Rethinking Cybersecurity: How Serverless Architecture Redefines Risk Management

Azar Mamiyev

Obuda University, Budapest, Hungary, azarmamiyev@stud.uni-obuda.hu

Abstract: Serverless computing is revolutionizing cybersecurity risk management with the introduction of new features—ephemeral functions, event-driven execution, and shared responsibility—that upend traditional security practices. In serverless environments, individual functions execute for a brief time to complete targeted tasks before they become non-existent, meaning security controls must quickly react to protect these fleeting processes rather than watch over static, long-lived systems. Additionally, since serverless applications react dynamically to events like user behavior or data triggers, they bring new vulnerabilities that require innovative threat detection and mitigation techniques. The shared responsibility model further complicates the matter by dividing security roles between cloud providers who secure the underlying infrastructure, and organizations who must protect and manage their own code and configurations. This work explores these singular attributes to illustrate how conventional security methodologies need to be reassessed and how more dynamic, recent models can protect against the novel types of attacks native to this adaptive computing paradigm.

Keywords: Serverless computing, serverless security, cybersecurity, cloud technologies

1 Introduction

Serverless computing is a cloud architecture model that provides a new paradigm in which developers write code without provisioning or managing the underlying infrastructure. The paradigm conceals the complexity of server management, resource scaling, and provisioning and allows organizations to deploy applications instantly and scale them dynamically according to demand. Major cloud computing service providers, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud, have introduced serverless computing platforms in the guise of AWS Lambda, Azure Functions, and Google Cloud Functions, which are gaining traction rapidly due to their ability to improve cost savings and performance. As companies increasingly adopt serverless models to automate and reduce overhead, it is important to take into account the security ramifications of this emerging architecture.

While serverless computing benefits are obvious, it also introduces new cybersecurity challenges that cannot be addressed by conventional security tools. In conventional systems, where the security controls are designed to protect long-running, static infrastructure and applications, serverless environments consist of short-lived, event-driven functions that are ephemeral and dynamically scaled. Such characteristics introduce new vulnerabilities in areas such as monitoring, data integrity, and access control. In addition, the shared responsibility between cloud providers in serverless architecture, where cloud providers are responsible for securing the underlying infrastructure while organizations must secure their applications and configurations, makes the security responsibilities harder to demarcate. With serverless environments changing increasingly on a daily basis, today's cybersecurity best practices that have been optimized for more conventional on-premises and cloud architecture are not adequate to address these emerging risks.

This paper discusses the cybersecurity challenges introduced by serverless architectures and argues that traditional risk management practices must be re-evaluated. By discussing the distinctive features of serverless computing—ephemeral functions, event-driven execution, dynamic scaling, and shared responsibility model—this paper aims to provide an end-to-end perspective on why existing security measures are insufficient. Besides, the goal is to identify and propose new, agile risk management techniques that can be adopted to protect organizations from the emerging threats of serverless environments. Through this research, the paper hopes to contribute to the current debate regarding how cybersecurity practices must evolve to cope with the rapid pace of development in cloud computing technologies.

2 What is Serverless Computing?

Serverless computing is a model of cloud computing whereby the cloud providers automatically take care of the infrastructure needed to run applications so that developers need not worry about server provisioning, scaling, and server maintenance. Under serverless architecture, the developers code which runs as a function of specific events or triggers rather than worrying about the life cycle of an application or system. The term "serverless" may be misunderstood as suggesting an absence of servers, but in reality, technically servers are used to execute the code; instead, the only variation is that the developers no longer have to care about the underlying hardware or infrastructure. All the issues related to management are taken care of by the cloud provider, and users only pay for actually consumed resources used in executing the code, rather than server running time.

Among the features of serverless computing are ephemeral functions. They are temporary and serve to conduct a specific job once they are presented with an event and proceed with termination as soon as they have completed the task. By being transient in nature, serverless functions prove extremely resource-optimized in usage since resources get used only as long as it takes for a function to run. Unlike legacy long-running processes, there is no requirement for servers to remain in operation all the time, which can translate into cost savings.

Serverless architectures are naturally event-driven. This implies that function execution is initiated by events, like an HTTP request, a file upload to cloud storage, or a database change. The system dynamically responds to such triggers by executing the corresponding function based on the event. This method facilitates live, scalable response to user interaction or system changes without constantly monitoring or manually responding to such interaction. Event-driven computing facilitates simple addition of flexibility with minimal overhead in the implementation of complex workflows and integrations.
Serverless platforms scale applications automatically as needed. When invoking a function, the cloud provider provisions adequate resources to execute the function. When there is high traffic or demand, additional instances of the function are created to serve the load. When there is low demand, resources are de-allocated to avoid any wastage of resources. This dynamic scaling model helps organizations achieve high performance and cost-effectiveness in the sense that they do not pay for any computing resources except when running.

Several major cloud providers offer serverless computing platforms, each of which has a different feature and capability:

- AWS Lambda is one of the best-known serverless platforms on which developers can execute code in response to a wide range of triggers, from data changes in Amazon S3 to HTTP requests via Amazon API Gateway or DynamoDB table updates. Developers pay only for compute time consumed in executing their functions on AWS Lambda, with events automatically scaling by frequency (Amazon Web Services, n.d.-a), (Amazon Web Services, n.d.-b).
- Similar to AWS Lambda, Google Cloud Functions is a serverless environment where code can be run by reacting to events such as file uploads, HTTP requests, or messages from other Google Cloud services like Pub/Sub. Google Cloud Functions shares tight integration with other Google Cloud services, and therefore it would be a viable choice for firms already established within the Google framework (Google, n.d.).
- Microsoft's Azure Functions is another serverless platform where developers can run code that executes as a consequence of events from Azure resources such as Blob Storage, Event Hubs, or HTTP requests. Azure Functions also provides multiple hosting plans for different workloads with both consumption-based and dedicated resource options (Microsoft, n.d.).

These platforms are designed to simplify application development and deployment by abstracting out infrastructure concerns so developers can focus on crafting code that responds to specific triggers in a scalable and cost-effective manner.

3 Traditional Cybersecurity and Its Limitations

Traditional cybersecurity models were designed with the idea of static, long-lived systems where resources are allocated to specific tasks for extended periods of time. In these settings, security controls are designed to protect against attacks on infrastructure components, such as servers, networks, and databases, that are intended to run all the time. Common elements of these types of systems include firewalls, intrusion detection systems (IDS), and antivirus software that are designed to protect well-defined, static environments.

For example, firewalls are configured to control incoming and outgoing traffic based on preconfigured rules, with the assumption that the network and the applications it supports are relatively static. Similarly, intrusion detection systems (IDS) monitor network traffic for signs of malicious behavior, based on the assumption that threats will manifest in detectable patterns and can be discovered through ongoing monitoring. This model aligns well with traditional, monolithic apps where the infrastructure below does not undergo dynamic change and risk can be addressed by layered security controls.

Risk in classical cybersecurity is based on a relatively static threat model. In this model, risks are contemplated based on a stable environment—always-on servers with known attack vectors and expected behavior. Security strategy is one of protecting these long-lived systems from attack that could exploit known weaknesses, such as unauthorized access, data theft, or denial-of-service attacks. Risk analysis puts a high priority on perimeter protection, endpoint security, and access control management.

In contrast, serverless computing is a dynamic environment where threats can quickly evolve. Serverless functions, as ephemeral and event-driven entities, don't conform to traditional models of perpetual operation. Instead of maintaining concerns over a running system, security analyses must look into how to secure code, data, and interactions triggered on specific events. This calls for a shift away from the static risk model to one that can keep pace with highly dynamic, unpredictable behavioral patterns that emerge with the instantaneous scaling and transient nature of serverless systems. Threats in this environment may arise from atypical sources or in unpredictable manners and therefore might be harder to assess and mitigate with traditional approaches.

The conventional security tools were built to address the specific risks associated with static environments, where the infrastructure, applications, and resources are relatively stable and predictable. Some of the most common tools include:

- Firewalls are one of the most fundamental security controls in traditional environments. Firewalls enforce network traffic policies, either allowing or blocking data packets based on IP addresses, protocols, and port numbers. While firewalls remain important in a serverless environment, they are not sufficient on their own because serverless functions are event-driven and do not require persistent inbound or outbound network connections.
- IDS and IPS try to detect and prevent malicious activity by inspecting network traffic for suspicious behavior or known attack signatures. These products utilize signature-based detection, anomaly detection, or heuristic analysis, and function well for established systems with predictable traffic flows. However, in serverless, where functions can have brief lifetimes and scale in unpredictable ways, these tools are not able to give visibility for ephemeral workloads or real-time threat detection (Abdulganiyu et al., 2023).
- Antivirus software existed to identify and remove threats in the form of viruses, malware, and other malicious code on long-lived systems. They are all focused primarily on detecting signatures of known malware and file system scanning for vulnerabilities. With the use of serverless architecture, focus is shifted from endpoint protection at an individual level to protection of the execution environment and ensuring that dynamically executed functions are not exposing vulnerabilities. Antivirus software is less effective because serverless functions do not always have a persistent filesystem and may live for the duration of the function call.
- Legacy systems rely on user authentication, authorization, and access control mechanisms (e.g., role-based access control or multi-factor authentication) to manage who is allowed to access resources. In serverless computing, although IAM remains core, dynamic scaling and the event-driven nature of serverless require more fine-grained and adaptive controls, e.g., fine-grained permissions that have the ability to secure the execution of individual functions or resources (Singh et al., 2023).

Traditional cybersecurity controls and practices have proven to be effective in safeguarding against threats in static, long-lived systems. However, the dynamic, short-lived nature of serverless computing creates significant gaps in these security models, since they fail to account for the quick, event-driven execution of functions. As serverless computing continues to evolve, the limitations of these traditional tools become increasingly apparent as new, dynamic approaches to cybersecurity are created that can more successfully protect these dynamic environments.

4 How Serverless Computing Redefines Risk

In traditional computing environments, applications run as long-lived, persistent processes that maintain state over time. Security controls for these environments are designed to monitor and protect these persistent systems from ongoing threats, usually through continuous scanning and endpoint protection. In serverless computing, however, the ephemeral function model is a serious threat. These short-lived, stateless functions are created, executed, and destroyed in rapid succession. A typical serverless function may only have a few milliseconds or seconds to live before it terminates. Because serverless functions are so short-lived, traditional security tools that work based on constant monitoring and inspection of long-lived processes are poorly equipped to respond to threats in real-time. An attacker could take advantage of a vulnerability in the short lifespan that a function is alive and evade traditional security systems that otherwise would be able to detect malicious behavior in a static system (Lynn et al., 2017).

Serverless functions have no lasting states or lengthy-lived data, which complicates it to detect an attacker's activity in the system. That means typical detection and logging facilities may not know if a serverless function is compromised or experiencing an attack. For example, if an attacker can inject dangerous code into a running ephemeral function, tracing out such an attack after the function has already ended becomes highly unlikely. Stateless ephemeral functions are typically stateless, i.e., they do not retain context of previous execution. While this design provides flexibility and scalability, it also limits the ability to trace or correlate attacks between different instances of a function. Since they do not have a "memory" of previous happenings, attackers at times are able to remain unnoticed between consecutive function calls. In order to safeguard transitory functions, security controls must be fast, agile, and operating within the brief execution windows such functions offer. These can include real-time threat analysis, function-level monitoring, and response capability that prevents security holes from being opened in such transient settings.

Serverless architecture is naturally event-driven, where functions are invoked by specific events, such as an HTTP request, a change to a cloud storage bucket, or a message in a queue. While this event-driven model introduces flexibility and scalability, it introduces new vulnerabilities as well. Since serverless functions execute in response to external triggers, attackers can likely utilize these triggers to launch attacks. For example, an attacker can form a malicious request or input specifically designed to exploit a bug in the code of the called function. Traditional security models that rely on continuous, static observation of systems may not take into account the dynamic nature of event-driven calls, and thus functions are exposed when they are called by malicious or malformed events.

In a serverless system, multiple events can trigger multiple functions that interact with each other. Without isolation, an attacker might exploit the interaction between

these functions. For instance, if a function writes data to a database and another reads data from the database, a malicious event can taint the data or establish unnecessary interactions between these functions. Event-oriented architectures make it more difficult to correlate between functions and identify malicious behavior. A series of what appears to be innocuous events could be leveraged by an attacker so that collectively, they may be damaging. These spread-out attacks then move beyond what traditional threat-detection systems, usually pattern-finding in nature within a particular universe, are designed to detect. Threat detection in a world of events requires sophisticated monitoring software that can watch events in real time, correlate them between functions, and recognize any abnormal or suspicious activity that deviates from the anticipated patterns of events.

Serverless computing platforms are designed to dynamically scale based on demand, allowing organizations to easily add or delete function instances based on traffic surges or declines. While this dynamic scaling capability is one of the most important benefits of serverless architecture, it also offers some security challenges. Because serverless functions automatically scale up and down based on event load, they can inadvertently bring new areas of attack surface. For instance, during traffic spikes, an increased number of serverless functions may lead to an increased number of entry points that attackers can take advantage of. Traditional security controls, such as firewalls or intrusion detection systems, may struggle to keep up with severely fluctuating environments and variable resources.

Demand-driven automated scaling can also lead to resource misuse. When an attacker sends a lot of requests, it may trigger the scaling activity, which may deluge the system and even lead to service downtime (denial of service). The scaling may also reveal public unnecessary or incorrectly configured services, which become possible points of entry for attackers. Classical security models rely on pre-set settings, static network addresses, and deterministic workloads. Dynamic scaling serverless environments, by contrast, create a context in which resources are constantly changing in amount, location, and configuration. It is difficult to enforce uniform security policies upon a changing environment.

To counter dynamic scaling risks, companies must deploy scalable, adaptable, and dynamic security controls alongside the application. Such controls could be real-time traffic monitoring, dynamic scaling triggers that are set through automation, and fine-grained security policy which can be adapted dynamically whenever the system is scaling. In a serverless environment, the shared responsibility model divides security duties between the customer and the cloud provider. While cloud providers manage the underlying infrastructure, such as the servers, networking, The security of the infrastructure is guaranteed by the cloud provider, including the servers, the data centers, and the serverless platform. The providers often include robust security controls, including encryption, IAM (Identity and Access Management), and access control, to allow customers to secure their functions and data. However, the security controls are contingent upon the customer's configurations and practices used for them to function.

While the infrastructure is kept secure by the cloud provider, the customer must keep their code, configurations, and application logic secure. This includes IAM role and permission management, event trigger configuration, function endpoint security, and encryption of sensitive data. Since serverless environments can scale automatically and involve event-driven interactions, customers must maintain strict control over the security of the functions themselves. The shared responsibility model is a double-edged sword. On the one hand, it encourages greater flexibility. On the other hand, it places great responsibility on organizations to configure and lock down their serverless environments properly. Misconfigurations such as too liberal access controls or inadequate logging leave organizations vulnerable to severe threats. To preclude these threats, organizations must create extensive security policies that set clearly the division of responsibility between the customer and the cloud provider. Regular security audits, security testing automation, and adherence to best practices are crucial to ensuring that the division of responsibility of security is being adequately fulfilled.

5 The Need for Agile Cybersecurity Strategies

As increasingly more organizations turn to serverless computing, the need for adaptive security models has never been more acute. Static environment-based security models, though, are not well-positioned to keep up with the dynamic and fleeting nature of serverless environments. Serverless functions are transient and event-driven, and thus there is a need for security to respond quickly to changing conditions. An adaptive security model is one that dynamically adapts with the system, continuously keeping an eye on the environment and adjusting security controls accordingly. With serverless architecture, the randomness and volume of workloads make it a necessity to have a more fluid security strategy. That is, a departure from traditional perimeter defense approaches and embracing continuous monitoring, rapid threat detection, and dynamic response capabilities. This type of response enables security controls to scale and evolve with the serverless application, reacting to danger in real time as opposed to static, preconfigured expectations. Adaptive security frameworks enable businesses to remain ahead of changing threats and new attack surfaces brought about by the serverless paradigm. Automation is at the core of serverless environment security due to the speed and volume at which serverless functions operate. In a traditional setup, manual intervention to patch vulnerabilities, observe system health, and neutralize threats may be effective. However, the transient lifecycle of serverless functions—coupled with dynamic scaling—makes it unfeasible to implement manual security components. Security automation provides the tools to detect, respond to, and neutralize threats in real-time.

Automation can be used in various key domains of serverless security:

- Automated solutions can scan for vulnerabilities on a continuous basis, monitor for suspicious activity, and identify possible threats. In event-driven architectures, security solutions can trigger responses as soon as suspicious activity is detected, preventing further damage.
- Automated incident response mechanisms can quickly quarantine or disable impacted functions, reducing response time and minimizing damage risk. For example, in case of malicious payload, automated security controls can halt further execution of the impacted function or roll back to a known good state.
- Automation also guarantees serverless applications conform to security best practices and regulatory requirements for compliance. Automated security audits can run continuously on code deployments, configurations, and access policies so that any divergence from compliance standards is detected and fixed in real-time.

Automating such critical security procedures allows organizations to keep a high level of protection in a highly dynamic environment while minimizing human error and operational overhead.

Two modern security models—Zero Trust and micro-segmentation—are particularly relevant in the serverless environment. Both are designed to address the challenges posed by dynamic, distributed environments, where traditional perimeter-based security falls short.

- Zero Trust security architecture is built on the "never trust, always verify" tenet. Under a Zero Trust architecture, no internal or external entity can ever be trusted implicitly. All access requests, internal to the organization's network or external, must be authenticated, authorized, and validated at all times. This is a perfect approach for serverless architectures, with functions called by many different sources and not bound to a network or resource pool (Rose et al., 2020).
- Zero Trust in the serverless case means every function being individually authenticated and authorized before execution. This can be achieved by imposing robust identity and access management (IAM) controls that restrict only the legitimate code to execute, regardless of where it is invoked. With the use of Zero Trust, organizations can contain the blast radius of any compromise such that even if a single function has been compromised, the attack cannot spread throughout the system (Rose et al., 2020).
- Micro-segmentation is segmenting the network and resources into isolated, small segments where security policies are applied at a granular level. In serverless architecture, it implies that each function or service is an isolated entity with its own security policies. Micro-segmentation minimizes the lateral movement within the application, where if an attacker breaks into one function, it is difficult to reach the rest of the application (Sheikh et al., 2021).

- Micro-segmentation in the serverless model ensures that security controls are being applied at each individual function rather than relying on the perimeter defenses. It provides more fine-grained access control and mitigates the effect that could be caused by an attack (Sheikh et al., 2021).

In adaptive environments like serverless environments, threat intelligence has a leading position in anticipatory security. Threat intelligence includes the gathering of information, its analysis, and use regarding potential threats and weaknesses. It can be used to predict and divert attacks before the resulting damage is inflicted. As serverless computing evolves quickly and the threat landscape evolves continuously, organizations must refresh their threat intelligence constantly in a bid to outsmart attackers.

With more sophisticated cyberattacks targeting serverless infrastructure, predictive risk management is not optional. Integrating threat intelligence into serverless security operations helps organizations identify patterns of behavior that can indicate an impending breach, even before one happens. Threat intelligence feeds can, for instance, provide real-time intelligence about changing attack vectors, which can help security systems automatically update detection mechanisms and adapt to changing threats.

Moreover, incorporating threat intelligence into security automation platforms allows for faster response times. For example, when a vulnerability is discovered in a popular serverless framework, security systems may automatically patch or reconfigure access controls ahead of time to prevent exploitation before the attack has registered its impact.

Conclusion

As serverless computing continues to transform application development and deployment, there is a need to acknowledge the unique cybersecurity risks that accompany this revolution. The event-driven, ephemeral nature of serverless functions in conjunction with auto-scaling and event-driven architectures defies the traditional security paradigms, which were designed for static, long-lived systems. The shared responsibility model only compounds this complexity, dividing security responsibility between the cloud providers and organizations, both having their respective roles to play.

To address these challenges, organizations must adopt agile and adaptive security strategies that can keep up with the rapidly changing serverless environments. This includes leveraging automation for threat detection and response, embracing new security models like Zero Trust and micro-segmentation, and integrating real-time threat intelligence for proactive risk management. It is only by knowing the particular vulnerabilities of serverless computing and taking appropriate, adaptive security steps that organizations can successfully protect their applications and data in this new environment.

Finally, serverless computing has huge benefits of flexibility, scalability, and cost-effectiveness. But at the cost of having to redesign and rethink cybersecurity processes to take into account the requirements of this rapidly changing and fast-paced technology. As increasing numbers of companies adopt serverless models, it is important that cybersecurity approaches move in tandem so security is never an afterthought, but embedded in serverless computing.

References

[1] Abdulganiyu, O. H., Ait Tchakoucht, T., & Saheed, Y. K. (2023). A systematic literature review for network intrusion detection system (IDS). International Journal of Information Security, 22(5), 1125–1162. https://doi.org/10.1007/s10207-023-00682-2

[2] Amazon Web Services. (n.d.-a). AWS Lambda Documentation. Retrieved November 11, 2024, from https://docs.aws.amazon.com/lambda/?icmpid=docs_homepage_featuredsvcs

[3] Amazon Web Services. (n.d.-b). What is AWS Lambda? - AWS Lambda. Retrieved December 23, 2024, from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[4] Google. (n.d.). Cloud Run functions documentation | Cloud Run functions Documentation. Google Cloud. Retrieved November 11, 2024, from https://cloud.google.com/functions/docs

[5] Lynn, T., Rosati, P., Lejeune, A., & Emeakaroha, V. (2017). A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms. 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 162–169. https://doi.org/10.1109/CloudCom.2017.15

[6] Microsoft. (n.d.). Azure Functions – Serverless Functions in Computing | Microsoft Azure. Retrieved November 11, 2024, from https://learn.microsoft.com/en-us/azure/azure-functions/

[7] Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero Trust Architecture. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

[8] Sheikh, N., Pawar, M., & Lawrence, V. (2021). Zero trust using Network Micro Segmentation. IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 1–6. https://doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484645

[9] Singh, C., Thakkar, R., & Warraich, J. (2023). IAM Identity Access Management—Importance in Maintaining Security Systems within Organizations. European Journal of Engineering and Technology Research, 8(4), Article 4. https://doi.org/10.24018/ejeng.2023.8.4.3074