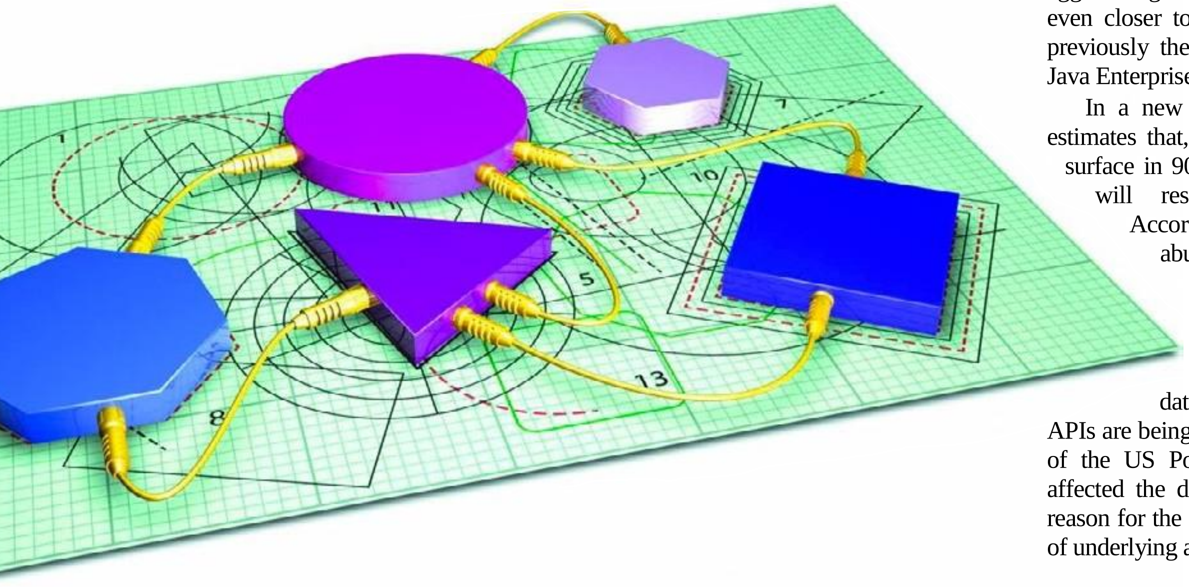
OWASP API Security Top 10

Well protected

Martin Burkhart



Hackers are increasingly setting their sights on APIs. A glance at the OWASP list of vulnerabilities shows the points at which developers are required.

Exposed APIs are becoming the most popular attack surface of web applications. OWASP (Open Web Application Security Project) is responding to this with a new and specialised top ten list for API security.

The original OWASP top ten is a widely distributed list of the ten most significant vulnerabilities in web applications. The list appeared for the first time in 2003 and is based on the data of hundreds of organisations worldwide. It describes every vulnerability, and possible countermeasures, in detail. Over the years, OWASP has also taken into account vulnerabilities of APIs (Application Programming Interfaces), which are becoming increasingly widely used in software development. OWASP therefore no longer talks about simply applications, but rather “Applications or APIs”. Likewise, the organisation has incorporated vulnerabilities that are API-specific, such as “A4 - XML External Entities” in the 2017 edition.

OWASP is now taking a further step and, in December 2019, released a separate list of top ten vulnerabilities for APIs. In this, OWASP emphasises the increasing importance of API security for companies.

Dedicated security products such as web application firewalls (WAF), API Gateways and CIAM systems (Customer Identity Access Management) may have some success in protecting programming interfaces. However, it would be negligent to rely exclusively on such products. The most important protection stems from API developers who know the vulnerabilities and can therefore prevent them.

SPAs are fuelling the spread of APIs.

Unlike even ten years ago, nowadays web applications are usually single page applications (SPAs) that integrate several APIs, for example in the form of micro-services. Web frameworks such as Angular or React are used in the development of user interfaces. The APIs encapsulate individual aspects of business logic. The inter setting the elements on the user interface is oriented to rich clients.

Modern APIs are typically implemented as RESTful web services. The SPAs call them up directly from the browser, making them equally vulnerable to attack as traditional web applications. Unfortunately, APIs of this kind often exhibit similar or even the same vulnerabilities. Another aggravating factor is that they are located even closer to the sensitive data than was previously the case, for example behind a Java Enterprise facade.

In a new API security study, Gartner estimates that, by 2021, the greatest attack surface in 90 percent of web applications will result from exposed APIs. According to this, by 2022 the abuse of these will be the most significant attack vector and will lead to data breaches globally.

Even today, significant data incidents owing to unsecure APIs are being reported, such as the hacking of the US Postal Service in 2018 which affected the data of 60 million users. The reason for the successful attack was the lack of underlying access controls for objects.

Comparison of charts

When comparing the new top ten list for APIs with the current one for web applications from 2017 (A1-A10), it is clear that several vulnerabilities are the same or at least similar:

- Broken Authentication (API2, A2)
- Security Misconfiguration (API7, A6)
- Injection (API8, A1)
- Insufficient Logging and Monitoring (API10, A10)

One reason for this is likely to be the fact that traditional web applications and SPAs fundamentally carry out the same tasks using APIs. Both approaches provide a user interface in web browsers, and have to authenticate users. They receive user data and manipulate datasets in databases. Both run on servers or in containers, and are therefore susceptible to configuration errors. In the past, and today, administrators monitor the operation and the security by monitoring log data.

The lists not only have overlapping topics, but also a similar priority of vulnerabilities. Only in the case of injection is it apparent that this is eighth place in APIs, but at the top of the web applications list.

You might assume that modern web frameworks are increasingly taking on functions for validating the input data, and therefore clients transfer fewer malicious strings to the APIs. However, it is necessary to take into account the fact that native mobile apps or IoT clients also use the APIs. Corresponding frameworks are missing here. Furthermore, one should never rely on validation on the client side, since hackers can directly attack the APIs, bypassing the frameworks.

API-specific vulnerabilities

Interestingly, "A4 XML External Entities (XEE)" does not appear in the API list. This may be due to the decreasing importance of SOAP web services. "A7 Cross-Site Scripting (XSS)" is also missing from the API list. OWASP appears to consider XSS purely as a browser problem. It is true that APIs do not interpret JavaScript, and are therefore not directly susceptible to XSS. However, API endpoints should validate their input data to ensure that they do not contain any JavaScript commands that an application could store permanently. Depending on the client, the commands, and thus XSS, can absolutely be a problem.

The generic topic "A5 Broken Access Control" is divided into different aspects in the API list. In this case, OWASP takes into account the fundamental structure of API calls, and the formats used, such as JSON. The organisation differentiates unauthorised access according to whether the access is to entire objects (API1) or whether individual attributes of objects are accessed. In the case of the attributes, OWASP also differentiates between reading (API3) and modification (API6).

Compared with the OWASP top 10, there are also two new additions in the form of "API 4 Lack of Resources and Rate Limiting" and "API9 Improper Assets Management". This is reasonable insofar as API endpoints are located closer to the effective infrastructure than the URL of a servlet that processes data from an HTML format.

The right security infrastructure

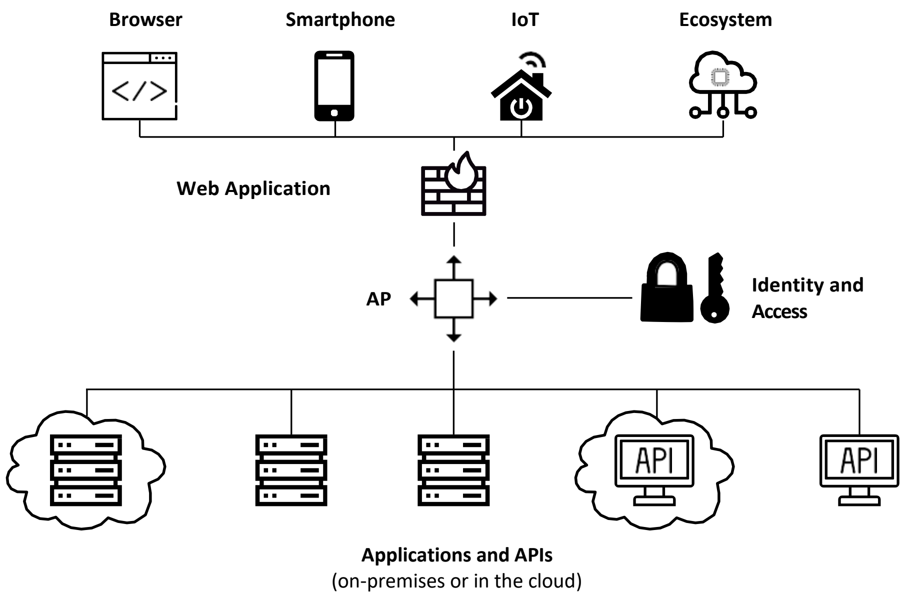
In current technology, security services are often designed to be implemented upstream, in order that they can benefit all applications and interfaces. The services include a combination of web application firewalls and API management, integrated with functions for access management (see Fig. 1).

The functions of an architecture of this kind vary depending on the product range. In principle, however, it provides the possibility of releasing new APIs in a targeted manner (API9). The fact that an API is available internally does not automatically mean that it is enabled for public access. API Gateways can issue API Keys which allow external developers and partners to establish clients on the basis of the public APIs.

In the case of access by a technical client with a valid key, the appropriate usage policy is applied. Restrictions such as throttling or quotas can be applied here (API4). If specifications for APIs are available, for example in the OpenAPI format, the API Gateways can read them and ensure that only compliant requests pass through. This prevents exploration or forceful browsing attacks on APIs owing to unprotected, undocumented endpoints, or by legacy endpoints or attributes (API9). This also allows for correct typing of the attributes on the gateway check and implement. If the specifications are concise and precise, injection attacks can thus be prevented (API8).

Summary: OWASP API Security Top 10

Number	Title	Description
API1	Broken Object Level Authorisation	API endpoints receive object IDs without checking whether the user/client is authorised to access said objects.
API2	Broken Authentication	Authentication logic is often implemented incorrectly. This allows attackers to compromise authentication tokens or to exploit errors in the authentication. If the identity of the client or of the user cannot be reliably ascertained, the basis for API security is lacking.
API3	Excessive Data Exposure	Developers generally tend to expose all object characteristics at endpoints, even the sensitive ones. They rely on the client, and hope that they will filter the data appropriately prior to displaying it to the user.
API4	Lack of Resources and Rate Limiting	APIs do not set any limitations with respect to the size and number of resources requested by a client. This can lead to impaired performance, or even denial of service (DoS). Furthermore, this can allow for brute force attacks, e.g. on passwords.
API5	Broken Function Level Authorisation	Complex access regulations with different hierarchies, groups, roles and an unclear separation between administrative and regulatory functions lead to authorisation errors. Attackers can thus gain access to resources for which they are not authorised.
API6	Mass Assignment	Data delivered by the client, e.g. in JSON format, is entered into the data model directly and unfiltered. Attackers can guess additional attributes, view them in the documentation, or find them out by means of exploration. They can thus modify object attributes to which they should not have access.
API7	Security Misconfiguration	Unsecure configurations usually result from unsecure default settings, incomplete configurations, publicly available cloud storage, incorrectly configured HTTPS headers and methods, CORS regulations that are too open, or error messages that give away too much.
API8	Injection	Injection vulnerabilities, e.g. for SQL, NoSQL, LDAP or OS commands, arise if unsecure input data is sent to an interpreter as part of a command. An attacker may thus be able to trigger particularly malicious actions and read or change data without authorisation.
API9	Improper Assets Management	APIs tend to expose more endpoints than traditional web applications. Correct and up-to-date documentation is therefore very important. An inventory of the hosts and API versions prevents, for example, the publication of APIs that are no longer supported, and debugging endpoints.
API10	Insufficient Logging and Monitoring	Insufficient logging and monitoring, associated with lacking or insufficient integration with incidence response, allow attackers to attack systems, become implanted, and attack further targets with the aim of extracting or modifying data. Studies show that breaches are often only discovered after 200 days, and even then only from external locations and not through internal processes.



Web Application Firewall, API Gateway and IAM work hand-in-hand for comprehensive API protection (Fig. 1).

CIAM systems are used to manage the identity and access rights. They authenticate the user (API2) and authorise them using standards such as OAuth, OpenID Connect or SAML, for access to applications and APIs (API5). This also allows for implementation of a comprehensive single sign on (SSO), and for development of standard tasks via user self-services.

Web application firewalls offer a large number of protection mechanisms against known attacks such as injections, XSS or CSRF (API8). Furthermore, they have secure basic settings for HTTP headers and TLS (API7), as well as functions for certificate management, using Let’s Encrypt, for example. For good API protection, it is necessary to ensure that the WAF effectively analyses JSON objects and can apply their rules to individual attributes. Otherwise, an appropriate API security gateway would have to take on the work. However, many API Gateways focus on API management, and put security aspects second.

A dedicated security infrastructure simplifies monitoring, troubleshooting and forensic analyses (API10). Supplemented by SIEM systems, information on different components is easy to compile and correlate. Furthermore, an alert in the case of anomalies makes it possible to leave the reactive mode and gain valuable time, in the event of an intrusion.

In the meantime, many security products now use machine learning (ML) in order to be able to react to previously unknown attacks and situations. For individual developers, the use of ML is difficult, because the methods require a large amount of data and benefit from a cross-service perspective.

The developers’ duty

Irrespective of external gateways, some tasks always remain in the hands of the developers. They should generally develop software so that it is secure without upstream security services. Among other things, they should validate inputs, irrespective of whether a WAF checks for injection attacks. WAFs always have to make a compromise between false positives and false negatives, and therefore do not have a one hundred percent identification rate. Changes to the infrastructure may also go unnoticed by the developer. Security by Design is recommended, rather than ensuring security ‘as an afterthought’. Vulnerability scanners integrated in the build pipeline can help to identify existing vulnerabilities in the finished code.

Developers must absolutely address the specialised authorisation of business objects. An API Gateway may know the endpoints and be able to distinguish between GET and UPDATE requests. However, the business objects and their attributes are unknown to them. Therefore, developers have to ensure that object IDs that the client delivers are effectively approved for the authenticated user (API1). This check is also necessary when modifying individual attributes (API6). When minimising the data transmission, developers cannot rely on the client, but have to filter out sensitive attributes on the API side (API3).

The way of dealing with API Keys is also important. These are explicitly not means for authentication, but rather serve merely to identify technical clients such as a mobile app or an SPA. An application must not only enable access to APIs on the basis of API Keys, but rather must ensure reliable user authentication and authorisation. Of course, API Keys are not part of public cloud storage. If clients require hard-coded API Keys teams must invest in client security in order to make attacks such as debugging and decompiling of the client codes more difficult.

Attack surface - APIs

APIs are likely to develop, over the coming years, into the main attack surface for web applications. OWASP is responding to this with a new and specialised top ten list for API security. Some topics from the new list, such as authentication and injection attacks, have been dominating web security since 2003, when OWASP published the first top ten list.

The vulnerabilities should be prevented in particular in the context of APIs. Dedicated security products such as web application firewalls, API Gateways and CIAM systems can contribute to the protection of APIs. However, developers must not rely solely on products. They are still responsible for certain topics, such as specialised authorisation of business objects, and secure handling of API Keys. (rme)

Dr Martin Burkhart

is Head of Product Management at Airlock, a security innovation of Ergon Informatik AG He initially ran IAM integration projects, and has been responsible for product management for the Airlock Secure Access Hub since 2012.