

About me

Trevor Bryant

- Security minded DevOps nerd
- Knight of NIST
- Auditor, Analyst, Engineer, Architect
- Tech policy
- Instructor @DC_TOOOOL
- Conference Organizer / Volunteer

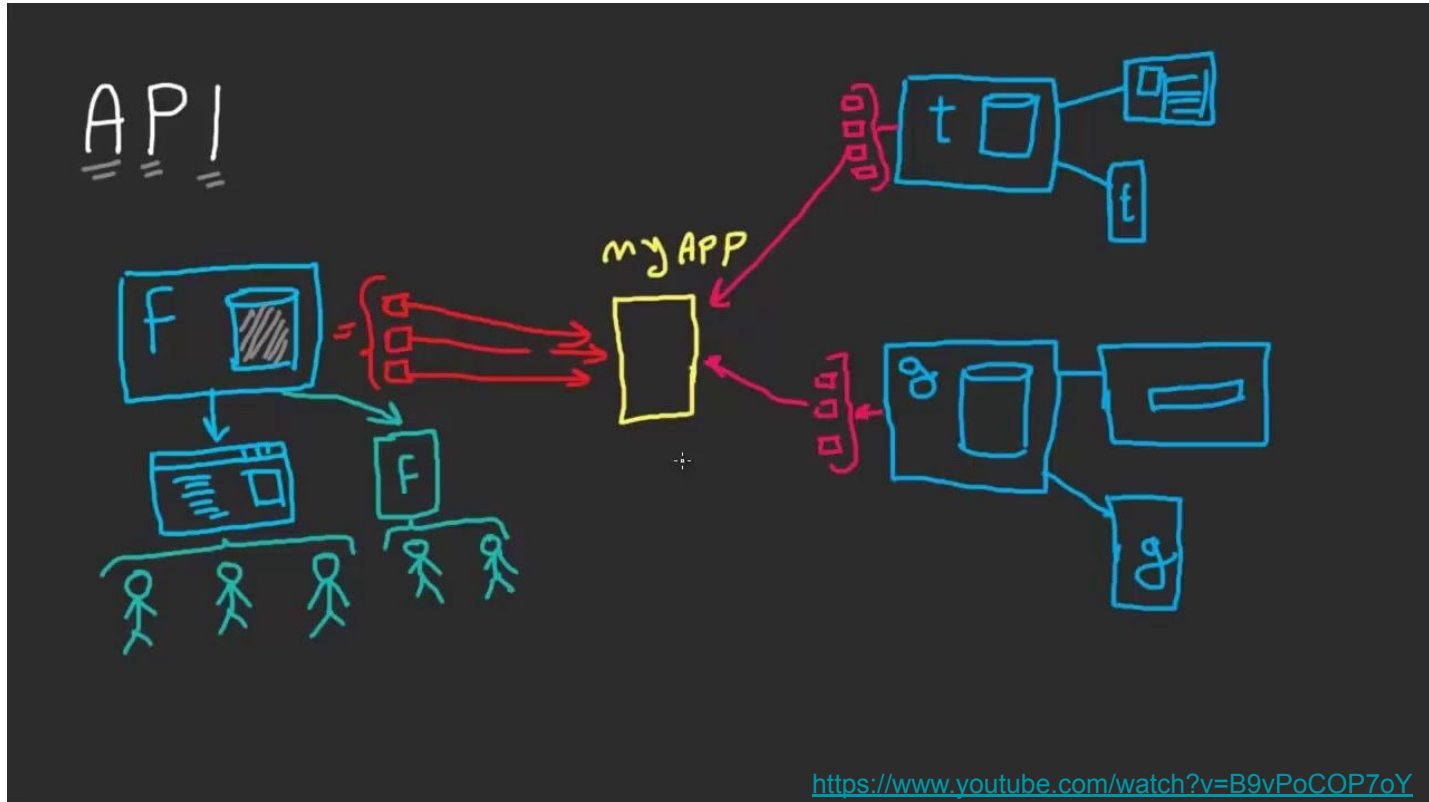
apporima.com



api_security

I know nothing, for I am Jon Snow

Google Image Search

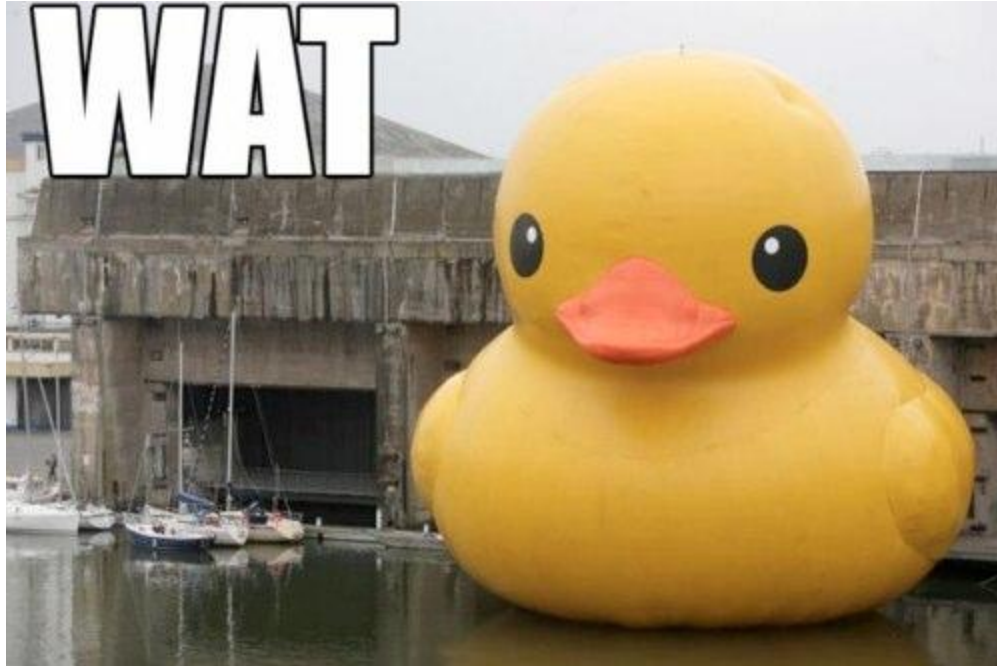


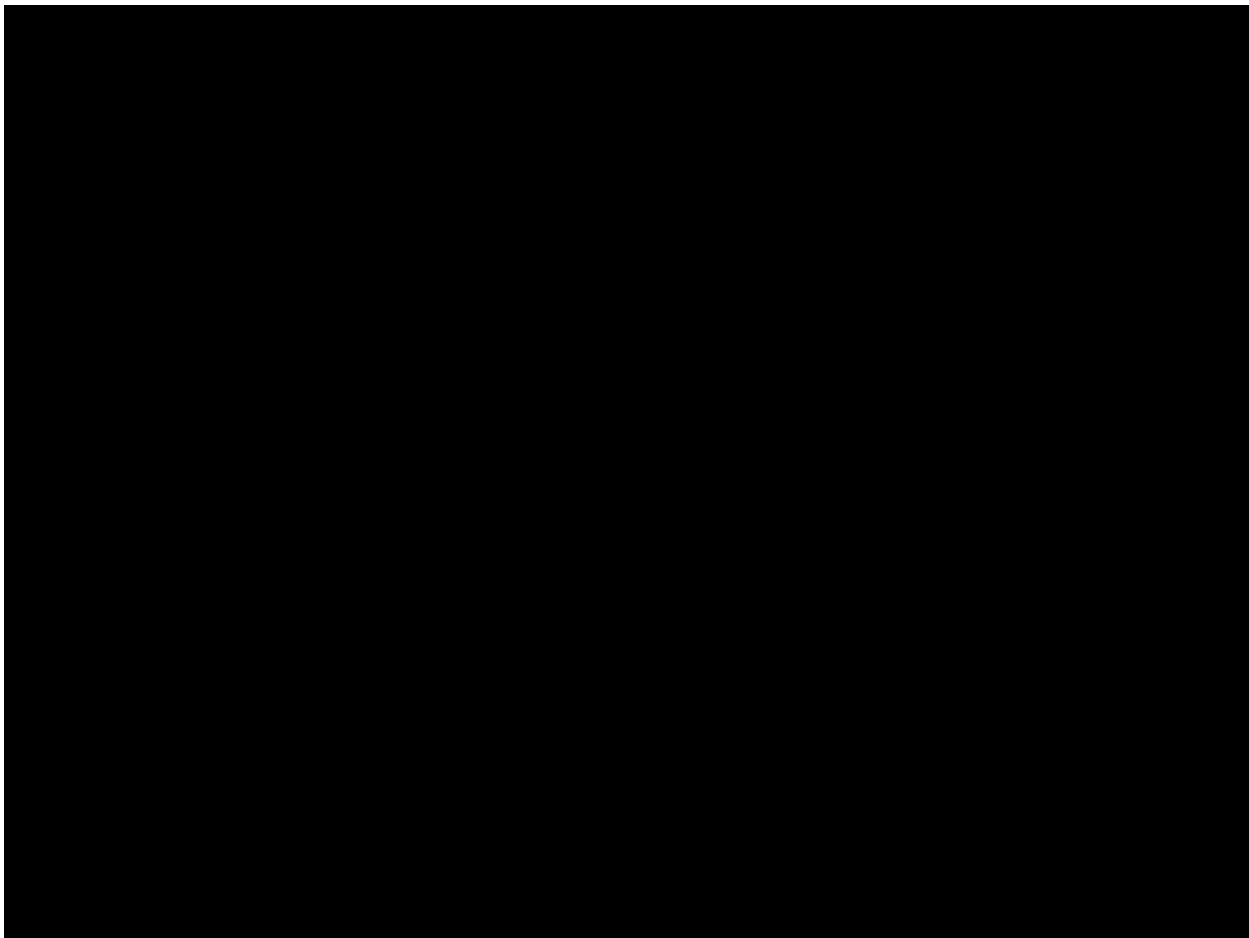
Searching NIST Glossary

Page Not Found

Trying to find a specific publication? Visit our [publications homepage](#) or see lists of [Draft Publications](#), [FIPS](#), [SP 800s](#), and [NISTIRs](#).

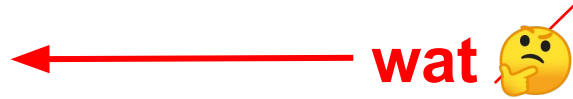
The page you were looking for cannot be found. If this was unexpected behavior, please send an email to webmaster-csrc@nist.gov. Make sure to include a detailed description of the actions you took and the page ultimately referred you here.





<https://github.com/shieldfy/API-Security-Checklist>

- Authentication
- JWT (JSON Web Token)
- OAuth
- Access
- Input
- Processing
- Output



OWASP API Security Project


What is API Security?

A foundational element of innovation in today's app-driven world is the API. From banks, retail and transportation to IoT, autonomous vehicles and smart cities, APIs are a critical part of modern mobile, SaaS and web applications and can be found in customer-facing, partner-facing and internal applications.

By nature, APIs expose application logic and sensitive data such as Personally Identifiable Information (PII) and because of this have increasingly become a target for attackers. Without secure APIs, rapid innovation would be impossible.

API Security focuses on strategies and solutions to understand and mitigate the unique vulnerabilities and security risks of Application Programming Interfaces (APIs).

hackthebox.eu invite



Hi!

Feel free to hack

Invite Code

Sign Up

If you are already a member click [here](#) to login

Input

length: 108
lines: 1

```
SW4gb3JkZXIgdG8gZ2VuZXJhdGUgdGhIGludml0ZSbjb2RlLCBtYWtLIgEgUE9TVCByZXF1ZXN0IHRvIC9hcGkvaW52aXRlL2d1bmVyYXRl
```

Output

time: 5ms
length: 81
lines: 1

In order to generate the invite code, make a POST request to /api/invite/generate

inviteapi.min.js

```
trevor@hecate:~$ curl -XPOST https://www.hackthebox.eu/api/invite/generate
{"success":1,"data":{"code":"RUlXUE0tVU1GVVktWEVFU0MtQkZJS1MtUERTUFE=","format":"encoded"},"0":200}trevor@hecate:~$
```

Input

```
RUlXUE0tVU1GVVktWEVFU0MtQkZJS1MtUERTUFE=
```

Output

```
EWIWM-UMFUY-XEESC-BFIKS-PDSPQ
```

```
> makeInviteCode()
< undefined
{0: 200, success: 1, data: {code: 'SW4gb3JkZXIgdG8gZ2VuZXJhdGUgdGhIGludml0ZSbjb2RlLCBtYWtLIgEgUE9TVCByZXF1ZXN0IHRvIC9hcGkvaW52aXRlL2d1bmVyYXRl', entype: 'BASIC', _proto_: 0b1}, _proto_: 0b1}
```


<https://cheatsheetseries.owasp.org/>



Life is too short • AppSec is tough • Cheat!

<https://cheatsheetseries.owasp.org/>

- Index ASVS
- Index Proactive Controls
- AJAX Security
- Abuse Case
- Access Control
- Attack Surface Analysis
- Authentication
- Authorization Testing Automation
- Bean Validation
- C-Based Toolchain Hardening
- C-Based Toolchain Hardening
- Choosing and Using Security Questions
- Clickjacking Defense
- Content Security Policy
- Credential Stuffing Prevention
- Cross-Site Request Forgery Prevention
- Cross Site Scripting Prevention
- Cryptographic Storage
- DOM based XSS Prevention
- Denial of Service
- Deserialization
- Docker Security
- DotNet Security
- Error Handling
- Forgot Password
- HTML5 Security
- HTTP Strict Transport Security
- Injection Prevention
- Injection Prevention in Java
- Input Validation
- Insecure Direct Object Reference Prevention
- JAAS
- JSON Web Token for Java
- Key Management
- LDAP Injection Prevention
- Logging
- Mass Assignment
- Microservices based Security Arch Doc
- OS Command Injection Defense
- PHP Configuration
- Password Storage
- Pinning
- Protect FileUpload Against Malicious File
- Query Parameterization
- REST Assessment
- REST Security
- Ruby on Rails Cheatsheet
- SAML Security
- SQL Injection Prevention
- Securing Cascading Style Sheets
- Server Side Request Forgery Prevention
- Session Management
- TLS Cipher String
- Third Party Javascript Management
- Threat Modeling
- Transaction Authorization
- Transport Layer Protection
- Unvalidated Redirects and Forwards
- User Privacy Protection
- Virtual Patching
- Vulnerability Disclosure
- Vulnerable Dependency Management
- Web Service Security
- XML External Entity Prevention
- XML Security

API Security Top 10 Release Candidate is Here!


A1	Broken Object Level Authorization	APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface Level Access Control issue. Object-level authorization checks should be considered in every function that accesses a data source using input from the user.
A2	Broken Authentication	Authentication mechanisms are often implemented incorrectly, allowing attackers to compromise authentication tokens or to exploit implementation flaws to assume other user's identities temporarily or permanently. Compromising system's ability to identify the client/user, compromises API security overall.
A3	Excessive Data Exposure	Looking forward to generic implementations, developers tend to expose all object properties without considering their individual sensitivity, relying on clients to perform the data filtering before displaying it to the user. Without controlling the client's state, servers receive more-and-more filters which can be abused to gain access to sensitive data.
A4	Lack of Resources & Rate Limiting	Quite often, APIs do not impose any restrictions on the size or number of resources that can be requested by the client/user. Not only can this impact the API server performance, leading to Denial of Service (DoS), but also leaves the door open to authentication flaws such as brute force.
A5	Broken Function Level Authorization	Complex access control policies with different hierarchies, groups, and roles, and an unclear separation between administrative and regular functions, tend to lead to authorization flaws. By exploiting these issues, attackers gain access to other users' resources and/or administrative functions.
A6	Mass Assignment	Binding client provided data (e.g., JSON) to data models, without proper properties filtering based on a whitelist, usually lead to Mass Assignment. Either guessing objects properties, exploring other API endpoints, reading the documentation, or providing additional object properties in request payloads, allows attackers to modify object properties they are not supposed to.
A7	Security Misconfiguration	Security misconfiguration is commonly a result of insecure default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers, unnecessary HTTP methods, permissive Cross-Origin resource sharing (CORS), and verbose error messages containing sensitive information.
A8	Injection	Injection flaws, such as SQL, NoSQL, Command Injection, etc. occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A9	Improper Assets Management	APIs tend to expose more endpoints than traditional web applications, making proper and updated documentation highly important. Proper hosts and deployed API versions inventory also play an important role to mitigate issues such as deprecated API versions and exposed debug endpoints.
A10	Insufficient Logging & Monitoring	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems to tamper with, extract, or destroy data. Most breach studies demonstrate the time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.


NIST SP 800-204: Security Strategies for Microservices-based Application Systems


Abstract


Microservices architecture is increasingly being used to develop application systems since its smaller codebase facilitates faster code development, testing, and deployment as well as optimization of the platform based on the type of microservice, support for independent development teams, and the ability to scale each component independently. Microservices generally communicate with each other using Application Programming Interfaces (APIs), which requires several core features to support complex interactions between a substantial number of components. These core features include authentication and access management, service discovery, secure communication protocols, security monitoring, availability/resiliency improvement techniques (e.g., circuit breakers), load balancing and throttling, integrity assurance techniques during induction of new services, and handling of session persistence. Additionally, the core features could be bundled or packaged into architectural frameworks such as API gateways and service mesh. The purpose of this document is to analyze the multiple implementation options available for each individual core feature and configuration options in architectural frameworks, develop security strategies that counter threats specific to microservices, and enhance the overall security profile of the microservices-based application.


Learn for API, not for other services


 **Trevor Bryant** 11:04 AM
API question: outside of SP 800-204, where can I find about how the gov protects APIs? specifically over the internet


 **9 replies** Last reply today at 11:50 AM


 **Aidan Feldman (Tech Portfolio, NYC, he/him)** 5 days ago
aside from FIPS stuff, i cant think of much that would be gov-specific... (edited)


 **John Jediny (TTS Tech Portfolio / DCA / he/him)** 5 days ago
not aware of any gov specific guides @gray might know best? <https://cheatsheetseries.owasp.org> seems a good checklist but is more basic hygiene than API specific FWIW


 **Gray Brooks (OPP - DCA - he/his)** 4 days ago
I'm not aware of anything else gov-wide, either.


 **Gray Brooks (OPP - DCA - he/his)** 4 days ago
GSA recently promulgated a big, long detailed API security guide internally and I'd pitched them on letting us publish it publicly but that hasn't happened yet.

 **Peter Burkholder (he/DCA/4th year 18F)** 22 hours ago
There's nothing in the document marking it as internal only, so it's okay to share with government contractors, right?

 **Gray Brooks (OPP - DCA - he/his)** 2 hours ago
if they are folks working at GSA, go for it. if outside, it'd be better for us to go ahead and engage with the GSA security team about just making it public.

 **Peter Burkholder (he/DCA/4th year 18F)** 2 hours ago
Well, it says "approved for distribution" so, it's okay to distribute, right? It's a non-sensitive work produced with taxpayer funds. I agree we should work with security to publish, but if there's a guidance that I *shouldn't* distribute, then I'll go ahead and share out with Trevor.

 **Gray Brooks (OPP - DCA - he/his)** 2 hours ago
I believe that's in reference to distribution internally. Yeah, please hold for the moment. @trevorbryant, I'll try to get it public sooner rather than later.

 **Peter Burkholder (he/DCA/4th year 18F)** 40 minutes ago
Meanwhile, you could also email SecEng@gsa.gov and ask them to share with y'all. (edited)

Summary

- OWASP API Security Project
 - https://www.owasp.org/index.php/OWASP_API_Security_Project
- OWASP Cheatsheet Series Project
 - <https://cheatsheetseries.owasp.org/>
- NIST SP 800-204: Security Strategies for Microservices-based Application Systems
 - <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-204.pdf>
- Gray Brooks @18F – GSA API Security Guide
 - `curl -XPOST graybrooks.com`

Drop Some Knowledge

