2025 Ebook

oligo

From Reachability to Exploitability:

A Practical Guide for Reducing CVE Fatigue

Introduction

Prioritizing and mitigating software vulnerabilities is a formidable challenge for most organizations. Tens of thousands of new CVEs are discovered every year. But most organizations lack a clear understanding of which vulnerabilities are genuinely exploitable within their applications and runtime environments.

Developers waste time and effort addressing CVEs that don't pose a legitimate threat to their applications - and their business at large. IT organizations waste time and effort patching vulnerabilities that pose no real-world risk.

Application security solutions can help organizations avoid wasted time and effort and reduce exposure by focusing on the CVEs that matter most. But not all application security solutions take the same approach to assessing risk and prioritizing CVEs.

This eBook explains the differences between traditional shift-left AppSec solutions that assess **reachability** and newer runtime AppSec solutions that assess **exploitability.**

You will learn:

1. The key differences between reachability analysis and exploitability analysis
2. The pros and cons of each approach
3. Best practices for strengthening your application security posture and eliminating CVE sprawl and fatigue

What is Reachability Analysis?

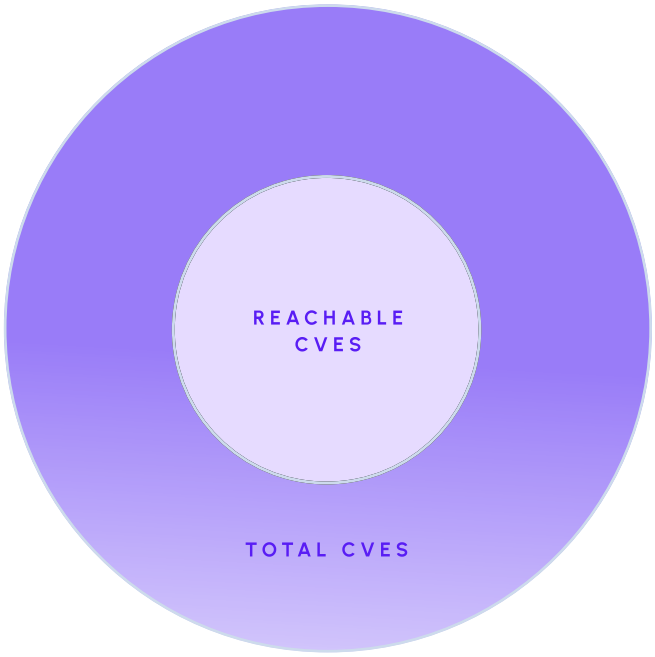
Many "shift-left" application security solutions use reachability analysis to help security analysts and developers prioritize CVEs, improve productivity, and reduce exposure. This approach examines an application's codebase, tracing function calls and dependencies to determine if vulnerable code is **theoretically** reachable (i.e., potentially called by the application.)

Reachability analysis helps developers save time and effort by deprioritizing CVEs associated with uncalled code or functions, but it is far from a perfect solution. Reachability analysis cannot conclusively determine if vulnerable code is ever called in a running application.

Developers are still overwhelmed by false positives. They still squander time and energy addressing CVEs that don't present an actual risk to the business.

Even worse, the assessment is performed at a single point in time during development. A lot can change after that assessment is performed. Vulnerable code can be added during the build phase or even dynamically loaded during runtime, leaving applications exposed to exploitation.

Reachability rules out many, but not all CVEs



Reachability Analysis Tradeoffs

PROS
Straightforward operation: easily integrated early in the development lifecycle
Improved visibility: provides useful insights during code reviews or CI/CD pipeline checks
Increased productivity: allows developers to rule out many, but not all inapplicable CVEs

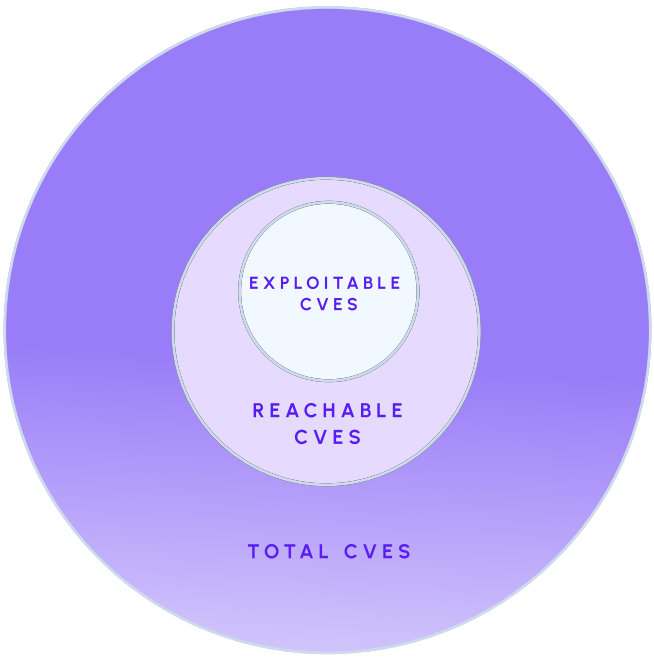
CONS
No runtime visibility: reachability identifies potential exposure but can't deterministically confirm if the vulnerable code is executed in the running application and if the CVE is truly relevant
High false positives: developers are still flooded with alerts for theoretical vulnerabilities that don't pose a legitimate threat to production systems
Single point in time assessment: reachability is evaluated at a single point in time during development; CVEs discovered or published any time after the initial reachability assessment go undetected in production
Inefficient: the approach consumes significant time and CPU resources during the CI/CD process
Limited scope: reachability analysis often only considers the code present in the repository; it can yield inaccurate results if the assessed code does not match the final production state

What is Exploitability Analysis?

Exploitability analysis is the logical evolution of reachability analysis. Unlike reachability analysis, which is performed at a single point in time during development, exploitability analysis is performed continuously on running code during execution, typically during staging or in production. Exploitability analysis overcomes the inherent inefficiencies and constraints of reachability analysis by **definitively** determining whether specific functions within a library or codebase are actively invoked by an application.Reachability analysis is performed on code repositories or within the CI/CD pipeline, but it often misses dynamically loaded code or changes during build and deployment. In contrast, exploitability analysis is performed at runtime, examining the actual running code, ensuring evidence is derived with certainty.

Modern runtime AppSec solutions leverage native kernel-level monitoring capabilities to assess exploitability, providing granular, real-time insights into application behavior without impairing application performance or operation.

Exploitability definitively rules out most CVEs



Exploitability Analysis Tradeoffs

PROS

- **Full runtime visibility:** provides deep, live observability into first-party, third-party, and open-source code
- **Definitive insights:** deterministically ascertains if vulnerable code is truly exploitable
- **Extreme precision:** eliminates false positives, CVE fatigue, and guesswork
- **Ultimate productivity:** removes developers from the vulnerability assessment process
- **Maximum risk mitigation:** reduces attack surface and strengthens security posture
- **Pre-production:** can be run within the CI/CD process whenever a fully running application is present

CONS

- **Special-purpose instrumentation:** requires a software-based sensor
- **Runtime-only insights:** does not scan code in an IDE or repository

oligo

Reachability vs Exploitability Comparison

	REACHABILITY	EXPLOITABILITY
SLDC Stage	Development	Runtime (typically during staging and/or in production)
Scope	Open source in first party applications with source code access	Open source in first and third party applications without source code access
Precision	Low - theoretical	High - deterministic
Real-time visibility into app behavior	Low - theoretical	High - deterministic
Eliminates a large percentage of false positives	No	Yes
Avoids access to source code	No	Yes

Best Practices for Strengthening Your Application Security Posture

Software vulnerabilities can overwhelm development, security, and IT organizations and expose business-critical systems to attack.

Follow these best practices to strengthen application security, reduce CVE sprawl, and mitigate risk:

1. Foster a security-first culture; promote collaboration and knowledge-sharing across development, AppSec, and IT teams
2. Augment existing shift-left tools and practices with a runtime AppSec solution that assesses exploitability
3. Look for a runtime AppSec solution that supports automated remediations to intelligently detect and block active exploits
4. Choose a solution with an extensive vulnerability knowledge base that includes vulnerable functions associated with CVEs

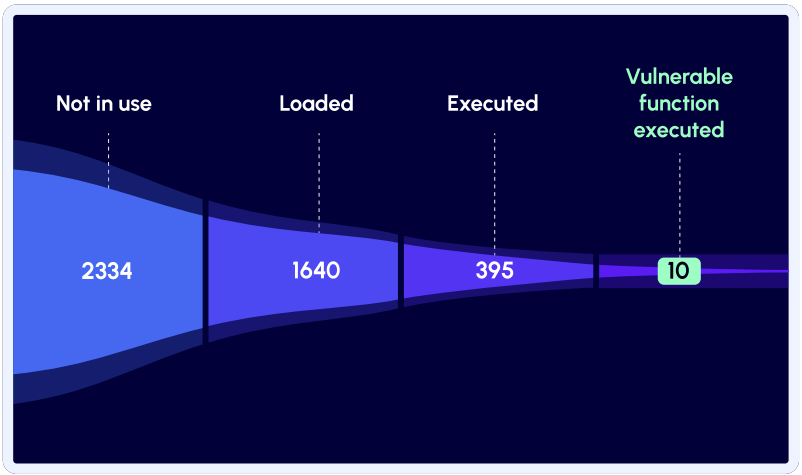
Take Reachability to the Next Level with Oligo Exploitability Analysis

Oligo goes beyond reachability, using exploitability analysis to definitively determine if a CVE presents an imminent threat to your running environment. Our innovative deep application inspection technology lets you observe the runtime behavior of every dependency in every application and system component you build, buy, or use.

With Oligo you can:

- **Discover genuine exposure:** conclusively determine if vulnerable code or functions are loaded and executed by your application at runtime
- **Detect active exploit attempts:** intelligently identify unusual activity symptomatic of an in-progress attack
- **Preemptively stop attacks:** automatically block suspicious actions at the function level, without compromising the rest of the code base and application

Discover genuine exposure with Oligo



Next Steps

oligo

To learn how Oligo exploitability analysis can help your organization improve observability and mitigate risk

See Oligo in Action