Contrast SECURITY

SOLUTION BRIEF

AppSec Solution Guide for Complying with New NIST SP 800-53 IAST and RASP Requirements

Executive Overview

New application security (AppSec) standards by the National Institute of Standards and Technology (NIST) are a recognition that legacy AppSec tools are inadequate for enabling timely delivery of secure applications that address the current advanced threat landscape. Specifically, the guidance calls for the adoption of security instrumentation in the form of interactive application security testing (IAST) and runtime application self-protection (RASP) tools. NIST recognizes that this technology is vital for reducing alert noise, minimizing interruptions to the development cycle, and prioritizing the vulnerabilities that pose the greatest risk for a particular organization. These benefits cascade to the everyday lives of developers and security team members, enabling them to be more effective in their jobs while improving their overall application security posture.

The new NIST Cybersecurity Framework contains a Special Publication, NIST SP 800-53 Revision 5[1] that includes the following:

- SA-11(9), Developer Security Testing and Evaluation: "Require the developer of the system, system component, or system service to employ interactive application security testing [IAST] to identify flaws and document results."[2]
- SI-7(17), Software, Firmware, and Information Integrity: "Implement [Assignment: organization-defined controls] for application self-protection at runtime."[3] This section mandates implementation of RASP technology to "reduce the susceptibility of software to attacks by monitoring its inputs, and blocking those inputs that could allow attacks."[4]

These requirements are a recognition that, as a result of an increasing attack volume[5] driven by automation, security instrumentation is critical to assessing the security risk of specific software vulnerabilities. In addition, instrumentation can improve DevOps efficiency by minimizing security-related delays to the development cycle.

> *When combined with analysis techniques, interactive application security identify a broad range of potential vulnerabilities and confirm control effectiveness.*[6]

Contrast SECURITY

The Influence of NIST Continues to Grow

These new requirements will have significant impact across all industries, as the NIST Cybersecurity Framework is quickly becoming the default standard in the United States. All U.S. federal government agencies are now mandated to comply with NIST, and many state and local governments have followed suit.[7] In the private sector, it is projected that 50 percent of U.S. organizations will follow the NIST framework by the end of this year.[8]

This wide adoption means that the new NIST guidelines will likely set a standard for other frameworks. For example, organizations that measure themselves against the North American Electric Reliability Corporation (NERC), the Federal Information Security Management Act of 2002 (FISMA), or the Federal Risk and Authorization Management Program (FedRAMP) can expect to have IAST and RASP requirements in the near future.

Implications of SA-11(9): Developer Security Testing and Evaluation

DEVELOPER TESTING AND EVALUATION – INTERACTIVE APPLICATION SECURITY TESTING

Require the developer of the system, system component, or system service to employ interactive application security testing tools to identify flaws and document the results.

Discussion: Interactive (also known as instrumentation-based) application security testing is a method of detecting vulnerabilities by observing applications as they run during testing. The use of instrumentation relies on direct measurements of the actual running applications, and uses access to the code, user interaction, libraries, frameworks, backend connections, and configurations to measure control effectiveness directly. When combined with analysis techniques, interactive application security testing can identify a broad range of potential vulnerabilities and confirm control effectiveness. Instrumentation-based testing works in real time and can be used continuously throughout the system development life cycle.[10]

IAST technology detects vulnerabilities by observing applications as they run during testing—and can be used throughout the software development life cycle. Additionally, using instrumentation-based testing works in real time and can be used continuously throughout the system development life cycle.

> *Runtime application self-protection [RASP] technology can reduce the susceptibility of software to attacks by monitoring its inputs, and defending APIs that are vulnerable to those inputs rather than using simple pattern matching.*[9]

Contrast SECURITY

TAKEAWAYS FOR SECURITY TEAMS

The increasingly complex threat landscape means that security teams must address significant risk across the organization. This requires a strategic approach that prioritizes threats and vulnerabilities according to the risk they pose to the organization. One thing that hampers such a strategic approach is alert noise in a world of increasing volume. It is no longer possible for security team members to comb through every alert manually. Complying with the new standards will make security teams more productive—and less stressed.

> *For 43% of organizations, false positives comprise >20% of alerts. 15% report that >50% of security alerts are false positives.*[11]

- **Continuous visibility.** The instrumentation approach ensures continuous monitoring throughout the software development life cycle, enabling real-time visibility for the security team without having to interrupt developers to do a scan.
- **Security accuracy eliminates false positives and identifies false negatives.** Security instrumentation identifies only vulnerabilities that pose risk. Traditional security measures that only look at code such as SAST or HTTP traffic such as dynamic application security testing (DAST) generate huge volumes of false positives that require security experts to resolve. IAST, on the other hand, has access to both of these datasets—plus libraries and frameworks, application state, data flow, control flow, backend connections, and configurations. The analysis engine then takes all this telemetry into account in assessing the risk of different vulnerabilities. The process is the same whether the application is containerized or not. SAST and DAST AppSec models rely on known signatures and do not account for unknown threats and zero-day attacks. As upwards of 50 percent of malware and attacks are unknown or zero day, this results in false negatives, which leave applications wide open to attacks—without the knowledge of the security team.
- **Full visibility and risk.** SAST and DAST AppSec models struggle to achieve full visibility across the full application attack surface. In particular, they frequently cannot see across all application programming interface (API) connections for each application. This results in missed vulnerabilities that can pose serious risk to an application.
- **Reduced security staff time spent on vulnerabilities.** IAST provides vulnerability telemetry across the entire application and API portfolio, and can help eliminate the majority of vulnerabilities without security involvement.

TAKEAWAYS FOR DEVELOPERS

For development teams, security is viewed as an impediment to getting their job done. As a result, compliance with NIST is not a big concern for many developers. However, addressing these new provisions in NIST can help eliminate the developer's biggest headache when it comes to security: delays caused by AppSec tools and processes.

Contrast SECURITY

SAST, DAST, and software composition analysis (SCA) tools can all create delays in the development process, and the need to do lengthy vulnerability scans can impact decisions on the timing of code changes and the methodology used at different phases of the development process. Deploying security instrumentation in accordance with NIST guidelines minimizes delays caused by security processes, while providing maximum flexibility for development teams to innovate. Following are some of the key takeaways for developers:

- **Elimination of vulnerability scans.** SAST scanning is time-consuming and creates code halts and other delays in the development cycle. With IAST, scanning is done continually in the background every time code is executed. Developers can continue their work and structure their processes in the most efficient way without having to worry about security delays.

- **Early and continuous detection of vulnerabilities.** Vulnerabilities in newly developed code will be identified immediately the first time the code is executed. This greatly reduces the likelihood that vulnerabilities are deployed in production. It also helps prevent the need for a major "rip and replace" of code late in the development process.

- **Fewer meetings.** With IAST, all standard usage becomes an additional security test. Security results appear as you walk through your application with adequate test coverage, either removing the need for a dedicated security test plan or letting it focus on the uniqueness of your application. This dramatically reduces the number of communications and meetings required between security and development staff to confirm vulnerabilities, trace their origins, and remediate them with SAST and DAST models.

- **Automated runtime verification of remediation.** When developers remediate a vulnerability that IAST has identified, the new code is immediately scanned so that developers have verification that the fix was successful.

- **Elimination of "tool soup."** Many organizations employ disparate SAST, DAST, and SCA tools to cover the software development life cycle. Often, these disparate tools will find the same issue in different ways, requiring duplicative manual work to review the issue. Running each of these tools and interpreting the results takes time from the development process, and requires a lot of security expertise not present on the development team. IAST combines the capabilities of these legacy tools and greatly expands them while automating many manual processes.

Implications of Software, Firmware, and Information Integrity [SI-7(17)]

SOFTWARE, FIRMWARE, AND INFORMATION INTEGRITY | RUNTIME APPLICATION SELF-PROTECTION

Implement [Assignment: organization-defined controls] for application self-protection at runtime.

Discussion: This control enhancement employs runtime instrumentation to detect and block the exploitation of software vulnerabilities by taking advantage of information from the software in execution. Runtime exploit prevention differs from traditional perimeter-based protection such as guards and firewalls, that can only detect and block attacks by using network information without contextual awareness. Runtime application self-protection technology can reduce the susceptibility of software to attacks by monitoring its inputs, and

blocking those inputs that could allow attacks. It can also help protect the runtime environment from unwanted changes and tampering. When a threat is detected, runtime application self-protection technology can prevent exploitation and take other actions (e.g., sending a warning message to the user, terminating the user's session, terminating the application, or sending an alert to organizational personnel). Runtime application self-protection solutions can be deployed in either a monitor or protection mode.[12]

Runtime exploit prevention is different from traditional perimeter-based protections, such as guards and firewalls, that can only detect and block attacks using network information absent contextual awareness. In contrast, runtime application self-protection (RASP) monitors software inputs and blocks those that could allow attacks. It also can take proactive actions to address attacks.

TAKEAWAYS FOR SECURITY TEAMS

Security teams benefit from RASP in varying ways:

- **Elimination of false positives and negatives.** Web application firewalls (WAFs) provide signature-based blocking of web requests that look like attacks. They are notorious for both false positives and false negatives. False positives in particular are very common: One study finds that 43 percent of organizations identify more than one in five alerts as false positives.[13] And 15 percent say that more than half of their alerts are false positives. Security team members can spend many hours combing through WAF alerts to distinguish legitimate threats from false positives.[14]

  On the other hand, RASP technology observes applications as they run, and analyzes whether a detected attack string would successfully execute. By performing runtime analysis of whether an attack will be successful, false positives are eliminated while helping ensure a secure application in production. This eliminates both false positives and false negatives, enabling increased efficiency and better protection.

- **Actionable alerts.** Beyond eliminating the false positives that hamper security team productivity, RASP technology distills the alert noise further, delivering highly accurate and relevant alerts based on an application's actual behavior.

- **Improved scalability.** RASP-instrumented "no-touch" application protection for built-in elasticity follows applications everywhere— from on-premises to the cloud. This fully portable AppSec approach is indifferent to network configurations, protocols, encryption, encoding, containers, microservices, and more. RASP affords security teams seamless adoption of and migration to digital transformation without operational disruptions or additional security expertise and skill sets.

- **Increased visibility.** RASP delivers 100 percent accurate runtime visibility that sees everything that is happening—data flows, frameworks, connections, and so forth. Security instrumentation enables stack-file-line, code-level visibility for the traceability of all code, and connections for both custom code and open sources. This enables security teams to orchestrate remediation with precision runtime-security telemetry and remediation prioritization, which accelerates secure code development. This raises the security awareness level across the entire software development life cycle.

TAKEAWAYS FOR DEVELOPERS

RASP technology can save developers significant time through its granular assessment process that eliminates false positives and prioritizes the risk of legitimate hits. Developers benefit in these ways:

- **Elimination of false positives.** False positives from a WAF can waste a developer's time in fielding their security colleagues' questions about alerts that they receive. By performing runtime analysis of whether an attack will be successful, it eliminates false positives while helping ensure a secure application in production.
- **Better detection through runtime analysis.** WAFs are also notorious for false negatives (viz., missed threats). Because they monitor applications as they are running, RASPs can identify virtually all attacks by how their code executes.

> *44% of companies have delayed moving an application into production due to security concerns.*[15]

Conclusion

The new AppSec guidelines found in NIST SP 800-53 are an acknowledgement that legacy tools are no longer doing the job. Security teams are overwhelmed by both increasing risk and alert noise, and developers are frustrated by security-related delays. The result for DevOps: a slower time to market and a larger potential for vulnerabilities and attacks in production.

The new standards will help organizations "right the ship" and achieve continuous assurance. Organizations should consider the following steps to accomplish this:

1. Adopt the NIST Cybersecurity Framework to minimize risk. NIST is considered the gold standard, and following it optimizes an organization's overall security posture by reducing risk.
2. Understand that security instrumentation is the key to meeting the new NIST requirements.
3. Eliminate security bottlenecks for the development team with DevOps-native and NIST-compliant security instrumentation. This unleashes developers to impact revenue rather than wait for time-consuming security testing and processes.
4. Optimize scarce cybersecurity talent by enabling developers to identify and remediate vulnerabilities in real time while coding.

When these steps are taken, security teams are freed to focus on vulnerabilities that actually pose risk to their organization, and developers are empowered to work efficiently, meet aggressive timelines, and deliver reliable and secure applications in production.

Contrast SECURITY

[1] "Security and Privacy Controls for Information Systems and Organizations," Draft NIST Special Publication 800-53, Revision 5, March 2020.
[2] Ibid., p. 271.
[3] Ibid., p. 339.
[4] Ibid., p. 339.
[5] Adam Shepherd, "500 New Cyber Threats Emerge Every Minute," ITPro, March 13, 2018.
[6] "Security and Privacy Controls for Information Systems and Organizations," Draft NIST Special Publication 800-53, Revision 5, March 2020, p. 271.
[7] Matthew Barrett, "Why Your State Should Join the 21 That Use the NIST Cybersecurity Framework," StateTech, September 27, 2018.
[8] "Cybersecurity Framework," National Institute of Standards and Technology, accessed March 19, 2020.
[9] Ibid., p. 339.
[10] Ibid., p. 271.
[11] Michael Hill, "Over a Quarter of Security Alerts Are False Positives," Infosecurity, March 17, 2020.
[12] "Cybersecurity Framework," National Institute of Standards and Technology, accessed March 19, 2020.
[13] Jeff Williams, "Why It's Insane to Trust Static Analysis," Dark Reading, September 22, 2015.
[14] The State of Container and Kubernetes Security, Winter 2020," StackRox, accessed March 20, 2020.
[15] Ibid.

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333
Fax: 650.397.4133

Contrast SECURITY

contrastsecurity.com