TRENDSIC
CASE STUDY

Enhancing Software Security
Through Advanced Scanning Techniques

EXECUTIVE SUMMARY

Trendsic Corporation, an industry leader in Software Engineering, Project Management, and IT Architecture Consulting, recently carried out an extensive project aimed at bolstering software security for one of our clients in the digital health sector. Recognizing the critical need for a more thorough and advanced approach to identifying software vulnerabilities, we transitioned a client from solely employing Project-Level Code Scanning to incorporating Container-Level Scanning and also scanning for often ignored Transitive Dependencies as well. This case study outlines the challenges faced, mitigation strategies employed, and the significant reductions in software vulnerabilities achieved as a result of this initiative. Not only that but we also worked closely with our clients security team to implement this strategy as policy to help enforce good security practice across the company. This project exemplifies Trendsic Corporation's commitment to delivering cutting-edge, secure software solutions to our clients.

OBJECTIVES

- ***Identify and resolve dependencies in source code and project solutions.***
- ***Differentiate between Project-Level and Container-Level Scanning.***
- ***Implement high-level mitigation steps for discovered vulnerabilities.***
- ***Implement solutions into Continuous Integration pipelines to mitigate known vulnerabilities from ever entering lower environments.***
- ***Implement solutions into daily container scanning pipelines to mitigate vulnerabilities in production deployments as they are discovered.***

Previous Implementation: Project-Level Scanning

High-Level Overview:

- **Top-Level Dependencies:** Dependencies that are declared by project maintainers within project files.
- **Build Step and "Containerization":** Once the build was successful, the code would be packaged into a container.
- **Ships to Production Environments:** The container would be shipped to various production environments without further security checks.

Challenges:

- Lacked visibility into transitive dependencies.
- Lacked coverage for container packages and libraries, which often run with higher privileges and are more susceptible to vulnerabilities.

New Implementation: Project-Level and Container-Level Scanning

Project-Level Scanning:

- **Top-Level Deps:** Continued scanning of top-level project dependencies.
- **Licensing Obligations:** Incorporated scanning for licensing compliance.

Build Step and "Containerization":

- After successful scanning, the project is built and containerized.

Container Ships to Production Environments:

- The newly built container is shipped to production environments only after passing the Container-Level Scanning.

Container-Level Scanning:

- **Top-Level Dependencies:** Dependencies that are declared by project maintainers within project files.
- **Transitive Dependencies:** Dependencies that are declared by top-level dependencies. As the case with .NET, these won't appear in your project until the application is published and these often go overlooked. Scanning has been expanded to cover the full manifest of dependencies.
- **Container Packages:** Containers are basically snapshots of an operating system's user space. They contain software packages and libraries that are unrelated to your application, but sit alongside your app and can provide a vector of attack. Scanning has been expanded to cover container libraries and packages.

Results of Container Scanning Before Mitigation

- **4 Vulnerabilities** discovered in transitive dependencies.
- **15 Vulnerabilities** discovered in container libraries code.

How do we resolve container and transitive vulnerabilities?

Container Dependencies:

- Switched from Debian based container to Alpine based container.
- Alpine is a minimal Linux distribution built specifically for production-level container orchestration.

Transitive Dependencies:

- Promoted vulnerable versions of transitive dependencies discovered to top-level dependencies. This must be maintained at project level until the package maintainer updates their library, as per Microsoft's best practices.

CONCLUSION

The transition to Container-Level Scanning offered a more comprehensive approach to security, drastically reducing the vulnerabilities from 19 to 0 in User Space code and transitive dependencies. This endeavor exemplifies Trendsic Corporation's commitment to maintaining cutting-edge practices in software security.

The benefits are manifold, including improved software reliability, fewer security risks, and compliance with the latest best practices. Future projects will continue to evolve and adapt these methods, affirming Trendsic Corporation's status as a leader in secure software engineering.