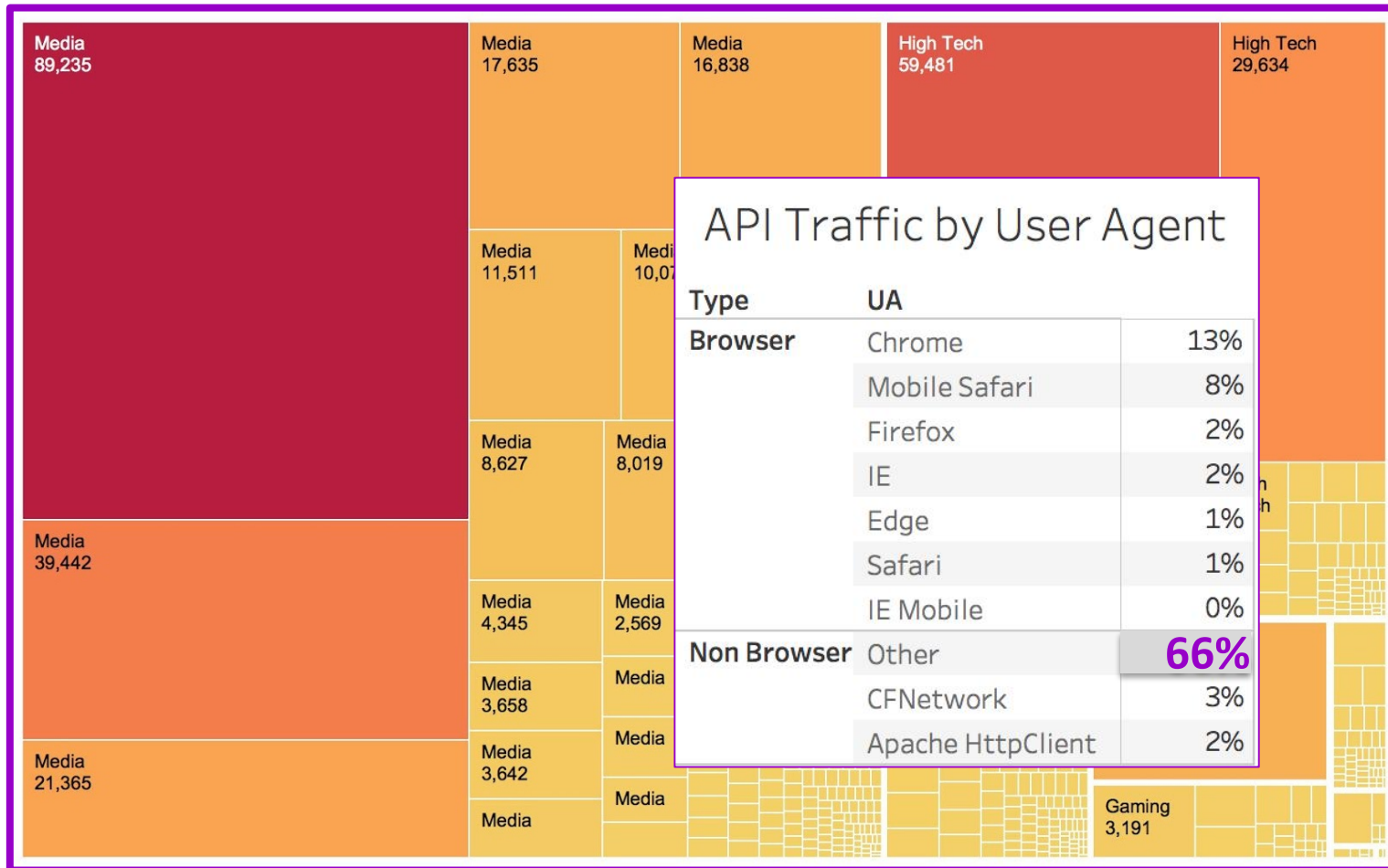
Growth of Web API Use: 2014 through 2018



API calls now dominate overall web hits

Source: Akamai ESSL Network, SOTI Q1 2019

Things On The Internet Make The Majority Of API Calls



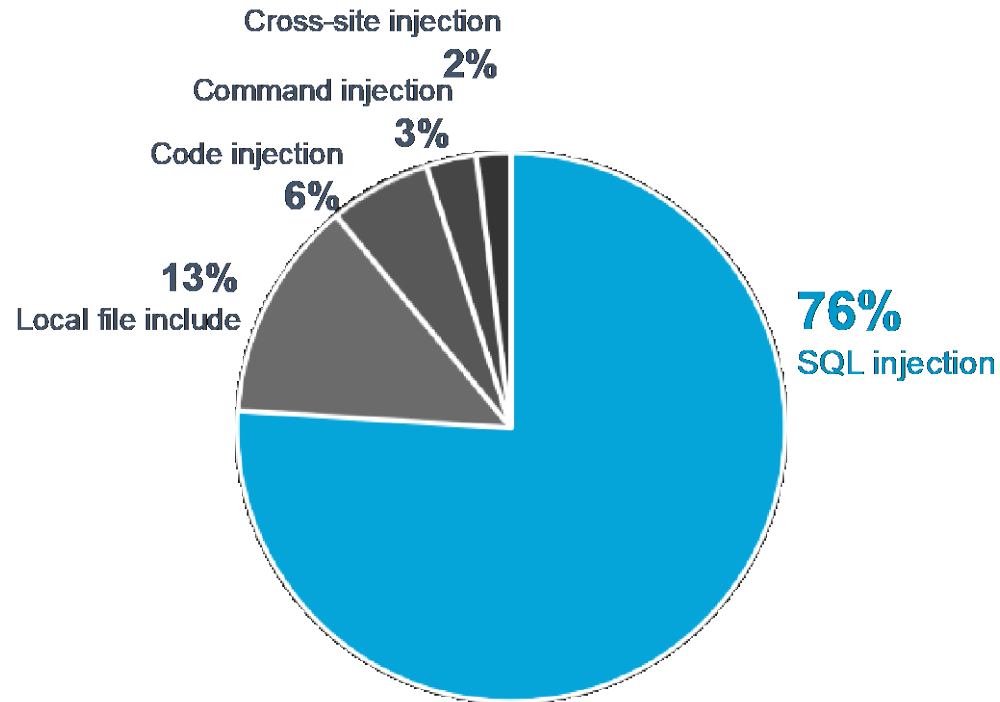
Source: Akamai SOTI Q1 2019

About 1/3rd of Web API calls come from browsers.

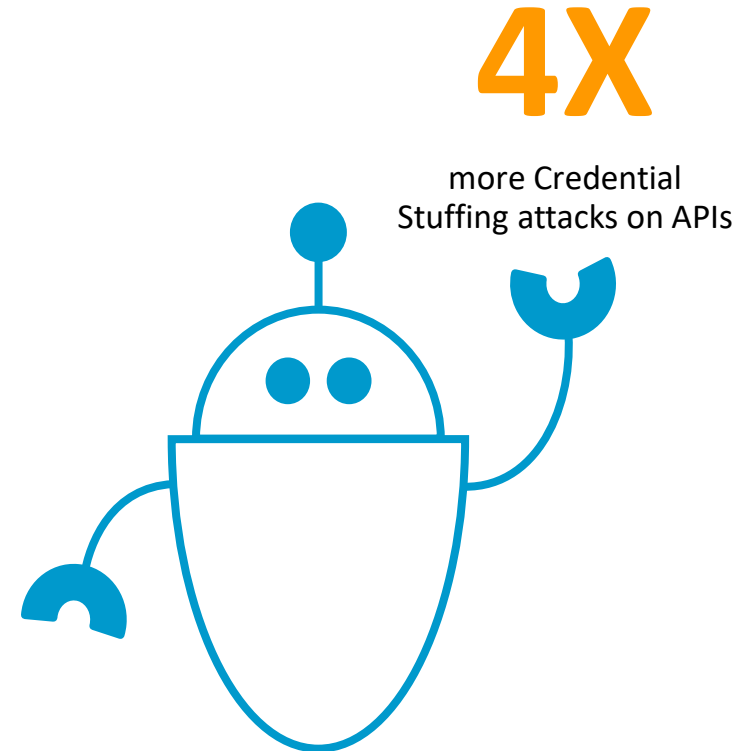
The other 2/3^{rds} come from mobile phones, gaming consoles, smart TVs, etc...

This is a huge challenge!

Web APIs Are A Primary Target For Attackers Today



Web sites & Web APIs share the same (old) attack vectors – but APIs are often unprotected



APIs are more performant **and** less expensive to attack compared with traditional web forms

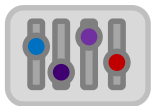
RSA®Conference2019

Let's Examine Some Real World Attacks



Example #1: Distributed Denial of Service (DDoS)

“It’s a little more challenging to identify these kinds of automated, high-bandwidth types of attacks against an API when the whole point is everybody goes faster and gets data faster.” Source: [API Security Trends for 2018](#)



Traditional request-rate-based controls are hard:

Need to validate API keys and track quotas to identify legit traffic



Positive security / Input validation is hard:

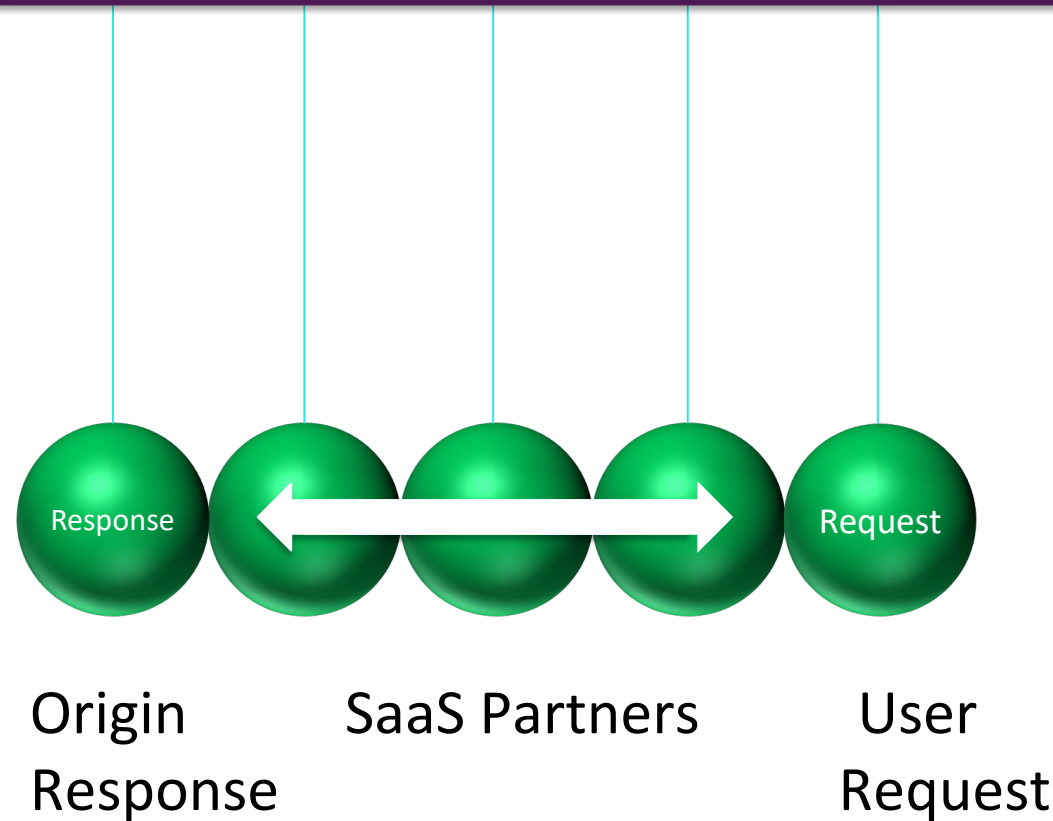
The realm of possibilities for a legit request can be very large



Understanding who and what to trust in a SaaS world is hard:

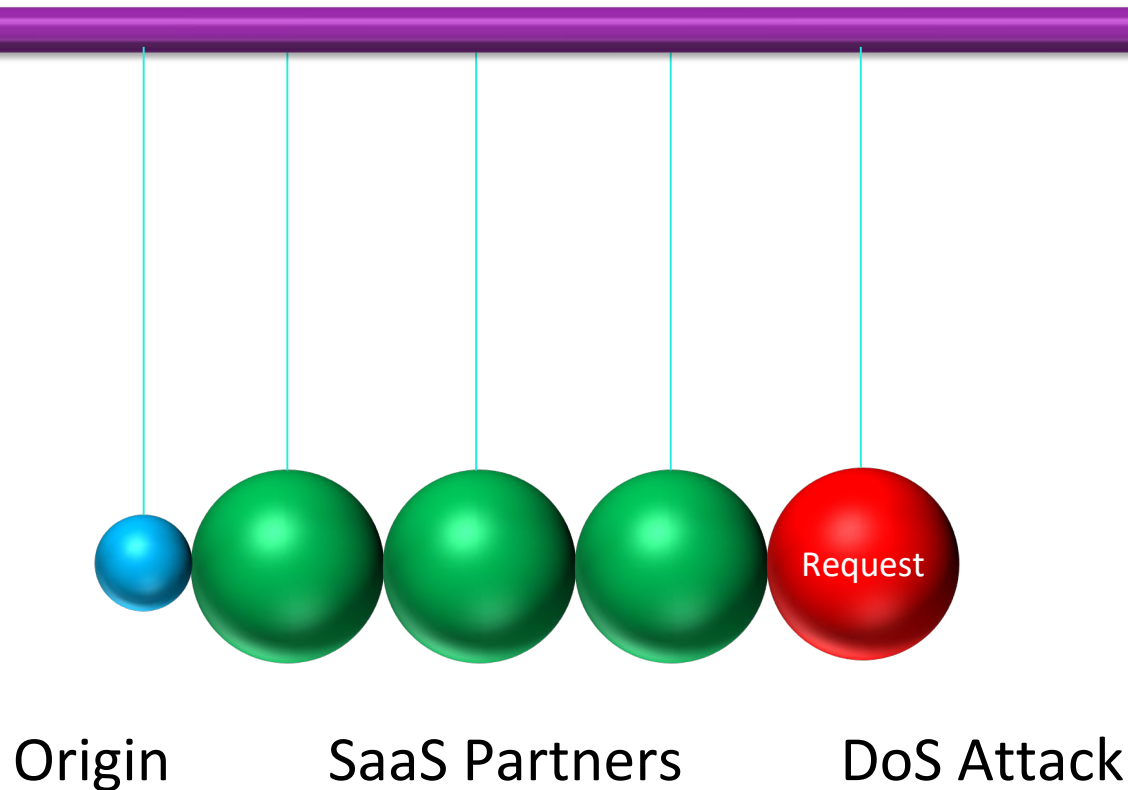
Trusted partners can be abused to cause unexpected harm

API Exposure Can Be Amplified By Business Ecosystems



API requests are within limits, all apps and SaaS partners perform

API Exposure Can Be Amplified By Business Ecosystems

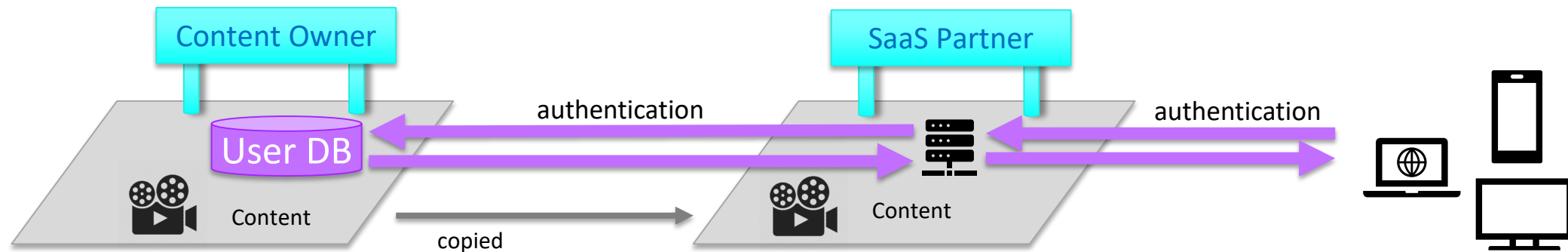


Flood of requests to distributed SaaS partner(s)

May drive a tsunami of requests to centralized origin (that's usually bad)

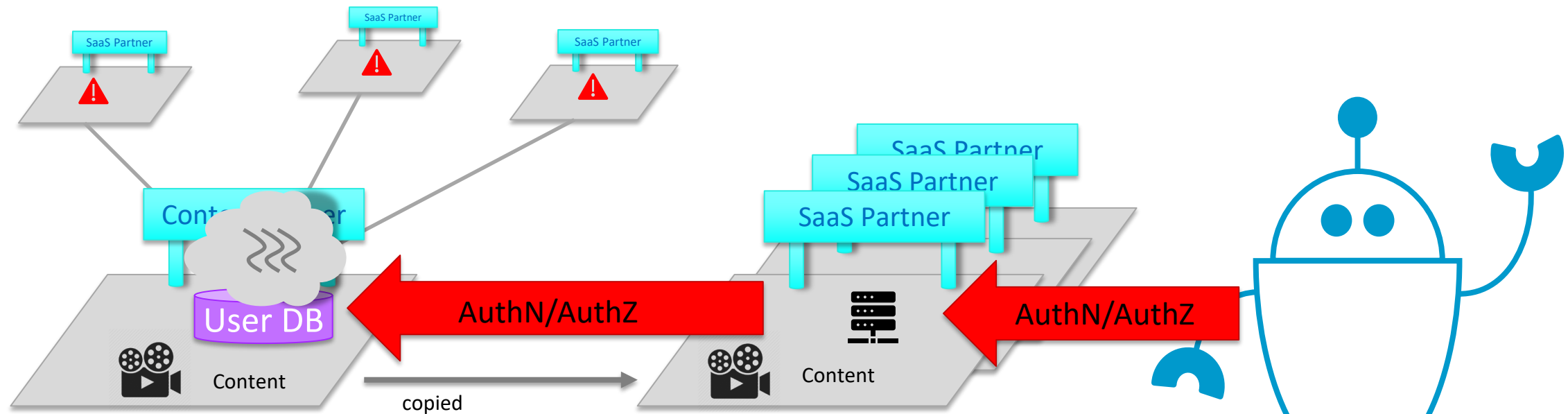
Back To Example #1: Media Distribution Ecosystem DDoS

- Victim #1 is a media company who leverages SaaS partners to adapt and distribute their video content online
- For security purposes, content owner (victim) performs all AuthN and AuthZ centrally in their own systems / data center

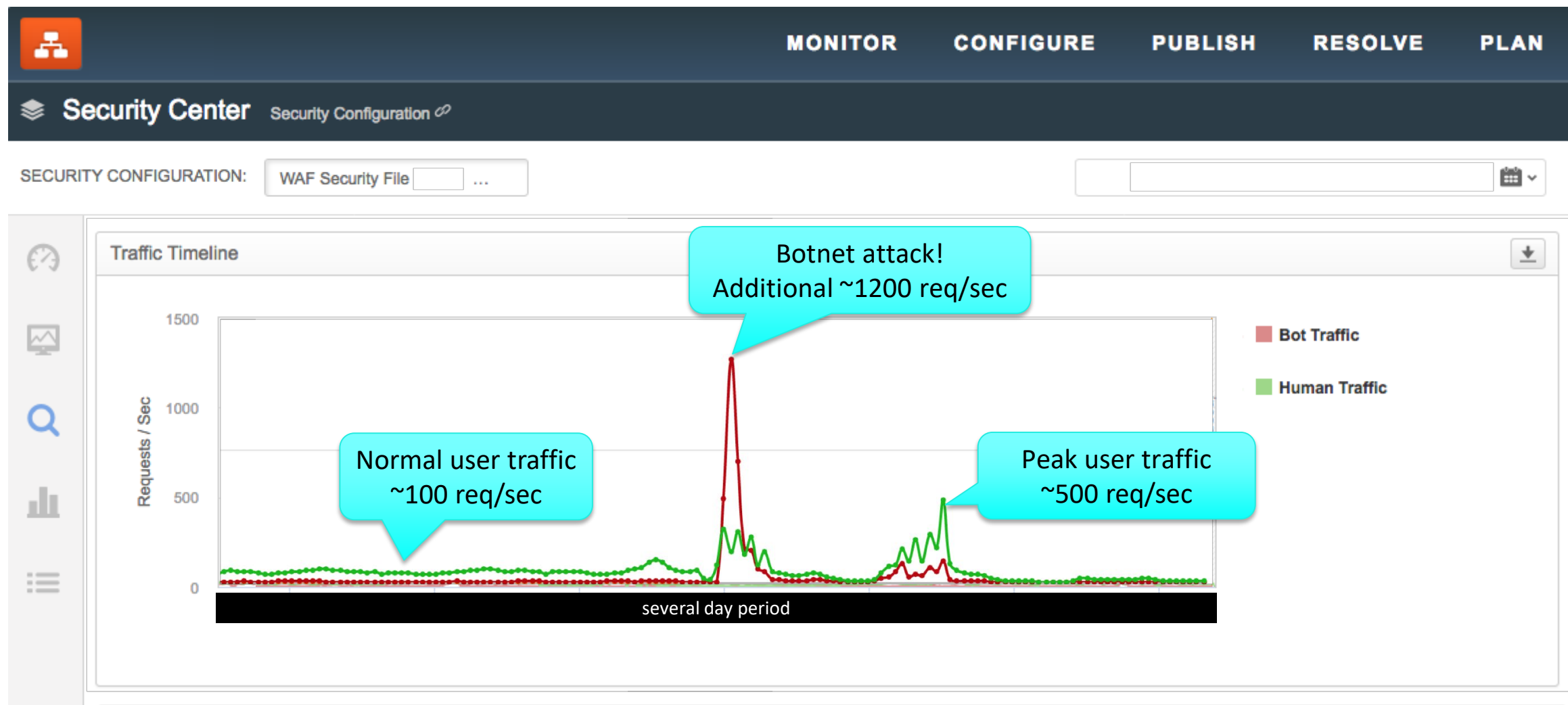


Example #1: A Request and Retry Storm

- Attacker uses botnet to request videos from content distributors
- AuthN and AuthZ requests and retries overwhelm the origin
- All end user access to content (regardless of distributor) is blocked

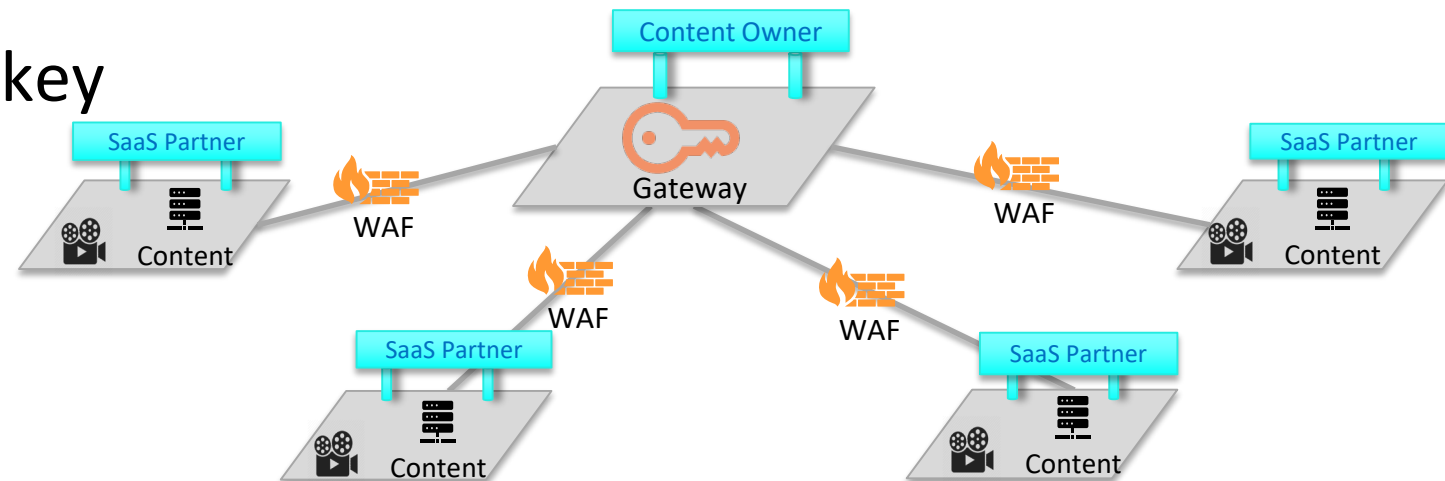


Example #1: Request Volume Experienced At Origin



Example #1: Lessons Learned

- Distribution and scaling strategies fail at bottlenecks
- Attackers will exploit architectural weaknesses to cause havoc
- Mitigations applied:
 - WAF: Stop malformed reqs
 - Bot Management: Block bots
 - WAF: Rate controls by API key
 - API Gateway: API key assignment and quota enforcement



Credential Stuffing On An API Looks A Lot Like DDoS

With clients that don't render JavaScript a lot of the typical credential stuffing defenses just don't work.

Aggressive botnets will overwhelm origin with login requests



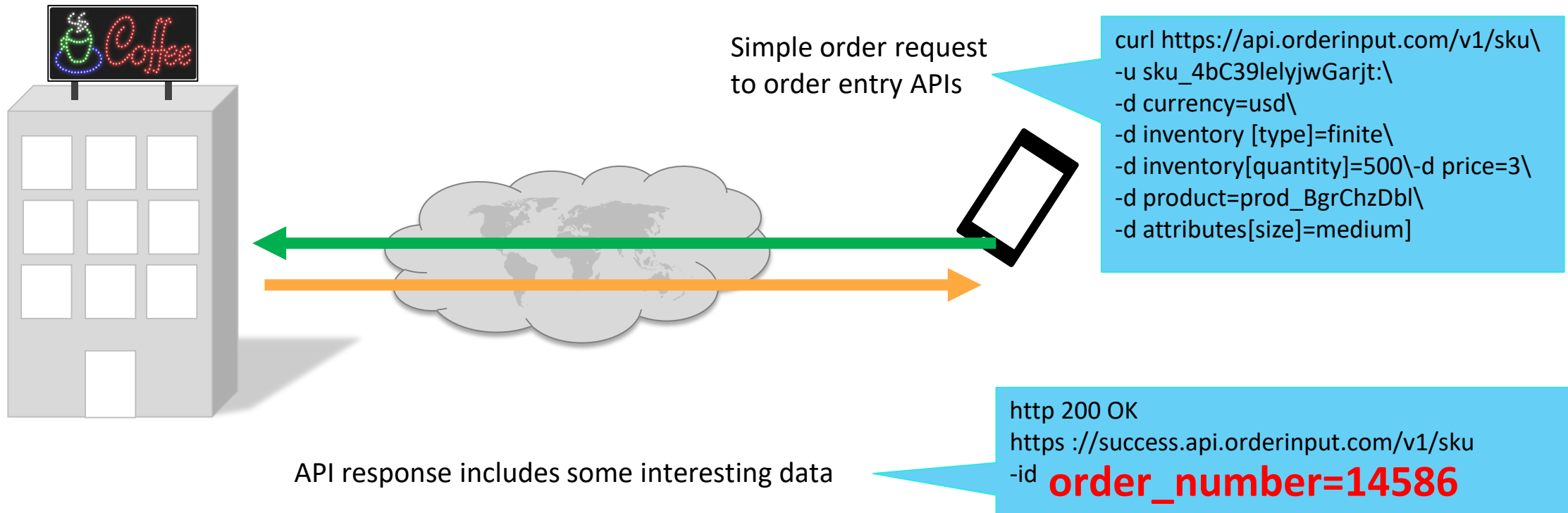
28 billion
credential stuffing attempts
in 8 month

(Observed on Akamai Intelligent Edge Platform, 2018)

Source: Akamai SOTI 1Q19

Example #2: What's In Your API Response?

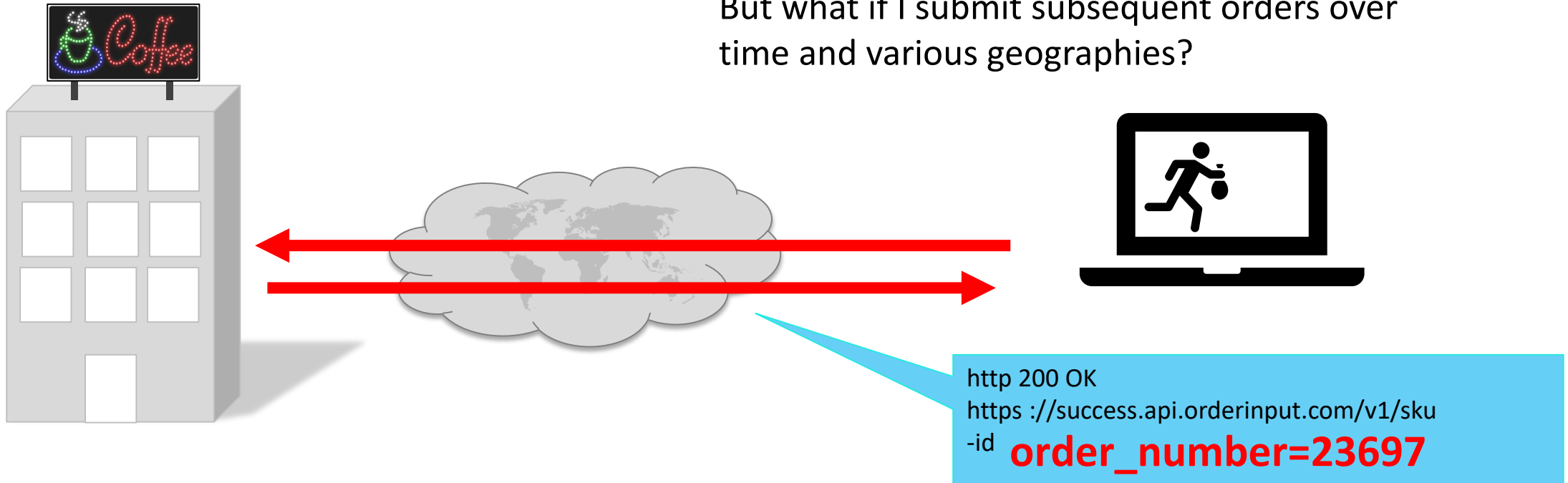
Developers often make assumptions that systems will be used as intended.....*"Only my mobile app will call my API"*



Example #2: What's In Your API Response?

It is rare for developers to consider attack scenarios, especially non-traditional ones.....*"Sequential order numbers makes sense"*

But what if I submit subsequent orders over time and various geographies?



Example #2: But Why?

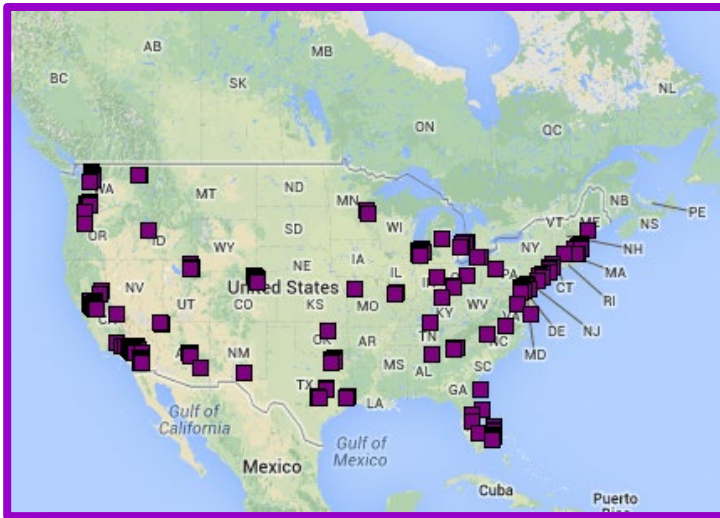
Honestly - We don't know. Same store sales data?



Competition?
Investor?

Example #2: Lessons Learned

- API responses can contain valuable information
- Restrict access to Web APIs to authorized apps only
- Mitigations applied:
 - Order number randomization
 - Mobile app authentication

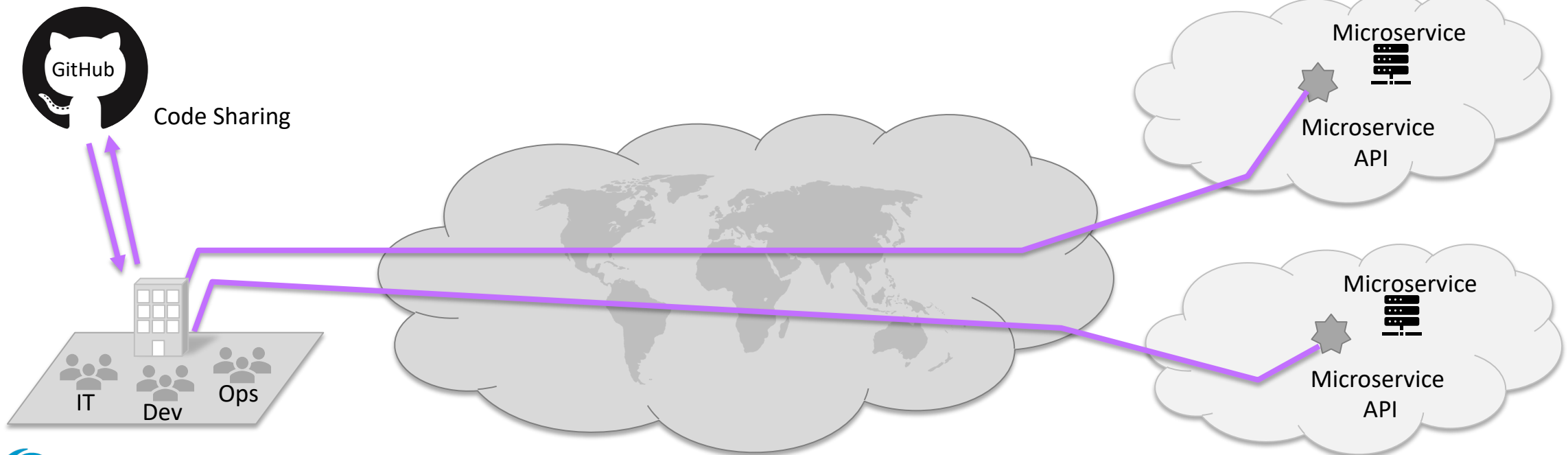


Example #3: You Put Your What? Where!?!

CI/CD processes have developers moving faster than ever

Developers use GitHub to share example code with others

Sensitive data can be accidentally disclosed



If You've Got Web APIs, I Can Find Them

Fierce is a domain discovery tool

Step 1

Scanning for
typical hostnames

Hostnames

auth.
api.
developer.
download.



IP Addresses

xxx.xxx.xxx.xxx
xxx.xxx.xxx.xxx
xxx.xxx.xxx.xxx
xxx.xxx.xxx.xxx



Step 2

Reverse lookup
+/- 10 IP Addresses



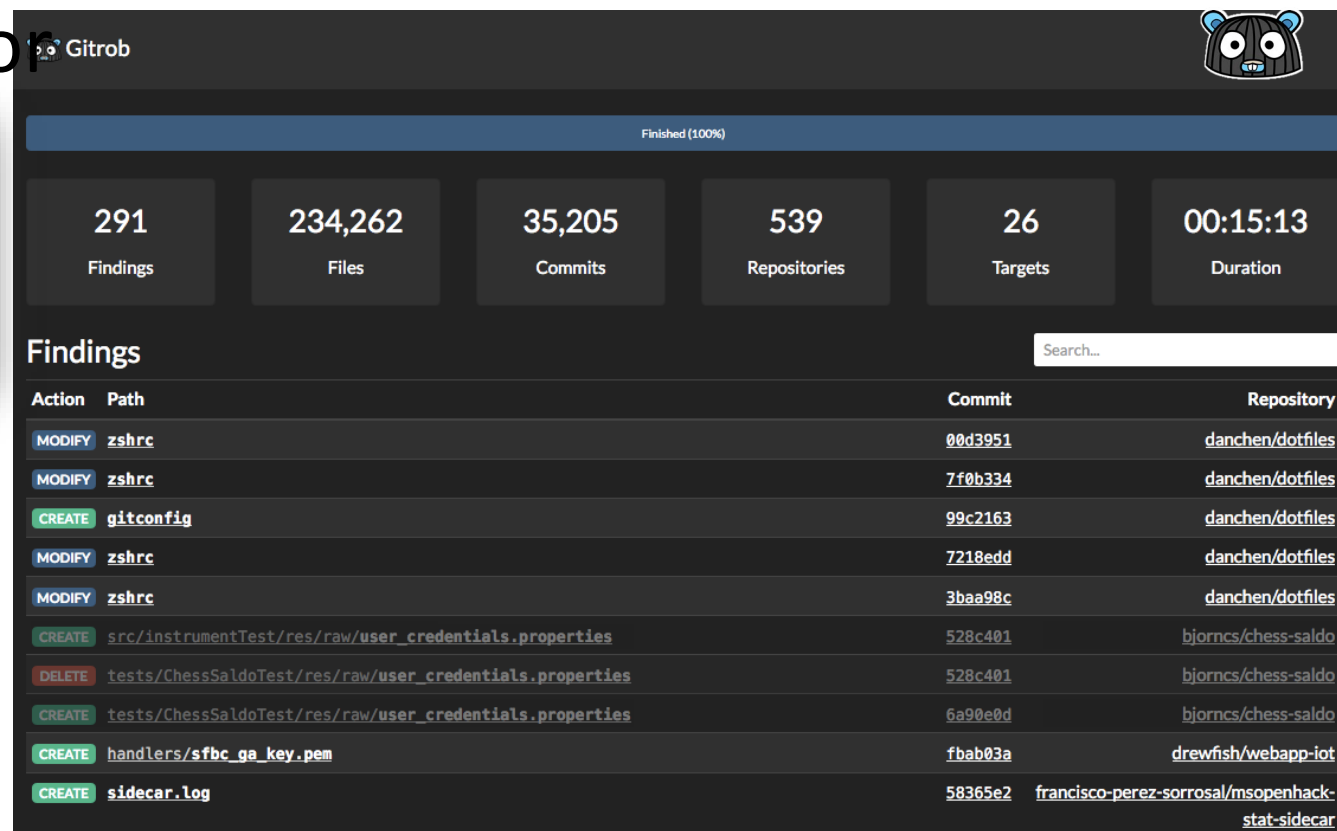
If You've Got Sensitive Data (On GitHub) I Can Find It

Gitrob Sensitive Data Search for Github



Search by organization

Flag interesting files:
Private Keys, usernames,
emails, internal system info,
etc...



The screenshot shows the Gitrob web application interface. At the top, there's a progress bar indicating 'Finished (100%)'. Below this, a summary dashboard displays six key metrics: 291 Findings, 234,262 Files, 35,205 Commits, 539 Repositories, 26 Targets, and a duration of 00:15:13. A 'Findings' section follows, featuring a search bar and a table of results. The table has four columns: Action, Path, Commit, and Repository. The findings list various sensitive files like 'zshrc', 'gitconfig', and 'user_credentials.properties' across different repositories, with actions ranging from 'MODIFY' to 'CREATE'.

Action	Path	Commit	Repository
MODIFY	zshrc	00d3951	danchen/dotfiles
MODIFY	zshrc	7f0b334	danchen/dotfiles
CREATE	gitconfig	99c2163	danchen/dotfiles
MODIFY	zshrc	7218edd	danchen/dotfiles
MODIFY	zshrc	3baa98c	danchen/dotfiles
CREATE	src/instrumentTest/res/raw/user_credentials.properties	528c401	bjorncs/chess-saldo
DELETE	tests/ChessSaldoTest/res/raw/user_credentials.properties	528c401	bjorncs/chess-saldo
CREATE	tests/ChessSaldoTest/res/raw/user_credentials.properties	6a90e0d	bjorncs/chess-saldo
CREATE	handlers/sfbc_ga_key.pem	fba03a	drewfish/webapp-iot
CREATE	sidecar.log	58365e2	francisco-perez-sorrosal/msopenhack-stat-sidecar

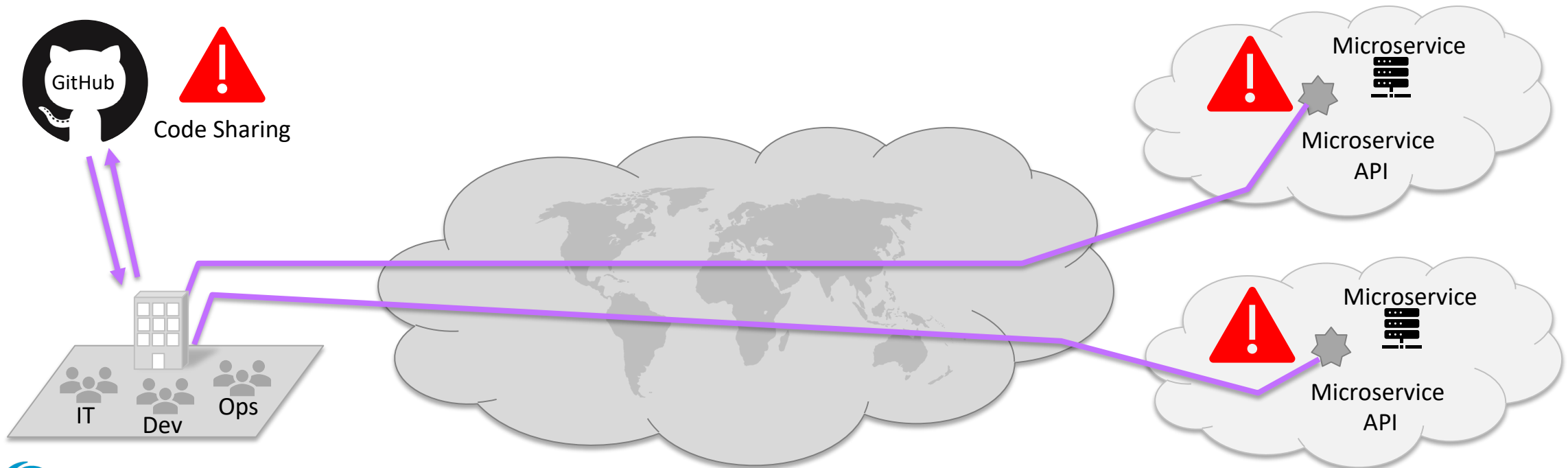
Source: <https://michenriksen.com/blog/gitrob-putting-the-open-source-in-osint/>

Example #3: You Put Your What? Where!?!

Sample code posted on Git

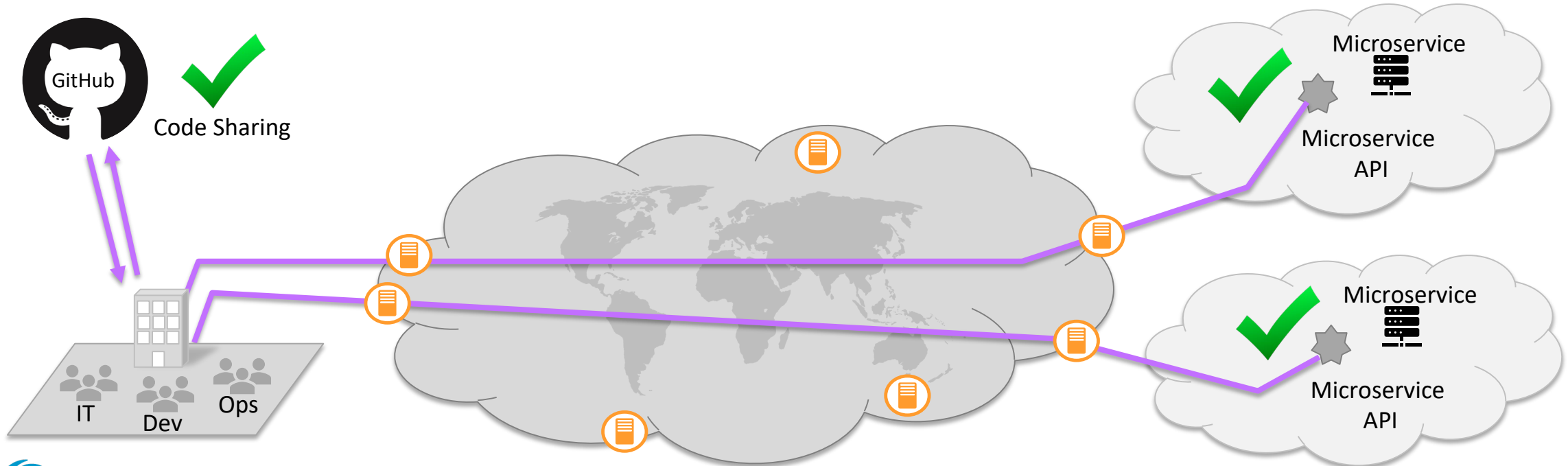
Sensitive API Keys were included in the sample

Attackers leveraged the API Keys to gain unauthorized access



Example #3: Lessons Learned

- Careful code sharing
- API Inspection & Validation
- Mitigations Applied:
 - API Gateway: Dynamically assign and easily revoke API Keys



Develop An API Protection Plan Today

Next week you should:

- Assess your API ecosystem and identify potential security risks

In the first three months following this presentation you should:

- Understand who is accessing your APIs from where and how
- Define appropriate API security measures

Within six months you should:

- Select a security solution which allows proactive API protection tailored to your organization's needs
- Drive an implementation project to protect all public and private APIs

RSA®Conference2019

Thank you !

