CLOUDVECTOR

WHITEPAPER

Beyond Common Gateway Protection: Advanced API Risks

www.cloudvector.com

Executive Summary

APIs create new risks that cannot be addressed by commoditized solutions alone. In particular this risk is caused by a lack of visibility into API specs, deep at the payload level.

DIGITAL TRANSFORMATION IS CREATING A NEW SET OF RISKS

Digital Transformation **makes "every company a software company."** Business applications are migrating to cloud infrastructure. DevOps and SecOps are adopting an agile Continuous Integration/Continuous Development (CI/CD) process. Rapid development and changes to business applications have become necessary to keep pace with changing business needs.

APIs have become the de facto data transport because its custom specification enables maximum flexibility for custom business applications to communicate.

However, this rapid evolution of apps has not translated to a rapid change in app infrastructure and security. On the contrary, rapid development of apps has been made possible by the c**ommoditization of infrastructures,** including core security functionality once considered advanced features.

For example, most cloud infrastructure providers now offer OS/container platform security, segmentation, and data encryption at rest and in transit. Common Web protection against DDoS, Bot, and OWASP Top 10 type attacks are now baseline features. However, a major limitation of gateways is that they are **only** useful for north-south traffic because they are only deployed at the perimeters.

When it comes to API security, solutions seem focused on protecting infrastructure risks. Some gateways offer protection for **common** API security risks, such as access management for API Gateways (e.g. Mulesoft, **Apigee**, Kong). Other gateways (e.g. L7 Defense and Salt Security) piggyback on Web Application Firewalls (WAF) to leverage AI models that recognize and protect **common API functions.**

One example of such a common function is the "login" action. Some advanced API protection solutions include inspection and monitoring of the login action to detect abuses such as credential stuffing. Another example is the detection of bad bot behaviors based on telemetries such as the source of calls and call volume. As these solutions are **only** mitigating risks common to all APIs, they are **not** addressing **risks specific to individual application API transport.** In essence, these solutions are only providing **API infrastructure** security.

However, even in a state-of-the-art cloud environment with all of its infrastructure security enabled, there are still risks specific to custom APIs themselves. Left unaddressed, this is the **new risk surface for data breaches.**

NEW CATEGORIES OF RISKS

Beyond the OWASP Top 10, there is now a new OWASP API Security Top 10[1] , which highlights specific risks at the custom API data transport level.

OWASP API SECURITY TOP 10	
A1	Broken Object Level Authorization
A2	Broken Authentication
A3	Excessive Data Exposure
A4	Lack of Resource & Rate Limiting
A5	Broken Function Level Authorization
A6	Mass Assignment
A7	Security Misconfiguration
A8	Injection
A9	Improper Asset Management
A10	Insufficient Logging and Monitoring

In the remainder of this document, we will dissect key API risks in the following categories:

- Risks due to unknown/outdated API blueprints
- Risks due to uninspected API calls
- Risks due to uncontrolled third-party APIs
- Risks due to lack of in-depth object/function inspection

RISKS DUE TO UNKNOWN/ OUTDATED API BLUEPRINTS

APIs are powerful because their data transport is highly customizable. Their custom attributes are usually called an API Specification or Spec for short. Popular spec formats are OpenAPI (formerly Swagger) and RAML. A complete, up-todate API spec is usually a requirement for any API security testing/assessment tools.

Application services exposing public APIs for third-party developers are usually quite good at keeping their published API specs up-to-date as external users serve a forcing function. However, this is not the case for APIs developed for private use, which are the majority of business app APIs.

The general lack of complete, up-to-date API specs is because they are often manually generated by developers at the beginning of a project. And as a manual process, there is little validation or enforcement to keep these specs up-to-date when the app evolves.

1 RISK 1: No API Specs

If business applications expose APIs without creating API specs, then the custom API transport represents an entirely undocumented data risk surface area.

SecOps realize the need to have an accurate API asset registry or a Live API Catalog, but the manual process and updates make it easier said than done.

2 RISK 2: Loosely Defined API Specs

A loosely defined API spec is not properly protected because it is missing information.

This information is an undocumented risk surface.

1 OWASP API Security Project

3 RISK 3: Out-of-Spec API Calls

An API spec can be incomplete or out of sync due to rapid changes during implementation. Bad actors can leverage out-of-spec API functions or parameters to extract data. Two recent examples include:

- A massive data leak was reported in a T-Mobile app. The app API inadvertently exposed an undocumented "shadow API parameter," which enabled external callers aware of the hidden parameter to access T-Mobile customer account data.[2]
- The Harbor Registry API vulnerability allowed a user to elevate a guest account registration to "admin" status by simply adding an out-of-spec parameter administration="yes".[3]

RISKS DUE TO UNINSPECTED API CALLS

Modern application architectures, such as containers and micro-services, have introduced the notion of "east-west" traffic: APIs are used by not only front-end web services but also back-end, service-toservice transactions. Conventional "ingress controller" ("north-south") type gateways are not deployed to inspect east-west API calls.

4 RISK 4: API Calls Not Inspected by a Conventional Gateway

Bad actors are known to use compromised servers to launch "lateral" attacks against services with custom APIs to bypass gateways. For example, a major data breach was reported by a hosting company, Hostinger, in which a lightly protected server was compromised, then used to exfiltrate data from another database service.[4]

5 RISK 5: API Calls Not Inspected by Network Detection Due to Encryption

There are network detection methods (e.g. those provided by ImVision) that can potentially inspect service-to-service (eastwest) API calls without relying on a gateway. However, the increased use of encryption makes such inspection less and less effective—in the case of the Hostinger breach, the API calls made by the bad actors used TLS encryption.

RISKS DUE TO UNCONTROLLED THIRD-PARTY APIS

APIs are not limited to ingress calls from enterprise users or partners, they are also made by business applications to external services. Public cloud infrastructure services expose API access as part of their standard offering. If not inspected, these API calls can be vectors of data leakage.

6 RISK 6: Egress API Calls to External Services

Business applications are the "clients" of external services, but the egress API calls they make can be abused to exfiltrate data. For example, storage service APIs exposed by Microsoft Azure have been reportedly leveraged by bad actors to exfiltrate data to unauthorized accounts.

2 ZDNet
3 Unit 42
4 Threat Post

7 RISK 7: Public API Access to Enterprise Private Resources

Many public cloud services enable standard, public API access. For example, unlike a private data center, AWS S3 API network access cannot be turned off for an enterprise customer. Authenticated user roles provide the only access control. In the case of CapitalOne[5], compromised credentials enabled access to all S3 buckets belonging to the company. Data exfiltration was not detected for months due to the lack of additional monitoring and alerting.

RISK DUE TO LACK OF IN-DEPTH OBJECT/FUNCTION INSPECTION

As discussed previously, object/function level inspection requires a complete and up-to-date API blueprint, which is often missing for privately used APIs. Lack of object/function inspection leaves application business logic exposed to exploits.

7 RISK 8: API Parameters Out of Critical Range

APIs, especially those used for control purposes (e.g. IoT), may be abused if certain parameters are set out of range. Conventional anomaly detection methods cannot detect small deviations (e.g. adding a '0' to a parameter) in an otherwise normal call. In one attack, an industrial furnace was physically damaged when a bad actor issued rouge API calls to sabotage its temperature control unit.[6]

9 RISK 9: API Session and Query Parameter Mismatch

It is a common but extremely damaging application logic mistake to miss crosschecking the login session and query parameter. Worst, if a front-end service is missing such cross check queries of a backend service on the user's behalf, it is impossible for the back-end service to detect such data exfiltration as the original caller credentials are no longer visible to back-end service.

Without ensuring the query parameters are properly "pinned" to the login user-session user, an API session belonging to user X can be abused to retrieve data belonging to user Y. There are numerous data breaches due to this kind of vulnerability. A few are listed below:

- US Postal Office's end-user privacy reporting API itself was found to have a vulnerability for more than 12 months, allowing a login user to retrieve any other user's private information.[7]
- The Facebook "ViewAs" privacy breach that exposed approx. 50 Million users' private data is another example of the lack of session and data retrieval "pinning."[8]

CONCLUSION

According to Gartner, "By 2022, API abuses will be the most-frequent attack vector resulting in data breaches for enterprise web applications."[9] Unfortunately, most solutions only address common threats. App-specific API data transport will become the major data risksurface.

5 Security Boulevard
6 Wired
7 Krebs on Security
8 Auth0
9 Gartner