Something Old, Something New - OWASP API Security Top 10 in 2023

Colin Domoney
Chief Technology Evangelist
About the Speakers

Colin Domoney
Chief Technology Evangelist
Editor of APISecurity.io
CyberProof, CA, Veracode, Deutsche Bank
Housekeeping Rules

- All attendees muted
- Questions via Q&A
- Recording will be shared
- Polling questions
Agenda

• Overview of research into API vulnerabilities of the last 12 months
• OWASP Top 10 methodology
• Two items dropping off the Top 10
• Items that stay the same and why they remain a concern
• Three new items on the Top 10
  • How to address them
  • How 42Crunch can help
• Questions and Answers
OWASP and the Top 10

The Top 10 in 2019 and now
### My own research via APISecurity.io

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue Number</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>26 May, 23</td>
<td>Issue 220</td>
<td>API flaw in Booking.com, apps leaking sensitive API data, API security testing checklist</td>
</tr>
<tr>
<td>12 May, 23</td>
<td>Issue 219</td>
<td>Money Lover app exposes user data, most web API flaws missed by standard testing</td>
</tr>
<tr>
<td>6 May, 23</td>
<td>Issue 218</td>
<td>Three Argo CD API exploits, distributed identity for modern API security</td>
</tr>
<tr>
<td>28 April, 23</td>
<td>Issue 217</td>
<td>Wordle API exposes answers, Twitter API breach updates, AWS exposed dangerous API</td>
</tr>
<tr>
<td>6 March, 23</td>
<td>Issue 215</td>
<td>API flaws in Lego marketplace, API style guides, 42Crunch joins MISA</td>
</tr>
<tr>
<td>9 February, 23</td>
<td>Issue 214</td>
<td>Google Cloud’s four pillars of API security, Cerbos for API permissions, attacking predictable GUIDs</td>
</tr>
<tr>
<td>26 January, 23</td>
<td>Issue 213</td>
<td>Supply chain vulnerability in IBM Cloud, hardcoded API keys in Algolia portal, JSON-based SQL attacks</td>
</tr>
</tbody>
</table>
## The OWASP Top 10 - process and scoring methodology

<table>
<thead>
<tr>
<th>Threat Agents</th>
<th>Exploitability</th>
<th>Weakness Prevalence</th>
<th>Weakness Detectability</th>
<th>Technical Impact</th>
<th>Business Impacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>API Specific</td>
<td>Easy: 3</td>
<td>Widespread 3</td>
<td>Easy 3</td>
<td>Severe 3</td>
<td>Business Specific</td>
</tr>
<tr>
<td>API Specific</td>
<td>Average: 2</td>
<td>Common 2</td>
<td>Average 2</td>
<td>Moderate 2</td>
<td>Business Specific</td>
</tr>
<tr>
<td>API Specific</td>
<td>Difficult: 1</td>
<td>Difficult 1</td>
<td>Difficult 1</td>
<td>Minor 1</td>
<td>Business Specific</td>
</tr>
</tbody>
</table>

# OWASP API Security Top 10 — Then and Now

<table>
<thead>
<tr>
<th>2019</th>
<th>#</th>
<th>2023</th>
</tr>
</thead>
<tbody>
<tr>
<td>API1:2019 - Broken Object Level Authorization</td>
<td>1</td>
<td>API1:2023 - Broken Object Level Authorization</td>
</tr>
<tr>
<td>API2:2019 - Broken User Authentication</td>
<td>2</td>
<td>API2:2023 - Broken Authentication</td>
</tr>
<tr>
<td>API3:2019 - Excessive Data Exposure</td>
<td>3</td>
<td>API3:2023 - Broken Object Property Level Authorization</td>
</tr>
<tr>
<td>API4:2019 - Lack of Resources &amp; Rate Limiting</td>
<td>4</td>
<td>API4:2023 - Unrestricted Resource Consumption</td>
</tr>
<tr>
<td>API7:2019 - Security Misconfiguration</td>
<td>7</td>
<td>API7:2023 - Server Side Request Forgery</td>
</tr>
<tr>
<td>API8:2019 - Injection</td>
<td>8</td>
<td>API8:2023 - Security Misconfiguration</td>
</tr>
<tr>
<td>API10:2019 - Insufficient Logging &amp; Monitoring</td>
<td>10</td>
<td>API10:2023 - Unsafe Consumption of APIs</td>
</tr>
</tbody>
</table>
Question One:

Which one of the OWASP Top 10 causes your organization the most amount of pain?

1. Broken Object/Function Level Authorization
2. Broken Authentication
3. Broken Object Property Level Authorization
4. Unrestricted Resource Consumption
5. Unrestricted Access to Sensitive Business Flows
Question One:

Which one of the OWASP Top 10 causes your organization the most amount of pain?

<table>
<thead>
<tr>
<th>Issue</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broken Object/Function Level Authorization</td>
<td>24%</td>
</tr>
<tr>
<td>Broken Authentication</td>
<td>34%</td>
</tr>
<tr>
<td>Broken Object Property Level Authorization</td>
<td>15%</td>
</tr>
<tr>
<td>Unrestricted Resource Consumption</td>
<td>15%</td>
</tr>
<tr>
<td>Unrestricted Access to Sensitive Business Flows</td>
<td>12%</td>
</tr>
</tbody>
</table>
Dropping out the Top 10 — API8:2019 — Injection

- Injection attacks affect all software systems, not only APIs
- Injection attacks are still very prevalent and affect APIs frequently
- Remediation advice as per OWASP Top 10 still applies (nothing API specific)
- 42Crunch offers protection for this category (relies on full specification of input data)

https://apisecurity.io/owasp-api-security-top-10/api8-injection/

Lack of injection in 2023 API10 #86

https://github.com/OWASP/API-Security/issues/86

Issue 200: Injection vulnerability in BitBucket, OAuth2 exploitation, and 200th issue prize giveaways

September 1, 2022
Dropping out the Top 10 — API10:2019 — Insufficient logging and monitoring

- Logging and monitoring affect **all** software systems, not only APIs
- Logging and monitoring are seldom reported in API breaches
- Typically they are a symptom rather than a cause
- Remediation advice as per OWASP Top 10 still applies (nothing API specific)

What’s new in 2023

- This entails using an API in a way in which it was not designed or abusing the underlying business flow or logic
- In a single word - BOTS
- Becoming the most common attack vector

<table>
<thead>
<tr>
<th>Threat agents/Attack vectors</th>
<th>Security Weakness</th>
<th>Impacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>API Specific : Exploitability Easy</td>
<td>Prevalence Widespread : Detectability Average</td>
<td>Technical Moderate : Business Specific</td>
</tr>
</tbody>
</table>

Exploitation usually involves understanding the business model backed by the API, finding sensitive business flows, and automating access to these flows, causing harm to the business.

Lack of a holistic view of the API in order to fully support business requirements tends to contribute to the prevalence of this issue. Attackers manually identify what resources (e.g. endpoints) are involved in the target workflow and how they work together. If mitigation mechanisms are already in place, attackers need to find a way to bypass them.

In general technical impact is not expected. Exploitation might hurt the business in different ways, for example: prevent legitimate users from purchasing a product, or lead to inflation in the internal economy of a game.

How To Prevent

The mitigation planning should be done in two layers:

• **Business** - identify the business flows that might harm the business if they are excessively used.
• **Engineering** - choose the right protection mechanisms to mitigate the business risk.

Common methods include:

• **Device fingerprinting**: denying service to unexpected client devices (e.g., headless browsers) tends to make threat actors use more sophisticated solutions, thus more costly for them
• **Human detection**: using either a captcha or more advanced biometric solutions (e.g., typing patterns)
• **Non-human patterns**: analyse the user flow to detect non-human patterns (e.g., the user accessed the "add to cart" and "complete purchase" functions in less than one second)
• Consider blocking IP addresses of Tor exit nodes and well-known proxies
How 42Crunch helps API6:2023 - Unrestricted Access to Sensitive Business Flows


Server-Side Request Forgery (SSRF) flaws occur when an API is fetching an unattended URL-based resource because of missing validation of user-supplied URL.

- Main target is internal resources
- Can also be external resources accessed taking advantage of trust relationships between server and targeted external resource
- Becoming a more common attack vector

How To Prevent

• Isolate the resource fetching mechanism in your network: usually these features are aimed to retrieve remote resources and not internal ones.

• Whenever possible, use allow lists of:
  - Remote origins users are expected to download resources from (e.g. Google Drive, Gravatar, etc.)
  - URL schemes and ports
  - Accepted media types for a given functionality

• Disable HTTP redirections.

• Use a well-tested and maintained URL parser to avoid issues caused by URL parsing inconsistencies.

• Validate and sanitize all client-supplied input data.

• Do not send raw responses to clients.
How 42Crunch helps API7:2023 - Server Side Request Forgery

- Use 42Crunch contract enforcement
- Restrict allowable input values to known paths such as “uploads/”

https://docs.42crunch.com/latest/content/extras/protection_allowlist.htm
New entry — API10:2023 - Unsafe Consumption of APIs

- Developers tend to trust data received from third-party APIs more than user input.
- APIs are part of a supply chain and need to be secured at every point.

---

[Table]

<table>
<thead>
<tr>
<th>Threat agents/Attack vectors</th>
<th>Security Weakness</th>
<th>Impacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>API Specific: Exploitability</td>
<td>Prevalence Common: Detectability Average</td>
<td>Technical Severe: Business Specific</td>
</tr>
<tr>
<td>Easy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exploiting this issue requires attackers to identify and potentially compromise other APIs/services the target API integrated with. Usually, this information is not publicly available or the integrated API/service is not easily exploitable.</td>
<td>Developers tend to trust and not verify the endpoints that interact with external or third-party APIs, relying on weaker security requirements such as those regarding transport security, authentication/authorization, and input validation and sanitization. Attackers need to identify services the target API integrates with (data sources) and, eventually, compromise them.</td>
<td>The impact varies according to what the target API does with pulled data. Successful exploitation may lead to sensitive information exposure to unauthorized actors, many kinds of injections, or denial of service.</td>
</tr>
</tbody>
</table>

---

How To Prevent

• When evaluating service providers, assess their API security posture.
• Ensure all API interactions happen over a secure communication channel (TLS).
• Always validate and properly sanitize data received from integrated APIs before using it.
How 42Crunch helps API10:2023 - Unsafe Consumption of APIs

- X Corporation
  - API 1
  - API 2
  - API 3

- 42Crunch API Protect (micro-firewall)

- X Corporation
  - API 1
  - API 2
  - API 3

- ACME Inc

- 42Crunch API Protect (micro-firewall)
Question Two:

How sure are you of your upstream API provider’s security posture?

1. Audited them and satisfied
2. Asked about their process
3. No idea
4. A growing concern
5. Experienced an incident
Question Two:

How sure are you of your upstream API provider’s security posture?

<table>
<thead>
<tr>
<th>Option</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audited them and satisfied</td>
<td>12%</td>
</tr>
<tr>
<td>Asked about their process</td>
<td>23%</td>
</tr>
<tr>
<td>No idea</td>
<td>49%</td>
</tr>
<tr>
<td>A growing concern</td>
<td>16%</td>
</tr>
<tr>
<td>Experienced an incident</td>
<td>0%</td>
</tr>
</tbody>
</table>
Tweaks in terminology and structure

- **API2**: The title was changed from “Broken User Authentication” to “Broken Authentication”
- **API4**: The title was changed from “Lack of Resources & Rate Limiting” to “Unrestricted Resource Consumption.”
- **API9**: The title was changed from “Improper Assets Management” to “Improper Inventory Management”
- **API3**: The 2019 categories “Excessive Data Exposure” (*read*) and “Mass Assignment” (*write*) are merged into “Broken Object Property Level Authorization”
Learning more and next steps

Where to get more information
Learning more

- https://danaepp.com/owasp-api-security-top-10-upcoming-changes-you-need-to-know-about
- https://danaepp.com/exploiting-ssrf-in-an-api
- https://portswigger.net/web-security/ssrf
Using the 42Crunch extensions

GitHub Action
42Crunch REST API Static Security Testing

GitHub Action: 42Crunch REST API Static Security Testing

The REST API Static Security Testing action locates REST API contracts that follow the OpenAPI Specification (OAS, formerly known as Swagger) and runs thorough security checks on them. Both OAS v2 and v3 are supported, in both JSON and YAML format.

https://github.com/marketplace/actions/42crunch-rest-api-static-security-testing

#1 API Security Newsletter
APISecurity.io

"Defending APIs against Cyber Attack"
by Colin Domoney

https://apisecurity.io/

https://amzn.to/3fHp8Mz
Audience Question One:

Are there any aspects you recommend someone who wants to start using automation?
Audience Question Two:

Does 42Crunch offer any security suggestions at API design beyond threat modeling?
Audience Question Three:

The approaches to manage authentication include JWT, Cookies, Session tokens. Internet is full of articles about them but not so much about what use cases each one of those fulfill. If you can share some resources that compare these approaches based on the use case that would be appreciated.

https://pragmaticwebsecurity.com/
Audience Question Four:

Does 42Crunch offer any solution on API discovery and inventory?
Something Old, Something New - OWASP API Security Top 10 in 2023

Colin Domoney
Chief Technology Evangelist