451 Research® | PATHFINDER REPORT

Securing Serverless Compute

COMMISSIONED BY

threat stack

NOVEMBER 2019

About this paper

A Pathfinder paper navigates decision-makers through the issues surrounding a specific technology or business case, explores the business value of adoption, and recommends the range of considerations and concrete next steps in the decision-making process.

ABOUT THE AUTHOR

FERNANDO MONTENEGRO

PRINCIPAL ANALYST, INFORMATION SECURITY

Fernando is a Principal Analyst on the Information Security team, based in Toronto. He has broad experience in security architecture, particularly network security for enterprise environments. He currently focuses on covering vendors and industry events in the endpoint security and cloud security spaces. Prior to joining 451 Research, Fernando worked in pre-sales and delivery roles with vArmour, RSA, SilverTail, Crossbeam and Hewlett-Packard. His areas of interest include security economics (particularly behavior economics), data science and network security. Fernando holds a BSc. in Computer Science and several industry certifications.

Executive Summary

To understand serverless security, it is important to frame it within the broader context of overall technology transformation. As many organizations modernize their IT approaches under the umbrella term of 'digital transformation,' there's innovation taking place across multiple dimensions. The types of workloads being modernized now range from analytics to customer experience and beyond, with increasing demand for faster time to value. The execution venue for the workload changes as part of that demand, from on-premises traditional IT to a multitude of cloud-based offerings. Lastly, newer options for compute bring container-based execution and 'serverless' function execution into the fold. All of this takes place as organizations also adapt their application delivery practices, broadly moving from more traditional 'waterfall' delivery into Agile models and, increasingly, doing so with a DevOps approach to team structure.

Serverless compute is not a wholesale replacement for all types of applications, and the majority of organizations will continue to utilize a hybrid environment that includes both serverless and container technology for the foreseeable future. Alongside its business benefits, serverless compute has the potential for security benefits as well. Compared with traditional compute options, the 'attack surface' that can be exploited in serverless is severely reduced both in scope and in time: only a small fraction of code is available, and the function execution itself is often limited to seconds or minutes.

This is not to say that organizations get to relinquish their security responsibility with serverless. There is still a multitude of security topics to be addressed – from risk assessments and threat modeling to application security practices, environment configuration, data handling practices, regulatory requirements, and business-level anti-fraud monitoring. These topics need to be addressed in a manner that preserves the business benefits of serverless but also accounts for the changes to the threat environment itself: highly automated attacks that may not exploit infrastructure flaws, but configuration and design issues instead.

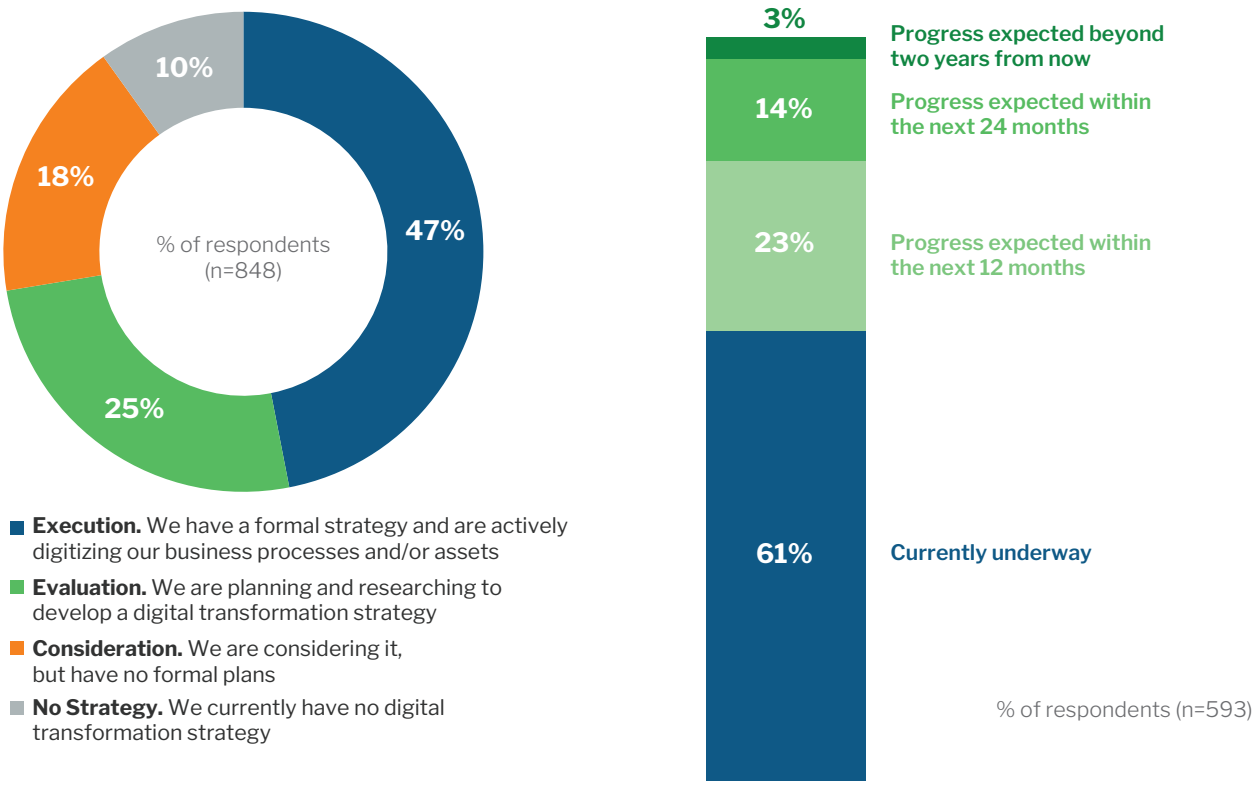
Organizations are urged to consider their broad capabilities – people, processes and technology – in light of demands from serverless security use cases and adapt accordingly.

Introduction

The mission of securing serverless compute begins with a need to clearly define it, and to understand where it sits within the broader technology landscape. To do so, one should understand that most organizations – regardless of size – are dealing with 'digital transformation' initiatives: according to 451 Research Voice of the Enterprise data, 72% of survey respondents indicated that a digital transformation initiative is being executed or evaluated, and 84% of those say that such an initiative is either under way or about to begin in the next 12 months.

Figure 1: Status of Digital Transformation

Source: 451 Research's Voice of the Enterprise: Digital Pulse, Workloads & Key Projects 2019



Within this scenario, enterprises are quickly refreshing their IT infrastructure along three dimensions: how IT projects are designed and delivered, with a strong focus on Agile and now DevOps; where the workloads are deployed, with a variety of environments including on-premises traditional IT, on-premises private clouds and variations of public clouds services and providers; and finally what the technology footprint itself is, with virtual machines giving way to containers and, increasingly, serverless compute, which is also referred to as 'event-driven computing' or 'functions as a service.'

This is the definition to be used in this report: serverless compute "is an abstracted model of cloud computing based on the execution of code and compute jobs, rather than server-based." While most cloud providers can now offer the delivery of services such as databases on demand, message queues and more – all of which require no server configuration – we restrict the analysis here to arbitrary code execution triggered by a variety of events.

Serverless compute offers organizations the potential to rethink several aspects of IT delivery, including changing long-running monolithic processes into event-driven disaggregated functions that can be developed and deployed independently. Depending on the type of workload and use case, this can significantly simplify and accelerate development, with the added benefits of more transparent costs and less infrastructure to manage.

For infrastructure teams, there is a now a continuum of options all the way from bare-metal servers through virtual machines, various options for containerized workloads, and now serverless compute.

Serverless Compute or Containers?

Serverless compute options include offerings from major cloud providers: AWS Lambda, Google Functions and Google Cloud Run, Azure Functions, IBM Cloud Functions, Alibaba Function Compute, Cloudflare Workers and others. In these scenarios, the providers create and manage the execution environment themselves, and clients only need to worry about their function package and responding to events.

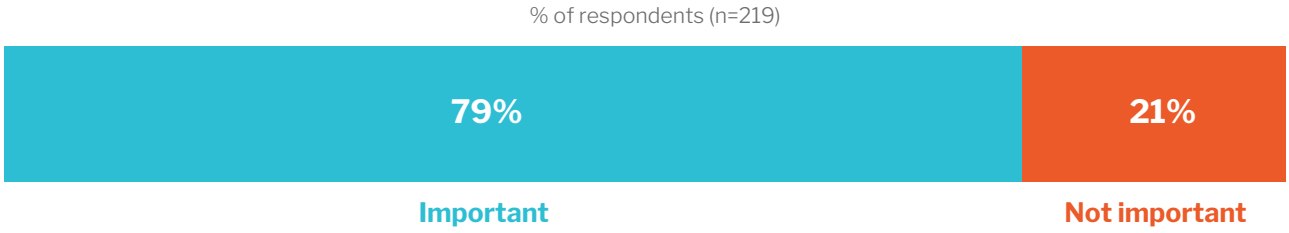
Open-source frameworks typically abstract away underlying infrastructure – usually Kubernetes – to provide serverless execution environments. Options for open-source frameworks include Apache OpenWhisk, Kubeless, OpenFaaS and others. The Knative framework, created by Google and Pivotal, appears to have momentum.

The typical execution profile for a serverless compute function is that the code will be loaded and executed in an isolated environment for the duration of the function – anything from a few milliseconds up to a few minutes – and then terminated. There is no state kept within the function, and the developer typically only needs to be concerned with the packaging of the dependencies and code: the provider is responsible for maintaining the actual language runtime and surrounding environment. There's significant variation in serverless offerings, as some support specific language runtimes, others offer a hybrid environment to execute functions packaged inside a container, and more.

For many organizations, containers are a much more popular technology. One of the factors that have so far tipped the scales toward container-based deployments has been standardization. As can be seen from the survey results in Figure 2, organizations responded overwhelmingly that portability of their code is key.

Figure 2: Importance of Portability for In-House Cloud-Enabled/Cloud-Native Software

Source: 451 Research's Voice of the Enterprise: Cloud, Hosting & Managed Services, Workloads and Key Projects 2019



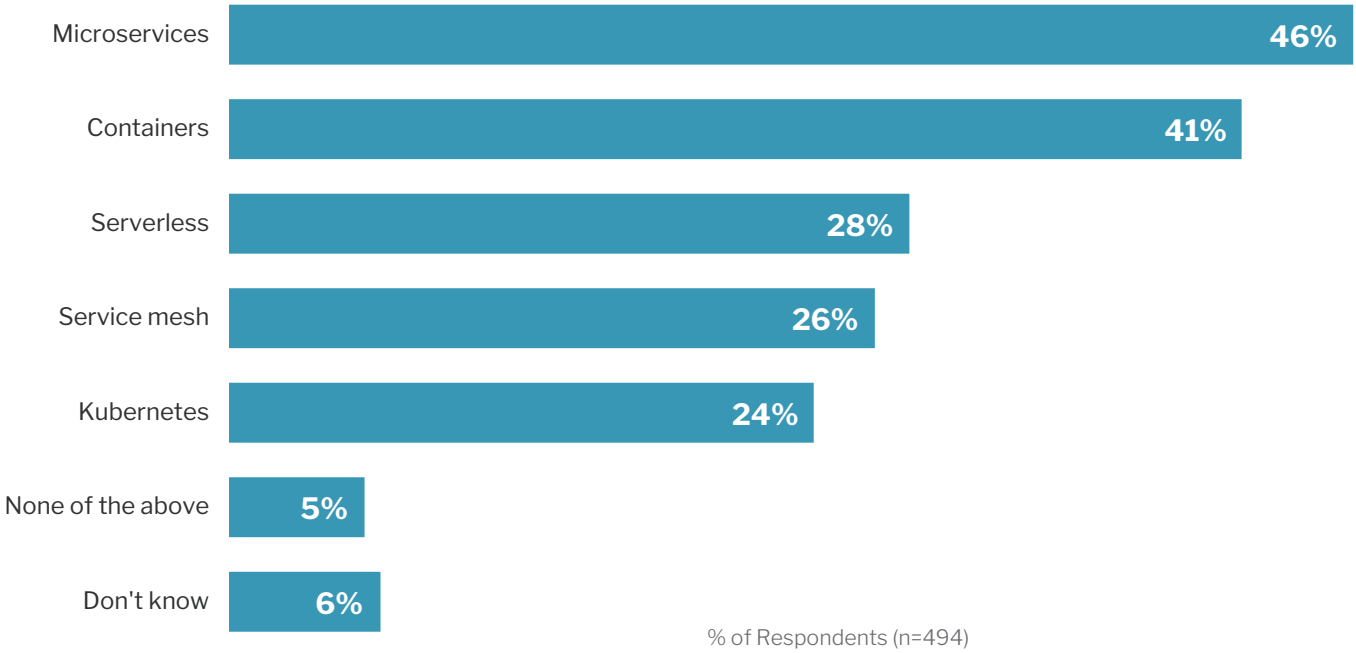
Containers have benefited from this demand, since the image specification and runtime specification have long been standardized, leading to significant portability for workloads and a raft of execution environments. Serverless compute, however, has not had a similar standardization offering. This has given rise to several independent 'serverless frameworks,' each aiming to abstract away details while offering consistency between key providers.

Adoption Trends for Serverless Compute

Looking into responses from the 451 Research DevOps survey, serverless is indeed one of the up-and-coming technologies. It is interesting to note that respondents indicated that serverless is the next most important technology after microservices – a well-known design pattern – and containers, the key technology for modern applications over the past few years.

Figure 3: Important Cloud-Native Technologies and Methodologies

Source: 451 Research's Voice of the Enterprise: Q1 2019 VotE DevOps survey



This by itself is interesting, but when looking at different correlations on this question based on attributes such as respondent title and experience with DevOps, an interesting pattern arises on the importance of serverless. In Figure 4, the percentages correspond to those that consider serverless important for their deployments among two categories of respondents.

Figure 4: Importance of Serverless by Role and Experience

Source: 451 Research

TRAIT	CATEGORY1	PERCENTAGE	CATEGORY 2	PERCENTAGE
Respondent Title	Technical Staff	34%	Non-Technical (usually management)	25%
Digital Transformation	Leaders	31%	Learners and Laggards	28% and 18%
Operational Involvement	Directly involved	33%	Others	20%
Level of DevOps adoption	Full	33%	Some Adoption	24%
DevOps Experience	More than 2 years	31%	0-2 years	23%

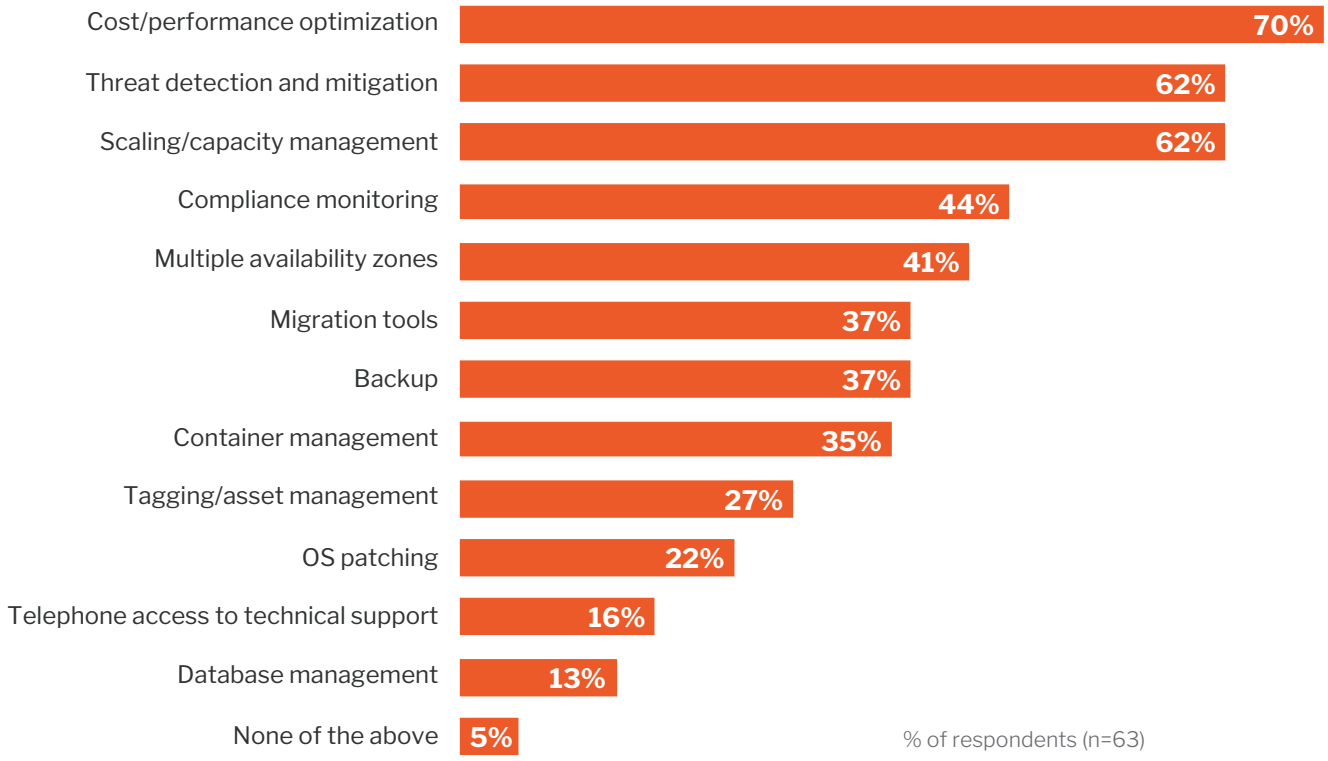
In short, it appears that more technical teams and those further along in their digital transformation and DevOps journeys have a deeper appreciation for the importance of serverless in modern deployments.

Use Cases for Serverless Compute

A 451 Research survey on broader cloud adoption highlighted areas where organizations feel that additional capabilities from service providers would yield better outcomes on cloud projects.

Figure 5: Additional Services Necessary for Public-Cloud Application Deployment

Source: 451 Research's Voice of the Enterprise: Cloud, Hosting & Managed Services, Organizational Dynamics 2019



The top three areas – cost/performance optimization, threat detection/mitigation and scaling management – are precisely areas where an organization may yield benefits from considering serverless compute concepts. For cost/performance optimizations, serverless compute is billed on a per-use basis and allows customizing other aspects of function execution such as memory size and maximum runtime for a specific function. Depending on the type of workload, this ties directly to optimization goals.

Security incident response is another area where serverless compute can provide much faster response to incidents: rather than wait for a human incident response team, the environment can be configured to automatically correct deviations or respond to specific threats via custom code written as serverless functions. Lastly, serverless compute offers elasticity and capacity management, with minimal intervention by operators.

Serverless compute is not a wholesale replacement for all types of applications. In broad terms, the key areas for serverless compute revolve around supporting event-driven applications. Applications that are developed with microservices architectures can also benefit. The types of events that each platform supports will vary, but generally include:

- React to a change in an underlying object – new record added to database, new object made available in a storage area.
- Invoked via a direct call from a remote client or via an API gateway.
- Scheduled events – serverless functions can, in many cases, replace regular 'cron' jobs, for example.

Taking these capabilities together, serverless compute can then be used to perform a variety of complex IT-centric tasks such as reconfiguring environments in response to system behavior, updating database records, dispatching and process requests from a message queue, performing transformation on objects, and implementing conversational interfaces (chatbots).

Within the context of a cloud provider, serverless compute is also often used as 'glue' code between the variety of services. These events can then be aggregated into higher-order business goals such as extract, transform, load (ETL) pipelines and processing for data analytics/AI, or processing events from large numbers of endpoints such as low-power IoT devices.

Securing Serverless Compute

With an understanding of how serverless compute is used and where it fits, comes the question of how to secure it. There is currently a significant gap between where typical security practices are and where they need to be regarding seamless integration with modern DevOps delivery – used here as an imperfect but valid proxy for serverless deployments. Figures 6 and 7 – from the 451 Research DevOps survey – highlight two findings that should give security teams some pause.

Figure 6 highlights how using 'vulnerability assessment' towers over other security practices. This leads us to consider that development is still happening independent of security, and that only at the end of the pipeline is a product scanned for security issues. This was typical of previous delivery methodologies such as waterfall, but is not suitable for DevOps processes.

Figure 6: Security Elements Critical to DevOps Workflows

Source: 451 Research's Voice of the Enterprise: Q1 2019 VotE DevOps survey

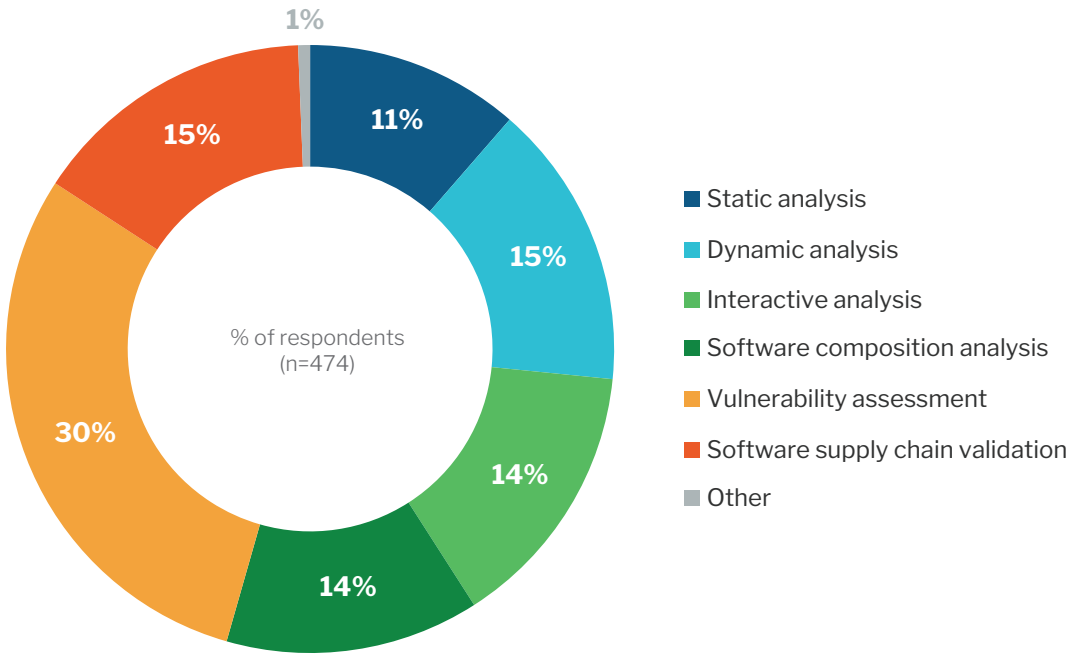
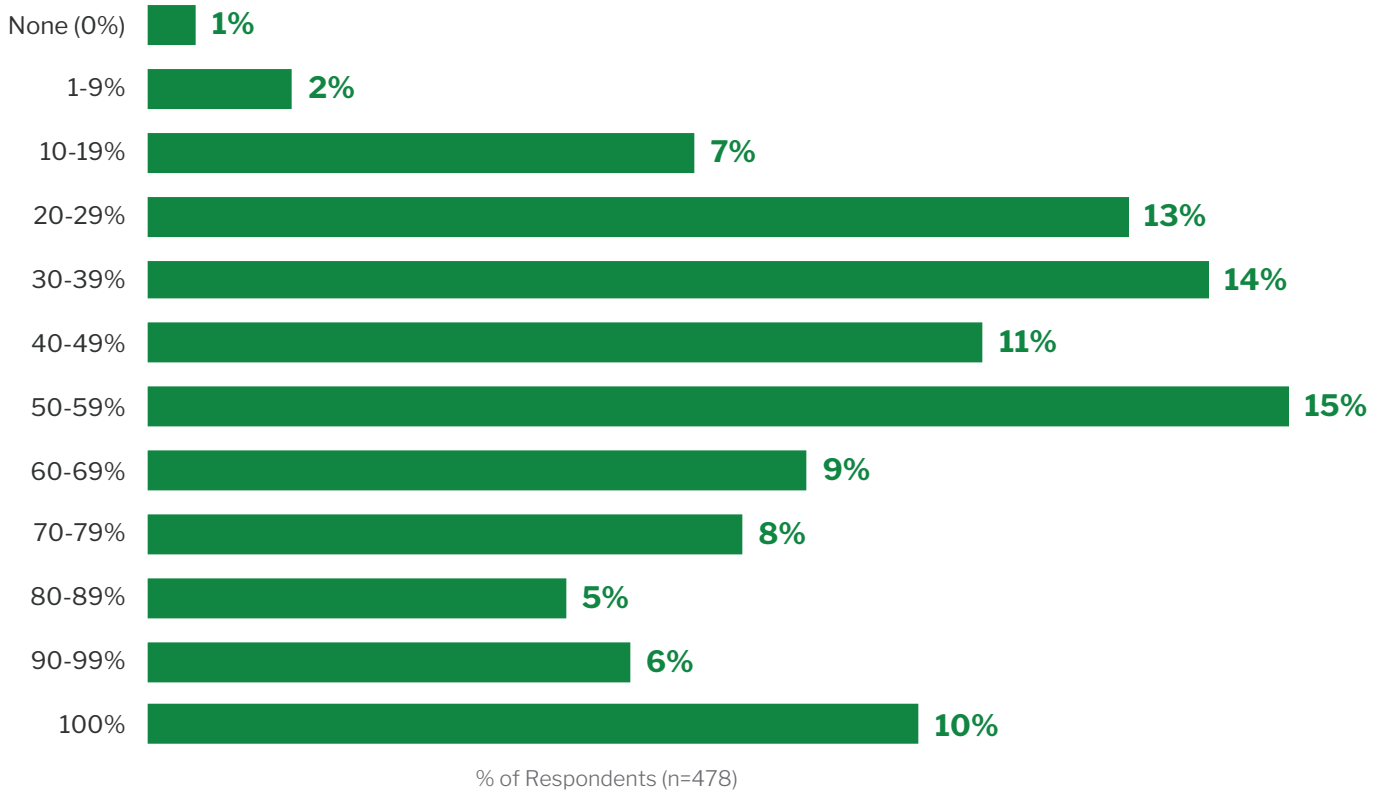


Figure 7 highlights another troubling finding: a significant number of respondents indicate that only a fraction of their DevOps deployments include security elements built-in. This means that development teams are moving ahead, often without waiting for security to catch up. This should be reviewed.

Figure 7: Percentage of DevOps Implementations With Security Elements

Source: 451 Research's Voice of the Enterprise: Q1 2019 VotE DevOps survey



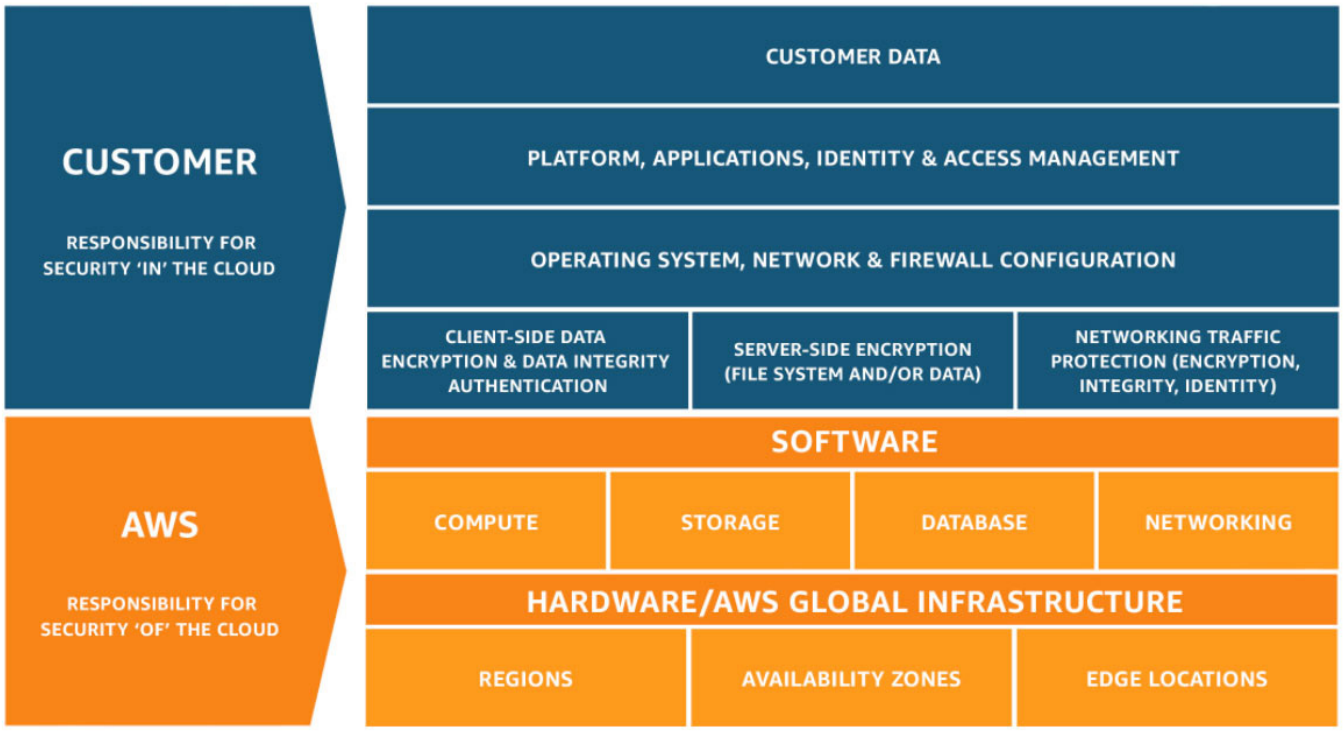
If security teams are to embrace securing serverless compute, having tighter synergy with DevOps teams (again, used as a proxy for serverless in this study), is mandatory. This synergy must include collaborating on how to translate security requirements into actionable directions for development teams, and potentially realigning some responsibilities between teams.

Once that internal hurdle has been addressed, the next step is to understand the scope of responsibilities with the cloud service provider. As organizations deploy their workloads – bare metal, virtual machines, containers or serverless – to the cloud, they do so under the scope of what's being referred to as the shared responsibility model (SRM). Each provider will have slightly different versions, but it's a basic understanding that the responsibility for implementing adequate security controls is divided as follows:

- The cloud provider is responsible for providing security for the layers used to implement the services contracted by the client.
- The client is responsible for the security of how those services are configured and used.

AWS's cloud responsibility model is shown in Figure 8.

Figure 8: AWS Cloud Responsibility Model
Source: AWS



Securing serverless compute then becomes an exercise in securing the various layers that do fall under the user's responsibility:

- The overall design must include proper security planning. This includes performing the necessary risk assessments and threat modeling exercises to understand what attack vectors must be considered and what controls will need to be applied.
- The execution environment must be secured. This is the key differentiation that a provider-based execution environment for serverless compute offers. All hyperscale cloud providers have taken on the responsibility for securing this execution environment, including enforcing segmentation, implementing least privilege and generating telemetry that can be used for monitoring. Naturally, if the serverless compute environment is based on an on-premises environment – or is even deployed as a service on top of a Kubernetes cluster instantiated by the organization – some of these responsibilities shift back to the teams responsible for those environments. Most of the security incidents associated with container and Kubernetes environments, for example, stem from misconfigurations of the environment.

- When coding the function, typical application security controls apply:
  - Validate the security of third-party dependencies – libraries, modules and function layers. With the increase in software supply-chain attacks, it is critical to ensure that there's as little a chance for underlying issues in external code as possible. Typical checks include validations against multiple vulnerability databases, license checks and more.
  - When coding, perform adequate input sanitization. This is critical when handling user-supplied input, including but not limited to strings, files, binary objects, messages from a queue and more.
  - Do not hard-code secrets, credentials or any identifiers that lock the code to any environment (dev, test, production). Rather, parameterize the necessary code so it can be integrated with secrets management tooling during deployment.
  - Generate meaningful function execution logs, but be mindful of not sending these logs to function output – send them to dedicated log collection functionality instead – and do not log sensitive data such as passwords, secrets or privacy-protected information.

For more information on application security practices for serverless compute, please refer to the Open Web Application Security Project (OWASP) initiative, particularly the 'OWASP Top 10' for generic security issues and the 'Serverless Top 10' for serverless-specific issues.

In terms of specific technical controls that apply to securing serverless, the list includes but is not limited to:

- Limit effective function permissions and actions: while this may be challenging to define in a rapid development pace, it is critical to ensure that functions have implemented as much of a 'least privilege' approach as possible. Telltale signs of misconfiguration include using wildcards ("*") when defining entities, permissions, actions or resources available to a function. Other issues include not restricting function invocation from selected environments (network segments, source hosts and others).
- Validate execution requests. As functions are triggered by a variety of events – including HTTP, Webhooks, changes to underlying assets and more – developers should carefully validate via authentication and authorization checks that the invocation is valid. Should an attacker exploit a flaw in the underlying event generation, they may be able to create unauthorized executions, which could have consequences in terms of data security or costs.
- Consider implementing runtime application protection capabilities within the application code – usually invoked with a couple of function calls – to add security capabilities such as whitelisting, injection protection and others. In some cases, beware of the fact that protection modules may increase function resource requirements.
- Manage credentials and secrets securely. Should functions require access to specific credentials or configuration parameters during execution, these should be provided securely via secure storage. These can include provider-offered services (all major cloud providers offer variations of secrets and configuration vaults) as well as independent third-party vendors.
- Other data security considerations – data governance issues, encrypting data at rest and in transit, privileged access management, data leakage protection and others – still apply for serverless compute.

- Finally, the execution of the functions generates significant telemetry that should be monitored for anomalies, at multiples layers of abstraction. This type of monitoring is needed to detect, for example, suspicious behavior from a function or potential denial-of-service attacks aimed at consuming resources, as well as business-level fraudulent transactions.

Conclusions and Recommendations

Serverless compute options have grown in popularity, and continue to find adoption in ever more use cases. It is expected that container technology will continue to be popular, but serverless workloads should grow quickly, particularly in environments where an event-driven microservices approach may be applicable.

Securing serverless compute environments means abstracting away some areas of concern – there's no patching of runtimes, for example – but there is a more complex network of functions and events that needs to be secured. Having a clear understanding of the shared responsibility model for security is essential.

Security teams looking to provide adequate protection for serverless compute deployments should consider performing adequate threat modeling and security architecture design, choosing the controls that align with their specific needs.

Ensure that security controls are applied across the environment, paying attention to effective permissions assigned to functions, and ensure that the event-generating services are properly secured.

Work to include adequate lifecycle security controls on the code used for the functions themselves. This includes leveraging proper application security mechanisms and practices, as well as providing the protection against malicious user input.

About 451 Research

451 Research is a leading information technology research and advisory company focusing on technology innovation and market disruption. More than 100 analysts and consultants provide essential insight to more than 1,000 client organizations globally through a combination of syndicated research and data, advisory and go-to-market services, and live events. Founded in 2000 and headquartered in New York, 451 Research is a division of The 451 Group.

NEW YORK
Chrysler Building
405 Lexington Avenue,
9th Floor
New York, NY 10174
+1 212 505 3030

SAN FRANCISCO
505 Montgomery Street,
Suite 1052
San Francisco, CA 94111
+1 212 505 3030

LONDON
Paxton House
30, Artillery Lane
London, E1 7LS, UK
+44 (0) 203 929 5700

BOSTON
75-101 Federal Street
Boston, MA 02110
+1 617 598 7200

451 Research®