# Exploring Serverless Security: Identifying Security Risks and Implementing Best Practices

Ramasankar Molleti, Independent Researcher, Email: sankar276@gmail.com , USA

Anirudh Khanna, Independent Researcher, Plano, TX, USA, Email: mailtoanirudh@gmail.com

**Abstract**
FaaS or Serverless computing extends the existing cloud computing by removing or abstracting the notion of a server and the need to scale up and down on demand. This paper seeks to discuss the security issues that are associated with serverless architecture, with special focus on authentication, data encryption, compliance issues and those risks that are unique to different vendors. It compares existing security architectures and measures and current industry benchmarks originating from premier cloud platforms such as AWS Lambda, Azure Functions, Google Cloud Functions. Some of the critical risks including injection attacks, insecure deployments, and operational monitoring have similar threat proneness models accompanied by secure coding, IAM integration, DevOps rules, and recommendations.

***Keywords - DevOps, IAM, Serverless Architectures, injection attacks, monitoring, FaaS.***

## I. Introduction

### A. Overview of Serverless Computing

Serverless computing or Function-as-a-Service (FaaS) is a new approach to Cloud computing which initially frees the developer from managing the cloud's servers [1]. Depending on the cloud type, the classic ones require developers to set up and manage servers while serverless hides the management and allocation of the server from a developer and just provides it as a service. This abstraction enables the developers to run the code in small event-coined functions while only paying for the amount of computing resources used during the execution.

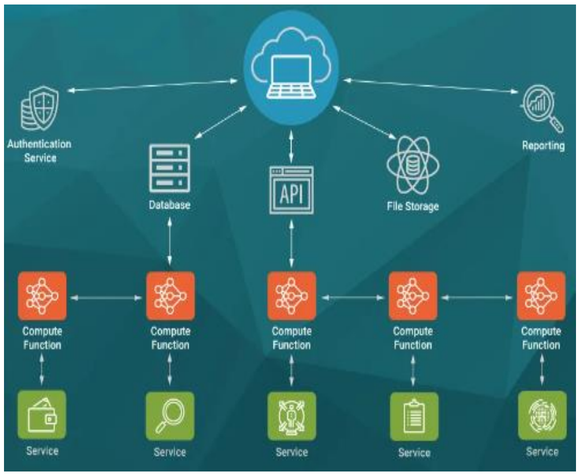

**Figure 1: Serverless architecture**
(Source: https://www.sentinelone.com)

### B. Importance of Security in Serverless Architectures

While the management of the infrastructure decreases, serverless architectures increase the exposure to threats. Data processing and business functions involve other cloud services like Amazon Web Services, APIs, and external systems, which increases the possible paths for an attack [2]. There is arguably no greater area than identity and access management to stress the importance of timely and efficient management. As for the security, serverless applications are safe due to the enforcement of the principle of least privilege, which adds limitations to the execution of unauthorized code/data. Serverless functions sometimes operate on the content of the data, and this content can be very sensitive. Data encryption when data is at rest and when moving across networks helps to reduce data losses which compromises the security of data [3].

Some of the metrics for monitoring will not be new but will not readily cover serverless computing paradigms. Incorporating effective logging systems and the real-time monitoring allows for early identification of the odd and/or security breaches. The use of serverless needs to ensure the compliance of it with industry rules like HIPAA [4]. There should be easily implementable and compatible compliance narratives that may support serverless technologies concerning data security and compliance to the rule of laws. bowl providers have different kinds of securities control and configurations when it comes to serverless services. It is therefore equally important to comprehend and manage these controls to eliminate specific risks related to certain vendors.

## II. Security concerns in Serverless Architectures

### A. Authentication and Authorization Issues

Security of the system is very important in serverless solutions which include; Authentication and authorization that will only allow rightly authorized entities to access resources and perform functions. Identity and access management controls in distributed serverless functions and cloud services are complex to design. Inability to configure the identity management properly implies that people with illicit intentions can gain access and maybe tamper with sensitive information. The IAM systems offered by cloud providers are some of the popular ones that interconnect with serverless functions. The integrations of services with the account management system have to be properly set to lock down the access rights to the minimal level possible to dodge privilege escalation vectors. Most serverless applications rely on Third-party identity providers like OAuth and OpenID Connect [5]. It is crucial for the tokens to be secure and for communication being correctly validated to counter the risks that are related to malicious tokens.

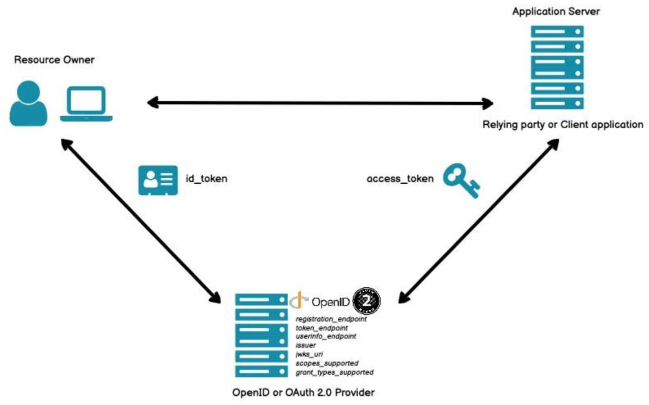

**Figure 2: OpenID Connect Discovery and OAuth2 Authorization Server Metadata**

(Source: https://miro.medium.com/)

### B. Data Security and Encryption Challenges

Serverless functions sometimes work with data which has to be encrypted while stored and during the transfer. Applying highly secure encryption guarantees data confidentiality and its integrity which minimizes risks of data leaks. In the consideration of using serverless, developers should embrace the appropriate method of coding, particularly in dealing with the sensitive data within the functions. This includes, limiting exposure of data when transmitting it, avoiding the act of coding credentials directly, and securely passing data to external services. Serverless applications have to meet the constraints related to data locality and corresponding regulatory standards (e. g. , EU GDPR, HIPAA) [6]. To minimize legal issues that arise from compliance with the legal aspects of data processing and storage, it is critical to align the location of these locations with the regulations of the law.

### C. Compliance and Regulatory Considerations

When it comes to GDPR, CCPA, consent management is key, along with minimizing the data and the ability to respond quickly to data subject access requests. Some industries are very specific about data security and comply with certain regulations (e. g. , PCI DSS for health care, HIPAA for finance) [7]. These standards are mandatory for serverless applications to deal with the sensitive data to avoid penalties and lawsuits. There is a need to have detailed audit logs of the system and make logging available for serverless functions for compliance. These logs permit visibility over the data access, processing activities and even the occurrence of security incidents facilitating compliance auditing and investigations.

## III. Evaluation of the existing Frameworks and Best Practices

### A. Review of the Security Frameworks

Many of the providers that offer cloud-based solutions have established their security models and regarded guidelines for serverless, with the purpose of eradicating some of the weaknesses and enhancing the security as an entire. An example of such a platform is AWS Lambda, the leading, serverless computing platform owned by Amazon Web Services. There are several main areas that are relevant with regards to security concerning AWS Lambda. First of all, AWS mandates IAM (Identity and Access Management) policies and roles. Next using least privilege permissions, it means the function can only perform an operation on a resource that it requires and nothing more, thus reducing the exposure points. Another crucial component of security that AWS Lambda uses is encryption. AWS has its Key Management Service (KMS) used for encrypting sensitive data both in storage and in communication within the scope of serverless architectures [8]. That way, the data is shielded from leakage and acts of espionage throughout the data's lifecycle. Also, AWS Lambda provides focus on the protection of environment variables including but not limited to API keys and database credentials. Secure environment variables are useful as they provide developers with a way to prevent normally exposed information from being seen and used wrongly.

### B. Comparison of best practices

Alerting and reporting are two fundamental features of security based systems and AWS Lambda utilizes AWS CloudWatch a lot in this regard. CloudWatch allows tracking function execution metrics in real-time, logs function results for traceability, and helps set up an alarm that will notify of suspicious activity. In addition, AWS Lambda encourages the formation of good coding principles regarding security, such as preventing injection attacks and avoiding inadequate exception handling, thus improving the anti-threat capabilities of serverless applications as a whole. Security frameworks with corresponding best practices are also provided by other key cloud service providers including Microsoft Azure and Google Cloud Platform. Azure Functions for example can directly use Azure Active Directory for identity, which aligns well with enterprise IAM. Azure also contains Azure Key Vault for managing the storage of keys and secrets as well as incorporation of cyclic encryption equivalent to AWS KMS [10]. On the other hand, Google Cloud Functions uses Google Cloud Key Management Service (KMS) for encryption requirements; the solution mainly focuses on the issue of securing data in serverless environments [9].

**Table for the comparison of the security based systems in serverless cloud environments**

| Feature | AWS Lambda | Azure Functions | Google Cloud Functions |
|---|---|---|---|
| Identity and Access Management | AWS IAM | Azure Active Directory | Google Cloud IAM |
| Encryption | AWS KMS | Azure Key Vault | Google Cloud KMS |
| Monitoring and Logging | AWS CloudWatch | Azure Monitor, Application Insights | Stackdriver Logging, Monitoring |
| Compliance Certifications | SOC 2, ISO 27001 | SOC 2, ISO 27001 | SOC 2, ISO 27001 |

As far as monitoring and logging are concerned, Azure Monitor and Application Insights are used by Azure Functions for continuous monitoring along with logging and diagnostics of function and application performance. Also, Google Cloud Functions work synergistically with Stackdriver Logging and Monitoring to generate real-time data on the Function's performance and operational statistics. AWS, Azure and Google cloud all provide compliance certifications and programs that can help organizations with compliance items including SOC 2 and ISO 27001. Such platforms offer effective features which include Auditing, Logging and reporting hence useful in showing compliance of an organization in their sectors of operation, also important in ensuring that an organization complies with set regulations in various industries.

## IV. Security risks

### A. Evaluation of the technical vulnerabilities in serverless environments

#### 1. Injection Attacks

Serverless computing brings in certain technical risks that the organization needs to manage to protect its applications. Among these the major concern is the injection attacks with reference to both structured query language injection and NoSQL injection [17]. Even though serverless platforms hide the server management from the developer, invalid input sanitization in serverless functions remains as one of the vulnerabilities in which attackers can inject codes to the queries or commands and, as a result, lead to data leakage or loss.

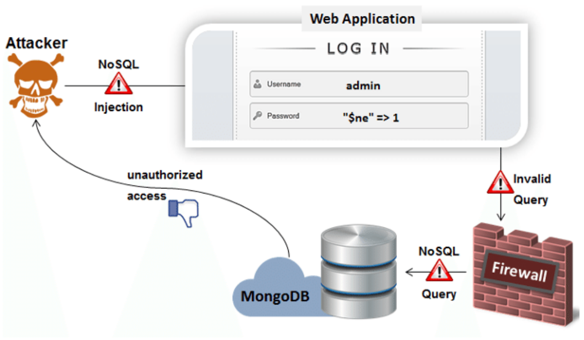

**Figure 3: NoSQL injection attack in web applications**

(Source: https://www.researchgate.net)

#### 2. Insecure Deployment Configurations

The second major threat source is the insecure settings when the application is deployed. Since serverless applications are mostly implemented on cloud providers, security controls including IAM roles, policies and environment variables are usually configured by the cloud providers. With poor configurations like extensive authorizations, lack of sufficient security for environment variables that contain sensitive information, there is propensity to unauthorized data and services access.

**3. Denial-of-Service (DoS) Attacks**

A classic threat, which is still very much present in serverless architecture, is the Denial-of-Service (DoS) attack [16]. Attackers can be considered as a load testing tool for serverless functions in which the legitimate client load is flooded, or the inputs purposely constructed to trigger a denial of service condition. Inherent in the serverless platforms is the capability of scaling the resources dynamically, based on the required amount; however, poorly designed functions or insufficient resource constraints contribute to vulnerability to DoS attacks.

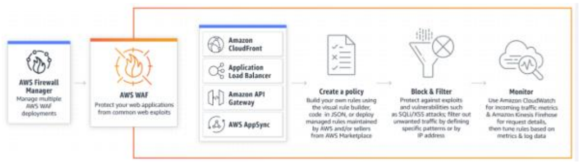
AWS Firewall Manager
Manage multiple AWS WAF deployments
AWS WAF
Protect your web applications from common web exploits
Amazon CloudFront
Application Load Balancer
Amazon API Gateway
AWS AppSync
Create a policy
Build your own rules using the visual rule builder, code in JSON, or deploy managed rules maintained by AWS and/or sellers from AWS Marketplace
Block & Filter
Protect against exploits and vulnerabilities such as SQLi/XSS attacks; filter out unwanted traffic by defining specific patterns or by IP address
Monitor
Use Amazon CloudWatch for incoming traffic metrics & Amazon Kinesis Firehose for request details, then tune rules based on metrics & log data

**Figure 4: AWS WAF Security Automations uses AWS CloudFormation to block SQL injection attacks**
(https://aws.amazon.com/waf/)

**B. Evaluation of the operational risks and management challenges**

**1. Monitoring and logging issues**

These operational risks coupled with managerial challenges present difficulties on the security and operational reliability in serverless structures. Specifically, monitoring and logging pose some of the most crucial issues because serverless functions are short lived. There is a weakness that occurs especially when using traditional monitoring tools because they may not be able to record transitory function execution or timely details of resource consumption and its associated performance characteristics. User organizations have to employ special monitoring tools and practices of logging to gain visibility into how functions work and when a security breach takes place.

**2. Incident Response and Recovery**

Control of a number of incidents and their subsequent recovery in serverless architectures differs from what occurs in traditional ones. Precisely, serverless functions are stateless, which makes the process of responding to incidents challenging because functions are created and deleted based on event occurrences. It is also important to define the adequate processes for such cases; this involves having a swift notification system, proper procedures for handling security incidents which are aligned with the serverless manner of working, as well as established practices for creating scripts for incidents and rapid deployment of patches or workarounds in case of a security incident.

**C. Vendor-Specific Risks**

Every serverless computing platform exposes vendor-specific threats that an organization has to manage depending on the cloud solution it employs. AW Lambda, Azure Functions, and Google Cloud Functions all have different forms of security with various settings as well. AWS Lambda, for instance, is pointed out as being integrated with AWS IAM for authorization and AWS KMS for encryption [11]. If an organization is using AWS Lambda, then they need to ensure that IAM roles are set up with appropriate permissions with low risk profiles applied to them, in addition to continuing to encrypt data to protect it [13]. Azure Functions utilize Azure AD for Identity and access management while the Azure Key Vault for storing securely and encrypting keys. Still, Azure Functions' application settings misconfiguration along with inadequate attention to the Azure Monitor and Application Insights may result in operational threats or future security breaches.

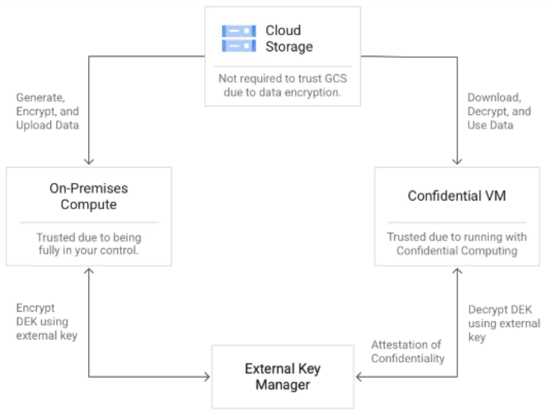
Cloud Storage
Not required to trust GCS due to data encryption.
Generate, Encrypt, and Upload Data
Download, Decrypt, and Use Data
On-Premises Compute
Trusted due to being fully in your control.
Confidential VM
Trusted due to running with Confidential Computing
Encrypt DEK using external key
Decrypt DEK using external key
Attestation of Confidentiality
External Key Manager

**Figure 5: Google Cloud Key Management**
(Source: https://lh3.googleusercontent.com/)

Google Cloud Functions uses Google Cloud IAM for authorization and Google Cloud KMS for encryption so that this option has a relatively high level of security [12]. Nonetheless, lack of logging and monitoring through Stackdriver Logging and Monitoring may prove a challenge in securing organizations' ability in detecting and responding to security incidents. Addressing these vendor-specific risks would need having specific security controls and practices for each vendor applied to the systems, processes, and people interfacing with any of the platforms. To safeguard and secure an organization's serverless environment, risks should be assessed, proper security measures should be put in place, and organizations should always update themselves on the new changes and developments of the serverless provider they are using.

**Table for the concise comparison of vendor-specific security features and potential risks**

| Vendor | AWS Lambda | Azure Functions | Google Cloud Functions |
|---|---|---|---|
| Identity Management | AWS IAM | Azure Active Directory | Google Cloud IAM |
| Encryption | AWS KMS | Azure Key Vault | Google Cloud KMS |
| Monitoring and Logging | AWS CloudWatch | Azure Monitor, Application Insights | Stackdriver Logging, Monitoring |
| Potential Risks | IAM role misconfigurations, inadequate encryption | Azure AD misconfigurations, insufficient monitoring | Lack of logging and monitoring |

## V. Security guidelines and recommendations

### A. Access Control Best Practices

Identity and access management is the first layer in the process of protecting serverless applications as only the right entities are allowed access to pieces of data and perform the right tasks. Some of these are; the principle of least privilege, which means that permissions are granted in accordance to the least privilege needed to perform duties. For serverless environments, this includes identity and access management roles/policies, which limit actions/operations on some resources. Recommendations for IAM configurations include the frequent performing of audits and reviews of the system as this will allow for early detection of rogue access settings or suspicious attempts at accessing the system.

### B. Secure Coding Practices for Serverless Applications

Lack of security in serverless applications is one of the major issues, and having secure codes help to eliminate the primary risks and threats.

Examples of measures that should be taken by the developers include input validation to pre-empt injection attacks for instance; SQL injection, NoSQL injection [15]. Thirdly, it is advisable not to hardcode such things as API keys in databases or credentials in the function's code, for example, helps to avoid such dangers. Validating function arguments before calling them, use of try-catch statements is effective in making functions respond well to invalid operations by preventing flaws that may be exploited by hackers.

**C. Integration with Identity and Access Management (IAM) Systems**

Hence, the IAM system integration is crucial for identity management and handling access control measures in the serverless domain. Organizations should employ IAM options offered by the cloud providers (AWS IAM, Azure Active Directory, Google Cloud IAM, etc.) for user, application, and service identities when requesting access to the serverless functions. Having identity management as a centralized system provides an efficient way of managing accesses and implementing the security policies in an organization. More secure ways of IAM include use of MFA, SSO, monitoring and logging to enable identification of the suspicious or unauthorized activity [14]. This, in turn, would make it easy for organizations to have robust access control measures, secure coding practices, and integrate secure IAM systems thus making the serverless applications to have a scaled up security. These measures not only assist in managing risks that pertain to unauthorized access of systems and data breaches but also help in meeting the regulatory and organizational requirements while enabling proactively secure and sound serverless computing.

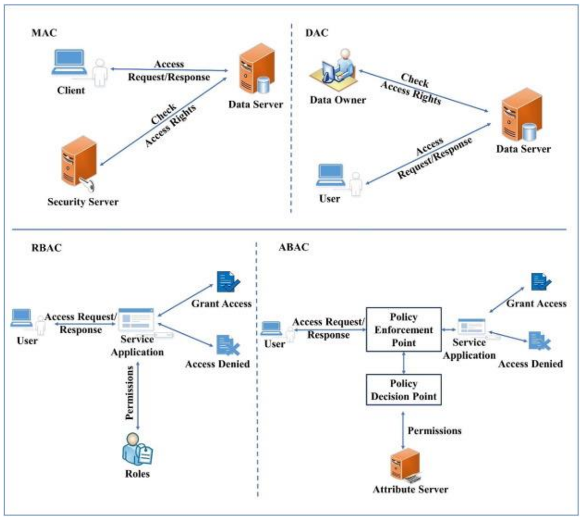

**Figure 6: Access Control Mechanisms in different cloud environment**

(Source: https://ars.els-cdn.com)

**VI. Challenges in serverless security**

**1. Addressing the Challenges of Serverless Security**

Serverless functions are stateless and triggered by events and thus, the more traditional monitoring methods do not suffice. Function management requires the application of specific technologies and approaches to increase awareness of the function's performance, resource consumption, and security issues, if any, in real-time. Security blunders that are related to serverless architectures include the formulation of incorrect identity and access management policies, including IAM roles, and the use of insecure environment values. To address such risks, strict deployment methods should be enforced and the security check conducted periodically.

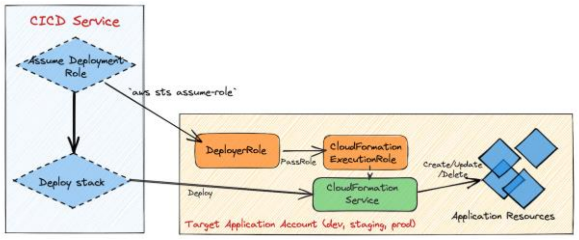

**Figure 7: Deployment of AWS Serverless application using IAM**

(Source: https://serverlessfirst.com)

Applications deployed on server-less architectures handle a lot of data that may be considered sensitive. Data confidentiality and integrity can only be guaranteed if the data is encrypted even when it is idle and data must be encrypted during transmissions to other systems, programmers should adhere to the best practices avoiding injection attacks, and sensitive information should be handled comprehensively. Third party and/or APIs are being used extensively in serverless applications. Validating the dependencies and using secure channels with correct security protocols is critical to avoid supply chain attacks and data breaches.

**2. Integration of Security into DevOps Practices**

Security issues can be detected in the early stages of the development lifecycle in DevSecOps by the embedded security as this can help in solving issues proactively. Static code analysis and detection of vulnerability can be conducted during the CI/CD pipeline which brings agility in the development process [18]. This helps in making repeatable and consistent configuration as serverless enhancements or deployments helps in enhancing security. This helps in detecting unauthorized changes and also helps in detecting vulnerabilities as it helps in automating the security tasks by the management of the configuration. It helps in compliance checks and manages security operations in serverless operations. This helps in getting automated response which enables mitigation and detection of the different security issues and this helps in minimizing the impact of the issues and downtime.

**3. Future Trends and Emerging Technologies**

The constant evolution of security solutions and products that would optimize for serverless environments, along with security monitoring, threat and runtime security tools. New adaptations of the zero-trust model in which users' identities and access are verified and authenticated one or more times according to strict policies, interrogated across the four dimensions of location irrelevance in serverless and hybrid cloud architectures to improve security postures. To build standard security practices, procedures and regulation particular to serverless computing, to champion the improvement of the standard guidelines and certifications across the industry. Real-time analysis of the huge amounts of telemetry data, through the application of artificial intelligence and machine learning techniques in order to identify security threats.

## VII. Conclusion

Security in serverless environments has a few peculiarities, which are associated with data protection and operational integrity. Some of the challenges include: visibility; deploying software securely; incorporation of highly secure security solutions into the DevOps process. In this case, they can lower risks by employing preventive measures, using automation, and having a pulse on new technologies. In the future, practical development of the specific security measures of serverless tools and standards will continuously improve the security environment of serverless, making it a credible and safe solution for current application development.

## Reference List

**Journals**

[1] Lynn, T., Rosati, P., Lejeune, A. and Emeakaroha, V., 2017, December. A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 162-169). IEEE.

[2] Ivan, C., Vasile, R. and Dadarlat, V., 2019. Serverless computing: An investigation of deployment environments for web apis. *Computers*, *8*(2), p.50.

[3] Hong, S., Srivastava, A., Shambrook, W. and Dumitraș, T., 2018. Go serverless: Securing

cloud via serverless design patterns. In *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*.

[4] Chris Munns 2018. Powering HIPAA-compliant workloads using AWS Serverless technologies. URL: https://aws.amazon.com/blogs/compute/powering-hipaa-compliant-workloads-using-aws-serverless-technologies/.

[5] Li, W., Mitchell, C.J. and Chen, T., 2019, November. Oauthguard: Protecting user security and privacy with oauth 2.0 and openid connect. In *Proceedings of the 5th ACM workshop on security standardisation research workshop* (pp. 35-44).

[6] Shah, V., 2019. Towards Efficient Software Engineering in the Era of AI and ML: Best Practices and Challenges. *International Journal of Computer Science and Technology*, *3*(3), pp.63-78.

[7] NetApp, Inc. 2019, Meeting Data Compliance with a Wave of New Privacy Regulations: GDPR, CCPA, PIPEDA, POPI, LGPD, HIPAA, PCI-DSS, and More. URL: https://bluexp.netapp.com/blog/data-compliance-regulations-hipaa-gdpr-and-pci-dss.

[8] Trach, B., Oleksenko, O., Gregor, F., Bhatotia, P. and Fetzer, C., 2019, May. Clemmys: Towards secure remote execution in faas. In *Proceedings of the 12th ACM International Conference on Systems and Storage* (pp. 44-54).

[9] Podjarny, G. and Ţal, L., 2019. *Serverless security*. O'Reilly Media, Incorporated.

[10] Hong, S., Srivastava, A., Shambrook, W. and Dumitraş, T., 2018. Go serverless: Securing cloud via serverless design patterns. In *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*.

[11] Rath, A., Spasic, B., Boucart, N. and Thiran, P., 2019. Security pattern for cloud SaaS: From system and data security to privacy case study in AWS and Azure. *Computers*, *8*(2), p.34.

[12] freeCodeCamp.org 2018, How to secure and manage secrets using Google Cloud KMS. URL: https://www.freecodecamp.org/news/securing-managing-secrets-using-google-cloud-kms-3fe08c69f499/.

[13] Müller, C., 2019. *Practical aspects of FaaS applications' migration* (Bachelor's thesis).

[14] Sikeridis, D., Papapanagiotou, I., Rimal, B.P. and Devetsikiotis, M., 2017. A Comparative taxonomy and survey of public cloud infrastructure vendors. *arXiv preprint arXiv:1710.01476*.

[15] Grumuldis, A., 2019. Evaluation of "Serverless" Application Programming Model: How and when to start Serverles.

[16] Zheng, S. and Yang, X., 2019. {DynaShield}: Reducing the Cost of {DDoS} Defense using Cloud Services. In *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*.

[17] Bhamra, K., 2019. *A Secure and Serverless Approach to Verification of Student Records* (Doctoral dissertation, California State University, Northridge).

[18] Buijtenen, R.V. and Rangnau, T., 2019. Continuous Security Testing: A Case Study on the Challenges of Integrating Dynamic Security Testing Tools in CI/CD. *17th SC@ RUG 2019-2020*, p.45.