Article
Yuri Marx · Mar 13, 2021 5m read

Protect your REST API applying OWASP Top Ten

Hi Community,

Did you know about OWASP and Top Ten Web Application security risks to your Web API or Web Apps?

OWASP is a community foundation created to help us to improve the security of web apps/web APIs. OWASP do the web apps more secure through its community-led open source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.

To summarize the top procedures to do your web app/web api more secure OWASP published the "Top 10 Web Application Security Risks" recomendations, see (source: https://owasp.org/www-project-top-ten/):

1. **Injection**. Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker ' s hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2. **Broken Authentication**. Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users ' identities temporarily or permanently.
3. **Sensitive Data Exposure**. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
4. **XML External Entities (XXE)**. Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
5. **Broken Access Control**. Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users ' accounts, view sensitive files, modify other users ' data, change access rights, etc.
6. **Security Misconfiguration**. Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.
7. **Cross-Site Scripting (XSS)**. XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim ' s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
8. **Insecure Deserialization**. Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
9. **Using Components with Known Vulnerabilities**. Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
10. **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, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Some techniques could be used to implement OWASP top ten, see the table:

OWASP Recomentation	
Injection	In SQL use input parameters specif example, :var). Avoid concatenate Use Sanitazer API like https://developer.mozilla.org/en-US
Broken Authentication	Create strong passwords, see Micr ws/create-and-use-strong-passwor Don't use passwords inside source Externalize your passwords using p Don't allow brute force attacks usin https://docs.intersystems.com/comp
Sensitive Data Exposure	Avoid return Credit card data, healt possible, encrypt message channel collect to the minimal as possible.
XXE	Use %XMLAdaptor
Broken Access Control	Use access management tools like https://docs.intersystems.com/comp
Security Misconfiguration	Update your software periodically a the appropriate procedures. Follow the Security Administration https://docs.intersystems.com/irisla
XSS	Do the input sanitization/validation, tips: https://cheatsheetseries.owasp
Insecure deserialization	Create hash to files, create safer fo validate files extensions
Using Components with Known Vulnerabilities	In the InterSystems stack Apache configure it to the production, follow tips/. Enable your OS firewall, to ub https://www.digitalocean.com/comm ubuntu-18-04)
Insufficient Logging & Monitoring	Use Log Monitor (https://docs.intersystems.com/irisla MONITOR) Follow it: https://docs.intersystems.com/irisla Monitor using SAM: https://docs.intersystems.com/sam/

You can use WAF software in alignment with OWASP Top ten, see: https://www.g2.com/categories/api-security or use InterSystems IAM. From G2 list, I like 42 crunch, that allows a free account and you appoint your OpenAPI file and it is analyzed to report recomendations, see in https://42crunch.com/owasp-api-security-top-10/.

#Security #InterSystems IRIS

Source URL:https://community.intersystems.com/post/protect-your-rest-api-applying-owasp-top-ten