Understanding Application Security: Risks, Tools, and Best Practices

Learn about the key risks facing software applications, all main categories of application security tools, and best practices for securing your applications.

What is Application Security?

Application security is intended to prevent and effectively respond to cyber security threats targeted against software applications. This includes security considerations and risks that arise during application development and design, as well as systems and methods for securing applications after application deployment.

With the advent of the DevSecOps organizational pattern, organizations are shifting application security left. This means there is additional focus on ensuring applications are secured at early development and testing stages.

Developers, security, and operations teams are collaborating to identify security issues at every stage of the development lifecycle, and fix them as part of normal development workflows. The earlier security flaws can be fixed, the easier they are to remediate, and the lower the risk of exploitation by attackers.

For example, it is now common for developers to receive alerts about security vulnerabilities as they are writing code, and remediate them before initially committing their code to source control through new CI/CD tools. After code is committed, initial build testing performs additional tests for security flaws and vulnerabilities, and so on. Application security is integrated into every stage of the pipeline, including monitoring and threat prevention for production applications at runtime.

In this article, you will learn:

Application Security Risks

Application security is growing in importance as the threat landscape evolves. Here are a few common risks facing most software applications:

- **Components with known vulnerabilities**—modern software applications can have thousands of components and dependencies, many of them open source. Developers use libraries, frameworks and other software modules, often without testing them for security issues. Software with untested components may contain severe vulnerabilities that can be exploited by attackers.
- **Data leakage and exposure**—while this applies to all applications, web applications are especially vulnerable. Many web applications do not properly protect sensitive data like personally identifiable information (PII), credentials, or financial information. Threat actors who compromise the initial lines of defense can steal this data, causing harm to the organization and its customers, and creating legal and compliance exposure.
- **Injection**—a common threat vector is the injection of malicious SQL statements, operating system commands, LDAP configurations, etc. If the web application does not properly sanitize data submitted by users, via web forms or other methods, attackers can use the same methods to inject malicious code. Injection attacks can have severe consequences, from data exfiltration to compromise of the entire server.
- **Security misconfiguration**—some web applications have security controls in place, but do not properly configure them. Applications, the operating systems they run on, and their supporting libraries and dependencies, should all implement security best practices, such as secure communications, closing unnecessary open ports, and limiting privileges. Failure to ensure secure configuration can expose the application to attack.
- **Authentication and authorization**—if application features related to authentication and session management are not implemented correctly, attackers can use them to compromise accounts, and perform privilege escalation to gain access to sensitive systems or administrator functions.
- **Insecure deserialization**—deserialization involves translating objects into a data format that can be stored in the file system. When serialized data is incorrectly converted into an object usable by the application, this can enable Remote Code Execution (RCE) attacks, which can allow attackers complete access to the compromised system.
- **Inadequate logging and monitoring**—even with all security measures in place, attacks will happen. Without comprehensive logging and monitoring of applications, attackers can perform reconnaissance of applications, attempt intrusion, and eventually find a way to bypass security controls. Monitoring enables security teams to detect these activities and mitigate the threat.
- **Ignoring security during early development**—many organizations have not adopted DevSecOps practices, and only include security at late stages of the development process. This makes it much more difficult to identify and remediate security weaknesses, and increases risk.

What is the OWASP Top 10?

The above threats are significant for all types of software applications. However, web applications are typically more sensitive to attack, because they are commonly exposed to public networks.

The Open Web Application Security Project (OWASP) performs regular research to identify the top ten security vulnerabilities facing web applications. In addition to the risks we listed above, the OWASP top ten identifies the following additional vulnerabilities commonly facing web applications:

- **XML External Entities (XXE)**—improper processing of XML documents, which allow attackers to create malicious references to external entities. XXE attacks can result in exposure of sensitive data on servers, internal port scanning, and denial of service (DoS).
- **Cross Site Scripting (XSS)**—exploitation of insecure session mechanisms, which allow attackers to impersonate users and perform activities on a web application without their consent. XSS attacks can be used to hijack user sessions, redirect users to malicious websites, steal personal data, and deface websites.

Application Security Tools

The following are major categories of tools used for application security. Most of these can also be considered as DevSecOps tools, because they promote ongoing security testing as part of development and deployment workflows.

Related content: read our guide to DevSecOps tools ›

Web Application Firewall (WAF)

business rules, and threat intelligence, to identify suspicious traffic sources or malicious user behavior, and prevent it from reaching the application.

A downside of WAFs is that they require heavy tuning to each web application's specific business rules. WAFs can block normal user behavior, unless the organization implements custom rules to specify which actions and activities are allowed. WAF rules based on static signature can often be bypassed by attackers.

Static Application Security Testing (SAST)

SAST tools analyze application source code to discover security vulnerabilities, and suggest remediations. They are a type of white-box testing, in which the testing mechanism is aware of the internal workings of the system under test.

SAST tools can:

- Detect and report on security vulnerabilities in source code
- Identify defects like improper input validation, numerical errors, race conditions, problematic pointers or references, path traversals, and more.
- Scan both non-compiled code (using static code analysis) and compiled code (using binary analyzers)

Drawbacks of SAST include that it typically requires heavy customization to fit the codebase under review. SAST solutions also tend to return a large number of false positives, and it takes time to review false positives and tune the solution to remove them.

Dynamic Application Security Testing (DAST)

DAST tools scan code running in production, to identify vulnerabilities and security weaknesses. They are a form of "black-box testing", because they operate without access to the source code or knowledge of software internals. For this reason, DAST tools can test software from the point of view of an attacker.

DAST tools can:

- Detect security vulnerabilities that can be exploited in a running application.
- Scan and identify security issues in APIs, HTTP/S requests and responses, JavaScript, sessions and cookies, authentication, and more.
- Simulate a variety of attacks like code injection, cross site scripting (XSS) and cross-site request forgery (CSRF) and test the application's response.
- Perform fuzz testing to see the application's response to random or malformed inputs.

A few drawbacks of DAST are that they return a large number of false positive alerts, and it is difficult to get them to follow complex application flows. Running DAST in production can have unexpected effects like crashing an application, or producing large numbers of new data records.

Interactive Application Security Testing (IAST)

IAST is a hybrid approach that combines SAST and DAST tools. They can scan applications both during development stages in a white-box approach, and during runtime, using a black-box approach.

The hybrid approach has many advantages. It can identify flaws and vulnerabilities early, allowing quick remediation during early development. But they also verify that vulnerabilities cannot be exploited when an application is deployed in testing or production environments.

IAST tools can:

- Analyze data flow to simulate complex attacks.
- Perform recursive dynamic analysis, seeing how the application reacts to specific tests and generating new tests accordingly—this process can continue until the tool identifies a vulnerability.
- Reduce false positives, which are common in traditional SAST/DAST tools, by combining and correlating data from static and dynamic testing.

Software Composition Analysis (SCA)

SCA tools test source code to create a bill of material (BOM) of software components, with a special focus on open source components. For each open source component, they can identify its full tree of dependencies, and scan the component and all dependent libraries for security vulnerabilities and license issues.

SCA tools can:

- Identify known vulnerabilities in open source components, according to the NIST CVE database and other open and commercial vulnerability databases.
- Identify components that need to be patched, or are end of life (EOL) and should be replaced.
- Scan binary code, byte code, and original source code.

Runtime Application Self-Protection (RASP)

RASP tools are considered an evolution of SAST, DAST and IAST. They are primarily relevant to applications deployed in production. RASP analyzes application traffic and user behavior at runtime to detect and prevent cyber threats.

RASP can review application source code (even for compiled applications), and analyze weaknesses and vulnerabilities. It identifies security vulnerabilities and proactively protects applications, by either terminating the session or issuing an alert.

RASP provides deep inspection and protection, which many argue reduces the importance of SAST, DAST, and IAST. Instead of having to modify applications to remediate security vulnerabilities, which is complex and time consuming, RASP can protect applications and prevent exploitation of those vulnerabilities. However, RASP cannot substitute for a comprehensive DevSecOps process and early detection of security vulnerabilities.

Application Security Testing Orchestration (ASTO)

ASTO is a new category of tools introduced by Gartner in 2017. It integrates security tools across the entire software development lifecycle, to support DevSecOps processes. The purpose of ASTO is to coordinate application security tools, such as the ones we described above, ensuring they are used appropriately at each stage of the development pipeline.

Tools are critical to an effective application security testing process. ASTO not only coordinates and automates tools, but also makes it possible to manage their data and insights in one place. This allows organizations to track all potential risks more easily and resolve them more quickly, without needing to switch between multiple consoles and dashboards.

Perform a Threat Assessment

When planning an application security program, map out applications by sensitivity, and investigate the key entry points an attacker can use to compromise each application. Identify security measures already in place, and evaluate if they are appropriate to protect against the threats. Set reasonable goals and milestones to improve protection and achieve the required level of security for each application.

Integrate Security into CI/CD Processes

Modern software development processes are managed using continuous integration / continuous delivery (CI/CD) tools, which automate the entire release process. Security testing tools should be fully integrated with CI/CD pipelines, from planning to development, testing, deployment and production environments.

Automated application security tools allow teams to test applications at multiple checkpoints throughout the CI/CD pipeline. For example, when a developer submits code and triggers a build, it should automatically undergo security testing, and return feedback to the developer, allowing them to quickly fix security issues in the code.

Prioritize Remediation

In most organizations, application security tools will identify a large number of application vulnerabilities. It is usually not possible to remediate all vulnerabilities, at least not immediately. Prioritization is very important—teams need to easily identify the most critical vulnerabilities. They should have efficient processes in place to remediate them without compromising developer productivity.

The security testing process should include automated indicators of the severity and potential for exploitation of each vulnerability. If necessary, a manual assessment can be performed, to understand whether the vulnerabilities are really a risk to the business. For example, a vulnerable component may not be used in the production application at all, or a vulnerable system may have other security measures which make it more difficult to exploit.

Manage Privileges

Tooling and data related to application security is highly sensitive, and can be very useful to an attacker. This includes security policies, processes, tool configurations, and credentials that can be used to access CI/CD tooling. Several catastrophic supply chain attacks, such as the global SolarWinds attack, were made possible by weaknesses in CI/CD pipeline security.

Use the principle of least privilege to ensure that each user has access to only the data and systems that are absolutely necessary for them to complete the task. Ensure CI/CD tooling uses multi-factor authentication and is closely monitored to detect anomalous behavior.

Use Penetration Testing

Automated testing can fix many security issues, but it can miss important vulnerabilities. Consider testing your application using human penetration testers. These are ethical hackers who evaluate the application from an attacker's perspective, detect vulnerabilities, find potential avenues for attack, and actually attempt to breach the application, without causing damage.

You can also perform "blind" penetration testing, conducted without the knowledge of security and operations staff, as a real-life test of your security practices and personnel.

Test Third Party Components

Modern software is assembled using a large number of third-party code components, many of them open source. Open source has many advantages, but can also expose an organization to security and compliance risks. Open source projects may not be properly maintained and may not implement secure coding practices. Even if they do, they must be regularly updated to prevent known vulnerabilities.

A common solution for scanning third-party components is Software Composition Analysis (SCA). SCA solutions scan open source components and their dependencies, identifying security vulnerabilities, and also license issues that can threaten a software development project. These solutions provide detailed recommendations that can help teams remediate issues or replace problematic open source components.

Securing Cloud Native Applications with Aqua Security

Aqua replaces outdated signature-based approaches with modern controls that leverage the cloud-native principles of immutability, microservices and portability. Using dynamic threat analysis, machine-learned behavioral whitelisting, integrity controls and nano-segmentation, Aqua enables modern application security protection across the lifecycle.

Aqua's full lifecycle security approach provides coverage for all clouds and platforms, integrating with enterprises' existing infrastructure and the cloud native ecosystem.

Secure the Build

Accelerate development by detecting security issues in your artifacts early and shortening time to remediate. "Shift left" security into the CI/CD pipeline, get full visibility into the security posture of your pipeline and reduce the application attack surface before application deployment.

Secure the Infrastructure

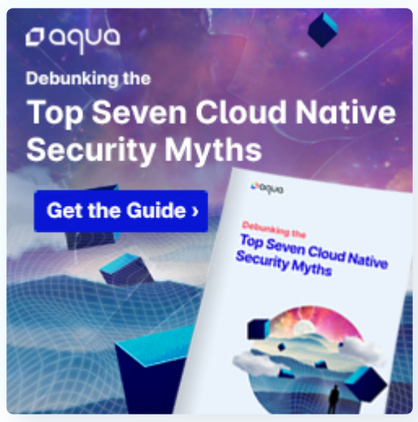
Enforce compliance across the stack, gain real-time visibility and control over your security posture. Monitor, detect, and automatically remediate configuration issues across public cloud services and Kubernetes clusters. Ensure conformity with CIS benchmarks, PCI-DSS, HIPAA, GDPR and other regulations.

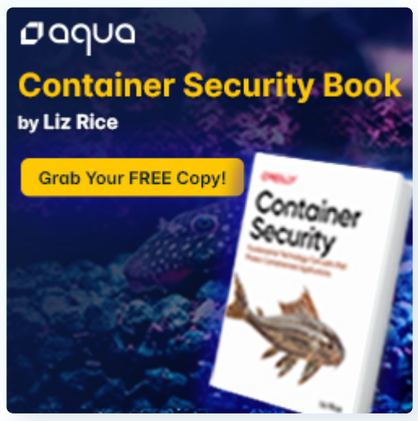
Secure the Workloads

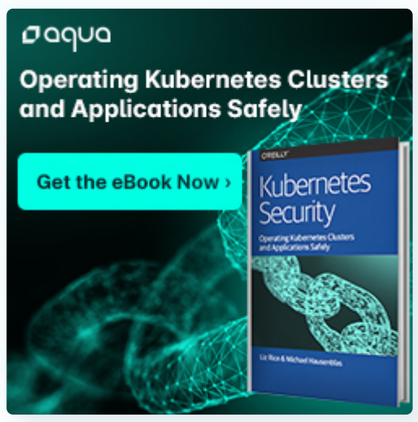
Protect applications in runtime using a zero trust model, with granular controls that accurately detect and stop attacks. Unify security across VMs, containers, and serverless on any cloud, orchestrator, and operating system. Leverage micro-services concepts to enforce immutability and micro-segmentation.

Key features:

- **Vulnerability scanning**: Scan CI pipelines and registries, container images, VM images, and functions. Find known vulnerabilities, malware, embedded secrets, OSS licensing, configuration, and permissions issues and prioritize based on potential impact
- **Dynamic Threat Analysis**: Detect and mitigate hidden malware and supply chain attacks in container images using a secure sandbox
- **Cloud Security Posture Management (CSPM)**: Continuously audit cloud accounts and services for security risks and auto-remediate misconfigurations







Aqua Blog
All about cloud native and security ›