

Securing Serverless Web Applications: Against Insider Attacks and Data Breaches

**Nabanita Pattnaik, Satyajit Barik, Nitesh Pandey and
Prof. Mr. Pradeep Bedi**

Dept. of SCSE (MCA) Galgotias University

Abstract

In this Article we propose a comprehensive security framework tailored to serverless web applications. Our framework incorporates principles of least privilege, continuous monitoring, and threat modelling to strengthen security posture. We also discuss the importance of encryption, authentication, and auditing in safeguarding sensitive data and operations. Furthermore, we evaluate various security tools and techniques that can be integrated into serverless web applications to detect and respond to insider threats and data breaches in real-time. We emphasize the importance of proactive security measures, including security training and awareness programs for development teams. In conclusion, this work aims to raise awareness of the security challenges specific to serverless web applications in cloud computing environments. By implementing the proposed security framework and best practices, organizations can enhance their ability to defend against insider attacks and data breaches, ultimately ensuring the confidentiality, integrity, and availability of their web applications and sensitive data. To identify potential insider attack vectors and vulnerabilities in serverless web applications. To assess the impact of data breaches in cloud computing environments. To propose security measures and best practices for mitigating these risks.

Keywords: Cloud Computing, Serverless Web Application, Security, Security Threats

Introduction

In today's cloud computing environments, serverless web applications have become increasingly popular due to their scalability and cost-effectiveness. However, these benefits come with security challenges. This project focused on securing serverless web applications against insider attacks and data breaches within cloud computing environments.[1] We aimed to identify vulnerabilities, assess risks, and propose security measures to mitigate these threats. Cloud computing offers numerous advantages, but it also exposes organizations to potential insider attacks and data breaches.[2] Serverless web applications are particularly vulnerable, as they rely on third-party cloud providers for infrastructure. The purpose of this project was to analyse and improve security in such environments.

Background

In this section, we look at the current serverless ecosystem. More specifically, we first introduce serverless computing, then analyse the five key elements that make up any serverless platform, and finally discuss existing security solutions.

Serverless Computing

In serverless computing, application logic is divided into a set of small, temporary, stateless functions, each running in a separate execution environment (e.g., container), which communicates with each other and with different cloud services (e.g., hosting services) to complete their tasks. By using stateless functions, serverless computing decouples memory from compute, allowing both to be provisioned, managed, and priced separately. Additionally, in this context, the cloud provider is now responsible for creating and managing function instances automatically and transparently in worker nodes, as well as executing all operational tasks (e.g., server and operating system maintenance, patching, logging, load balancing or auto-scaling). Finally, serverless computing also significantly reduces application deployment costs through a pay-as-you-go model, where users are charged only based on the resource (e.g., CPU, network or memory) that they consume.

In addition to the obvious benefits that serverless brings to software developers (in terms of flexibility, scalability, performance and cost), it is worth noting that cloud providers can also enjoy benefit from its use. Because functions are called only occasionally and are executed for very short periods of time, cloud providers can achieve a higher degree of co-location on their servers and further optimize usage. These last two points, when carefully planned and orchestrated, can result in a more profitable model for cloud providers.

Serverless Ecosystem:

1) Function. Functions are the core component of a serverless platform. They can be written in many different programming languages (e.g. JavaScript, Python, and Go).[3] Software developers can write them themselves, rely on third-party open source functions or use proprietary functions for which they pay licensing fees. Functions are typically executed in a newly created isolated execution environment (e.g., container) in the worker node. Quoted functions are executed in response to external and/or internal events specified by the application owner (e.g., HTTP requests, modifications to stored objects, table updates, or transfers). change function).[3] It should be noted that not all functions defined need to communicate directly with the outside world (it may happen that some functions can only communicate with cloud functions and services other and not directly accessible from the outside).

2) Cloud service. Current serverless platforms integrate a variety of cloud services that are used to extend the capabilities of their functions, such as to collect various types of data (e.g., using Amazon Kinesis).[4] to react quickly to events (e.g. using Google Cloud Pub /Bus secondary messages).[1] system gateway or API), to manage the entire application lifecycle and enable DevOps capabilities (e.g., using Microsoft Azure DevOps), or to achieve long-term and short-term storage (e.g., using Amazon S3 and DynamoDB).

3) Security tools. Cloud providers offer software developers a set of tools and services to help manage workflow security. Some of these tools and services are also used as part of microservices; However, the task of setting them up correctly becomes much more difficult in serverless mode. The Identity and Access Management (IAM) service, for example, allows configuring granular access controls to authenticate and restrict the resources that functions have access to.[4] Another widely used security service is Virtual Private Cloud (VPC), which allows the creation of private, isolated networks for secure communication between applications belonging to the same organization. In addition to the services mentioned, we believe that other services and tools, such as those used for runtime application self- protection (RASP), infrastructure analysis layer-as-code (IaC) and component analysis of the source code, can play an important role.[2] important role in protection. Serverless applications and platforms are resistant to attacks.

4) Control plane features. Serverless platforms often include many control plane features that allow cloud providers to operate, manage, and monitor their infrastructure. For example, there is an orchestration component that manages the process of assigning functions to worker nodes. Similarly, a monitoring component is used to periodically check the status of worker nodes, the software they are running, and the runtime environment running on them. To achieve this purpose, the monitoring component collects measurement data, logs, and some metrics emitted by worker nodes. This way, if an error is detected, the affected functions can be quickly instantiated on other worker nodes. While features may vary slightly across many serverless platforms, what they have in common is that the data plane will receive periodic configuration updates from the control plane and control plane. The controller will regularly receive (or collect) the operating status of the control plan. . data plane. So it is essential that the control plane remains synchronized withthe data plane.

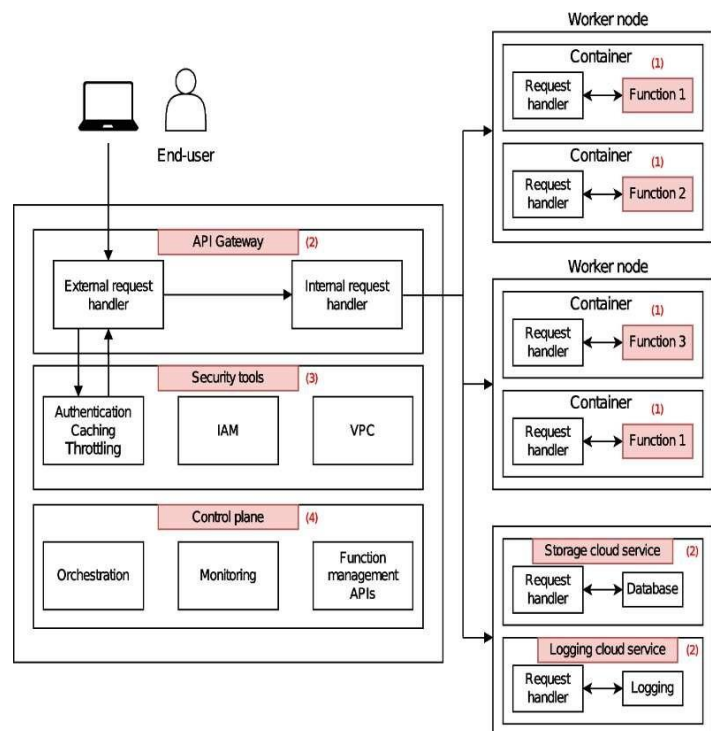


Fig-1

2. LITERATURE SURVEY

The area of serverless applications is still quite new and developing rapidly.[1] Cloud providers offering serverless services are increasing and evolving. The literature search includes the Diva portal in order to find out what earlier exam works has been conducted in the area of “serverless” and “serverless application”. [2] BTH library’s own search tool has been used to further examine what studies, books and articles have been released in the area of “serverless” and “serverless application” globally.[4] By re-iteration from a recent article in the serverless area more theses (published by Chalmers) connected to the field were found through the library web page of Goteborg’s University. [3]As this area is developing rapidly, only references from 2019 and onwards will be considered. Finally, the Serverless Framework and the serverless providers that will be examined in the report all have their own guides and information that will be utilized.[3]

Security controls exist at the infrastructure level

Current serverless platforms typically run functions in containers (or similar execution environments) protected by various opensource security mechanisms and services (some built into the kernel). Linux) combined with security mechanisms developed by the cloud providers themselves. In the following, we focus only on widely used opensource security mechanisms (as mechanisms developed by cloud providers are often ad-hoc and often not publicly available or well documented). enough).[1] Note that these security mechanisms play an important role in securing runtimes used in production environments today such as g-Visor and Firecracker; g-Visor is essentially an additional layer of security developed on top of Linux security mechanisms, while

AuthorName	Methodology	Conclusion
Bishop M., Gates C., “Defining the Insider Threat”, in Proc. of the 4th Annual Workshop on Cyber Security and Information Intelligence Research, Tennessee[4]	The main goal of the proposed hierarchy is to produce a security index that describes the security accomplished by an evaluated cloud computing environment to secure data breaches	In conclusion, the proposed hierarchy serves as a valuable framework designed with the primary objective of generating a comprehensive security index. This index is intended to effectively characterize the security measures implemented within a given cloud computing environment, specifically in the context of safeguarding against data breaches. By employing a systematic approach, the hierarchy offers a structured and evaluative methodology to assess and quantify the security achieved in such environments. The emphasis on preventing data breaches underscores the importance of maintaining the confidentiality, integrity, and availability of data in the rapidly evolving landscape of cloud computing.

<p>Kelly D, Glavin FG, Barrett E (2021) Denial of wallet– Defning a looming threat to serverless computing. Journal of Information Security and Applications [2]</p>	<p>Successful analyses of insider threats hinge on the availability of appropriate data sources. Several enterprises utilize log data for insider threat analytics. The report also provides a comprehensive list of logs which can serve as essential data sources for insider threat detection processes.</p>	<p>In conclusion, the success of insider threat analyses is intrinsically tied to the accessibility of relevant data sources. This report has shed light on the prevalent practice within enterprises of leveraging log data for the purpose of insider threat analytics. Recognizing the critical role of data in fortifying security measures, the report goes further to furnish a comprehensive list of logs that stand as indispensable data sources for enhancing the efficacy of insider threat detection processes.</p>
<p>Theoharidou M., Kokolakis S., Karyda M., Kiountouzis E., "The insider threat to Information Systems and the effectiveness of ISO 17799", Computers & Security, Vol. 24</p>	<p>The main goal is to provide a secure website by providing OTP or biometrics in the registration process.</p>	<p>In conclusion, implementing a robust security mechanism in the form of OTP (One-Time Password) or biometrics during the website registration process is a crucial step toward enhancing overall security and safeguarding user accounts. The primary objective of this approach is to fortify the authentication process, mitigating the risk of unauthorized access and potential security breaches.</p>

Firecracker sandboxes (running in user space) are also limited by security mechanisms of Linux like seccomp, c group and name spaces.

These security mechanisms can be grouped into the following four categories: (i) server hardening; (ii) process isolation; (iii) network security; and (iv) access control.

For an overview of the security mechanisms in the first three categories, we refer the reader (as these mechanisms are generally applicable to containers regardless of what content is executed within them). When it comes to access control, cloud providers typically provide some mechanism built into the API gateway to limit, cache, authenticate, and authorize external API calls before transferring them.[3] forward requests to the respective functions, for example, by relying on external identity providers or specifying a scope. IP address from which legitimate requests may come.

Security of current mechanisms. In recent years, researchers have delved into the real security guarantees provided by existing mechanisms used to protect container-based infrastructure.[1] This led to the identification of critical weaknesses in the security mechanisms used for process isolation and network security. Furthermore, previous research has shown that server hardening mechanisms, such as seccomp, App Armor, and SELinux, require cloud operators to manually configure them, which is the labour intensive and error-prone.

Mechanism of Serverless Security:

Serverless security is an additional layer of protection added directly to the application to secure code functions. Thus,[1] it provides developers with a compliance and security position on their applications. Here's how serverless security works:

1. It is based on event architecture

Most serverless architectures provide a multitude of event sources that can trigger the execution of serverless functions. Specific events or triggers, such as file downloads, database modifications, or newuser registrations, trigger automatic code execution.

2. Infrastructure is managed by the cloudprovider

The provider ensures that when a function is called, the necessary resources are transparently allocated to execute that function. Cloud providers, such as AWS with Lambda or Google Cloud with Cloud Functions, do the heavy lifting. They manage server provisioning, maintenance, and scaling. This allows developers to focus on writing and deploying code.

3. Key components of Serverless Security

Function. These are blocks of code, often designed to perform a specific task.[2] They are called or executed in response to an event Cause. The entity or action that initiates a function. For example, an HTTP request can act as a trigger for a function designed to retrieve data. Event source. These are AWS services or applications created by developers that generate events and can act as triggers.[2] Popular event sources include Amazon S3, DynamoDB, and API Gateway.

Working Of Serverless Architecture

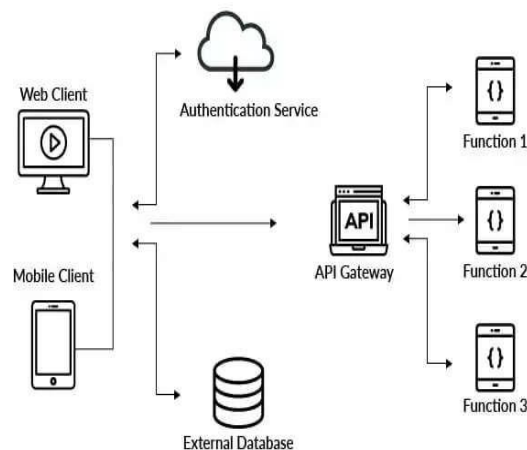


Fig-2

Threat Model:

Serverless platforms are complex and dynamic ecosystems with many distinct components. To design a secure serverless ecosystem, one must consider the security provided by each component and their interactions. Additionally, to properly shape the serverless security ecosystem, as will be done next, we must first define the corresponding threat model. To do this, we mainly distinguish between two types of opponents: i. outside; and ii. internal, will be discussed later. [4]

External adversaries often execute their attacks from outside the cloud by leveraging user-controlled input fields in one of the existing APIs provided to handle events. The same goes for serverless platforms. These attacks could allow an adversary to execute arbitrary commands inside a function to retrieve sensitive data (e.g., session tokens stored in an environment table) or to tamper with execution. In any function (or cloud service that receives malicious input and does not) apply appropriate input data hygiene techniques). While some code injection attacks are widely known because they can be applied to standard web applications (e.g., applications that exploit cross-site scripting or applications that rely on code injection), /command), however, serverless functions can be triggered from a variety of event sources: this latest attack feature significantly expands their attack surface. Internal adversaries refer to adversaries that have full control over one (or more) functions and launch attacks from within the cloud. In the case of public clouds, it is relatively easy for these adversaries to deploy malicious functions in an attempt to launch attacks from the inside. These competitors may attempt to: i. create secret channels; ii. conduct privilege escalation attacks (e.g., to compromise other co-resident functions or worker nodes); iii. recover or tamper with sensitive data (e.g., data in storage services); iv. gain knowledge about the execution environment and infrastructure; or v. carry out various types of denial of service (DoS) attacks (including so-called denial of wallet attacks). In another area of research, researchers also point out that if there are subscription services where serverless functions developed by other software developers can be found,[2] adversaries Having access to the registry can carry out so-called stealth attacks. The goal of these attacks is to spread malicious container images by exploiting potential typos made by container users.

Similarly, there are attacks where the attacker's goal is to influence the scheduler to co-locate the attacker's application with the targeted victim applications. It is worth mentioning that collocation is an important prerequisite for performing certain attacks such as Rowhammer, Specter or Meltdown.

While privacy issues are beyond the scope of this article, it is worth mentioning that, from a privacy perspective, there is growing concern that cloud providers may accidentally or intentionally unintentional disclosure of sensitive data to third parties (e.g., through malicious insiders). Because of this second threat, it is common in the research community to view cloud providers as honest but curious entities. In this model, cloud providers are expected to perform customer functions as expected, but at the same time, they may try to learn as much information as possible about the calculations being performed. is performed and the data is stored. [1]

In the following sections, we analyse the impact of serverless computing on security, discussing the advantages and disadvantages of this paradigm in relation to its contribution to the security posture of the ecosystem. status is supported.

Insider Threat In cloud computing:

Regardless of the technical and operational countermeasures deployed in the infrastructure, it is difficult to defend against accidental or intentional human actions. The Insider threats affect virtually all infrastructure and remain an open research topic for decades. problem when it comes to traditional IT infrastructure, although the manifestation of insider threats in cloud computing has not yet been fully researched. Give

In the functional context of cloud computing, a malicious insider with access to cloud resources can cause more damage to the organization. Furthermore, as If an attack can affect a large number of cloud users, the impact of that attack will be significant.

Cloud outsourcer: An insider is an employee of an organization that has outsourced part or all of its infrastructure to the cloud.

This is the worst-case scenario for both the cloud provider and the cloud customer, i.e. malicious system administrator working for a cloud provider. Due to their business role within the cloud provider, Insiders may use their authorized user rights to access sensitive data. For example, administrators are responsible for performing regular backups of systems where client resources (virtual machines, data stores) are stored, could exploit the fact that it has access to backups and thus filter out sensitive user's data. Detecting such indirect data access can be a difficult task. Depending on the motives of the insider, the results of such an attack on cloud infrastructure will vary, from data leakage to severe failure of affected systems and data. Regardless, the commercial impact on suppliers will be significant. All common Cloud types (IaaS, PaaS, SaaS) are also affected by insider attacks as long as insiders have (or could have) access to a data centre or cloud management system. One could argue that the aforementioned impact of insider threat in the cloud is similar to the internal impact in the classic outsourcing model. That is partly true because the decision to outsource comes with inherent risks disclose sensitive data to third parties. In fact,[3] it provides a global solution for outsourcing through IaaS and PaaS. Therefore, the cloud computing model can be used to outsource much of the work infrastructure rather than specific services, such as web hosting or application hosting.

Effective insider threat mitigation requires defence in depth and volume
Countermeasures are taken by both the cloud provider and the customer. Client side

- Confidentiality/integrity

Even in IaaS, where the customer has ultimate access to the cloud infrastructure (administrative access to the virtual operating system), it can be difficult for cloud customers to detect this. Someone has gained unauthorized access to their data using operating system-level security mechanisms such as IDS/IPS. The reason is that an intern is working for a cloud service provider (e.g., malicious administrator) has access to physical infrastructure that the customer does not control. Clients can use cryptographic techniques, for protection purposes the security and integrity of their external data. However,[1] encryption is one Practical solution mainly for bulk data storage and especially for static data. Warehouse data in encrypted form and decrypt it every time it is needed to access it (one common technique), does not constitute a sufficiently effective defence against insiders, because the decryption key must also be stored somewhere in the cloud. Considering that the insiders can have access to the physical server and thus can access physical memory used by the customer's virtual system, any encryption key stored in memory can be obtained. A strong solution to this problem is to not store the encryption key in the cloud but perform data operations directly on encrypted data. Some techniques have been proposed to solve this problem. However, the overhead of these techniques is often so high that it causes they are currently[2] impractical for real-world applications.

- Availability

When it comes to availability, using multiple data centres, ideally located in different regions, is the only effective solution, as long as the cloud provider does not experience global outages. Some vendors offer such options to their customers, including automatic failover to a backup data centres, in case one instance of the primary data centres fails. Failure, such geographic redundancy protects customers as long as malicious insiders cannot interfere with multiple data centres at the same time.

Data Breaches:

As per Google, this year's research shows that more than a third (39%) of businesses experienced a data breach in their cloud environment last year, up from 35% reported in 2022. In fact, human error is believed to be the main cause. more than half (55%) of respondents experienced a cloud data breach.

1. File-based malware

Most cloud storage providers today offer file synchronization, which is when files from your local device are automatically uploaded to the cloud as they change.

File sync is an ideal solution for businesses because it creates a “central hub” of files that teams can access and work on across different devices. But it's great for file- based malware for the same reason. Cloud storage providers like OneDrive or Drop Box are attached to a local folder on your computer, and files stored in the cloud are synced to that folder.[1] To your device, these cloud folders are just like any other folder. So, if you download a malicious file to your local device, there is a path to your company's cloud, where it can access, infect, and encrypt company data. This type of ransomware attack is also known as “Ransomcloud”.

2. Weaknesses of IAM policy

Each user in a cloud environment has their own roles and permissions that govern the access they have to certain parts of the cloud, and because cloud workloads are accessible online, hackers just need your security information to get the “keys to the kingdom”.

This is why strong identity and access management (IAM) policies are essential to cloud security.

Identity and access management is a way to control user permissions and access to cloud resources. You can think of IAM less as a single piece of software and more as a framework of processes, policies, and technologies.[1] According to Palo Alto Networks, the most common cloud data breaches start with misconfigured IAM policies or credential leaks.

Specifically, the researchers found that poor IAM configuration caused 65% of detected cloud data breaches, followed by low password usage (53%) and allowing Password reuse (44%).[4]

3. API is not secure

Many companies use application programming interfaces (APIs) to connect applications and data to the cloud. At a high level, APIs allow different applications to communicate with each other over the network.

Because APIs provide a way to query, access, and modify critical data, cloud threat actors are constantly looking for vulnerabilities in it. Here we go: In a 2021 analysis of affected customers, IBM's X- Force IR team found that two-thirds of cloud data breaches were caused by compromised APIs. wrong configuration.

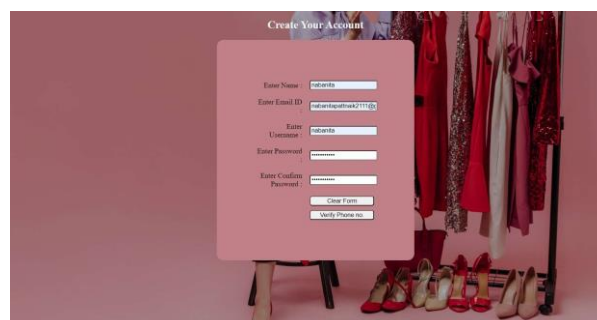
4. Poor configuration

In VMware's 2021 State of Cloud Security Report, one in six companies surveyed experienced a cloud data breach due to misconfiguration in the past year. Researchers have found that among all cloud services, cloud storage has the highest misconfiguration rate.

Given this, it's no surprise that there have been many cloud storage data breaches in recent years.

Last year, misconfigured Amazon S3 buckets exposed more than 1,000 GB of data and more than 1.6 million files from dozens of cities across the United States. Microsoft Azure isn't much better: in 2021, misconfigured Azure storage accounts exposed millions of files containing sensitive information.[1]

Authentication



The image shows a registration form titled "Create Your Account" overlaid on a background image of a clothing rack with red and white garments and several high-heeled shoes. The form contains the following fields and buttons:

- Enter Name:
- Enter Email ID:
- Enter Username:
- Enter Password:
- Enter Confirm Password:
- Buttons: "Clear Form" and "Verify Profile Info."

Fig-3 Registration Form

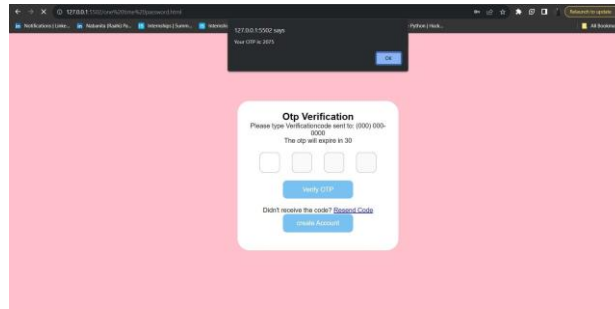


Fig-4 OTP verification within 10sec

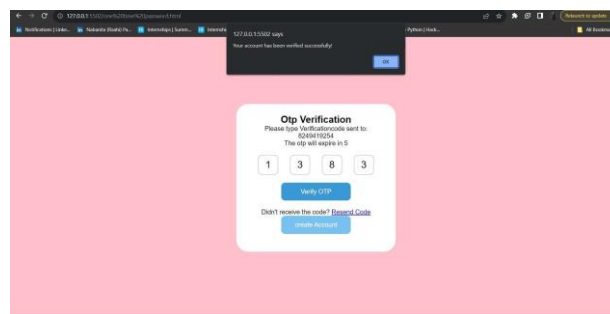


Fig- 5 OTP verified

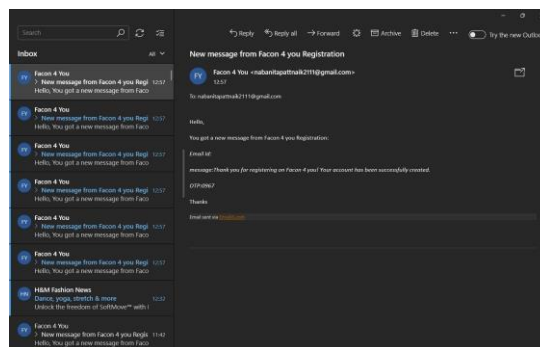


Fig-6 Received OTP via E-Mail -Account Signed in successfully

Conclusion:

In this article, we have shown that, on the one hand, serverless computing offers additional security features, but on the other hand, it also introduces its own threats and security challenges, helping clearly distinguishes it from current virtualization technologies. In particular, we examined current serverless architectures, classified current security threats, presented practical tips to improve the current state of security, and highlighted research directions security to make serverless the model of choice when looking for virtualization solutions where security is paramount. We believe that our contribution, in addition to being valuable, paves the way for further research in this area, which is challenging and relevant for practitioners, industry, and academia.

Future Work:

This thesis focused on deploying a serverless back-end API to different providers through the Serverless Framework. There were many similarities between the deployments but also differences to be aware of. Best practices in serverless are constantly adopted to new solutions. AWS is currently the most popular choice for serverless applications but this might change over time. Multiple large vendors might over time streamline code structure and the usage of BaaS services which might decrease vendor lock-in. Conducting a similar study as this thesis in a few years time might render a different result. Perhaps the deployment configurations would be more similar even without the usage of a framework. The Serverless Framework (or a new framework) might be updated to better support other providers than AWS. Another angle would be to implement the same non proprietary solutions for database and authentication on all three providers. By examining the code it could tell how large part of the code base that can be reused between the providers. Security and permissions are two important areas when working with serverless applications. These are important parts of the serverless architecture and it would be interesting to compare how this is handled by different vendors.[3] The Serverless Dashboard supports monitoring and troubleshooting of AWS deployments. The solutions to logging and troubleshooting had many differences between the Serverless Dashboard, Azure and Google. Logging can be a difficult to gain a deeper insight into and has not been further explored in this thesis. Perhaps an investigation into this area could help set guidelines for a more generic solution.[3]

References:

1. Nam J, Lee S, Seo H, Porras P, Yegneswaran V, Shin S (2020) BASTION: A Security Enforcement Network Stack for Container Networks. In: USENIX Annual Technical Conference (USENIX ATC). pp 81–95. USENIX Association[1]
2. Kelly D, Glavin FG, Barrett E (2021) Denial of wallet–Defning a looming threat to serverless computing. Journal of Information Security and Applications [2]
3. Theoharidou M., Kokolakis S., Karyda M., Kiountouzis E., "The insider threat to Information Systems and the effectiveness of ISO 17799", Computers & Security, Vol. 24 [3]
4. Bishop M., Gates C., "Defining the Insider Threat", in Proc. of the 4th Annual Workshop on Cyber Security and Information Intelligence Research, Tennessee[4]