Refreshing its inaugural 2019 list, the 2023 API Security Top-10 list highlights the ten most common and serious security risks created when developing applications that expose or use APIs. Issues such as Broken Object-Level Authorization, a superset that includes IDOR vulnerabilities, remains the same from the prior list. Yet, new categories—or reorganized categories—now highlight issues overlooked in the past, such as Server-Side Request Forgery (API7:2023) and Unrestricted Access to Sensitive Business Flows (API6:2023).

“By nature, APIs expose application logic and sensitive data such as Personally Identi able Information (PII) and because of this, APIs have increasingly become a target for attackers,” the OWASP group stated in its announcement[9]. “Without secure APIs, rapid innovation would be impossible.”

API Security Cheat Sheet

De nitions

API Endpoint—The point of communication between two systems, typically a URL of a container or server running a microservice. Using an URL, an application or developer can request information from the server or execute an action on the API server or microservice.

API-Related Tra c—Internet tra c that consists of an HTTP or HTTPS request and has a response content of XML or JSON, indicating that data is being passed to an application, usually through SOAP, WSDL, a REST API, or gRPC (see below)

Dynamic Application Security Testing (DAST)—The process of analyzing an application or API server by using the interface, whether the user interface for an application, a web front end for a web application, or URLs for API endpoints. At type of black-box testing, this approach evaluates an application from the “outside in” by attacking an application in the same way as an attacker, usually without knowledge of internal processes.

Static Application Security Testing (SAST)—An approach to application security that scans the source, binary or byte code for recognized patterns of errors or vulnerabilities. Sometimes referred to as white-box testing, SAST uses an “inside-out” approach that identi es potential vulnerabilities and errors that may, or may not, be exploitable by an external attacker. Lightweight static tools can provide real-time feedback to developers in their IDE.

SOAP/WSDL—An XML-based protocol for creating Web APIs. SOAP is the protocol itself and WSDL (Web Service De nition Language) is the format used to formally describe services. Due to the heavy overhead, this API style has become unpopular for new developments.

SOAP2h6 TD [(RES)10EESxta40e0aD (de IDE.((er)83)7n Sech)is APdid 0 -2.736 TD [(SO)3sluates anrahtw.t6(esr bdba(xSEES) >>5

2023 API Security Top 10	Analogous 2019 API Security Entry
API1:2023—Broken Object Level Authorization	

How to Prevent It as a Developer?

Developers prevent insecure access to objects by enforcing strict controls, assigning unpredictable user identi ers to dissuade enumeration of accounts, and checking object-level authorization for every function that accesses a data source. Developers should encapsulate such checks, especially if based on user input, to remove the possibility that inadvertent errors could undermine security. Application-security and operations professionals should require authorization checks for each request to backend data.

How Can Fortify Help?

Fortify SAST and DAST by OpenText can detect a broad range of vulnerabilities in the Insecure Direct Object Reference (IDOR) category. IDOR can include vulnerabilities such as Directory Traversal, File Upload, and File Inclusion. More generally, IDOR also includes classes of vulnerabilities where identi ers can be modi ed via URL, Body, or Header manipulation. The system will alert developers to cases where the user can directly choose the primary key in the API request for a database or storage container, a problem that often leads to this class of vulnerabilities. The system will also warn when an expected authorization check is missing.

API2:2023—Broken Authentication

What Is It?

Authorization checks limit access to data based on speci c roles or users, but those limitations are not su cient to protect systems, data, and services. Developers and application-security teams also must properly implement capabilities to check user identity through authentication. Despite the critical nature of authentication, the components are often poorly implemented or improperly used—the root causes of Broken User Authentication. Broken user authentication allows attackers the ability to assume other user's identities temporarily or permanently by exploiting insecure authentication tokens or compromising implementation aws.

What Makes an Application Vulnerable?

This common and easy-to-exploit

In February 2022, a miscon gured cloud storage bucket left 1 GB of sensitive data from email

In addition, Fortify SAST has knowledge of the most important JSON and XML serialization and deserialization mechanisms. Using this, the tool can detect code that does not properly deserialize the domain transfer objects (DTOs), which could allow mass assignment of its attributes. Some cases of information exposure and mass assignment can also be detected using Fortify WebInspect. Finally, some countermeasures can be implemented through adding rules to the web application rewall (WAF).

API4:2023—Unrestricted Resource Consumption

Applications that do not limit the resources assigned to satisfy a request can be vulnerable, including those that fail to restrict allocable memory, number of les or processes accessed, or the allowed rate of requests, among other attributes.

What Is It?

APIs expose many useful business functions. To do so, they use computing resources like database servers or may have access to a physical component through operational technology. Because systems have a nite set of resources to respond to API calls, attackers can specially craft requests to create scenarios that result in resource exhaustion, denial of service, or increased business costs. In many cases, attackers can send API requests that tie up signi cant resources, overwhelming the machine or bandwidth resources and resulting in a denial-of-service attack. By sending repeated requests from di erent IP addresses or cloud instances, attackers can bypass defenses designed to detect suspicious spikes in usage.

What Makes an Application Vulnerable?

API requests trigger responses. Whether those responses involve accessing a database, performing I/O, running calculations, or (increasingly) generating the output from a machine-learning model, APIs use computing, network, and memory resources. An attacker can send API requests to an endpoint as part of a denial-of-service (DoS) attack that, rather than overwhelm bandwidth—the goal of a volumetric DoS attack—instead exhaust CPU, memory, and cloud resources. Applications that do not limit the resources assigned to satisfy a request can be vulnerable, including those that fail to restrict allocable memory, number of les or processes accessed, or the allowed rate of requests, among other attributes.

The server processing APIs needs to have limits in place to prevent excessive allocation of memory and workloads, excessive requests for API-triggered operations, or excessive charges for a third-party service without spending limits.

A common attack is to modify the arguments passed to the API endpoint, such as increasing the size of the response and requesting millions of database entries, rather than, say, the rst ten:

```
/api/users?page=1&size=1000000
```

In addition, if the attacker can access a backend service that charges for usage, resource consumption attacks can be used to run up charges for the application owner. Another OWASP example points to a reset-password feature that uses an SMS text message to verify identity and which could be called thousands of times to increase expenses for the victim.

```
POST /sms/send_reset_pass_code

Host: willyo.net
{
 "phone_number": "6501113434"
}
```

How to Prevent It as a Developer?

DevSecOps teams should design a standard approach to authentication and authorization that prevents access to requests by default, enforcing a default of "deny all." From this default, always apply the principle of least privilege when determining access for roles/groups/users. Developers should ensure that authentication and authorization are in place for all relevant HTTP verbs/methods (e.g., POST, GET, PUT, PATCH, DELETE) related to each API endpoint. Irrelevant verbs should be disallowed. In addition, developers should implement a base class for administrative access and management, using class inheritance to ensure that authorization controls check the user's role before granting access. All critical administrative functions should use the authorization mechanism to prevent privilege escalation.

DevSecOps teams should design a standard approach to authorization and authentication that prevents access to requests by default, enforcing a default of "deny all."

How Can Fortify Help?

By combining the static code and API analysis features of Fortify SAST with the runtime checks of the Fortify WebInspect dynamic application security testing (DAST) suite, DevSecOps teams can evaluate their application for broken function-level authorization issues and continuously test production code for security weaknesses before deploying. To detect Broken Object Function Authorization issues, Fortify SAST uses rules specifying when an authorization check would be expected in certain programming languages and frameworks, and the absence of such a check is reported.

API6:2023—Unrestricted Access to Sensitive Business Flows

What Is It?

From sneakerbots to ticket bots, attacks on the inventory of online retailers through their APIs has become a signi cant problem for e-commerce sites. By understanding the business model and the application logic, an attacker can create a series of API calls that can automatically reserve or purchase inventory, thus preventing other, legitimate consumers from gaining access to the businesses' products or services. Any API that allows access to a business process can be used by an attacker to impact the business and falls under the de nition of Unrestricted Access to Sensitive Business Flows.

What Makes an Application Vulnerable?

Application control and logic ows are the heart of any online businesses, and as companies move more of their operations to the cloud, those ows can be exposed and exploited. This excessive access may harm the business, when attackers automate the purchase of products, create bots for leaving comments and reviews, or automate the reservation of goods or services.

Application control and logic ows are the heart of any online businesses, and as companies move more of their operations to the cloud, those ows can be exposed and exploited. This excessive access may harm the business.

If an application o ers an endpoint that has access to the company's business ow without limiting access to the business operations behind the endpoint, then the application will be vulnerable. Protections include limiting the number of access attempts from a single device through ngerprinting, detecting whether the activity originates from a human actor, and detecting whether automation is involved.

Attack Examples

When Taylor Swift tickets went on sale on Ticketmaster in November 2022, 1.5 million customers had pre-registered, but more than 14 million requests—including three times as much bot tra c—swamped the purchasing links and APIs as soon as ticket sales opened. The site crashed, preventing many customers from purchasing tickets.[19]

The onslaught of reseller bots resembled those that ruined the launch of the PlayStation 5 in November 2020. Supply-chain issues had already limited supply prior to the launch of the latest Sony gaming console, but the automated bots made nding available units even harder and led to astronomical resale prices. In one e-commerce site's case, the number of "add to cart" transactions grew from an average of 15,000 requests per hour to more than 27 million, using the store's API to directly request products by SKU number.[20]

How to Prevent It as a Developer?

Developers should work with both the business-operation and engineering teams to address issues of potential malicious access to business- ows. Business teams can identify which ows are exposed through APIs and conduct threat analyses to determine how attackers could abuse those endpoints. Meanwhile, developers should work with engineering operations as part of a DevOps team to establish additional technical defensive measures, such as using device ngerprinting to prevent automated browser instances from overwhelming and identifying patterns in behavior that di erentiate between human and machine actors.

Operations teams should also review any APIs designed to be used by other machines, such as for B2B use cases, and ensure that some defenses are in place to prevent attackers from exploiting machine-to-machine interactions.

How Can Fortify Help?

Catching vulnerable and sensitive business ows often relies on doing the basics. Companies need to document and track all of their functioning APIs and determine which ones expose sensitive processes and data to potential attackers. Application logic also needs to be analyzed for logic aws that could be exploited by attackers.

Overall, preventing Unrestricted Access to Sensitive Business Flows is more about a holistic approach to application security and less about nding a speci c technology.

API7:2023—Server Side Request Forgery

What Is It?

Backend servers handle requests made through API endpoints. Server-Side Request Forgery (SSRF) is a vulnerability that allows an attacker to induce a server to send requests on their behalf and with the server's level of privilege. Often the attack uses the server to bridge the gap between the external attacker and the internal network. Basic SSRF attacks result in a response returned to the attacker, a far easier scenario than Blind SSRF attacks, where no response is returned, leaving the attacker with no con rmation whether the attack was successful.

19. Steele, Billy. "Ticketmaster knows it has a bot problem, but it wants Congress to x it." Engadget. News Article. 24 Jan 2023. www.engadget.com/ticketmaster-live-nation-senate-judiciary-aP 41 >>BDC -0

What Makes an Application Vulnerable?

Server-Side Request Forgery (SSRF) aws essentially are a result of a lack of validation of user-supplied input. Attackers are able to craft requests and include a URI that supplies access to the targeted application.

Modern concepts in application development, such as webhooks and standardized application frameworks, make SSRF more common and more dangerous, according to OWASP.

In an example cited by OWASP, a social network that allows users to upload pro le pictures could be vulnerable to SSRF, if the server does not validate arguments sent to the application. Rather than a URL pointing to an image, such as:

In 2022, a vulnerability management rm discovered that 12,000 cloud instances hosted on Amazon Web Services and 10,500 hosted on Azure continued to expose Telnet, a remote access protocol considered “inappropriate for any internet-based usage today,” according to a 2022 report.[25] The inclusion of unnecessary and insecure features undermines these security of the APIs and applications.

How to Prevent It as a Developer?

DevSecOps teams need to understand the steps necessary to create secure con gurations for their applications and use an automated development pipeline to check con guration les before deployment, including regular unit tests and runtime checks to continuously check the software for con guration errors or security problems. Security-as-code can help, by making con gurations repeatable and giving application-security teams the ability to set standard con guration sets for speci c application components.

Security-as-code can help, by making con gurations repeatable and giving application-security teams the ability to set standard con guration sets for speci c application components.

As part of their secure development lifecycle, developers and operations teams should:

- Establish a hardening process that simpli es the repeatable creation and maintainance of a secure application environment,
- Review and update all con gurations across the API stack to incorporate the new standard consistently, and
- Automate the assessment of the e ectiveness of the con guration settings across all environments.

How Can Fortify Help?

Fortify SAST can check con gurations during the development process and spot many types of weaknesses. Because Security Miscon gurations occur at both the application-code level and at the infrastructure level, di erent Fortify products can be used to catch miscon gurations.

Fortify SAST scans can check application code for miscon guration issues. During the static analysis check, Fortify SAST can evaluate con guration les for security errors, including those for Docker, Kubernetes, Ansible, Amazon Web Services, CloudFormation, Terraform, and Azure Resource Manager templates.

Con guration errors can also be caught during runtime. Fortify WebInspect allows DevSecOps teams to regularly test for common security miscon gurations. One of the biggest strengths of DAST scanning is that it runs on the application server in a con gured environment, which means that the full environment—application, server, and network—are tested all at once, giving the dynamic analysis platform a comprehensive view of the production environment is con gured.

25. Beardsley, Todd. “2022 Cloud Miscon gurations Report.” Rapid7. PDF Report. p. 12. 20 Apr 2022. Accessed through: www.rapid7.com/blog/post/2022/04/20/2022-cloud-miscon gurations-report-a-quick-look-at-the-latest-cloud-security-breaches-and-attack-trends/.

API9:2023—Improper Inventory Management

What Is It?

Like most software assets, APIs have a lifecycle, with older versions replaced by more secure and e cient APIs or, increasingly, using API connected to third-party services. DevSecOps

How to Prevent It as a Developer?

DevSecOps teams should document all API hosts and focus on maintaining visibility into the data ows between APIs and third-party services. The primary way to prevent Improper Inventory Management is the detailed documentation of the critical aspects of all API services and hosts, including information on what data they handle, who has access to the hosts and data, and the speci c API versions of each host. Technical details that should be documented include the authentication implementation, error handling, rate limiting defenses, the cross-origin resource sharing (CORS) policy, and details of each endpoint.

The signi cant volume of documentation is di cult to manage manually, so generating documentation through the continuous integration process and using open standards is recommended. Access to API documentation should also be limited to those developers who are authorized to use the API.

During the application building and testing phases, developers should avoid using production data on development or staged versions of the application to prevent data leaks. When new versions of APIs are released, the DevSecOps team should do a risk analysis to determine the best approach to upgrading applications to take advantage of increased security.

How Can Fortify Help?

Organizations can manage, monitor, secure, and document their API usage using the NetIQ Secure API Manager by OpenText, which allows application-security teams to maintain an up-to-date inventory of API assets. The Secure API Manager provides an authoritative repository where your DevSecOps team can store and manage all of the APIs used by the organization, allowing an easy-to-manage life cycle from API development to retirement. The software helps improve compliance with regulations and licensing by allowing detailed analytics.

API10:2023—Unsafe Consumption of APIs

What Is It?

With the increasing use of native cloud infrastructure to create applications, APIs have become the point of integration between application components. However, the security posture of third-party services accessed through APIs is rarely clear, allowing attackers to determine on which services an application relies and whether any of those services have security weaknesses. Developers tend to trust the endpoints that their application interacts without verifying the external or third-party APIs. This Unsafe Consumption of APIs often leads to the application's reliance on services that have weaker security requirements or lack fundamental security hardening, such as input validation.

What Makes an Application Vulnerable?

Developers tend to trust data received from third-party APIs more than user input, although

The API Security Top-10 Is Not Su cient!

For cloud-native developers speci cally focused on creating APIs to o er services to other parts of an application, internal users, or for global consumption, the OWASP API Security Top 10 list is an important document to read and understand.

However, the OWASP API Security Top 10 is not a standalone document. Developers also need to make sure that they utilize other sources of best practices, such as the OWASP Top 10, that are relevant to their current application and architecture. Common application vulnerabilities -SQL injection, data exposure, and security miscon guration- continue to be common ways that cyber threat groups can compromise corporate infrastructure and should be remediated quickly. In addition, some API-based applications, such as mobile apps, require di erent appsec hardening steps than a stand-alone web-app, and di erent from what may be required for connect and IoT devices. Overall, the API Security Top 10 list is important, but it remains only a facet of the overall secure software development lifecycle. The list, and the OWASP Top 10 list, should be used in conjunction with any other relevant standards and best practices that are required for the solution under analysis.

The OWASP API Security Top-10 is crucial for cloud-native developers building APIs. Yet, addressing common application vulnerabilities like SQL injection, data exposure, and security miscon guration should take priority, as they are frequently exploited by cyber threats. The API Security Top-10 is an essential part of secure software development but should be secondary to addressing general application vulnerabilities.

Conclusion

As applications increasingly rely on cloud infrastructure, web application programming interfaces (APIs) have become the foundation of the Internet. Companies typically have hundreds, if not thousands, of API endpoints in their environment, dramatically increasing their attack surface and exposing applications to a variety of weaknesses.

The release of the 2023 OWASP API Security Top 10 list is a good starting point for companies and developers to educate themselves on the risks of API-based infrastructure and to assess their own applications. Along with the more well-known Application Security Top-10 list, the pair of rankings can help DevSecOps teams toward developing a holistic approach to the overall security of their applications.

DevSecOps teams need to be aware of the security implications of APIs, how to reduce an implementation’s vulnerabilities and security weaknesses, and how to harden their development pipeline and the resulting API server to make it more di cult for attackers to compromise an application through its APIs.

Where to Go Next

Here are the products mentioned in this document:

- Fortify API Security
- Fortify Static Code Analyzer (SAST)
- Fortify WebInspect (DAST)
- NetIQ Secure API Manager

Additional Resources

- OWASP Top 10 API Security Risks—2023
- Gartner Magic Quadrant fo Application Security Testing
- Fortify Code Security Webinar Series
- Fortify Application Security

Connect with Us
www.opentext.com

opentext™ | Cybersecurity

762-000081-004 | O | 11/23 | © 2023 Open Text