SALT

OWASP API Security Top 10:

Insights from the API Security Trenches

E-BOOK

Why do you need to know about the OWASP API Security Top 10?

APIs have become critical for today's organizations. They connect modern applications, enable business innovation, and allow companies to meet their customers' increasingly high expectations.

The explosion in API use has not gone unnoticed by cybercriminals. In fact, APIs have become a primary target for attacks in recent years, in line with Gartner's prediction that "by 2022, API abuses will move from an infrequent to the most-frequent attack vector, resulting in data breaches for enterprise web applications." The analyst firm has since acknowledged that the prediction was right and recognizesd the need for dedicated API protections by placing API security in its own category in its latest security architecture reference.

With 94% of survey respondents in the most recent State of API Security report by Salt Labs admitting that they have experienced API security problems in production and Salt Labs' researchers finding a 400% increase in API attacks targeting Salt Security customers, it's unsurprising that API-related security incidents and breaches have been getting more and more attention in recent years. At the end of 2019, the Open Web Application Security Project (OWASP), an open-source community that focuses on improving the security of software, released the first-ever API Security Top 10 to raise awareness about the most common API security threats back in 2019. The top 10 has been updated in June 2023 to reflect the changing API security landscape.

This ebook explores the top vulnerabilities listed in the OWASP API Security Top 10, what they look like in the real world, and how you can protect your organization from the threats targeting your APIs and API-based applications.

Why is the OWASP API Security Top 10 so important?

The OWASP API Security Top 10 2023 published in June 2023 provides an updated version of the OWASP API Top 10 2019. While the two lists have many similarities, the changes introduced in 2023 are aimed at reflecting the evolving API security landscape and addressing new attack vectors that have been emerging in recent years.

With API attackers increasingly targeting the business logic in APIs and undertaking malicious activities that can go on for weeks and even months, being aware of the main threats faced by today's complex API ecosystems is a key step toward developing more mature and comprehensive API security strategies.

The OWASP API Security Top 10 provides the most up-to-date list of existing API security threats and highlights the need for dedicated API protections that go beyond traditional cybersecurity approaches.

OWASP API Security Top 10: inside each threat

API1:2023 Broken Object Level Authorization

Broken object level authorization (BOLA) represents around 40% of API attacks and is the most common API threat. But why is this the case?

As APIs frequently expose endpoints that handle object IDs, this creates a large potential attack surface. Object level authorization is an access control method that is typically implemented at the code level to verify a user's ability to access a certain object. Modern applications use a variety of intricate and pervasive authorization and access control systems. Developers frequently neglect to apply these checks before accessing an object, even when an application includes a robust infrastructure for authorization checks. Attackers can easily exploit API endpoints that are vulnerable to broken object level authorization by manipulating the ID of an object that is sent within an API request. These vulnerabilities are extremely common in API-based applications because the server component usually does not fully track the client's state. Instead, the server component usually relies on parameters like object IDs sent from the client, to decide which objects can be accessed.

Regardless of the level of sensitivity of the data, any unlawful access to data needs to be taken seriously and the fact is that automated static or dynamic testing cannot readily find these kinds of authorization problems.

Object level authorization checks should be implemented in every API endpoint that receives an object ID and executes any type of operation on the object. To ensure that the logged-in user has authorization to perform the desired action on the requested object, these checks should be carried out continuously over the course of a particular session.

BOLA authorization flaws can lead to data exfiltration as well as unauthorized viewing, modification, or destruction of data. Ultimately, BOLA can lead to full account takeover (ATO). Traditional security controls, such as WAFs and API gateways, can't detect these types of attacks because they cannot baseline normal API behavior or recognize unusual strings of behavior over time.

You must be able to recognize when an authenticated user is attempting to access another user's data without authorization in order to stop BOLA attacks. This requires the analysis of massive volumes of API traffic over time to create a detailed baseline of typical activity over millions of API calls.

What does Broken Object Level Authorization (BOLA) look like in real life?

Several high-profile automotive brands

One prominent example of BOLA vulnerabilities was found by researchers in early 2023.

In this instance, the researchers managed to find vulnerabilities in the API endpoints that several prominent automotive brands, such as Ford, Hyundai, Honda, Nissan, Rolls Royce, BMW Mercedes and Ferrari use for their vehicle telematics systems. The vulnerabilities found meant that an attacker could potentially lock, unlock, engine start, engine stop, precision locate, flash headlights, and honk vehicles, and even successfully takeover customer accounts and access personal identifiable information (PII).

The vulnerabilities were reported to the automotive manufacturers, and patched, but they could've had catastrophic consequences, had they been found by hackers first.

Verizon

Another public BOLA breach happened at Verizon, where a cyber researcher managed to expose the vulnerability by changing a contract number to access a customer account.

While authentication was needed to access the files, the researcher could successfully brute force the URL's GET parameters and modify one of the parameters to gain access to a different customer contract. This is a very difficult type of attack to detect and it allowed access to Verizon's entire database of contracts.

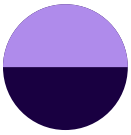
API2:2023 Broken Authentication

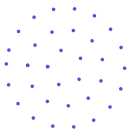
Attackers can easily target authentication processes, especially if they are fully exposed or accessible to the public. The second most frequent vulnerability reported by OWASP is broken user authentication, which enables attackers to utilize credential stuffing, stolen authentication tokens, and brute-force attacks to obtain unauthorized access to apps. Attackers are then able to control users' accounts, gain unlawful access to other users' data, and conduct unauthorized transactions. Technological issues, such as inadequate password complexity, missing account lockout criteria, overly long rotation times for passwords and certificates, or the usage of API keys as the only authentication method, can result in faulty authentication in APIs.

Traditional security controls cannot tell the difference between the various types of sophisticated attacks that target authentication, such as credential stuffing and credential cracking, since they are unable to follow attack traffic over time. To prevent sophisticated attacks that target authentication, an API security solution must be able to profile the typical authentication sequence for each API flow in order to detect abnormal behavior, such as missing credentials, missing authentication factors, or authentication calls that are out of order.

Attackers who can successfully take advantage of weaknesses in authentication procedures may be able to access another user's data without authorization and carry out illicit transactions using that user's account. Similarly, APIs can be specifically created for machine-to-machine connection or for direct API connectivity. All data that a machine identifies as authorized to view could be accessed by an attacker who compromises the authentication method or the authenticated session. There are also variations of broken authentication attacks that compromise workload authentication and server-side API metadata services.

What does Broken Authentication look like in real life?





Booking.com

In March 2023, researchers from Salt Labs, the research arm of Salt Security, identified several security vulnerabilities in the world-renowned vacation booking platform Booking.com. These authentication flaws could have enabled attackers to take over users' accounts, exfiltrate private account data, and cancel or book reservations and perform other actions on their behalf.

The flaws were found in the implementation of the Open Authorization (OAuth) social-login functionality utilized by Booking.com, which had the potential to affect users logging into the site through their Facebook account. The OAuth misconfigurations could have allowed for large-scale account takeover (ATO) on customers' accounts, enabling bad actors to manipulate platform users to gain complete control over their accounts, leak personal identifiable information (PII) and other sensitive user data stored internally by the sites, and perform any action on behalf of the user, such as booking or canceling reservations and ordering transportation services.

After discovering the vulnerabilities, Salt Labs' researchers followed coordinated disclosure practices with Booking.com, and all issues were remediated with no evidence that these flaws were ever exploited by malicious actors.

Parler

Another prime example of a broken authentication attack took place in January 2021 targeting social media platform Parler.

This was a politically charged incident, as Parler played an important part in enabling people to coordinate their actions during the storming of the Capitol on January 6, 2021. As a response to the event, digital activists scraped as much data as possible using Parler's APIs prior to it being shut down. Parler users experienced an alarming case of privacy erosion and Parler was effectively put out of business shortly after. The incident clearly exposed the fact that Parler APIs and data were inadequately secured in a number of ways, namely by allowing access without proper authentication.

When analyzing the Parler data breach, Salt Security found that Parler's authentication was at least partially absent. This flaw, along with other security vulnerabilities, allowed for the scraping of 70 TB of data from the platform, with at least one API endpoint requiring no authentication at all to access user data.

The shutdown of Parler was a valuable learning experience for social media services and any organization offering a public web service that stores and presents personally identifiable information (PII).

NEW

API3:2023 Broken Object Property Level Authorization

Broken Object Property Level Authorization merges attacks that happen by gaining unauthorized access to sensitive information by way of Excessive Data Exposure (previously listed as number 3 in the 2019 OWASP API Security Top 10) or Mass Assignment (previously in sixth place in the 2019 list). Both techniques are based on API endpoint manipulation to gain access to sensitive data. This new category deals with Object Properties level authorization as opposed to Object level authorization (number 1 on the list since 2019).

APIs usually expose endpoints that return all object's properties. This is particularly true for REST APIs. For other protocols such as GraphQL, the API may require crafted requests to specify which properties should be returned. Identifying these additional properties that can be manipulated requires effort, but is essential to prevent the exploitation of this type of vulnerability.

The main reason for introducing this new threat on the list is that, even if an API can enforce sufficient object-level authorization security measures, this might still not be enough to protect it. More specific authorization that covers the objects and their characteristics is often required. The varying access levels within an API object must also be taken into account, as an API object often has both a public property and a private one.

What does Broken Object Property Level Authorization look like in real life?

Twitter

According to a statement released by Twitter in August 2022, a broken object properly level authorization vulnerability was initially spotted by their bug bounty program in January 2022. As a result of the flaw, if someone submitted an email address or phone number to Twitter's systems, they would be informed of what Twitter account the submitted email addresses or phone number was associated with, if any. The issue was investigated and fixed by Twitter at the time with no evidence to suggest it had been exploited in the wild. However, in July 2022, it came to light that a bad actor had taken advantage of the issue before it was addressed by the company and was now offering to sell the data they compiled.

This very public incident brought to light the risk that this type of security vulnerabilities can pose, even for a well-established brand with vast resources and supposedly robust security programs.

3Fun

3Fun, an adults-only dating website, relied on client-side data filtering and had privacy filtering for personal data in place only at the application level. This meant that the APIs and URLs associated with users' private pictures and other personal details had no protections in place, meaning anyone could circumvent the privacy flag at an application level and access the data – as Pen Test Partners found out in 2019.

Additionally, the GPS locations of users were exposed since the filtering of the data was limited to the mobile app. Anyone could query the 3Fun server for the location of other users.

This vulnerability became even more scandalous because some of the GPS locations found by researchers pointed to the White House and the US Supreme Court.

BOLA, Broken User Authentication and Broken Object Property Level Authorization in real life

Peloton's API vulnerabilities' trifecta

What started out as a broken authentication vulnerability identified by Pen Test Partners at Peloton in 2021, soon became a fully-fledged nightmare for the sporting gear brand. The first vulnerability that was exposed pertained to the Peloton app's workout details API. Due to insufficient authentication, the API exposed personal details for workout participants, including their location, even if their profile was set as private. The user search API also showed an authentication flaw, where it was possible to search for any users on the Peloton app without being authenticated. The functionality could also be used by unauthenticated users to access user IDs and gather more information via the workout details APIs. The pen testers also found out that a data access API (GraphQL) also lacked proper authentication.

When contacted by the cybersecurity researchers that identified the broken authorization flaws, Peloton implemented authentication on the app for each API call and made it mandatory. However, they did not implement authentication at the object level, giving way for a BOLA vulnerability. In practice, this meant that, while users now had to create an account and log in to access the app, once logged in, they could still call any API endpoint and access other users' information, including PII, even for profiles marked as private.

The Peloton case is a great example of how an API flaw can ultimately trace back to inadequate authentication and authorization and result in the exposure of customers' private data.

While Peloton fixed the identified API flaws quickly, the company didn't adhere to its own documented vulnerability disclosure program, resulting in the disclosure of the breach within mainstream media by the security researcher and the pen test firm that alerted them.

API4:2023 Unrestricted Resource Consumption

The Unrestricted Resource Consumption vulnerability has replaced the previous number 4 in the OWASP API Security Top 10, Lack of Resources and Rate Limiting. However, while the name changed, this vulnerability remains the same overall.

Resources like the network, CPU, memory, and storage are used up by API calls. The user's input and the endpoint's business logic have a significant impact on the number of resources needed to fulfill a request. The size or quantity of resources that a client or user may request may not necessarily be constrained by APIs. This not only has the potential to negatively affect API server performance and cause Denial of Service (DoS), but it also makes APIs that support authentication and data retrieval vulnerable to brute-force and enumeration assaults, including token and credential cracking. The impact of this threat can be broken down into two components:

- Because there is no resource restriction, an attacker can overwhelm an application with a single API call, which will affect its performance and responsiveness or make it unresponsive. The term "application-level DoS" is sometimes used to describe this kind of assault. Yet, these attacks have an effect beyond availability. Additionally, they can put the system, application, or API at risk for excessive data leaks and authentication assaults.

- Due to the absence of rate restriction, an attacker can create and send several API queries at once to exhaust system resources, brute force login information, swiftly scan huge data sets, or exfiltrate significant amounts of data.

What does Unrestricted Resource Consumption look like in real life?

Poland's tax portal

A very popular example of this type of attack are distributed denial-of-service (DDoS) attacks targeting APIs. One recent example shows how Poland's key tax portal was rendered unavailable to Polish citizens due to an attack of this category, where hackers were able to exploit the resources that support the APIs behind the central tax service and make it unavailable for several hours.

With Polish government officials quickly pointing the finger at Russian hackers, in light of the Ukraine war, this attack highlights how today's social and political landscape makes cybersecurity, and API security in particular, even more crucial for businesses and government institutions alike.

SoundCloud

In 2020, a cyber research team at Checkmarx found that SoundCloud had not properly implemented rate limiting for the /tracks endpoint of their api-v2.soundcloud.com API.

Since no validation was performed for the number of track IDs in the IDs list, an attacker could manipulate the list to retrieve an arbitrary number of tracks in a single request and overwhelm the server.

Under normal conditions, the request issued by the SoundCloud WebApp should include 16 track IDs in the IDs query string parameter. The researcher was able to manipulate the list to retrieve up to 689 tracks in a single request causing the service response time to increase by almost nine times.

According to Checkmarx, "This vulnerability could be used to execute a Distributed Denial of Service (DDoS) attack by using a specially crafted list of track IDs to maximize the response size, and issuing requests from several sources at the same time to deplete resources in the application layer will make the target's system services unavailable."

API5:2023 Broken Function Level Authorization (BFLA)

Authorization issues are frequently the result of incorrectly configured or poorly implemented authorization. Because contemporary systems often have a wide variety of roles, groups, and user hierarchies, including sub-users and users with multiple roles, implementing suitable authorization procedures is a challenging undertaking. Distributed application architectures and cloud-

native design make this even more difficult. In this respect, BOLA and broken function level authorization (BFLA) are quite similar, except BFLA targets API functions rather than the objects that APIs interact with. While attacking APIs, attackers would try to take advantage of both flaws in order to increase their privileges either horizontally or vertically.

Attackers find these problems in APIs because, even in REST architectures, API calls are structured and predictable. Even without API documentation or schema definitions, this can be accomplished by deconstructing client-side code and eavesdropping on application traffic. Moreover, some API endpoints are accessible to common, non-privileged users, making it simpler for attackers to find them.

These flaws can be exploited by sending legitimate API requests to an API endpoint that an attacker should not have access to or by intercepting and manipulating API requests originating from client applications. For example, an attacker could change an HTTP method from GET to PUT or alter a query parameter or message body variable such as changing the string “users” to “admins” in an API request.

Attackers can access restricted resources, hijack another user’s account, create or delete accounts, or escalate privileges to acquire administrative access by taking advantage of broken function level authorization flaws.

What does Broken Function Level Authorization (BFLA) look like in real life?

Reddit

In July 2022, a researcher at HackerOne found that after an ad campaign was created on Reddit’s site, there needed to be approval from Reddit admins to verify that a payment was verified. However, by sending a certain PATCH request regarding the ad, the researcher was able to bypass this admin approval and simply activate the team’s own ad campaign, which broke the standard access control.

This vulnerability meant that by sending a PATCH request, any user could approve their own ad campaigns without paying for them with critical business implications for Reddit in lost ad revenue.

The HackerOne researcher earned a $5000 bounty from Reddit for finding and communicating this flaw, which allowed Reddit to fix the issue and

potentially save millions if the BFLA vulnerability had been exploited by an attacker.

New Relic

In 2018, researcher Jon Bottarini found that a user could make changes to alerts on Synthetics monitors at software vendor New Relic without the proper permissions to do so. In fact, even a restricted user could make changes with no permissions at all because of a privilege escalation weakness that was present in the product at that time. To explore this vulnerability, a hacker simply needed to send a legitimate request to an API endpoint that was otherwise not visible to the restricted user.

As part of his security research, Bottarini captured the traffic of a privileged session using an intercepting proxy tool. This traffic included a POST request to an API endpoint and function that creates alerts on Synthetics monitors. He found that you could trap a GET request from the non-privileged session, retain the tokens and cookies for that restricted user, and alter the remainder of the trapped request to resemble the privileged POST request. This manipulation of API traffic to access functionality not visible in the UI is a common technique attackers use to exploit function-level authorization weaknesses and achieve privilege escalation.

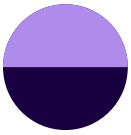
API6:2023 Unrestricted Access to Sensitive Business Flows

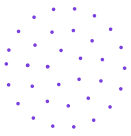
This threat has replaced Mass Assignment as number 6 on the OWASP API Security Top 10 list. It occurs when an API exposes a business flow without compensating for how the functionality could cause harm if used excessively through automation.

To exploit this vulnerability, a bad actor will need to understand the business logic behind the API in question, find sensitive business flows and automate access to these flows in a way that can harm the business.

This issue usually stems from a lack of holistic view of an API. Understanding which business flow an API endpoint exposes and how sensitive that business flow is is essential in preventing this vulnerability. An API endpoint is vulnerable to this risk if it exposes a sensitive business flow, without appropriately restricting access to it.

Common examples of sensitive business flows and risk of excessive access associated with them include:





- An attacker can buy all the stock of a high-demand item at once and resell for a higher price.
- A bad actor can create a comment/post flow, spamming a company's system.
- An attacker can reserve all the available slots for a given service and prevent other users from using the system.

This type of attack is notoriously hard to detect and protect against. Attacks in this category derive from a series of requests, in which each individual request is entirely legitimate. This attack can be detected only with a security architecture that can look at a series of API requests over time, which depends on cloud-scale storage of API traffic. Proxy architectures like a WAF or short analysis windows such as on-premise API security solutions cannot catch this type of attack.

What does Unrestricted Access to Sensitive Business Flows look like in real life?

Airline Company

A good example of how an attacker could exploit an Unrestricted Access to Sensitive Business Flows vulnerability would be by booking 90% of the seats on a flight online, taking advantage of the fact that the airline would charge no cancellation fee.

The attacker could then cancel all tickets simultaneously at no expense just a few days before the flight date, forcing the airline to put the tickets back on sale at a discounted price in order to fill the flight. The malicious user would then be able to buy a ticket at a much cheaper price, benefitting from the discounted price and causing financial damage to the airline.

NEW

API7:2023 Server Side Request Forgery

Server Side Request Forgery (SSRF) can occur when a user-controlled URL is passed over an API and is honored and processed by the back-end server. The risk for the environment materializes if the back-end server tries to connect to the user-supplied URL, which opens the door for SSRF.

SSRF can come in different shapes and forms, which include:

- The back-end server establishes a connection to a domain outside the control of the attacker, and while doing so, it can reveal internal credentials that could be used to intensify an attack.
- A port scan service discovery attack against the back-end server is possible since it connects to its loopback interface over a variety of TCP ports.
- The back-end server links to internal services that an attacker would not otherwise be able to access, increasing the attack surface and allowing for more chained attack paths.

As a result of a successful SSRF attack, attackers can gain access to internal network resources within a web-based environment, compromising security mechanisms within the web service.

What does Server Side Request Forgery (SSRF) look like in real life?

LEGO

Research published by Salt Labs in December 2022 found that it was possible to gain access to the internal network resources of a major LEGO-owned website, which could potentially compromise the whole security infrastructure at this web service.

Salt Labs' findings show that the security vulnerabilities found at LEGO's online services could have allowed an attacker to manipulate service users to gain complete control over their accounts, leak PII and other sensitive data stored internally by the service and gain access to internal production data, which could lead to full compromise of the company's internal servers.

The issues were disclosed to the security team at the LEGO Group and further testing showed that the issues have since been resolved.

API8:2023 Security Misconfiguration

The security misconfiguration threat represents a catch-all for a variety of security setup errors that frequently have a detrimental influence on API security as a whole and unintentionally expose vulnerabilities. Insecure default configurations, incomplete or ad-hoc configurations, open cloud storage, incorrect HTTP headers, unneeded HTTP methods, excessively

permissive Cross-Origin resource sharing (CORS), and verbose error messages are a few instances of security misconfigurations.

During their reconnaissance phase, attackers can use security flaws to learn about the application and API components. Specific faults, such as stack trace problems, can reveal private user information and system specifics that can help an attacker locate exploitable technologies, such as out-of-date or improperly configured web and application servers, during their reconnaissance phase. Attackers often use misconfigurations as a launching pad for assaults on APIs, like in the case of an authentication bypass brought on by improperly configured access control systems.

Although there are many automated tools available to find and exploit common or known misconfigurations like unused services or outdated settings, where you find them in the technology stack varies widely. Widely used vulnerability scanners can only check a running server for published software flaws and known vulnerabilities, which are typically listed as CVE IDs. They do not, however, give a complete picture because errors can occur in the underlying code, in dependence on outside parties, or in interactions with other enterprise architecture. Thus, organizations frequently use a flurry of security testing tools when constructing pipelines in order to catch as much as they can before the production release.

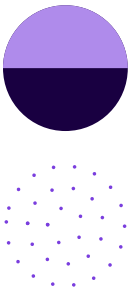
What does a Security Misconfiguration look like in real life?

IBM Cloud

In December 2021, the Wiz research team discovered a serious security misconfiguration in IBM's cloud infrastructure platform. The vulnerability allowed Wiz researchers to access sensitive information, including API keys and passwords of IBM Cloud users who had used the affected service.

The security flaw was caused by an insecure default setting in a widely used open-source software package called "etcd," which was used by IBM Cloud to manage its Kubernetes clusters. This misconfiguration allowed anyone with access to the internet to access and download the entire database containing the credentials of all IBM Cloud customers who used the affected service.

The IBM Cloud flaw highlights the severity of security misconfigurations in APIs, particularly in cloud-based services that host sensitive data by demonstrating that even a small misconfiguration can have severe consequences, such as data breaches.



Wiz disclosed their findings to IBM Cloud in a three-part report and stated that the issues were investigated and fixed promptly by their security team.

Capital One

In 2019, Capital One suffered a chained attack, for which the primary vector was a misconfigured WAF. The WAF was not appropriately configured for Capital One's AWS environment and an attacker was able to bypass the WAF's content inspection and message filtering using a well-crafted injection that targeted the backend AWS cloud metadata service.

Gathering metadata that should only be available to running workloads, the attacker was able to compromise several systems within Capital One's AWS cloud environment. This is commonly known as a server-side request forgery attack.

API9:2023 Improper Inventory Management

Improper Inventory Management has replaced Improper Assets Management as number 9 in the OWASP API Security Top 10 and, while the name has been changed to emphasize the importance of an accurate and up-to-date API inventory, the threat remains the same.

Understanding potential exposure and risk depends on maintaining an accurate, complete and up-to-date API inventory. Older API versions that need to be decommissioned are difficult to find because of unexpected holes in the API attack surface caused by an outdated or incomplete inventory. Similar to how faulty documentation makes it difficult to find vulnerabilities that need to be fixed, it also exposes hazards like unknown disclosure of sensitive data.

Security tools often do not monitor or defend unknown APIs, also known as shadow APIs, and abandoned APIs, also known as zombie APIs. Even well-known API endpoints may have shadow parameters or unspecified or undocumented functionality. These APIs and the infrastructure that supports them are frequently vulnerable to attacks because of this.

Via outdated, unpatched, or vulnerable versions of APIs, attackers may obtain unauthorized access to sensitive data or even full server access.

What does Improper Inventory Management look like in real life?

Optus

In September 2022, media reports indicated that the second largest telecommunications company in Australia, Optus, had suffered a data breach caused by a security vulnerability in one of the company's APIs and resulting in the exposure of 11.2 million customer records, including personal identifiable information. The API targeted by the attackers was used to manage customer accounts, and the vulnerability allowed unauthorized access to sensitive information, including names, addresses, and phone numbers.

The Optus breach highlights the importance of proper API inventory management, specifically the need for regular auditing and risk assessments. When APIs are not properly managed, vulnerabilities can go undetected, and unauthorized access can occur. In this case, the vulnerability was not discovered until it had already been exploited, putting thousands of customers' personal information at risk.

Adding to the severe reputational damage caused by the high-profile breach, Optus has reportedly put aside A$140 million to cover the costs of the incident.

NEW

API10:2023 Unsafe Consumption of APIs

Unsafe Consumption of APIs has come to replace Insufficient Logging and Monitoring as number 10 in the OWASP API Security Top 10 and it contains a mix of two common API issues:

- The consumption of API data itself, which was largely addressed in the Injection category of the 2019 list section. However, the Unsafe Consumption of APIs category goes further than the previous Injection threat to include attacks that are not explicitly injection-related, such as de-serialization issues, some types of desync attacks, and others. What all of these attacks have in common is the fact that a back-end service is too permissive when accepting user-controlled input carried over APIs and sometimes even blindly uses them without applying any proper validations.
- Integrations, which can include any third-party service or functionality embedded into the API implementation or in their supporting back-end

services. While it is closely related to the data consumption issue, this issue deals with another type of attack that abuses integration - a weak link in almost every modern system design. Integrations are usually written by a third party and often contain a large amount of codebase that can be very complex to understand. They can, however, be applied to your own service in just a few clicks or with a few lines of code. When a web service contains large amounts of unverified, third-party code, attackers can try to leverage this vulnerability to access sensitive information.

Although not exclusive to this category, API-based supply-chain attacks serve as a good example of this category's danger.

What does Unsafe Consumption of APIs look like?

Log4Shell

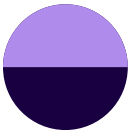
The Log4Shell incident made headlines in December 2021. This critical vulnerability allowed users to run arbitrary code on almost every web service using the very popular Apache Log4j logging library, potentially leading to full control of the system.

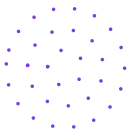
The vulnerability behind the attack was caused by the unsafe consumption of APIs in Log4j, specifically the ability to deserialize user-supplied data without proper validation. Attackers could exploit this vulnerability by sending requests containing malicious code that could then be executed on the server.

Thanks to the widespread use of the affected logging library, this high-profile attack highlighted the importance of properly consuming APIs in software development and the need to validate and sanitize user input, especially when deserializing data.

Fishpig

In September 2022, eCommerce malware detection platform Sansec discovered that Fishpig had been hacked. The attack targeting the popular Magento-Wordpress integrations' vendor consisted in injecting malware code into License.php which is normally used to validate a Fishpig license. This meant that when a Magento staff user visited the Fishpig control panel in the Magento backend, the malware downloaded a Linux binary that looked like a license asset, but was actually a Rekoobe trojan that granted administrator store access to the hackers.





This attack was possible due to a vulnerability whereby the Fishpig Magento WordPress Integration extension failed to properly sanitize user input before using it in API requests, making the extension vulnerable to this type of API parameter tampering attack.

Shortly after the attack was made public, Fishpig acknowledged the incident and published a status page recommending that customers upgraded or reinstalled all FishPig modules to ensure all extensions were secured.

How can Salt protect your APIs against the OWASP API Security Top 10 threats?

With API attacks on the rise, organizations need to take a new approach to API security. Traditional security tools that use only authentication, authorization and encryption to protect web applications are not enough to protect APIs against the OWASP Top 10 vulnerabilities, and API gateways and WAFs can do very little to stop increasingly sophisticated attacks. It's also important to remember that there are elements of API security that can't be addressed in code or even tested for and validated prior to deployment. Many elements of an API attack are only revealed in runtime.

The Salt Security API Protection Platform secures the APIs at the heart of today's complex and widespread application ecosystem. The platform collects API traffic across the whole application landscape, making use of big data, AI, and ML to discover all APIs and their exposed data, stop attacks during the reconnaissance stage and eliminate vulnerabilities at the source.

Salt enables today's organizations to do three key things to protect their APIs throughout their entire lifecycle:

Discover all APIs and the data they expose

Salt automatically inventories all APIs, including shadow and zombie APIs, across all application environments. It also shows all instances where APIs expose sensitive data, including PII. This continuous and automated discovery ensures full API protection, even as environments change and scale.

Stop API attacks with runtime protection

Salt's big data and patented artificial intelligence (AI) technology allow organizations to pinpoint threats and stop attacks by baselining legitimate

behavior and identifying attackers in real time, preventing them from advancing during the reconnaissance stage. The Salt platform correlates all API activities back to a single entity, sends one consolidated alert to avoid alert fatigue, and blocks the attacker.

Improve overall API security posture

Salt focuses on discovery and threat protection at runtime because they provide the greatest risk reduction in the shortest time. But all organizations should look to harden their APIs, removing gaps learned at runtime. The Salt platform stops attackers but also uses them like pen testers, capturing their minor successes during reconnaissance. Salt packages up those insights into prescriptive instructions for dev teams to apply to update their APIs and improve their security posture.

SALT

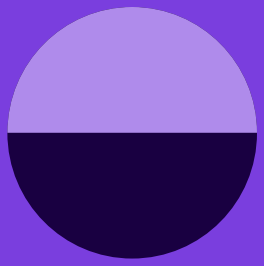
Salt Security protects the APIs that form the core of every modern application. Its patented API Protection Platform is the only API security solution that combines the power of cloud-scale big data and time-tested ML/AI to detect and prevent API attacks. By correlating activities across millions of APIs and users over time, Salt delivers deep context with real-time analysis and continuous insights for API discovery, attack prevention, and shift-left practices. Deployed in minutes and seamlessly integrated within existing systems, the Salt platform gives customers immediate value and protection, so they can innovate with confidence and accelerate their digital transformation initiatives.

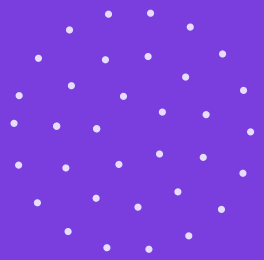
Request a demo today!

info@salt.security

www.salt.security

EB-OWASP-06062023





SALT

Securing your Innovation.