This is a Title

Richard Paul
rdpaul@umass.edu
University of Massachusetts Amherst

April 4, 2024

1 Introduction

As cloud computing evolves, serverless architectures have emerged as a paradigm shift, allowing developers to build and deploy applications without the costly overhead of managing servers. This allows developers to abstract away the infrastructure, focusing on code and functionality. However, this introduces new challenges, including in the realm of security [1]. One important concern is dependency management. Dependencies, or external libraries and packages for an application, can contain vulnerabilities that pose security risks [2]. Recognizing this issue, some serverless cloud providers have begun to integrate automatic dependency vulnerability checking and updating. This paper aims to scrutinize the benefits and drawbacks of those features, to provide a detailed analysis of their impact on serverless computing.

The introduction of automatic dependency vulnerability checking and updating is seen as a benefit for the security of serverless applications. By incessantly scanning for and remedying known vulnerabilities within dependencies, cloud providers can curb the likelihood of security breaches [3]. This preventive measure can bolsters application security and alleviate developers from the cumbersome task of manually tracking and updating their dependencies. As Buckholz notes, this can align with the philosophy of serverless development by allowing developers to focus more on innovation than maintenance [4].

However, this innovation has it's limitations. The automatic alteration of dependencies can lead to unintended changes or incompatibilities, which will potentially compromise application performance [5]. Additionally, the lack of transparency in automatic dependency checking could result in a diminished understand and oversight of faults in the application. It may also force developers to rely upon a certain cloud provider to maintain the dependencies, leading to vendor lock-in [6]. Furthermore, the thoroughness and reliability of automated vulnerability checks is still in question, and may lead some developers to believe their application is protected when it still contains security vulnerabilities.

2 Background

Serverless computing has dramatically changed the landscape of cloud services, offering a new paradigm for application deployment and management. Serverless computing allows code to execute in response to events without the need for provisioning or managing individual servers. This abstracts away the infrastructure layer, significantly simplifying the administrative responsibilities upon application developers [7]. This model, predominantly offered through Functions-as-a-Service (FaaS) platforms like AWS Lambda, Azure Functions, and Google Cloud Functions, allows developer to focus on code that contributes directly to their application, rather than on the underlying operational complexities [8].

A critical aspect of developing in a serverless environment is the management of dependencies, external libraries or packages that an application uses to perform functions. Dependencies can include many things from frameworks for web applications to libraries for accessing cloud services. Dependency management is not a trivial task, they must be kept up-to-date to incorporate bug fixes, new features, and, most importantly, security updates [5]. The National Vulnerability Database, a U.S. government repository of vulnerability management data, has shown an increasing amount of vulnerabilities among common dependencies, underscoring the importance of well thought out dependency management [].

This has led to rise in automatic dependency checking and updating software to help manage these dependency issues. Tools like Snyk, Dependabot, and Renovate can automate the process of detecting vulnerabilities in a projects dependencies and automatically update those dependencies to a new secure version. This is vital in a serverless environment where the agility and scalability of application can be compromised by vulnerable dependencies [].

However, the implementation of automatic dependency management in serverless computing still has significant challenges. The automated update process must be carefully managed to ensure that updates do not introduce changes that break the application. In addition, reliance on automated tools can lead to a decreased understanding of dependency management among developers, potentially leading to complacency and increased security risks when automated systems fail [].

The security risk in dependencies primarily come from two distinct factors: the external nature of the dependencies and the complex and opaque dependency trees that they can form. A single application may directly depend on hundreds of external libraries, which may themselves depend on thousands more. This complexity significantly increases the attack surface of the applications, as each external dependency is a point of failure for attackers []. For example, there was a notable incident where a popular NPM package, event-stream, was compromised to steal cryptocurrency for the attackers, showcasing the cascading risk a single vulnerable dependency poses to thousands of different applications [].

Moreover, the ephemeral and stateless nature of serverless function can signficantly increase these risks. Since serverless applications scale dynamically with the resources demanded from them, a vulnerability in a single dependency can rapidly propagate across multiple instances, amplifying the potential impact of any security brach [].

To counteract these risks, there exists several notable mitigation strategies and best practices that have been recommended and adopted:

- Implementing regular vulnerability scanning and dependency review. Using tools like Snyk and OWASP Dependency-Check can provide automated vulnerability scanning and are important to ensure a robust security management.
- The 'Shift-Left' security approach, which argues that security needs to be the a focus from the start of development, or else you will end up chasing your tail after the vulnerabilities that will almost certainly exist.
- Principle of Least Privilege (PoLP) is a foundational security principal to ensure that programs only have access to the permissions they need. This can be implement with dependencies to help box them in and not give them undue access.
- Use versions of dependencies that have been thoroughly vetted. This can avoid new vulnerabilities that could come up in new versions, but may delay the ability for your application to use new features/ get updates to old unseen vulnerabilities.
- Developer education has been cited as a very important part of dependency management. Ultimately, the buck stops with the development team and it's vital to insure all developer are cognizant of the risks involved with using dependencies, and best practices for doing so.

References

[1] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, "Serverless computing: Current trends and open problems," 2017.

[2] E. Marin, D. Perino, and R. Di Pietro, "Serverless computing: a security perspective," *Journal of Cloud Computing*, vol. 11, p. 69, 2022. [Online]. Available: https://doi.org/10.1186/s13677-022-00347-w

[3] Snyk, "2023 open security report," https://go.snyk.io/state-of-open-source-security-report-2023-dwn-typ.html, 2023, accessed: 2024-04-02.

[4] G. Buckholz, "The pros and cons of a serverless devops solution," *AgileConnection*, April 2018, accessed: 2024-04-02. [Online]. Available: https://www.agileconnection.com/article/pros-and-cons-serverless-devops-solution

[5] G. Benischke, "The case against automatic dependency updates," https://beny23.github.io/posts/automatic_dependency_updates/, 2023, accessed: 2024-04-02.

[6] M. J. Kavis, *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and Iaas)*. Wiley, January 2014.

[7] M. Roberts and J. Chapin, *Programming AWS Lambda: Build and Deploy Serverless Applications with Java*, 1st ed. O'Reilly Media, April 2020.

[8] M. Villamizar, O. Garcés, H. Castro, M. Verano Merino, L. Salamanca, R. Casallas, and S. Gil, "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," 10 2015.