Cheat sheet: 10 Serverless Security Best Practices

snyk

1. Patch function dependencies

Track the libraries you use in each function, flag vulnerabilities in those libraries, and monitor them continuously.

Use Snyk via the CLI, GitHub, Serverless Framework plugin or direct FaaS hooks.

2. Adopt the principle of least privilege

Managing granular permissions for hundreds or thousands of functions is very hard to do, but important nonetheless.

- Avoid globally defined roles and resource access permissions for functions.
- Minimize access rights to resources for functions using fine-grained permissions for each function with Serverless Framework manifests.
- Enforce the runtime least-privilege principle by making use of security libraries that disable access to system resources.
- Use dedicated users and minimalistic permissions when accessing third-party services.

3. Maintain isolated function perimeters

Treat every function as its own security perimeter to ensure that a compromise in one function doesn't escalate to other functions and resources:

- Do not rely on function access and invocation ordering
- Sanitize function input and treat event data as untrusted
- Adopt, mandate and re-use security libraries across your functions

4. Deploy functions in minimal granularity

Deploy functions in minimal granularity to avoid implicit global roles for all functions and bundling unnecessary code and dependencies.

5. Sanitize event input to avoid injection

The input received by a function may originate from different data sources, which makes it complicated to track and ensure that untrusted data has been validated before use.

Due to the event-driven nature of serverless architecture, properly validate and sanitize data that is received through a function to avoid injection attacks.

6. Employ API gateways as a security buffer

Don't expose functions directly to user interaction. Instead leverage your cloud providers' API gateway capabilities to add another layer of security in front of your function. API gateways can be used for:

- Filtering input data based on request and response mapping models
- Offloading authentication concerns from your functions core business logic
- DDOS protection, traffic throttling and rate limiting.

7. Monitor and log functions

Audit and monitor how and what functions are accessing to ensure no illegal paths are taken, and monitor security vulnerabilities in functions:

- Cloud provider tools like AWS X-Ray and Azure Monitor help monitor resources accessed by functions such as CPU, memory, function run time, functions data flow, and alike to create a baseline, alerts and pro-active mitigation.
- Implement verbose and safe logging of function events, and use a central logging system to gain better observability.
- Use function tags to provide increased visibility and easier maintenance
- Use Snyk to integrate with your cloud provider's deployed functions and continuously monitor for security vulnerabilities in deployed code.

8. Follow secure coding conventions for application code

With no servers to hack, attackers will shift their attention to the application layer, so take extra care to secure your code. The OWASP Top 10 is a good place to start.

9. Secure and verify data in transit

Small functions and stateless apps lead to more use of third party services, raising the risk of those functions and apps being attacked and of MITM attacks. Be sure to:

- Leverage HTTPS for a secure communication medium
- Verify SSL certificates to ensure the remote identity; halt communication upon failure to verify
- Treat responses from 3rd party services as untrusted user input

10. Manage secrets in secure storage

Sensitive information can easily be leaked and out-of-date credentials are prone to rainbow table attacks if you fail to adopt proper secret management solutions.

- Do not store secrets in application code, environment variables or in a source code management system, either encrypted or otherwise.
- For sensitive information, utilize a secret storage that enables both runtime access, as well as easy and routine key rotation.

Authors

@liran_tal
Node.js Security WG & Developer Advocate at Snyk

@guypod
Co-founder at Snyk