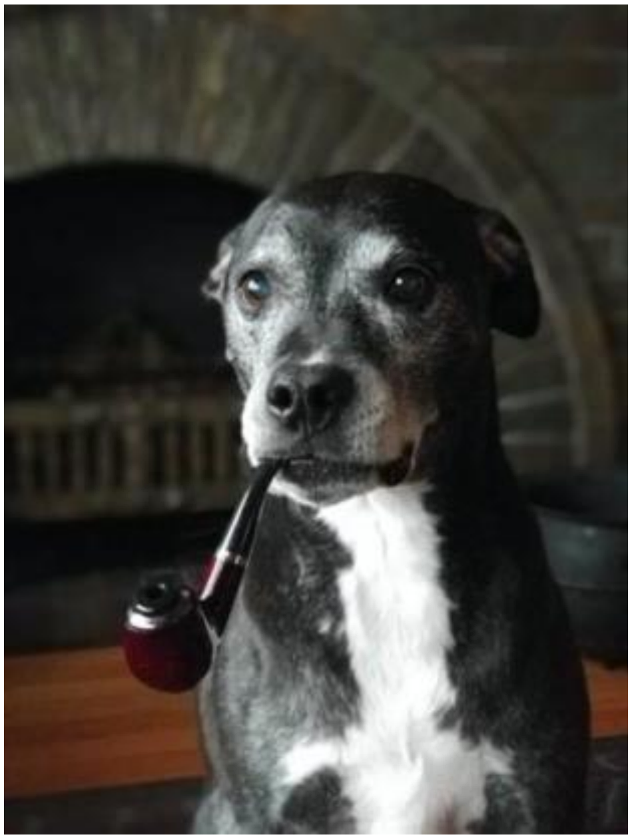
Hacking and Defending APIs 1.1

#whoami

Robert Wagner

@mr_minion

- 15+ years in Security
- Hak4Kidz Co-Founder
- ISSA-Chicago Board
- Dogs, motorcycles & scuba
- Honorary wolf pup dad
- Field CISO @apisec.ai

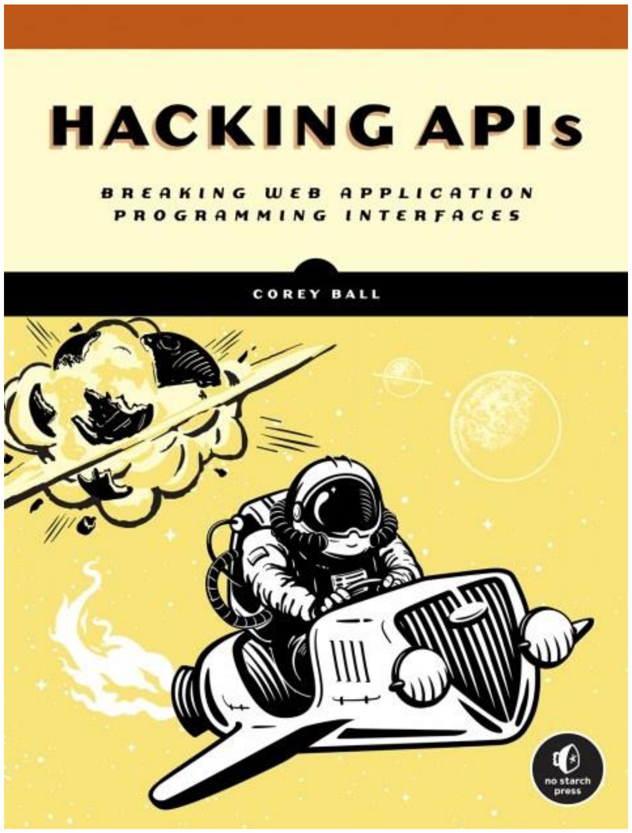


#su hapihacker|whoami

Corey Ball

@hAPI_hacker

- 12+ years in IT & Cyber
- Cyber Security Consulting Manager, Moss Adams
- Author of Hacking APIs (No Starch Press, 2022)
- Apisec.ai Evangelist
- Board game enthusiast


HACKING APIs
BREAKING WEB APPLICATION PROGRAMMING INTERFACES
COREY BALL
no starch press

API Top 10 Overview

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

API1: Broken Object Level Authorization

The most prevalent API vulnerability is BOLA

BOLA vulns are present when an API provider allows consumers access to resources that they should not be authorized to access. Two things are key to finding BOLA: resource ID and a lack of security controls.

Request

GET /api/resource/1

GET /user/account/find?user_id=15

POST /company/account/Apple/balance

POST /admin/pwreset/account/9001

Attack

GET /api/resource/5

GET /user/account/find?user_id=10

POST /company/account/Google/balance

POST /admin/pwreset/account/1337

API1: Testing and Defense

Create an access policy! (if you haven't already)

Design access the same way a Domain Admin would – put users into access groups, and assign permissions to the groups – for EVERY Attribute

Require ALL access be based on authorized requests to an ATTRIBUTE, and DENY everything else.

Testing isn't "hard". You're typically just replacing one value for another. But it can be time consuming, when you have 1000s of endpoints

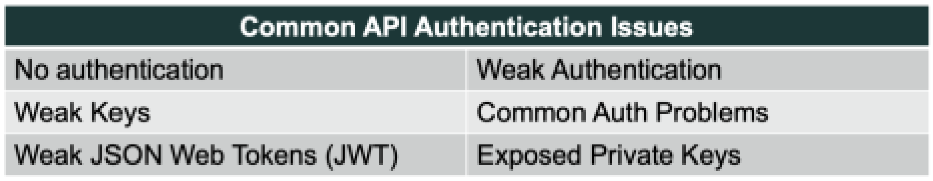
Use OWASP ASVS as a guide: https://github.com/OWASP/ASVS/blob/master/4.0/en/0x12-V4-Access-Control.md

And NIST 800-162 http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf

API2: Broken Authentication

Broken Authentication is a catch-all for all API authentication vulnerabilities.

API authentication is only unique, in the fact that it takes place beyond the GUI. Somehow, it is often overlooked and lacks security.



Common API Authentication Issues	
No authentication	Weak Authentication
Weak Keys	Common Auth Problems
Weak JSON Web Tokens (JWT)	Exposed Private Keys

API2: Mitigations

- This vector is all about attackers abusing your authentication flows. Credential stuffing works really well here.

- Get MFA already!!
- Just like encryption, don't roll your own Auth! This work has already been done for you.
- Remember that your "I Forgot My Password" mechanism can be abused just as easily.
- OAuth and API keys will not prevent these attacks.
- Brute force attack your own systems!

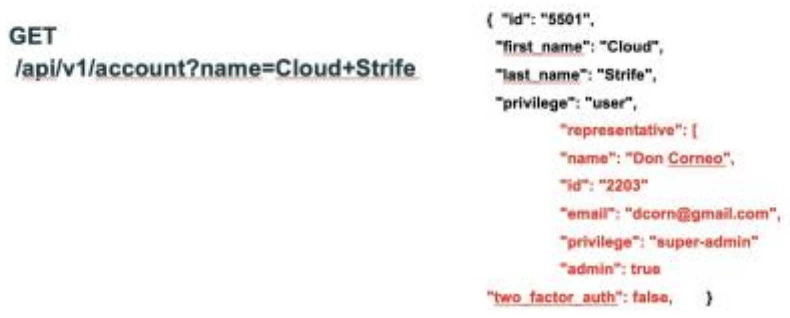
API3: Excessive Data Exposure

Excessive data exposure occurs when a consumer makes an API request for data and the provider responds with more information than requested

Excessive data exposure typically takes place because the API developers are expecting/trusting their consumers to parse out the data.

This vulnerability is the equivalent of asking someone for their name and they start telling you their DOB, home address, email address, SSN, and whether they use MFA

Example:

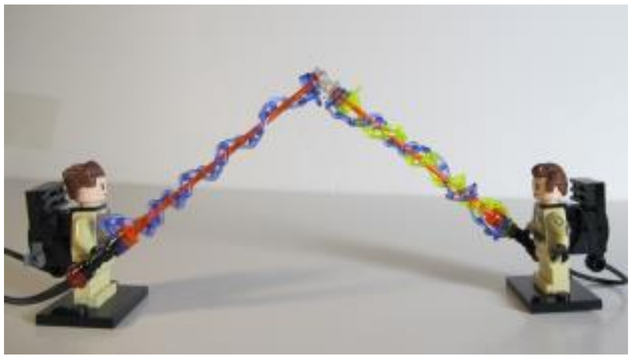


```
GET
/api/v1/account?name=Cloud+Strife
```

```
{ "id": "5501",
  "first_name": "Cloud",
  "last_name": "Strife",
  "privilege": "user",
          "representative": [
          "name": "Don Corneo",
          "id": "2203"
          "email": "dcorn@gmail.com",
          "privilege": "super-admin"
          "admin": true
"two_factor_auth": false,     }
```

API3: Watch Those Streams

- *OWASP recommends not relying on clients to filter data --*
  - *Clients should never be allowed to enforce ANY security policy. Ever!*
- *Check responses for things like:*
  - *Sequential IDs*
  - *More data than the user is authorized for (see API:1)*
- *If your API returns PII, you'll need to classify that data and apply policy to it.*
- *Apply concepts of least privilege, just like all other data access.*
- *Check your freakin' logs!*



API4: Lack of Resources & Rate Limiting

The API is not sufficiently protected against excessive amounts of requests or payload sizes.

Rate limiting is how some organizations monetize their APIs, bypassing this restriction can have serious financial impacts.

If the affected infrastructure does not have enough resources then a certain amount of requests will lead to a denial of service.

The correct rate limit is really up to the API provider.

Request

POST /admin/pwreset/account/9001

Attack

POST /admin/pwreset/AccoUnt/9001

POST /admin/pwreset/account/9001?

POST /admin/pwreset/AccoUnt/9001%00

API4: Defensive Throttling

Reach out to your SREs; they are your friends!

There are multiple ways to throttle requests, responses, and the resources allocated

Even Docker has built in limiting functions.

Logs, logs, logs!

Standard deviation is easy to compute in all the popular indexing tools

Checkout SciPy or R-project if you want to get really fancy

API5: Broken Function Level Authorization

Like BOLA, except instead of an authorization problem with resources it is an authorization problem for privileged actions.

Valid Request

PUT /api/v1/account/update

Token:UserA-Token

--snip--

{"username": "Ash","address": "123 C St","city": "Pallet Town""region": "Kanto"}

Attack

DELETE /api/v1/account/**UserB**

Token:UserA-Token

--snip--

Response

HTTP/1.1 200 OK

API:5 from goto import goto, API1

Just like with BOLA (API1) you need an access policy designed to stop imposters

Access needs to be controlled down to the attribute level, based on the users session. DENY everything else

Enforced only on the server side

Test to make sure your policy is working

Logs FTW again! Every indexing tool has some sort of "Rare" value call. Use it to surface rare methods to an endpoint. You won't prevent the attack, but at least you'll have visibility.

API6: Mass Assignment

Mass Assignment is the binding of consumer input to data models without filtering properties. In other words, a consumer can pass addition parameters to an application that are automatically added to code or other internal objects

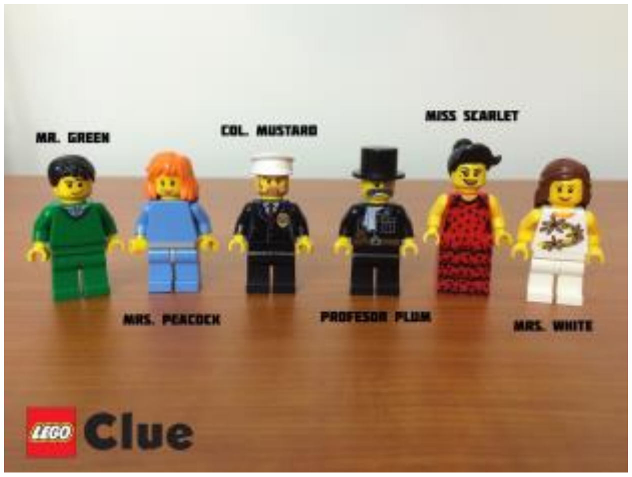
- Can take place if developers depend on obscurity as a security control.
- If documentation is shared between public users, partners, and administrators then mass assignment weaknesses will be even easier to discover/exploit

POST /api/v1/register

--snip--

{"username":"hAPI_hacker","email":"hapi@hacker.com",**"admin": true,"admin":1,"isadmin": true,"role":"admin","role":"administrator","user_priv": "admin"**,"password":"Password1!"}

API6: Looking for Clues



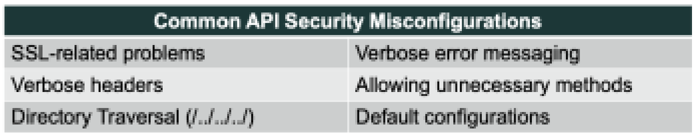
Your attacker is for any clue about properties or fields they can manipulate

You have to either disable mass assignment (yikes), or

Restrict and define the exact parameters the endpoint will accept (yay)

API7: Security Misconfiguration

Security Misconfiguration is a catch-all for generic security issues like:



Common API Security Misconfigurations	
SSL-related problems	Verbose error messaging
Verbose headers	Allowing unnecessary methods
Directory Traversal (/../../../)	Default configurations

API7: Security's Murphy's Law

Literally, anything that can go wrong, will go wrong

Forgot to patch, update, harden? Oops!

Fortunately(?) Infosec and IT have been dealing with this for decades.

Change management and compliance frameworks already cover almost all these issues.

Enforce them.

API8: Injection

Injection flaws include: SQL Injection, NoSQL Injection, XXS, and system command injection.

```
PUT /api/user/edit_info HTTP/1.1
Host: 192.168.195.132:8090
Content-Type: application/json
x-access-token: eyJhbGciOiJIUzI1NiIsInR5cCI...
--snip--

{
    "user": "§<email>§",
    "pass": "§<string>§",
    "id": "§<number>§",
    "name": "§<string>§",
    "is_admin": "§<boolean>§",
    "account_balance": "§<number>§"
}
```

API8: Still the Bane of Our Existence

Seriously, is this a joke? It's been over 20 years now!

The solutions are well documented. Sanitize. Validate. Filter.

Test early, test often. It's too easy for these flaws to get missed.

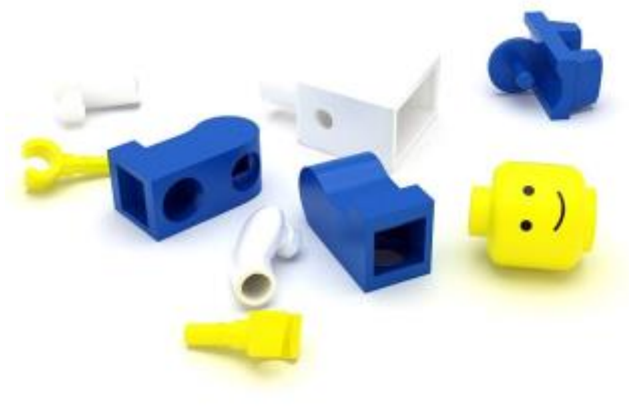
API9: Improper Assets Management

Improper Assets Management takes place when an API provider exposes a retired or in development API.

A common practice for APIs is to include versioning within the URL path: /v1, /v2, /v3, /test, /uat etc.

If an API provider allows access to an unsupported or incomplete version of the API then they run the risk of having additional weaknesses within the affected version.

Weaknesses can often be found in patch notes and all over Github. If an API provider had a weakness in a previous version, then the access to that version should be removed across the entire API.



API9: Where are those Crown Jewels, again??

Document your APIs. Yes, all of them.

Enumerate your APIs. You shouldn't *have* to use Kiterunner, but you probably will. See Katie!

Implement a robust change control and asset management process

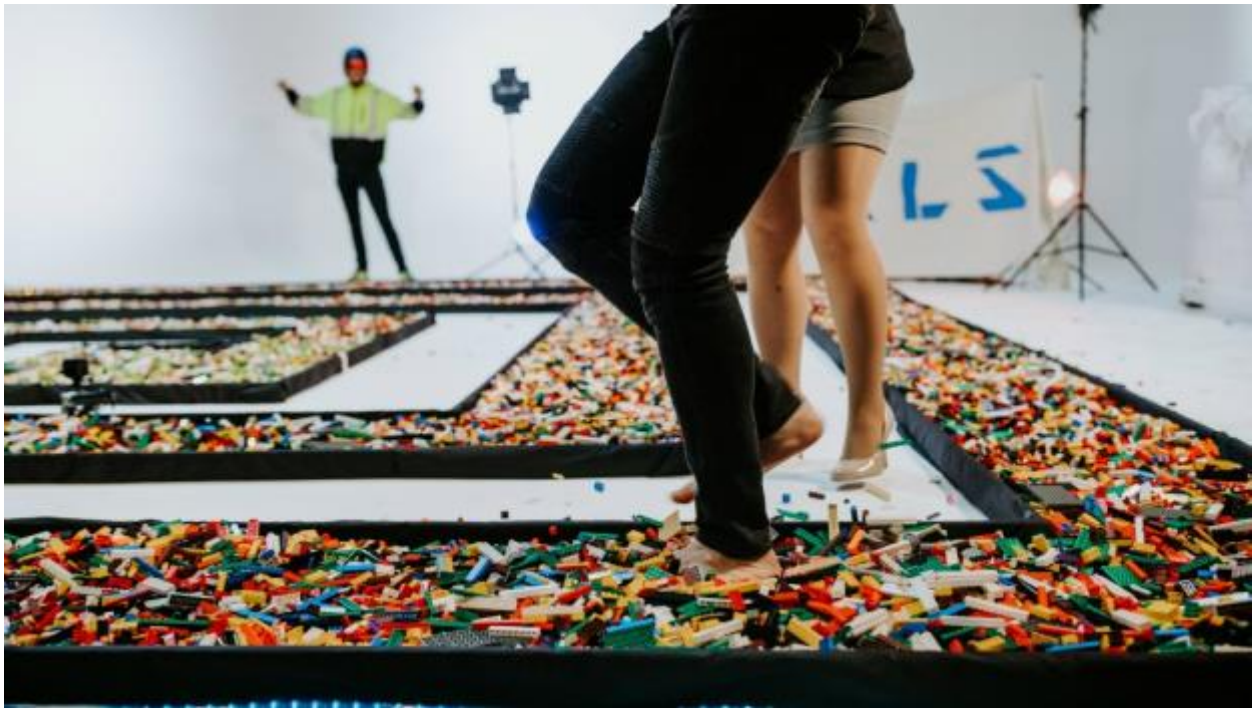
CI/CD tools have made it so much easier to automate this

API10: Insufficient Logging & Monitoring

Lack of logging, monitoring, and alerting will allow attacks to run rampant, go unnoticed, and there will be little to no chance of attribution.

API10: Log Your Apps or I Will...

Make you walk over all the legos used in this presentation. Barefoot.



The Danger of False Negatives

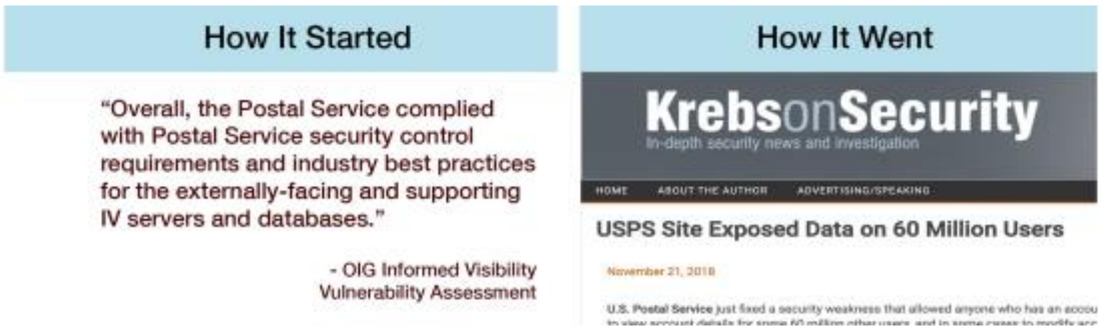
- *Essentially a false negative is when vulnerabilities are present, but are not detected.*
- *Vulnerability testing that results in false negative findings provides a risky sense of being secure.*
- *APIs that are not tested with the right tools and techniques often result in false negatives. False negatives leave API providers the idea that they are safe and secure, when they are not.*

False Positive
You're pregnant!
False Negative
You're not pregnant!

Case Study: USPS Informed Visibility (2018)

- *One of the features that drove the innovation behind the USPS Informed Visibility program was the API.*

- *The USPS Office of Inspector General performed a vulnerability assessment and the report was released on October 12, 2018. This report is public and can be found here https://www.uspsoig.gov/document/informed-visibility-vulnerability-assessment*



How It Started

"Overall, the Postal Service complied with Postal Service security control requirements and industry best practices for the externally-facing and supporting IV servers and databases."

- OIG Informed Visibility Vulnerability Assessment

How It Went

KrebsonSecurity

In-depth security news and investigation

HOME ABOUT THE AUTHOR ADVERTISING/SPEAKING

USPS Site Exposed Data on 60 Million Users

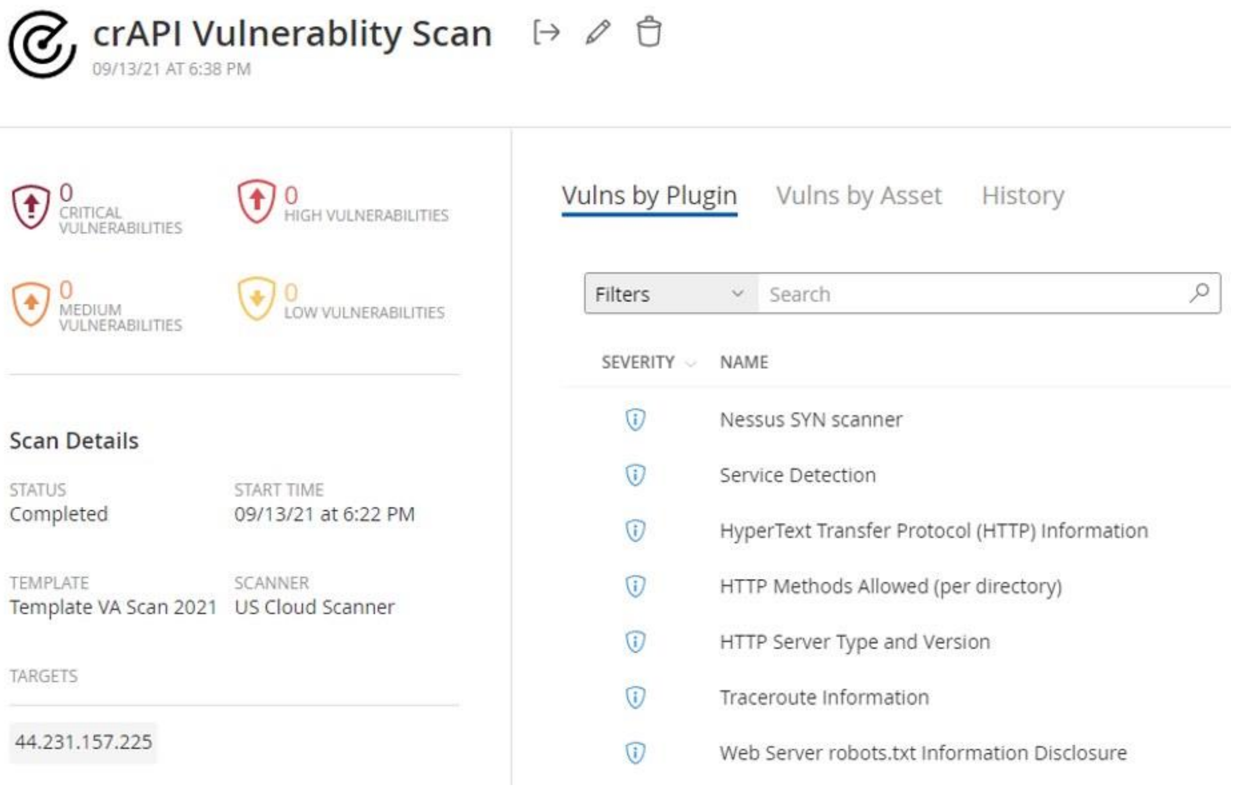
November 21, 2018

U.S. Postal Service just fixed a security weakness that allowed anyone who has an accou
to view account details for some 60 million other users, and in some cases to modify acc

- *The OIG Vulnerability Assessment was released on October 12, 2018. The Krebs article exposing the data exposure was released on November 21, 2018.*
- *The report fails to mention APIs altogether.*
- *This is not to drag the Inspector General through the mud, but instead to emphasize that they were using classic tools and techniques that provided them with false negative results. These results were passed on to the USPS.*

API Vulnerability Management

- *APIs do not fit the mold of generic vulnerability assessments.*
- *If you use the wrong vulnerability assessment tools then you will receive false-negatives.*
- *The OIG stated that they used Nessus and HP WebInspect. (Instead of HP WebInspect: Nikto, OWASP ZAP, Burp Suite Pro)*



crAPI Vulnerablity Scan

09/13/21 AT 6:38 PM

0 CRITICAL VULNERABILITIES

0 HIGH VULNERABILITIES

0 MEDIUM VULNERABILITIES

0 LOW VULNERABILITIES

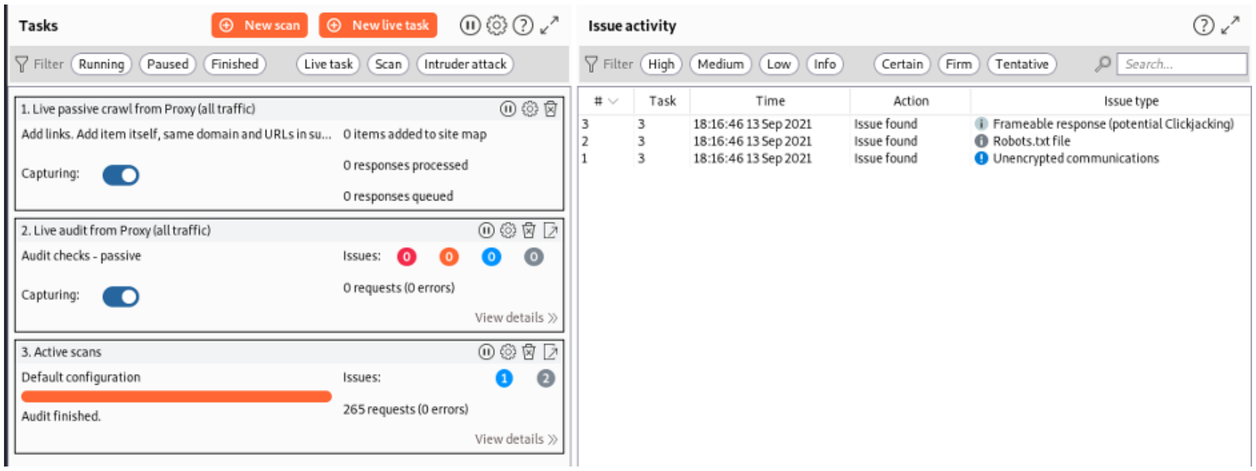
Scan Details

STATUS	START TIME
Completed	09/13/21 at 6:22 PM
TEMPLATE	SCANNER
Template VA Scan 2021	US Cloud Scanner

TARGETS

44.231.157.225

SEVERITY	NAME
ⓘ	Nessus SYN scanner
ⓘ	Service Detection
ⓘ	HyperText Transfer Protocol (HTTP) Information
ⓘ	HTTP Methods Allowed (per directory)
ⓘ	HTTP Server Type and Version
ⓘ	Traceroute Information
ⓘ	Web Server robots.txt Information Disclosure



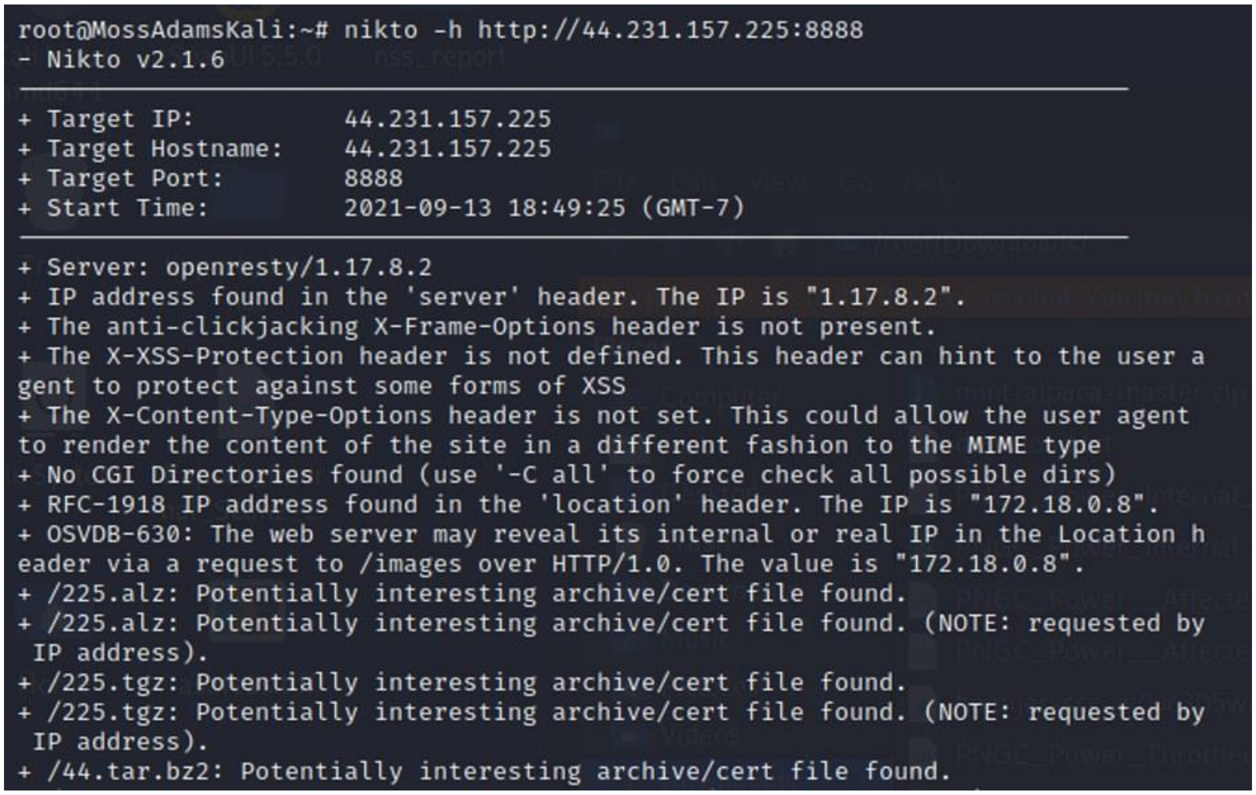
1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domain and URLs in su... 0 items added to site map
0 responses processed
Capturing:
0 responses queued

2. Live audit from Proxy (all traffic)
Audit checks - passive
Issues: 0 0 0 0
0 requests (0 errors)
Capturing:
View details »

3. Active scans
Default configuration
Issues: 1 2
265 requests (0 errors)
Audit finished.
View details »

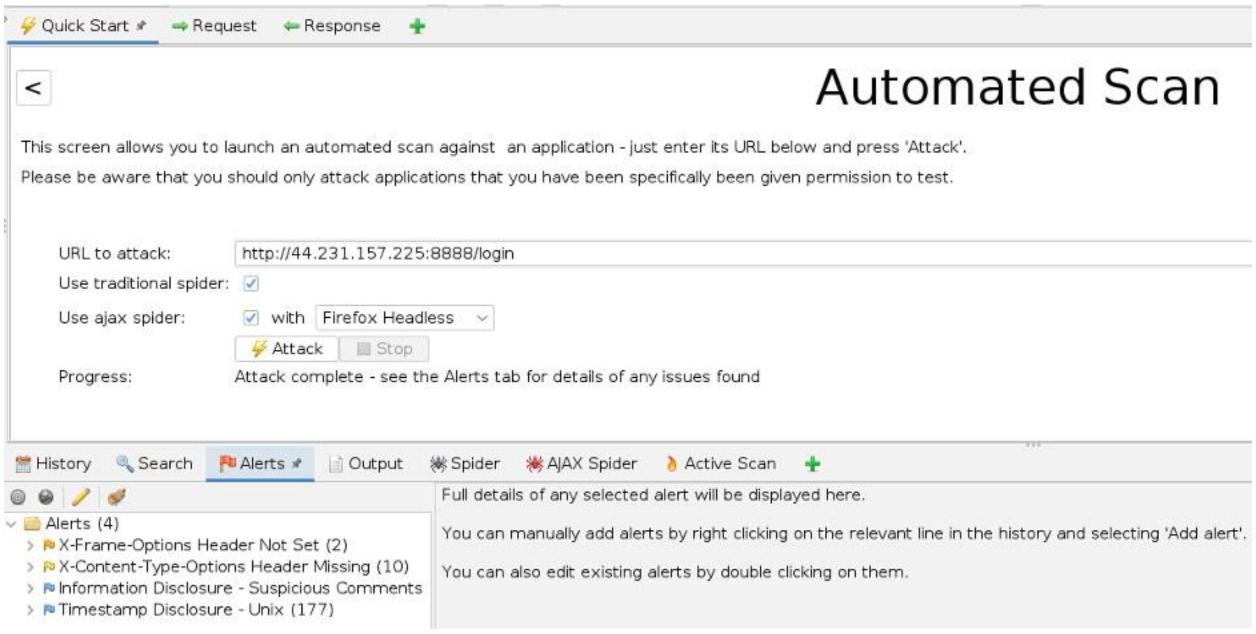
#	Task	Time	Action	Issue type
3	3	18:16:46 13 Sep 2021	Issue found	Frameable response (potential Clickjacking)
2	3	18:16:46 13 Sep 2021	Issue found	Robots.txt file
1	3	18:16:46 13 Sep 2021	Issue found	Unencrypted communications

1 - crAPI Burp Suite Automated Scan



```
root@MossAdamsKali:~# nikto -h http://44.231.157.225:8888
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          44.231.157.225
+ Target Hostname:    44.231.157.225
+ Target Port:        8888
+ Start Time:         2021-09-13 18:49:25 (GMT-7)
---------------------------------------------------------------------------
+ Server: openresty/1.17.8.2
+ IP address found in the 'server' header. The IP is "1.17.8.2".
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user a
gent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent
to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ RFC-1918 IP address found in the 'location' header. The IP is "172.18.0.8".
+ OSVDB-630: The web server may reveal its internal or real IP in the Location h
eader via a request to /images over HTTP/1.0. The value is "172.18.0.8".
+ /225.alz: Potentially interesting archive/cert file found.
+ /225.alz: Potentially interesting archive/cert file found. (NOTE: requested by
 IP address).
+ /225.tgz: Potentially interesting archive/cert file found.
+ /225.tgz: Potentially interesting archive/cert file found. (NOTE: requested by
 IP address).
+ /44.tar.bz2: Potentially interesting archive/cert file found.
```

2 - crAPI Nikto Scan



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://44.231.157.225:8888/login
Use traditional spider: ✓
Use ajax spider: ✓ with Firefox Headless
Attack Stop
Progress: Attack complete - see the Alerts tab for details of any issues found

Alerts (4)
- X-Frame-Options Header Not Set (2)
- X-Content-Type-Options Header Missing (10)
- Information Disclosure - Suspicious Comments
- Timestamp Disclosure - Unix (177)

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

3 - crAPI OWASP ZAP

Value of Good Penetration Testing and Bug Bounty

Penetration testing: Is currently a point in time adversarial test of your web apps and APIs.

- *When the right tools and techniques are used, penetration testing will not only reveal vulnerabilities, but can provide proof-of-concept for what can be achieved when those vulnerabilities are exploited.*

- *Penetration tests can be used to simulate different types of threats.*

- *There is a decent chance that your API's security is being tested whether or not you want it to be, at least with penetration testing you will get a report when we are done.*

Bug Bounty programs: Typically, bug bounties are a continuous test of your API. This is another valuable layer of testing.

- *Bug bounty communities often contain a mix of awesome hackers at various levels of expertise.*
- *A bounty program will often provide your API security testing with a variety of skillsets testing them.*
- *You can lure API hackers to target your APIs with additional incentives ($$$).*

Both of these types of testing offer advantages:

- *Business Logic Testing: Using features of an API against the API provider.*
- *Leverage low level findings for more complex attacks.*
- *Both provide you with an additional layer of assurance by delivering results your organization can use for remediation.*

How to Test Your API Vulnerability Management Program

Securely test your current API vulnerability management program against deliberately vulnerable APIs.

If your current program does not detect any vulnerabilities against these deliberately vulnerable applications then your API vulnerability management program needs improvement!

The most likely problem is that you are not using the correct tools or techniques in order to detect API vulnerabilities.

crAPI

https://github.com/OWASP/crAPI

VAmPI

https://github.com/erev0s/VAmPI

vAPI

https://github.com/roottusk/vapi

Pixi

https://github.com/DevSlop/Pixi

- *When testing APIs for vulnerabilities make sure to:*

  - *Configure tools and use techniques that test the API from the standpoint of an authorized user.*

  - *Do not trust automated scans that show that your API is secure (they are probably lying, configured incorrectly, or just currently incapable).*

- *Add layers to your API security testing by including penetration testing, bug bounty programs, and make sure that you are using the right tools configured appropriately.*

- *Use the OWASP API Security Top 10 (2019) to guide your vulnerability assessment testing.*

API Security Discovery Tips

- If you're concerned about unknown APIs, you can discover API routes using Kiterunner
- If you're not allowed to use Kiterunner to scan, you can still leverage it's "word lists" with a log indexing tool.
  - Download the word lists (.kite files), then convert to text
  - Load into lookup tables in your indexing tool
  - Sniff the wire for content-type: application/json
  - Filter/search using the word lists

API Security Deception Tips

- Deception practices work with APIs too (but more advanced)
- Create deceptive comments/responses with fake links or fake credentials
  - Create code to replace 4xx and 5xx with a fake 200 if more than x number of errors, with new (but unproductive) paths to follow.

Want to Learn More?

Coming soon: Hacking OWASP API 1 with Corey Ball

hAPI Hacking!

rwagner1@gmail.com

@mr_minion

www.linkedin.com/in/robertwagner2

corey.ball@mossadams.com

@hAPI_hacker

www.linkedin.com/in/coreyball