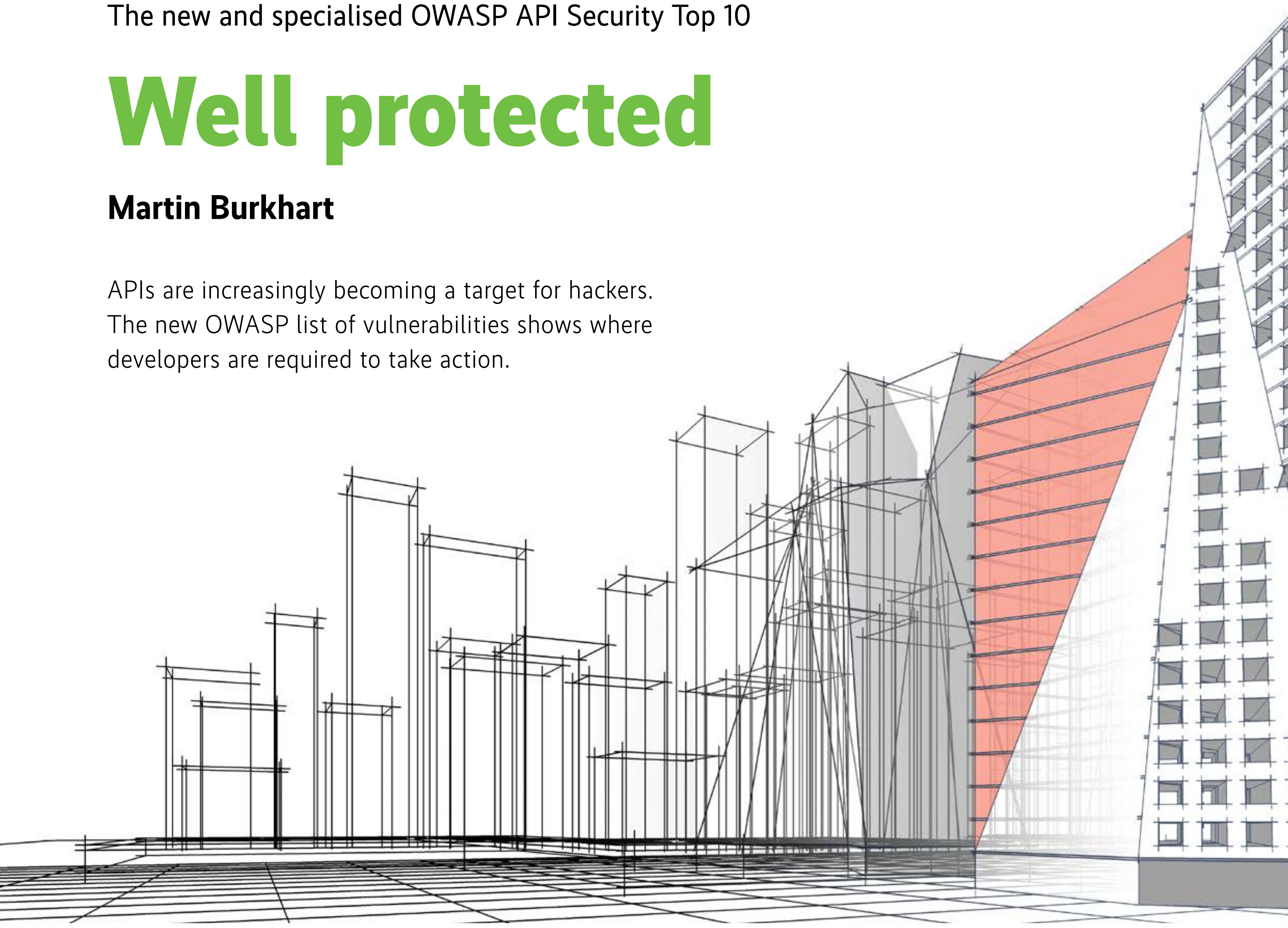
The new and specialised OWASP API Security Top 10

Well protected

Martin Burkhart

APIs are increasingly becoming a target for hackers. The new OWASP list of vulnerabilities shows where developers are required to take action.



Exposed APIs are fast becoming the most popular place in which to attack Web applications. In response to this situation, the Open Web Application Security Project (OWASP) has published a new and specialised API Security Top Ten. The original Top Ten is a widely used list of the ten principal vulnerabilities to be found in Web applications. The list was published for the first time in 2003 and is based on data supplied by hundreds of organisations worldwide. It provides a detailed description of each vulnerability as well as potential countermeasures. Over the last few years, OWASP has increasingly included vulnerabilities of APIs (Application Programming Interfaces), which are now becoming more and more widespread in software development. Consequently, OWASP no longer spoke of applications but of “applications or APIs”. The organisation also included API-specific vulnerabilities such as “A4 – XML External Entities“ in the 2017 edition. In 2019, OWASP set an example by publishing its own Top Ten of vulnerabilities to be found in APIs [a]. With it, the security project emphasises the increasing importance of API security for companies.

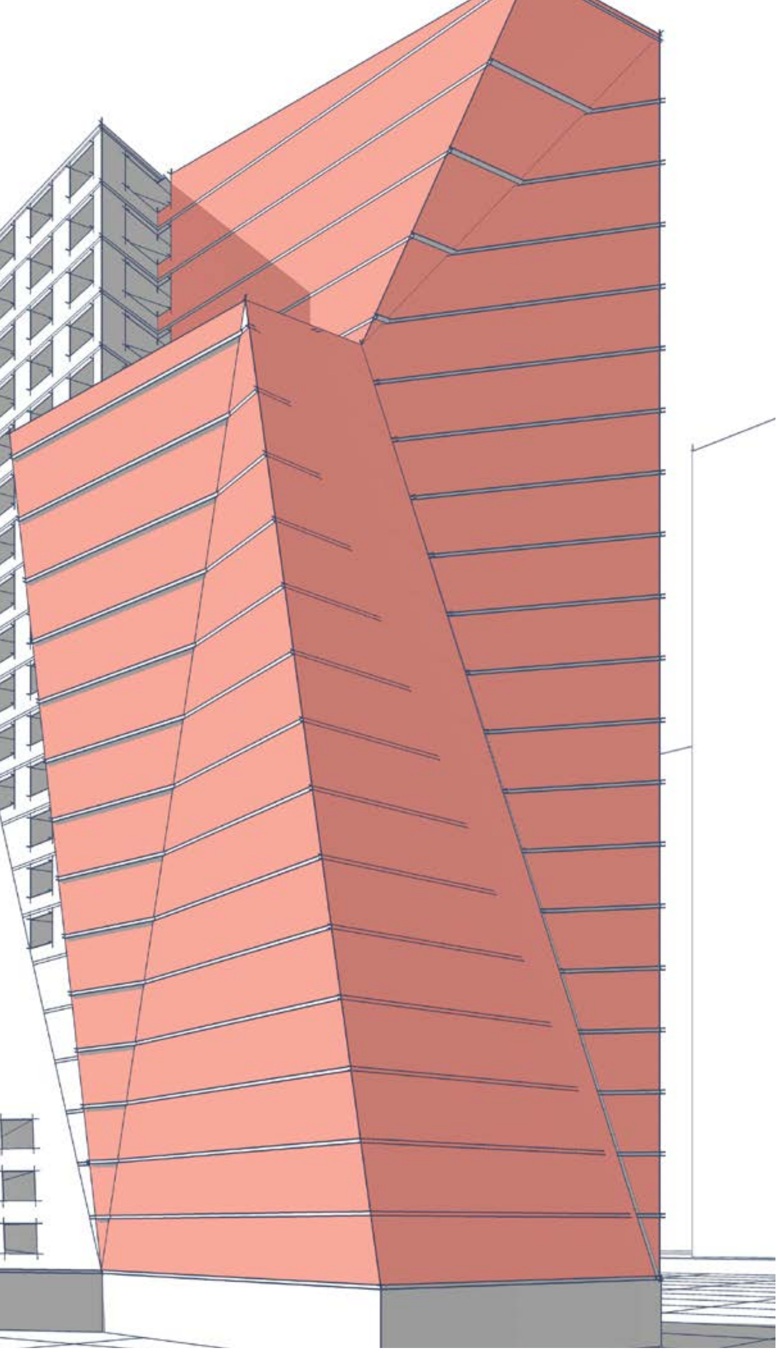
Dedicated security products such as Web application firewalls (WAF), API gateways and CIAM (Customer Identity Access Management) systems can do a lot to protect the programming interfaces. However, it would be grossly negligent to rely on products of this kind only. The most important protection is provided by the developers of APIs as they know the vulnerabilities and how to avoid them.

SPAs fuel the spread of APIs

In contrast to just ten years ago, Web applications are now usually single-page applications (SPA) which integrate a multitude of APIs, in the form of microservices for example. Web frameworks such as Angular or React are used in the development of the user interface. The APIs encapsulate individual aspects of the business logic. The configuration of the elements on the user interface is geared towards “rich clients”.

Modern APIs are typically implemented as RESTful Web services. The SPAs access them directly from the browser, making them as prone to attacks as traditional Web applications. Unfortunately, APIs of this kind often exhibit similar or even identical vulnerabilities. To make matters worse, they are located even closer to the sensitive data than they for example were when situated behind a Java enterprise facade.

In a new API security study [b], Gartner estimates that, by 2021, the greatest vulnerabilities in 90% of Web applications will be due to exposed APIs. In line with this, the abuse of APIs will become the principal attack vector by 2022 and will lead to world-



wide data breaches. The press is currently reporting prominent data breaches caused by insecure APIs - such as the hacking of the US Postal Services in 2018 [c], which involved the data of 60 million users. The reason for the successful attack was the absence of fundamental control mechanisms regulating access to objects.

Comparing the charts

When we compare the new Top Ten for APIs with the previous Top Ten for Web applications from the year 2017 (A1-A10) [d], we notice that a number of vulnerabilities are identical or at least similar:

OWASP Top 10 from 2017

Number	Title
A1	Injection
A2	Broken Authentication
A3	Sensitive Data Exposure
A4	XML External Entities (XXE)
A5	Broken Access Control
A6	Security Misconfiguration
A7	Cross-Site Scripting (XSS)
A8	Insecure Deserialization
A9	Using Components with Known Vulnerabilities
A10	Insufficient Logging & Monitoring

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

This is no doubt partly because traditional Web applications and SPAs with APIs basically perform the same tasks. Both approaches provide a user interface in Web browsers, and both have to authenticate users. They both receive user data and manipulate data records in databases. Both approaches run on servers or in containers, making them prone to errors in configuration. Now as before, administrators supervise operation and security by monitoring log data.

The lists not only have overlapping subjects – they also share a similar prioritisation of the vulnerabilities. Injection is the only exception: it is in eighth place on the API list, whereas it is right at the top of the Web application list. This may be based on the assumption that modern Web frameworks increasingly perform functions for validating the input data, so that clients hand over fewer malicious strings to the APIs. However, we should take into account that native mobile apps or IoT clients use the APIs too. The necessary frameworks are lacking there. Also, we should never rely on client-side validation as hackers can circumvent the frameworks and launch direct attacks on the APIs.

Interestingly, "XML External Entities (XEE)" is not included in the API list. This is probably due to the dwindling importance of SOAP Web services. "A7 Cross-Site Scripting (XSS)" is also missing from the API list. OWASP apparently sees XSS as being a browser problem only. Admittedly, APIs do not interpret JavaScript, so they are not immediately prone to XSS. However, API end points should validate the input data in a way ensuring that they do not contain JavaScript commands which could be stored permanently by an application. Depending on the client, the commands - and with them XSS – could well pose a problem.

API-specific vulnerabilities

The generic subject "A5 Broken Access Control" is divided up into different aspects in the API list. OWASP takes the fundamental structure of API calls as well as the formats used (such as JSON) into account. In the case of unauthorised accesses, the organisation makes a distinction as to whether an access involves whole objects (API1) or individual attributes of objects. In terms of attributes, OWASP also distinguishes between reading (API3) and modifying (API6).

In comparison with the OWASP Top 10, there are also two new accesses – "API 4 Lack of Resources & Rate Limiting" and "API9 Improper Assets Management". This is understandable as API end points are closer to the effective infrastructure than the URL of a servlet that processes data coming from an HTML form.

The appropriate security infrastructure

State-of-the-art security services are often installed upstream so that all applications and interfaces can benefit from them. The services consist of 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. Fundamentally, however, it provides the option of publishing new APIs in a targeted way (API9). The fact that an API is available internally does not automatically mean

OWASP API Security Top 10

Number	Title	Description
API1	Broken Object Level Authorization	API end points receive object IDs without checking whether the user/client is authorised to access these objects.
API2	Broken Authentication	Authentication logic is often implemented incorrectly, allowing hackers to compromise authentication tokens or exploit errors in authentication. If the identity of the client or user cannot be reliably determined, there is no foundation for API security.
API3	Excessive Data Exposure	Developers tend to generally expose all object properties on end points, including the sensitive ones. They rely on the client to filter the data in a suitable way before they are displayed to the user.
API4	Lack of Resources & Rate Limiting	APIs do not limit the size and number of the resources requested by a client. This can lead to bad performance or even to Denial of Service (DoS) in extreme cases. It can also make brute force attacks on passwords etc. possible.
API5	Broken Function Level Authorization	Complex access rules with different hierarchies, groups and roles and an unclear separation between regular and administrative functions lead to authorisation errors, allowing hackers to access resources without being authorised to do so.
API6	Mass Assignment	Data provided by the client, in the JSON format for example, are filled into the data model directly and with no filtering. Hackers can guess at additional attributes, view them in the documentation or find them out using exploration, allowing them to modify object attributes to which they should not have access.
API7	Security Misconfiguration	Insecure configurations usually result from insecure default settings, incomplete configurations, publicly available cloud storage, incorrectly configured HTTPS headers and methods, excessively open CORS rules or error messages which reveal too much.
API8	Injection	Injection vulnerabilities for SQL, NoSQL, LDAP or OS commands are the result when insecure input data are sent to an interpreter as part of a command. Potentially, this can allow a hacker to trigger malicious activities and read or change data without being authorised to do so.
API9	Improper Assets Management	APIs tend to expose more end points than traditional Web applications do. This makes correct and up-to-date documentation extremely important. For example, an inventory of the hosts and API versions prevents the publication of APIs and debugging end points which are no longer supported.
API10	Insufficient Logging & Monitoring	Insufficient logging and monitoring in conjunction with lacking or inadequate integration with incident response allows hackers to attack systems, implant themselves there and attack further targets with the aim of extracting or modifying data. Studies show that breaches are often only discovered after 200 days, and usually first by external instances rather than by internal processes.

that it is approved for public access. API gateways can provide API keys which allow external developers to build clients on the basis of the public APIs.

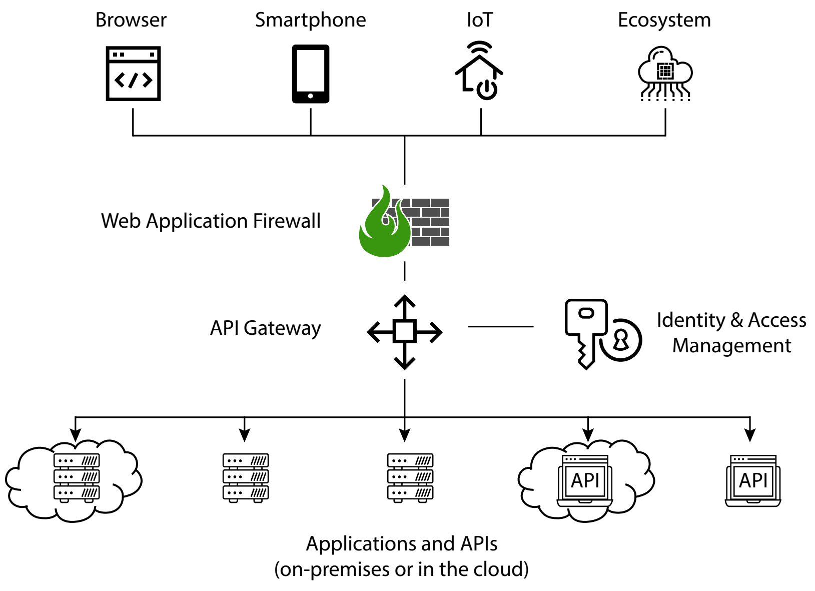
When a technical client accesses the system using a valid key, the appropriate usage policy is applied. Restrictions such as throttling or quotas can be implemented here (API4). If there are specifications for APIs – in the OpenAPI format, for example – the API gateways can read them and ensure that only conformal requests get through. This prevents exploration attacks or forceful browsing attacks on APIs because of unprotected, undocumented end points or due to legacy end points or attributes (API9). It also allows the correct typing of the attributes on the gateway to be verified and implemented. If the specifications are practical and precise, they can be used to prevent injection attacks (API8).

CIAM systems are intended for the administration of identity and access rights. They authenticate the users (API2) and use standards such as OAuth, OpenID Connect or SAML to authorise them for access to applications and APIs (API5). This also allows the implementation of an overarching single sign-on (SSO) and the processing of standard tasks by means of user self-service.

Web application firewalls offer a multitude of protective mechanisms against known attacks such as injections, XSS or CSRF (API8). In addition, they include secure basic settings for HTTP headers and TLS (API7) as well as functions for certificate management, for example using Let's Encrypt [e]. For good API protection, it is important to make sure that the WAF analyses JSON objects effectively and is able to apply its rules to individual attributes. If this were not the case, an appropriate API security gateway would have to do the job instead. However, a number of API gateways focus on API management and neglect security aspects.

A dedicated security infrastructure facilitates monitoring, troubleshooting and forensic analyses (API10). When SIEM systems are used too, information on different components can be easily compiled and correlated. Moreover, alerting when anomalies occur permits the user to exit the reactive mode and gain valuable time when infiltration occurs.

To allow them to react quickly to previously known attacks and situations, many security products now apply machine


Browser
Smartphone
IoT
Ecosystem
Web Application Firewall
API Gateway
Identity & Access Management
API
API
Applications and APIs (on-premises or in the cloud)

Web application firewall, API gateway and IAM work hand in hand to ensure comprehensive API protection.

learning (ML). ML is difficult for individual developers to use as the methods require a lot of data and benefit from an overarching service perspective.

The duties of developers

Independently of external gateways, some tasks are always left up to the developers. In general, it can be said that they should develop software in a way that makes it secure without the need for upstream security services. Among other things, they should validate inputs regardless of whether a WAF checks for injection attacks. WAFs always have to achieve a compromise between false positives and false negatives, so they do not have a 100% detection rate. In addition, changes to the infrastructure can happen without the developer noticing. It is recommendable to ensure security by design and not as an afterthought. Vulnerability scanners integrated into the build pipeline can help to reveal existing vulnerabilities in the finished code.

Developers are obliged to take care of the technical authorisation of business objects. An API gateway knows the end points and is able to distinguish between GET and UPDATE calls. However, it is not familiar with the business objects and their attributes. For this reason, developers have to ensure that object IDs provided by the client are effectively permitted for authenticated users (API1). This check is also necessary when individual attributes are modified (API6). When minimising data delivery, developers should not rely on the client, but should filter out risky attributes on the API side instead (API3).

The handling of API keys is important too. They are not explicitly a means for authentication, instead only allowing the identification of technical clients such as mobile apps or SPAs. An application must not only approve accesses to APIs on the basis of API keys, but should also ensure solid user authentication and authorisation. Of course, public cloud storage is no place in which to keep API keys. If clients require hard-coded API keys, teams have to invest in client security in order to make attacks entailing the debugging or decompiling of the client code more difficult.

Online sources

[a] OWASP API Security Project www.owasp.org/index.php/OWASP_API_Security_Project

[b] Gartner-Studie: API Security: What You Need to Do to Protect Your APIs www.gartner.com/en/documents/3956746/api-security-what-you-need-to-do-to-protect-your-apis

[c] USPS Site Exposed Data on 60 Million Users krebsonsecurity.com/2018/11/usps-site-exposed-data-on-60-million-users/ OWASP Top Ten owasp.org/www-project-top-ten/

[d] Let's Encrypt letsencrypt.org/de/

APIs as the principal attack vector

In the next few years, APIs are set to become the principal places in which to attack Web applications. In response to this, OWASP has published a new and specialised API Security Top Ten. Some subjects in the new list, such as authentication and injection attacks, have been predominant in Web security since 2003, the year in which OWASP published the first Top Ten.

It is especially important to prevent the vulnerabilities in the context of APIs. Dedicated security products such as Web application firewalls, API gateways and CIAM systems can do a lot to ensure the protection of APIs. However, developers must not rely on products only. As before, they are responsible for certain subjects such as the technical authorisation of business objects and the secure handling of API keys. (ane@ix.de)

Martin Burkhart

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