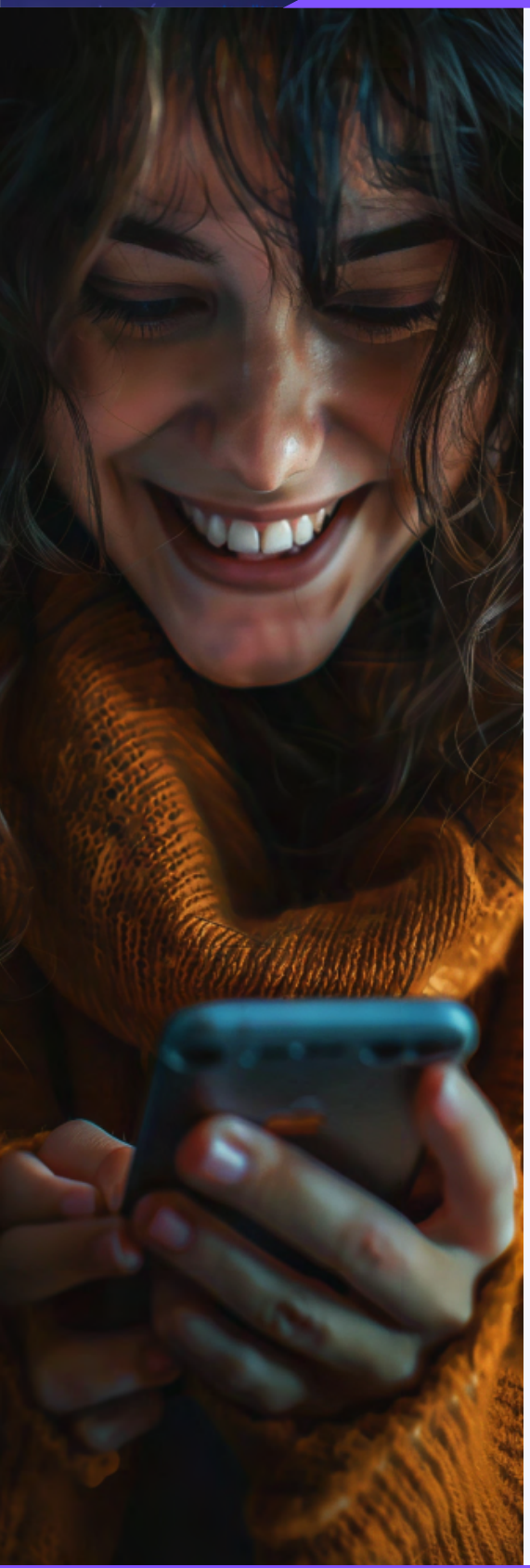
BUYER'S GUIDE TO
MAYHEM & COMPREHENSIVE

# APPLICATION SECURITY TESTING

Mayhem

# TABLE OF CONTENTS

# EXECUTIVE SUMMARY



**Applications are the heart of every organization,** pumping information throughout the business to provide users with the data needed to drive their objectives. In the past decade, applications became a driving force in all aspects of enterprises.

Due to this pivotal role in organizations, **applications are the target of 85% of cyber security attacks**, according to SAP. A multitude of security testing solutions fail to evolve with the everchanging cybersecurity landscape. To protect your company from exploitable vulnerabilities in applications, you'll need advanced forms of testing that grow with you.

The Buyer's Guide to Mayhem and Comprehensive Application Security provides an overview of different application security solutions. You will learn how different solutions excel and where gaps are left behind. A combination of tools is recommended to help organizations implement a comprehensive application security testing program that adapts to ever-changing security needs.

With this newfound application security testing knowledge, Mayhem can protect your company against cyber security threats. **Mayhem is known to prevent exploitables leading to security attacks.**

# WHAT MAKES SOFTWARE VULNERABLE?

**Software bugs are born when detailed programs are built with mistakes.** These bugs are viewed as defects until malicious activity exploits them as vulnerabilities. Though, applications may have software vulnerabilities for a variety of reasons including:

1. **Mistakes**
   Developers live and die by the roadmap, but their mission to quickly deliver creative features may make mistakes that harm users.

2. **Misconfiguration**
   Application stacks are complex. This makes misconfigurations of application components lead to vulnerabilities. An instance of these components is frameworks to databases to servers to platform.

3. **Inherited Vulnerability**
   Free, open source and third-party code is more popular than ever with development demands. Free, however, is never free. An organization's benefit of speed and scale could be paying a price in security. Organizations can inherit security risks from the unchecked application supply chain.

4. **Neglect**
   Application verification and validation best practices call for positive testing and negative testing. The objective of negative testing is to ensure the software remains stable in unexpected use cases and is free of vulnerabilities.

**The application attack surface is growing by 111 billion new lines of code every year, with newly reported zero-day exploits rising from one-per-week in 2015 to one-per day in 2021.**

# HOW ARE VULNERABILITIES IDENTIFIED?

**Application vulnerabilities fall into one of three risk categories:**

**The Spectrum of Risks**

| Known Knowns | Known Unknown | Unknown Unknown |
|---|---|---|

## Known Known

Identifiable risks known to lead to compromise (CVE) with 100s to 1,000s of vulnerabilities in a given software.

Known Knowns are **identifiable risks that are known to lead to compromise.** These risks are identified through a Common Vulnerabilities and Exposure (CVE) ID.

## Known Unknown

Identifiable risks that could potentially lead to a compromise (CWE) with 10,000s to 1Ms of defects in a given software.

Known Unknowns are **identifiable risks that could potentially lead to compromise,** and these risks exhibit software flaw patterns that are likely to create exploitable vulnerabilities. These risks are identified through a Common Weakness Enumeration (CWE) ID.

## Unknown Unknown

Unidentifiable risks not detectable by CVE or CWE and an unknown quantity of risks.

Unknown Unknowns are **risks that cannot be identified.** Unknown Unknowns present the greatest risk, because they enable adversaries to operate unnoticed for an extended period of time.

# NAVIGATING APPSEC SOLUTIONS

**A myriad of security tools creates the foundation software security best practices.** Organizations need different security tools to satisfy their unique needs. By and large, most software security solutions fall into one of two categories:

1. **Secure the Source**
   Integrate application security testing tools through the SDLC to find vulnerabilities and weakness before deployment. An example of Secure the Source is Static Code Analysis (SAST), which works as prevention during the development stage.

2. **Find and Fix**
   Identify security vulnerabilities and misconfigurations in production applications for development teams to fix later. An example of Find and Fix is Dynamic Analysis (DAST), which works as detection during quality assurance.

In the sections following, consider the detailed technologies that fall within these two categories and outline **how these tests could minimizing software security risk in your organization.**

# (SAST) STATIC APPLICATION SECURITY TESTING

**Static application security testing** (also known as static analysis or static code analysis) tools **uncover bugs by analyzing source code.** The defects they identify are known unknown risks. SAST vendors dissect each CWE to uncover specifications. Those specifications are then implemented to detect code flaws and weaknesses that could lead to vulnerabilities. This is a long, laborious process. Thus, SASTs, as with most application security testing (AST) tools, are better as they age. **More experience and knowledge built into SASTs products can catch more defects.**

SASTs operate in the world of “what-ifs”, taking in potentially relevant information to make assumptions on what “could-be”. The cautious nature of SASTs processes with good intent, but this approach leads to high false-positives and inaccuracies for many organizations.

# (SAST) STATIC APPLICATION SECURITY TESTING

**Many organizations gripe about SAST's inaccuracy.** All businesses lack time and resources, and therefore, it is unrealistic for them to chase every potential issue.

**Despite its shortcomings, SASTs have their place in the SDL as recommended, preventative practice.** Simply put, it's good hygiene. Since static analysis requires source code, SASTs are able to provide prescriptive remediation advice, down to the line of code. Static analysis can also be introduced earlier than most tools in the software development lifecycle, lowering cost and effort for remediation.

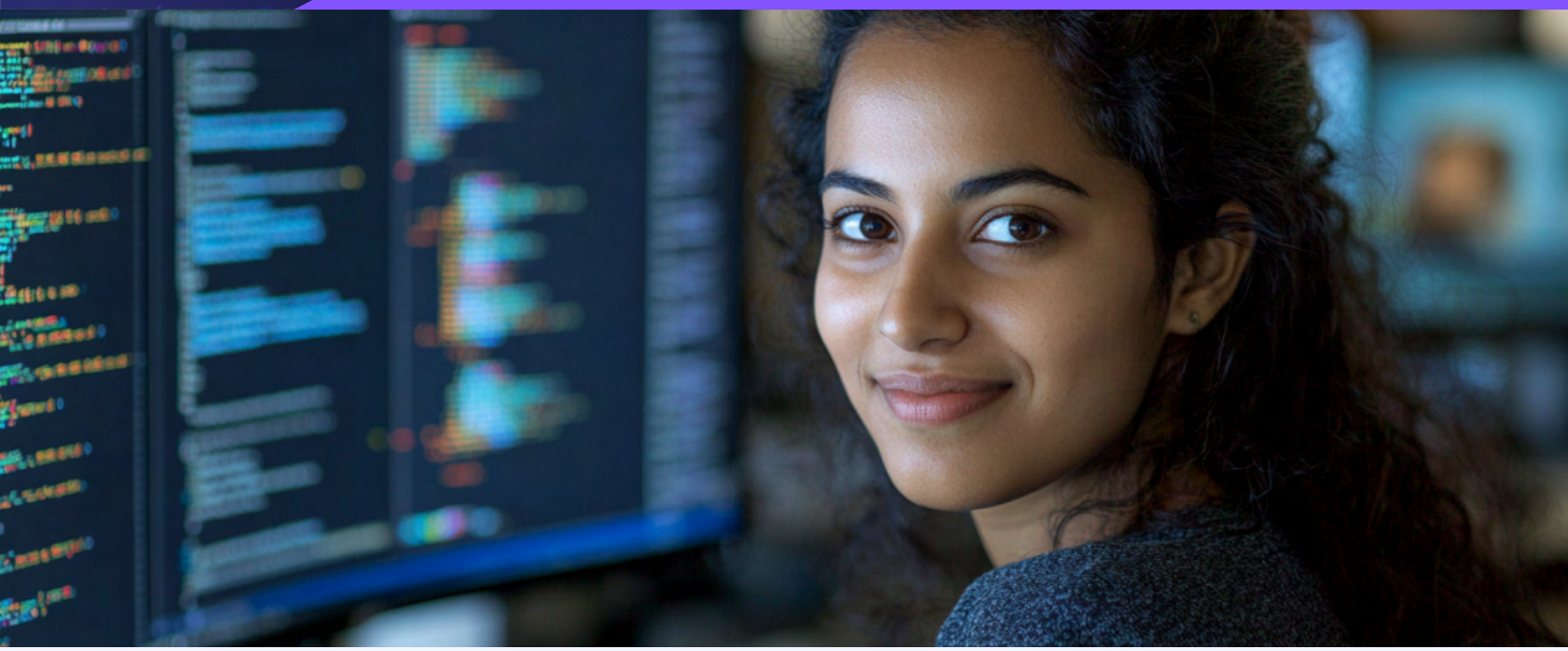
**A vendor with a market leading SAST solution revealed that, on average, it takes an enterprise approximately 8 years for their code to be deemed "clean".**

# (DAST) DYNAMIC APPLICATION SECURITY TESTING

**Dynamic application security testing** is also known as dynamic analysis. Unlike SAST, **DAST serves as an overarching category for a variety of techniques.** To provide more specificity, we'll focus on a DAST technique known as software composition analysis (SCA), a software security testing solution that nicely complements SAST.

Some, though not all, **SCAs can uncover vulnerabilities without access to source code.** The vulnerabilities they identify are known known risks. When vulnerabilities become known, SCAs have automated capabilities that pull information from the National Vulnerabilities Database (NVD) to detect them in the software they're scanning.

# (DAST) DYNAMIC APPLICATION SECURITY TESTING



**SCAs operate in the world of "what-is", relying on accepted and publicly known information to uncover vulnerabilities.** While they prevent adversaries from exploiting low-hanging fruits, they foster a reactive security approach -- meaning they mitigate risk after the window of exposure has opened. When vulnerabilities become known, they are publicly disclosed to organizations and adversaries at the same time. By nature, vulnerabilities are frictionless weapons. Vulnerabilities can be turned around and used in nefarious ways, at scale. Thus, time is of the essence when remediating these vulnerabilities.

**SCAs offer prescriptive advice, listing the affected component and providing a link to its patch.** This level of actionability is unique in DAST solutions. However, bear in mind, not all known vulnerabilities flagged by an SCA solution is exploitable. Additionally, each patch must be tested to ensure interoperability with the entirety of the application and/or the ecosystem they live in.

# MAYHEM FOR SECURITY TESTING

**Mayhem is a new technique under the dynamic testing and grey-box testing category.** The defects Mayhem identifies are unknown unknown risks. **Mayhem uses advanced fuzz testing** techniques to uncover defects utilizing unknown or uncommon attack patterns. After each simulated attack, it monitors and leverages its target reactions, or behaviors, as feedback to **autonomously generate new test cases that are increasingly likely to uncover more defects and new code edges.**

Mayhem operates between the world of "what-if" and "what-is". It uncovers unknown defects, enabling organizations to be preventative and proactive. Interestingly, continuous fuzz testing like Mayhem has been a proven and accepted software security practice for years. However, fuzzing has been exclusive to technology behemoths, such as Google, Microsoft, Apple, Nvidia, and more, who have the technical savvy and budget to implement and maintain such technologies.

**Latest advancements in this field of study have dramatically improved usability, making Mayhem increasingly accessible to everyone.** Although dynamic testing has required complete builds for testing, Mayhem has defied these inherent limitations with its capability to conduct regression and component testing during development, allowing organizations to find issues early, often, and continuously as a part of their DevOps pipeline.

Security engineers of Google's OSS-Fuzz team have disclosed that while it is possible to bootstrap and operate high-performance fuzzers, people often underestimate the complexity of upstanding such solutions. Yet, the benefit of fuzzing is undeniable.

**Teams at Google report that fuzzing uncovers 80% of their bugs, with the other 20% uncovered by other testing techniques, or naturally in production.**

# MAYHEM FOR SECURITY TESTING

**True to the DAST technique, Mayhem is accurate and precise, uncovering defects with zero false positives.** As mentioned prior, Mayhem is a grey-box solution, which means it conducts dynamic testing with visibility into source code. It can pinpoint the exact line of affected code and provides expert remediation advice, making fixes as simple as possible for developers.

## Mayhem in Action:

Award Winning TechnolZogy at DARPA Cyber Grand Challenge


The 2016 DARPA Cyber Grand Challenge first put to the test an early prototype of Mayhem. At the time, Mayhem and six other finalists competed in an open source operating system extension built exclusively for computer security research and experimentation. **In this environment Mayhem was able to perform and execute tasks autonomously and without human intervention including automatically patching vulnerabilities -- something that has yet to be done in the real world today.**

It's important to point out the evolution of Mayhem from DARPA CGC-winning prototype to the advanced testing technology it is today. By competing in the DARPA CGC, the ForAllSecure researchers were able to iterate on Mayhem and bring lessons learned from the DARPA CGC into real-world situations.

DARPA CGC set the stage for Mayhem and autonomous security. **Since then, Mayhem continues to revolutionize the application security world and solve real-world problems.**

# ALL TOGETHER NOW

Secure development practices call for the use of various testing techniques throughout the development lifecycle. **SAST, SCA, and Mayhem strategically offer strength in each technique's limitations, offering comprehensive application testing across the spectrum of software security risk.**

## Solutions for Unknown Risks

### SAST for Known Unknown Risk

SAST directly analyzes the code to detect coding and design vulnerabilities and/or weaknesses. There is a focus on quality with a low number of false-positives. Test during the SDLC development phase with manual test case creation.

### SAST Alone

Non-Running State During Testing

Low to Medium Accuracy

Low Automation

No Supply Chain Management

Unconfirmed Exploitability of Found Bugs

Doesn't Check Every Line of Code

Fast Interaction of Loop

# ALL TOGETHER NOW

Secure development practices call for the use of various testing techniques throughout the development lifecycle. **SAST, SCA, and Mayhem strategically offer strength in each technique's limitations, offering comprehensive application testing across the spectrum of software security risk.**

## Comprehensive Security

- Running state during testing
- High Accuracy
- High automation
- Supply Chain Management
- Confirmed Exploitability of Found Bugs
- Doesn't Check Every Line of Code
- Fast Interaction of Loop

# KNOWING WHAT'S IN IT FOR YOU

**There are many benefits to employing a multi-pronged software security approach:**

1. Lower costs due to early vulnerability prevention and/or detection.
2. Greater software quality, security, and stability.
3. Security testing for right- and left-of-ship.
4. Secured and hardened attack surface for minimized risk.
5. Detection of known, unknown, and zero-day vulnerabilities.

# NOW, IT'S YOUR TURN

Software is everywhere. It permeates nearly every aspect of our lives and has proven itself to be a productivity enhancer. As our dependency on software increases, we must ensure their reliability.

A multi-pronged application security strategy comprised of static analysis, software composition analysis, and continuous fuzz testing offer comprehensive security testing coverage across the spectrum of software risks. Well-known testing techniques like SAST and SCA are not enough as they can leave significant gaps behind.

Mayhem's continuous fuzz testing is a proven and accepted method for uncovering unknown defects automatically. It offers strengths in SAST and SCA's limitations. Utilizing a combination of these three security testing techniques not only ensure that organizations are enabled to deliver safe, secure, reliable software and service, but also implements predictability in today's unpredictable, ever-changing threat landscape.

**Interested in trying a security testing solution that increases app security?**

**SCHEDULE A DEMO**

mayhem.security/demo