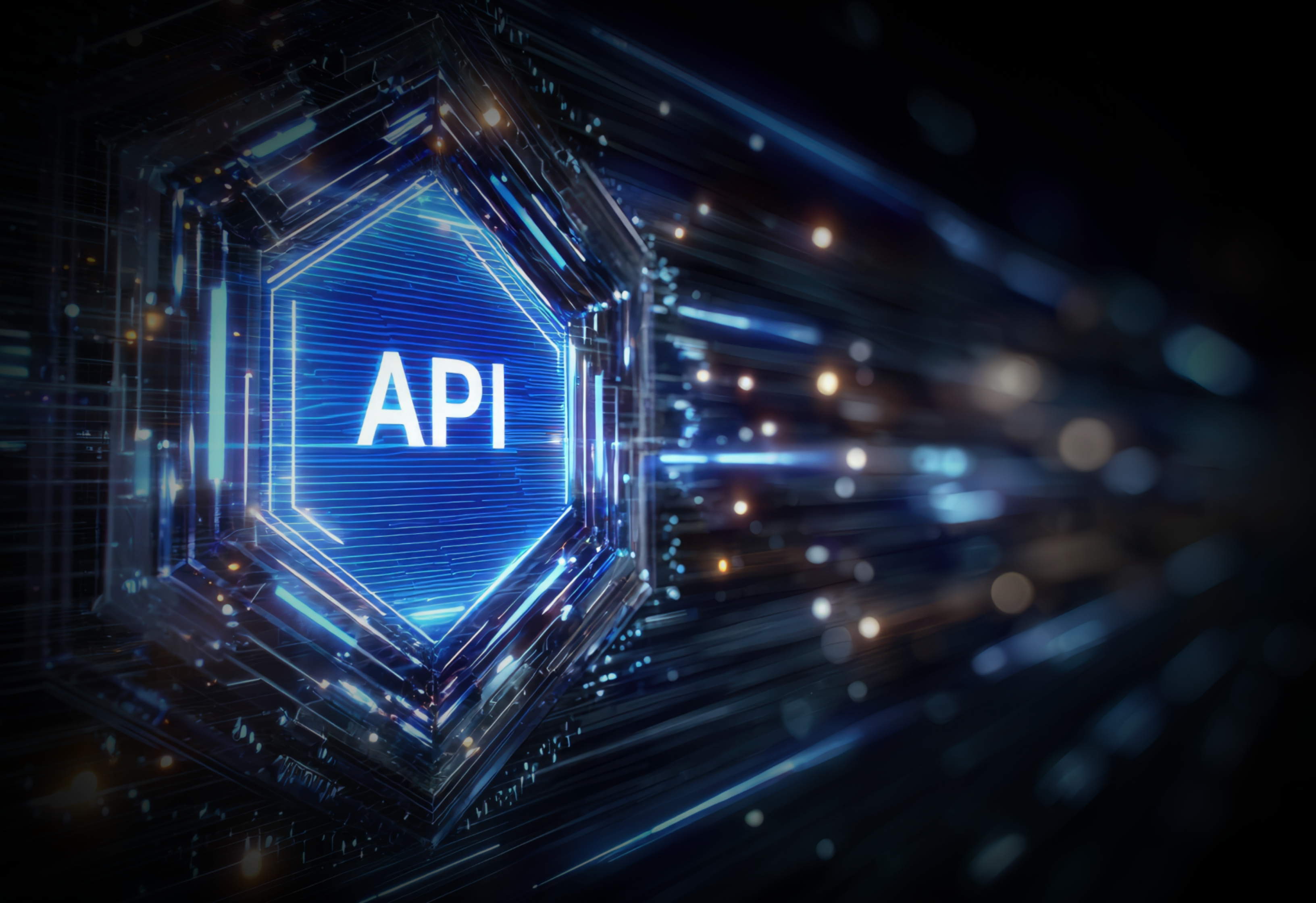
TRACEABLE by harness

WHITEPAPER

OWASP API Security Top 10...Or Should It Be 4?



How to approach risks during API security tool evaluations

Organizations rely on APIs to power their applications, but the unique nature of APIs and their distribution across environments pose significant challenges for security teams. The expanding footprint of AI/ML and Generative AI services further complicates the API landscape, introducing new elements that organizations are unprepared for or that existing tools cannot adequately protect. While the OWASP API Security Top 10 is a common starting point for researching solutions, it can be misleading when viewed in isolation. Many vendors claim to provide full coverage, but they may not address all risks adequately, or their solutions may require excessive manual intervention.

Automation, a void rapidly filled by Agentic AI, is a necessity for even basic API security. The rapid evolution of APIs and the increasing complexity of security threats make manual security measures insufficient and impractical.

Four areas of pervasive API risk typically remain for security teams even after vendor selection: improper authorization, business logic abuse, inadequate governance, and unchecked third-party services. To effectively address these risks, organizations must look for vendor solutions that provide context-aware, multi-tiered API security that can operate anywhere. This comprehensive approach is crucial to safeguarding their APIs and AI-native applications.

Dig Deeper on OWASP Risk Priority

To form the Top 10 list and rankings, OWASP uses data samples of discovered vulnerabilities and augments with anecdotal evidence. Security teams should view the numbering as subjective and not a prescription for risk prioritization in their security programs. The prevalence of risks based on large data samples doesn't equate to the actualized risk for your organization. There are other compounding risk management factors, such as the likelihood of vulnerability exploitation or difficulty in mitigation.

Security-related data, especially for APIs and AI, may be spotty. How can security teams produce accurate risk measures if the number of APIs or related exposure across the application portfolio is unknown? How do the teams discern between appropriate and inappropriate authorizations of APIs and functionality use? Security teams spend much of their effort chasing authentication hygiene, secure build practices, and vulnerability remediation, none of which may even properly account for APIs. They may not even have insight into API build artifacts or suitable runtime telemetry.

Use the OWASP API Security Top 10 list to identify pitfalls with tools, but it should not be your final criterion in vendor evaluations.

Focus on Four Areas of Pervasive API Security Risk

OWASP uses "broken" to qualify half of the items on the list that ultimately revolve around weaknesses in access control. The labeling adds confusion, even amongst experienced practitioners. Security teams also find that certain risks are difficult to mitigate entirely. Instead, approach the list as four categories of frequent gaps for security tools: improper authorization, business logic abuse, inadequate governance, and unchecked third-party services. These gaps represent areas where security tools often fail to procide comprehensive protection, leaving organizations vulnerable to attacks.

Improper authorization

APIs require appropriate access control just like other systems, and the fundamental elements involve:

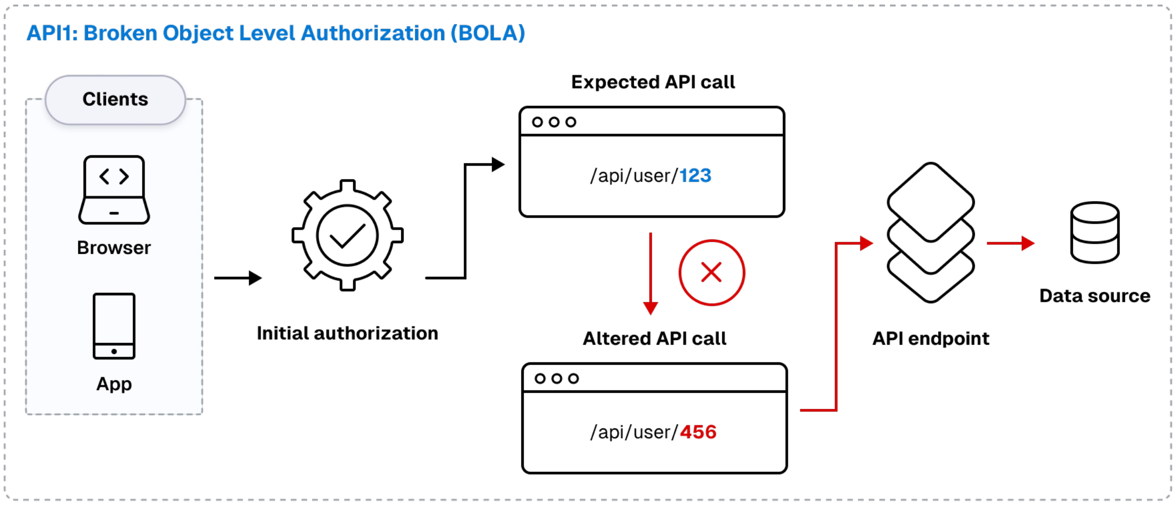
- **Identification -** who the user or API caller claims to be, such as username or account ID
- **Authentication -** how you validate identity to grant access, like a password or 2FA token
- **Authorization -** whether a user should have access to a given resource

These elements are straight out of security practitioner training and quickly enter the domain of Identity and Access Management (IAM). Still, they're tough to implement effectively when considering all the first- and third-party applications and identity types most organizations contend with. Cloud infrastructure and microservice architectures introduce further complexity. Authorization is typically federated, or distributed, in most designs, which requires pre-configuration between parties. "Users" may also be other services or AIs, not just employees or consumers. Your API pathways are numerous and ephemeral, and not all live within the confines of an organization-owned datacenter.

Authorization risks manifest because organizations lack situational awareness around data or functionality sensitivity in a complete system to inform access decisions.

OWASP split API authorization risks into three distinct items in the 2023 list due to the inner workings of APIs and where authorization issues can arise.

- **API1: Broken Object Level Authorization (BOLA) -** APIs may use weak object-level checks, which allow for manipulation of identifiers (e.g., changing /api/user/123 to /api/user/456), a form of unauthorized access. Addressing the risk requires workload-centric validation using role-based access control (RBAC) or attribute-based access control (ABAC) to verify permissions against business rules. For example, a banking API must ensure that **UserA cannot access UserB's account details.**



API1: Broken Object Level Authorization (BOLA) Mitigation

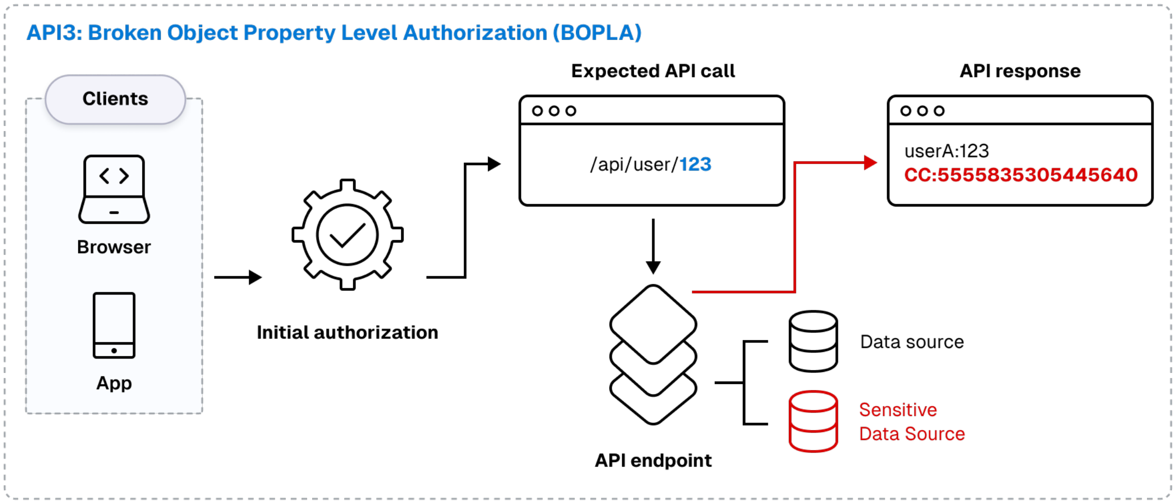
Mitigation typically involves:

- **Server-side validation:** Use middleware to check user permissions against object IDs, leveraging OAuth 2.0 scopes or JWT claims.
- **Randomized identifiers:** Replace sequential IDs with higher-entropy IDs (e.g., GUIDs, UUIDs) to reduce predictability.
- **Runtime monitoring:** Analyze request patterns in real-time for anomalies, to detect unauthorized access attempts.

Why it's a gap for most security tools:

- Servers are ephemeral, and API traffic is distributed
- Not all middleware implements strong authorization checks
- Requires continuous analysis for all API call flows

- **API3: Broken Object Property Level Authorization (BOPLA) -** APIs may expose sensitive fields (e.g., credit_card) without proper access control. Addressing the risk requires workload-level schema enforcement, using JSON Schema or OpenAPI definitions, to filter responses and include only authorized fields. For example, a healthcare API might return patient_id only and exclude any diagnosis data unless explicitly authorized.



API3: Broken Object Property Level Authorization (BOPLA) Mitigation

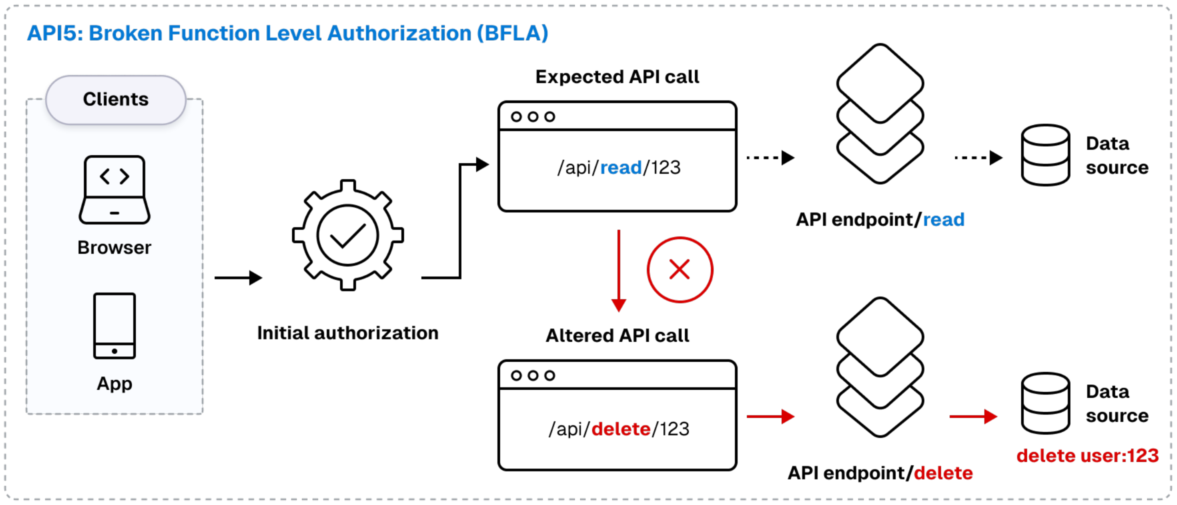
Mitigation typically involves:

- **Response filtering:** Strip unauthorized fields at API gateways based on user roles.
- **Schema validation:** Define strict response schemas at API gateways to enforce data boundaries.
- **Logging and auditing:** Track field access by API callers to detect potential overexposure.

Why it's a gap for most security tools:

- APIs may not be mediated with API gateways by the organization
- API schema definitions may be unavailable or changing too rapidly
- Logging is spotty and consumes excessive storage for "busy" APIs

- **API5: Broken Function Level Authorization (BFLA) -** Privileged API functions (e.g., /api/admin/delete) may employ weak authorization or inadequate role checks. Addressing the risk requires workload-centric controls, integrated with gateways, and enforcing function-level RBAC, ensuring only admins can invoke privileged endpoints. An example might be an eCommerce API restricting access to /api/discounts/apply to managers only.



API5: Broken Function Level Authorization (BFLA) Mitigation

Mitigation typically involves:

- **Role-based checks -** Validate user roles against endpoint permissions using RBAC policies.
- **Endpoint segregation -** Separate privileged or sensitive functions into distinct API paths that require stricter authentication, like MFA or certificate authentication.
- **Behavioral analysis -** Monitor function invocations to detect anomalies.

Why it's a gap for most security tools:

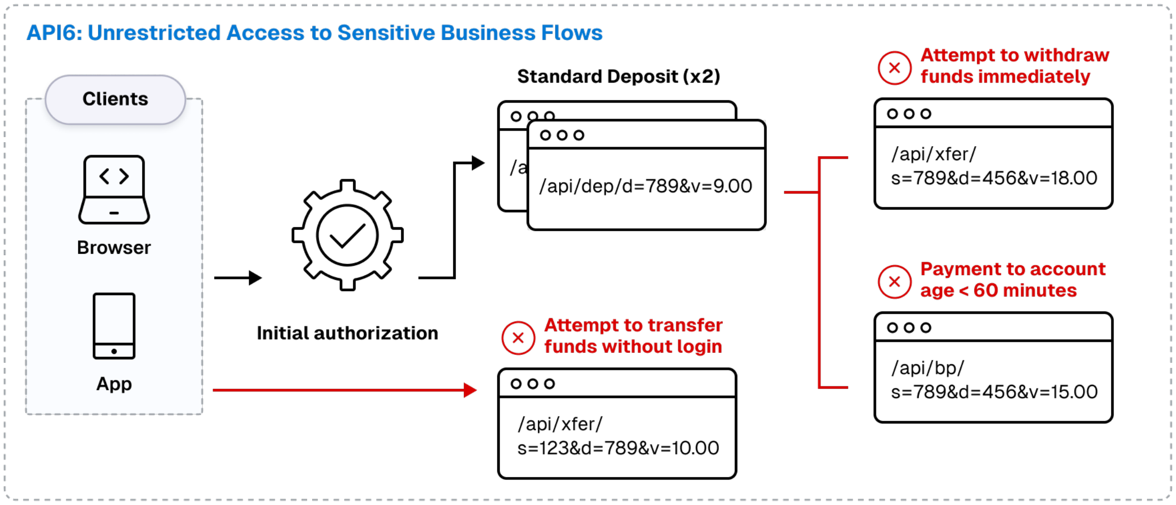
- Access control specifics are unknown to the API security control.
- Tools can't make dynamic access decisions based on API paths alone.
- Requires continuous analysis to form baselines and identify anomalies.

Business logic abuse

Business logic attacks get a treatment in the 2023 list, but they're arguably one of the most damaging for organizations. Business logic is specialized and distinct, making defenses seem impossible for many security teams. Security controls may address this risk in the most basic manner using rate limits or usage policies, but these often fail. API security tools must provide intelligence around the expected behavior of users. What do the normal authorization flows look like? What is the expected sequence of function calls for a complete application interaction, like a funds deposit or an item sale? Should those API calls be purely internal? Or do API calls also need to flow external to a third-party service? Security teams need better telemetry to uncover logic abuse based on how the organization builds or consumes APIs.

API call sequences determine overall business logic, not just one API or its underlying code.

- **API6: Unrestricted Access to Sensitive Business Flows -** Attackers automate their abuse of legitimate API functions for critical operations, causing financial or operational harm. Protection must move beyond standard authentication and authorization, focusing on usage patterns. For example, an online event promotion service API might be prone to scalping since it allows unlimited reservations, or a customer banking bill payment API might allow transactions with an account opened same-day.


API6: Unrestricted Access to Sensitive Business Flows
Clients
Browser
App
Initial authorization
Standard Deposit (x2)
/api/dep/d=789&v=9.00
Attempt to transfer funds without login
/api/xfer/ s=123&d=789&v=10.00
Attempt to withdraw funds immediately
/api/xfer/ s=789&d=456&v=18.00
Payment to account age < 60 minutes
/api/bp/ s=789&d=456&v=15.00

API6: Unrestricted Access to Sensitive Business Flows

Mitigation typically involves:

- **Rate limiting:** Enforce business-specific thresholds on endpoints, such as limiting users to 5 API calls per minute
- **Behavioral analysis:** Monitor for anomalies like rapid sequential requests at an API mediation point
- **Human verification:** Integrate CAPTCHAs or device fingerprinting for high-risk flows

Why it's a gap for most security tools:

- Rate limits are static, or business awareness is lacking to institute them
- Third-party APIs and inner APIs that enable business logic go unchecked
- CAPTCHAs damage user experience and don't address machine identity

Inadequate governance

Organizations rarely fully document all their APIs, particularly when the scope includes all first- and third-party APIs. Engineering teams often prototype new APIs that fly under the radar of standard enterprise release processes or asset management. Asset management processes are usually rooted in configuration management databases (CMDB) that don't support sufficient detail or context about APIs. CMDBs fit the bill for traditional systems and organizational people structures, not the world of applications, APIs, and AI. The organization might drive an API catalog through its API management platform, but this is only a subset of its API portfolio for partner integrations or API products.

Development practices and acquisition outpace many organizations' ability to document and audit everything.

- **API9: Improper Inventory Management -** Undocumented APIs are a common side effect of rapid development or third-party acquisition. Shadow APIs can appear from unsanctioned tool use or out-of-band releases. Zombie APIs may have been spun up for temporary use and quickly abandoned. These types of APIs are frequently unmonitored and uncontrolled. Debug tools can catalog endpoints by analyzing runtime traffic, but such approaches don't often scale.

API9: Improper Inventory Management

Mitigation typically involves:

- **Automated discovery -** Scan application traffic at multiple points throughout operating environments to gather API metadata.
- **Version control -** Track and maintain API versions using API management, an Internal Developer Platform (IDP), or an API catalog
- **Centralized inventory -** Maintain a dynamic catalog integrated with CI/CD build pipelines instead of traditional asset management

Why it's a gap for most security tools:

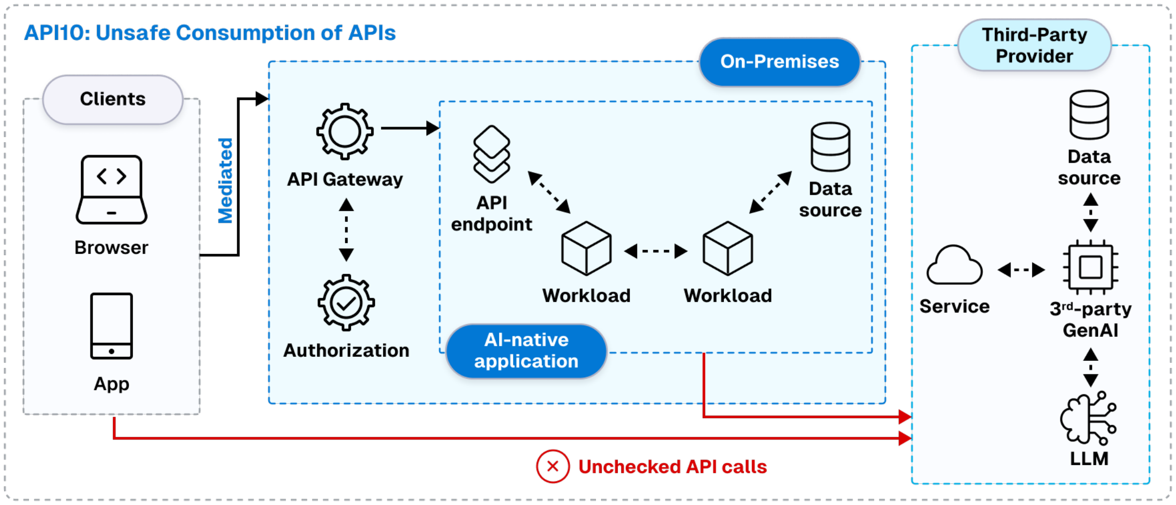
- Tools are run from a fixed network or endpoint position, reducing visibility
- API management isn't enterprise-wide and doesn't mediate all APIs
- Vulnerability management focuses on infrastructure, not API context

Unchecked third-party services

Most applications function by invoking other services, or APIs, regardless of organizational preference to build or acquire software. Cloud service providers (IaaS), application platforms (PaaS), and SaaS business applications are commonplace. "Consumption" can be misleading because this isn't a quota management problem, and APIs must communicate to function. The organization is likely unaware of its many third-party integrations; thus, engineering teams aren't properly validating all data or functionality.

Procurement of third-party tools often happens without any Enterprise IT oversight, resulting in API blind spots and improperly hardened resources.

- **API10: Unsafe Consumption of APIs -** An organization's systems often call external and third-party APIs, which may serve up malicious, unsanitized, or inaccurate data. They can also create unexpected impacts on business logic when integrated with other systems. To ensure safe data handling, sanitizing input and checking against API schemas at the workload level is necessary.



API10: Unsafe Consumption of APIs

Mitigation typically involves:

- **Input validation -** Sanitize external API responses using allow-lists or schema checks within data flows to the organization's systems.
- **Error handling -** Parse responses consistently and reject any malformed data.
- **Monitoring -** Analyze third-party API requests and responses for anomalies

Why it's a gap for most security tools:

- Allow-lists don't scale for cloud, and schema validations may be too rigid
- Data may be unstructured, making it difficult to identify malicious payloads
- Can't assess all upstream and downstream impacts from a single API call

Security Must Keep Up with Your API Footprint

Any vendor tool you consider for API protection must tune itself for your environments and system designs. The solution should directly support or work with existing mechanisms to address the OWASP API security risks we've spotlighted:

- **Improper authorization -** implement directly or work in tandem with IAM systems in the organization to:
  - Enable logic-specific access control so attackers can't manipulate object IDs, mitigating BOLA (API1). For example, a retail API must prevent unauthorized access to other users' carts.
  - Detect and prevent exposure of sensitive fields to mitigate BOPLA (API3). For example, a healthcare API must avoid the exposure of a patient's SSN without the patient's consent to a provider.
  - Ensure privileged API endpoints are accessible to a subset of users and perform role checks to mitigate BFLA (API5). For example, an HR endpoint for salary updates should be accessible only to HR admins or account provisioning functions.
- **Business logic abuse -** analyze API traffic through all operating environments, piece together API call sequences, and detect patterns of misuse that violate expected behaviors, mitigating API Unrestricted Access to Sensitive Business Flows (API6).
- **Inadequate governance -** ensure that API endpoints are continuously documented using traffic from operating environments, API code is maintained in sanctioned repositories, and APIs are delivered through controlled build pipelines that generate API schema automatically, effectively mitigating Improper Inventory Management (API9).
- **Unchecked third-party services -** irrespective of API schema, identify third-party API integrations in real-time, document complete API sequences with expected data, and ensure that API calls flow through infrastructure elements with monitoring in place so that visibility and control are maintained, mitigating Unsafe Consumption of APIs (API10).

Attackers perpetuate business logic abuse by chaining legitimate API calls, possibly out of the expected sequences, evading traditional security control detection. Combining edge-positioned and workload-positioned controls is critical for supporting behavioral analysis and anomaly detection. Any vendor solution under consideration must also handle this reality of distributed deployment. For example, a retail services provider might use an API security platform to organize a combination of rate limiting at the edge, data validation within inner API traffic to detect manipulation of coupon discount fields, and RBAC to prevent unauthorized bulk order discounting.

Conclusion

APIs are proliferating, and so are the security challenges. The introduction of AI-native design, ever-increasing cloud consumption, and broad adoption of DevOps practices have significantly expanded API attack surfaces for organizations. The OWASP API Security Top 10 highlights common resulting risks, and many security solutions still leave gaps. Be cautious of residual risk around improper authorization, business logic abuse, inadequate governance, and unchecked third-party services when considering solutions. Organizations must look for vendor solutions that provide context-aware, multi-tiered API security to safeguard their APIs and AI-native applications effectively.

TRACEABLE by harness

Traceable is the industry's leading API Security company that helps organizations achieve API protection in a cloud-first, API-driven world. With an API Data Lake at the core of the platform, Traceable is the only intelligent and context-aware solution that powers complete API security – security posture management, threat protection and threat management across the entire Software Development Lifecycle – enabling organizations to minimize risk and maximize the value that APIs bring to their customers.

To learn more about how API security can help your business, book a demo with a security expert.

www.traceable.ai