Security analysis document

OFS Platform

By Mohammed Al Harbi

Contents

1. Introduction 3
2. What is serverless and why serverless security 3
3. Security threat model of serverless 4
4. Serverless security measurements best practices 5
5. Proof 7
References 10

1. Introduction

The primary goal of this document is to demonstrate what is serverless and why we treat serverless differently especially from security perspective. We will draw attention to the typical best practices for monolithic and microservices systems and then switch to the best practices for serverless applications. Then, we will look at the security threat of serverless which implies what hackers can utilize for penetrating serverless applications or cause extra billing to the business owner. Finally, will dive into best practices to secure serverless applications and show proof of integrating solutions into existing development processes Oman Freelancing Platform conduct.

2. What is serverless and why serverless security

The definition of serverless: It is a cloud computing service which allocates resources based on demand allowing business to scale out or scale down on fly, it enables businesses develop their software solution very quickly as it advocates software engineers or developers to focus only on business logic. Unlike typical monolithic and microservices applications where servers need to be paid attention to, the cloud provider manages the infrastructure, patching and maintenance of the machines.

The key difference in terms of how system should be in serverless environment and that are typically monolithic or microservices lies the payment and how the code is being managed. In a serverless application the payment model is pay-to-go where businesses are billed for what the customers consume in their website. For example, clicking on my profile tab causes a machine to spin up and run in the cloud to obtain that user profile details then business is billed for that request and amount of time the execution took to perform retrieving of those details. Thus, the code is managed in separate functions known also as “FaaS” that implies function as a service. This result in a very competitive and cost-efficient model aside from enabling developers to focus on delivering features fast. On the other hand, monolithic or microservices applications where managing the infrastructure need to be drawn attention to, looking at the example of the previous feature of retrieving, the user clicks on profile details and an idle server receives the request and send the user details then stays available online. Businesses would need to pay for space and capacity they deploy their business logic code on, and this implies the paying for idle if not provisioned properly.

Serverless security: In non-serverless applications face security challenges namely cross-site scripting, broken access control, database injection such as SQL Injection, sensitive data exposure and the list goes on. Typically, to overcome those challenges numerous actions are required to take in place like installing and configuring firewalls, the utilization of IPS tools and noticeably those actions focus on network inspection protection.

Serverless applications does not benefit of solutions that are network-inspection based; the challenges are on a completely different level of security threat model, the security solutions need to concentrate on behavioural protection, permission configurations.

3. Security threat model of serverless

There are multiple security risks which we highlight in this section namely, attacks through vulnerable dependencies, increased attack surfaces, security misconfiguration, broken authentication and the threat of privileged functions.

1. **Attacks through vulnerable dependencies:** Serverless applications business logic code gets executed in separate time and space in the cloud provider this is known as FaaS and each function code may use different dependency to perform the business logic which sometimes happen to be doing the job as the developer needed but can be exploitable and thus a weakness in the architecture bad actors might utilize to install malicious code. As a consequence, hackers could control the command line via this malicious code then it is over.

2. **Increased attack surfaces:** As the application grows, so does the number of serverless functions and every line of code can potentially be a weakness, usually those serverless functions consume data from various sources including HTTP APIs, Internet of Things and this increases the chance of vulnerability by quite a bit since some of these parties may contain untrusted malicious messages and can be utilized to install malicious code in the serverless function.

3. **Security misconfiguration:** Once a serverless application is configured insecurely this allows bad actor(s) to perform for instance 2 popular kind of attacks namely known

as DoS (Denial-of-Service) and DoW (Denial-of-Wallet). The first attack aims to get some part of the serverless online application service or entire unavailable by simulating huge traffic all focused to the application entire. The second attack aims to increase the cost of the business to the moon by increasing the time needed for a serverless function to get executed (Cloud providers charge for the time a function took to get executed.)

4. **Broken authentication:** Serverless applications offers the use of microservices and allowing businesses to take event-driven architecture approaches which result in very cost-efficient, fast development pace, the ability to work in different autonomous teams in parallel and enables utilization of multiple programming languages. However, in such architectural design there are many serverless functions which need to be performed based on some authentication and authorization mechanism and if this mechanism is broken in one of those functions the rest of the functions are reachable to the hacker.

5. **Threat of over-privileged functions:** Many independent functions are used in serverless ecosystem each having their own role and permission. The interaction among those functions can cause intended privileges to some functions which can be cause the reachability to the database(s) by hackers and exploiting such interaction possible.

4. Serverless security measurements best practices

There are multiple security actions and best practises we can perform to avoid risks mentioned in the previous section including the utilization of API Gateway as security buffer, the data separation and secure configurations, ensuring secure authentication, sufficient monitoring and logging and minimizing privileges of serverless functions.

1. **Using API Gateway as a security buffer:** Separating the data from functions via ***API HTTPS*** endpoint gateways is one technique to stop event-data injection in serverless applications. Utilizing ***HTTPS*** endpoints leverages built-in security protocols like data encryption and assist in protecting sensitive data.

2. **Data separation and secure configurations:** As mentioned in the serverless risk section serverless application are prone to cyper attacks namely Denial-of-Services (DoS) and Denial-of-Wallet (DoW), to avoid the risk of those attacks it recommended to perform a code scan in the development process to identify potential security risks prior to deployment which typically can be placed in the continuous integration and continuous deployment pipeline, separating commands and queues. It is vital to configure function timeouts properly as it can be exploited by bad actors to increase the bill.

3. **Ensuring secure authentication:** To minimize the risks of broken authorization and authentication, there are multiple solutions for serverless applications to utilize, using the cloud provider access control solutions and in my case OFS platform run in amazon cloud services which uses Cognito access control service, the use of multi-factor authentication and implementing strong password policy requirements regarding length and symbols of passwords.

4. **Monitoring and logging:** Utilizing cloud provider monitoring most of the time is not enough since it does not cover application layer, and for in-depth vision over serverless functions it is recommended to use monitoring tools so it is possible to look at application event data as it can be a potential entry point for hackers. Example of those tools are: SLS-dev-tool, Sense Deep, Lumigo, serverless framework, Dashbird, IOPipe. OFS platform utilizes serverless framework.

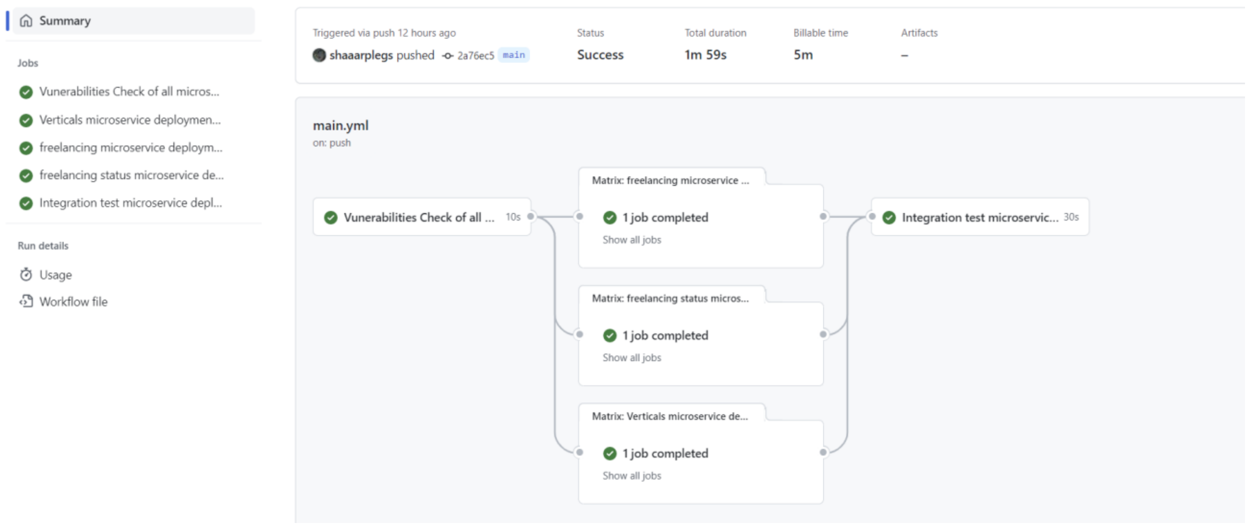
5. **Minimize serverless functions privileges:** It is considered best practice to give each serverless function an IAM Role and only needed permissions to perform their tasks to avoid possibility of having over-privileged functions as much as possible so that bad actors have hard time reaching sensitive data if they managed to install malicious code in the code base.

5. Proof

As of right now I have applied measures number 2,3 and 4 from the previous best practises against potential serverless threats and below are screenshot proofs of utilizing them

2. Data separation and secure configurations:

I use Synk tool which works well with serverless applications to automatically scan code for potential vulnerable dependency and security threats, and I integrated this tool in OFS Platform CI/CD pipeline in GitHub.



Summary

Jobs

- Vunerabilities Check of all micros...
- Verticals microservice deploymen...
- freelancing microservice deploym...
- freelancing status microservice de...
- Integration test microservice depl...

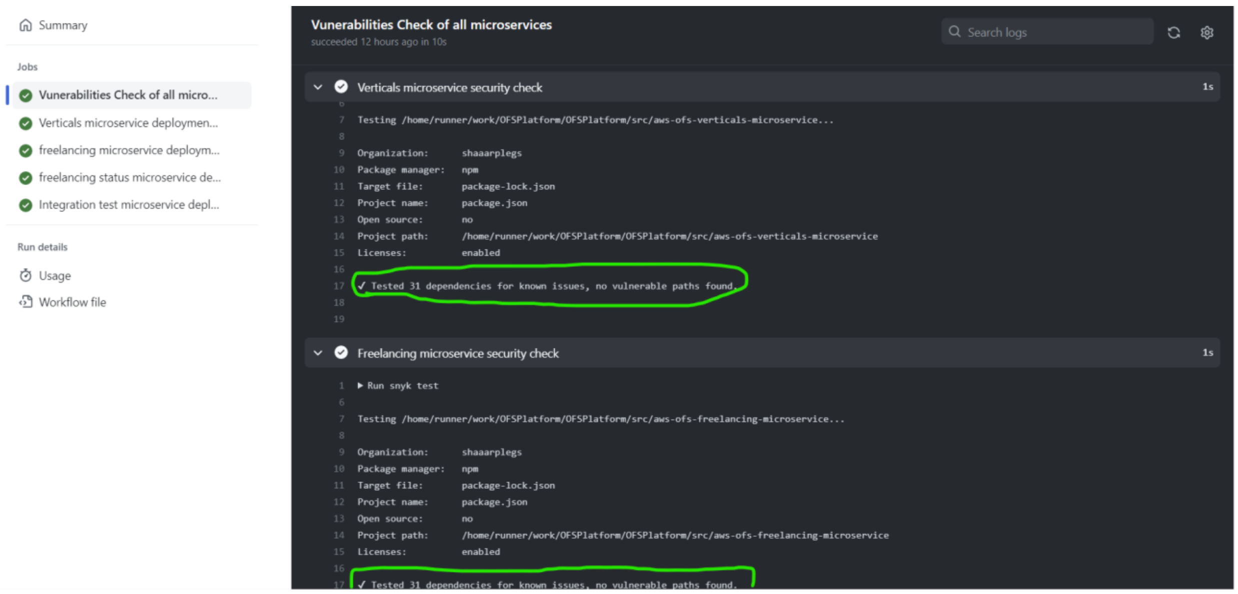
Run details

- Usage
- Workflow file

Triggered via push 12 hours ago	Status	Total duration	Billable time	Artifacts
shaaarplegs pushed 2a76ec5 main	Success	1m 59s	5m	–

main.yml
on: push

Vunerabilities Check of all ... 10s
Matrix: freelancing microservice ...
1 job completed
Show all jobs
Matrix: freelancing status micros...
1 job completed
Show all jobs
Matrix: Verticals microservice de...
1 job completed
Show all jobs
Integration test microservic... 30s


Summary
Jobs
Vunerabilities Check of all micro...
Verticals microservice deploymen...
freelancing microservice deploym...
freelancing status microservice de...
Integration test microservice depl...
Run details
Usage
Workflow file

Vunerabilities Check of all microservices
succeeded 12 hours ago in 10s

Verticals microservice security check — 1s

```
Testing /home/runner/work/OFSPlatform/OFSPlatform/src/aws-ofs-verticals-microservice...

Organization:      shaaarplegs
Package manager:   npm
Target file:       package-lock.json
Project name:      package.json
Open source:       no
Project path:      /home/runner/work/OFSPlatform/OFSPlatform/src/aws-ofs-verticals-microservice
Licenses:          enabled

✓ Tested 31 dependencies for known issues, no vulnerable paths found.
```

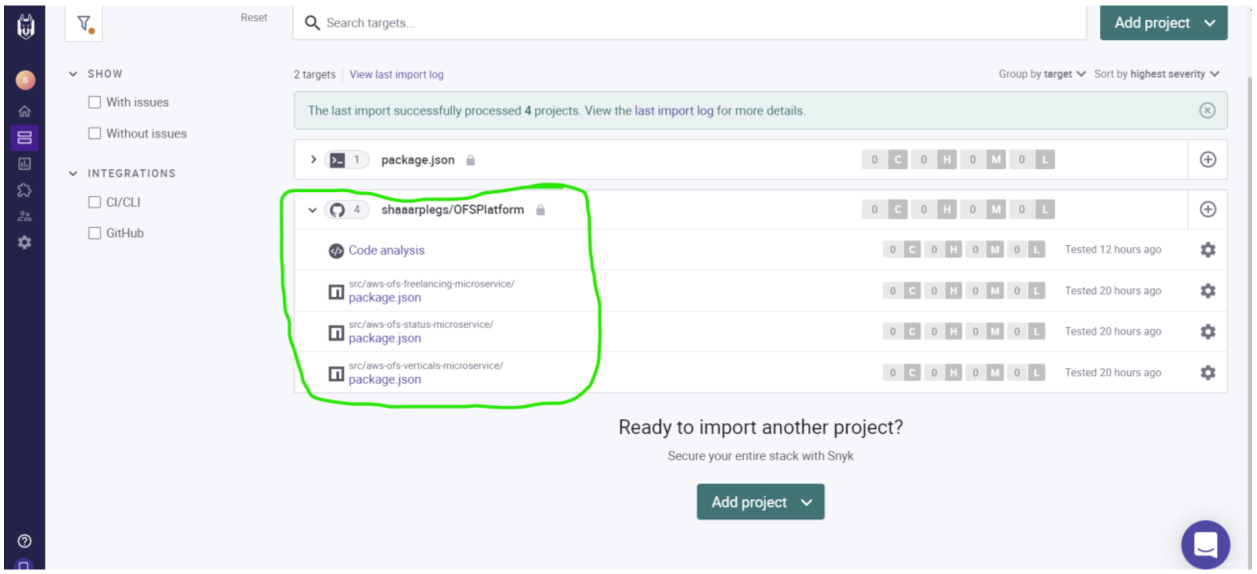
Freelancing microservice security check — 1s

```
▶ Run snyk test

Testing /home/runner/work/OFSPlatform/OFSPlatform/src/aws-ofs-freelancing-microservice...

Organization:      shaaarplegs
Package manager:   npm
Target file:       package-lock.json
Project name:      package.json
Open source:       no
Project path:      /home/runner/work/OFSPlatform/OFSPlatform/src/aws-ofs-freelancing-microservice
Licenses:          enabled

✓ Tested 31 dependencies for known issues, no vulnerable paths found.
```



2 targets | View last import log

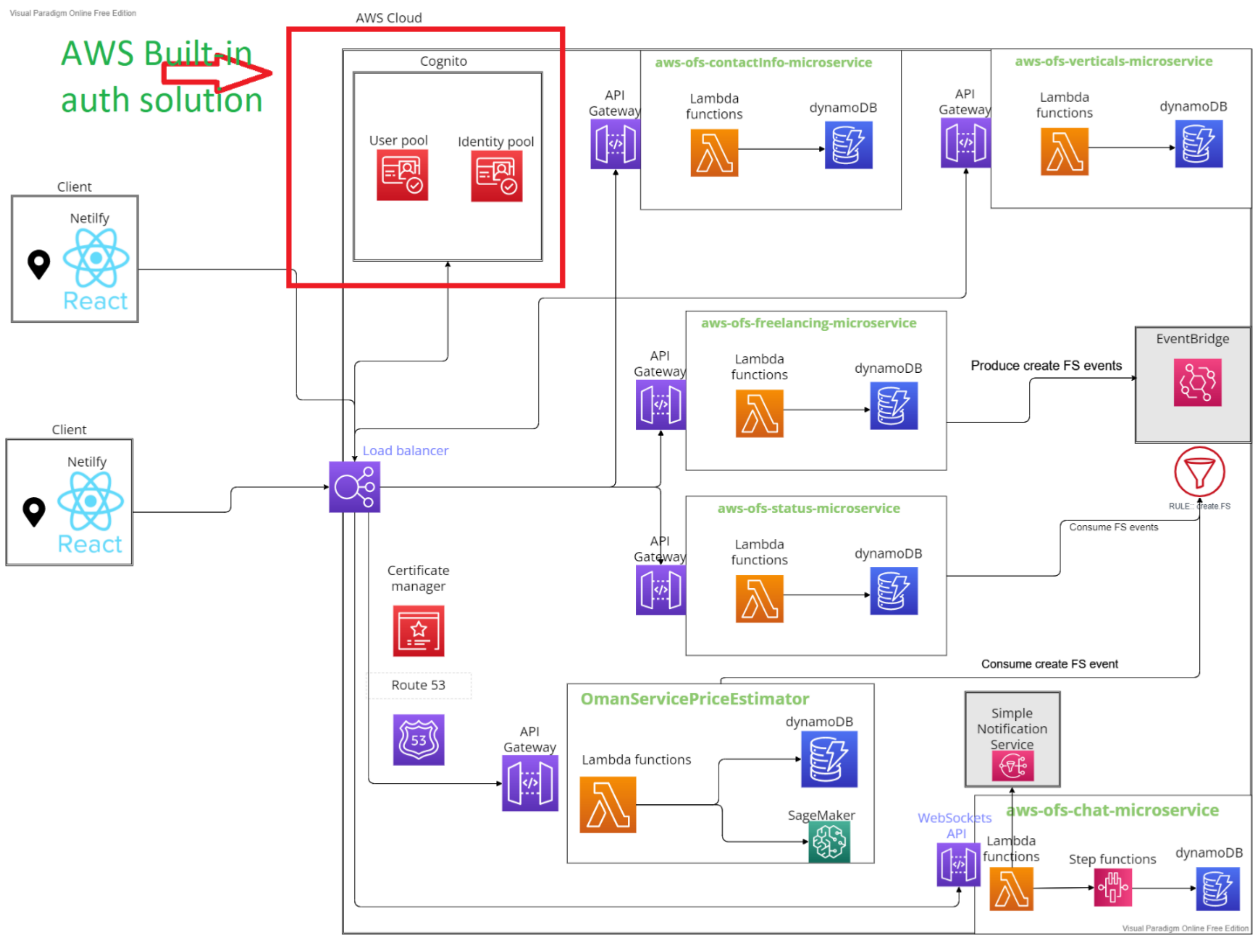
The last import successfully processed **4** projects. View the last import log for more details.

1 package.json	0	C	0	H	0	M	0	L	
4 shaaarplegs/OFSPlatform	0	C	0	H	0	M	0	L	
Code analysis	0	C	0	H	0	M	0	L	Tested 12 hours ago
src/aws-ofs-freelancing-microservice/ package.json	0	C	0	H	0	M	0	L	Tested 20 hours ago
src/aws-ofs-status-microservice/ package.json	0	C	0	H	0	M	0	L	Tested 20 hours ago
src/aws-ofs-verticals-microservice/ package.json	0	C	0	H	0	M	0	L	Tested 20 hours ago

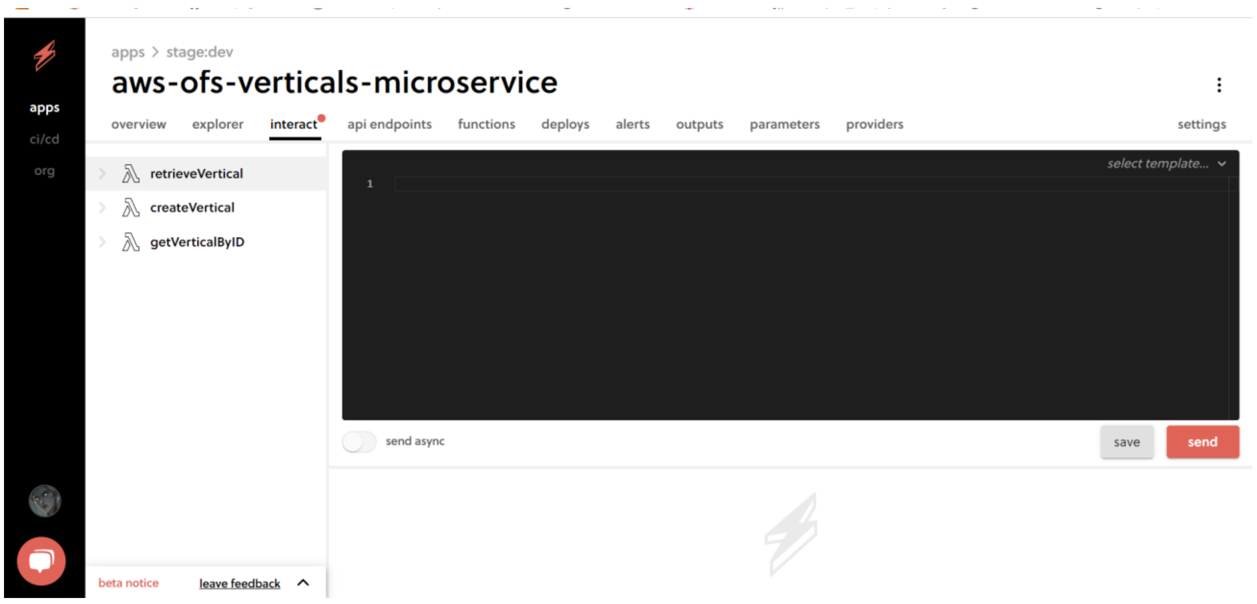
Ready to import another project?

Secure your entire stack with Snyk

3. Ensuring secure authentication:



4.Monitoring and logging:



References:

Serverless Computing Security in 2021 | CSAHow to Design a Secure. (n.d.). https://cloudsecurityalliance.org/artifacts/serverless-computing-security-in-2021/

Serverless Threat Model. (n.d.). https://www.iriusrisk.com/resources-blog/serverless-threat-model

Sysdig. (2021, December 23). *Serverless Security: Risks and Best Practices –*. https://sysdig.com/learn-cloud-native/kubernetes-security/serverless-security-risks-and-best-practices/

Kumar, C. (2022, September 22). *10 Best Tools to Monitor and Debug Serverless Applications*. Geekflare. https://geekflare.com/serverless-monitoring-troubleshooting-tools/