WHAT IS STATIC APPLICATION SECURITY TESTING

kiuwan

INDEX

WHAT IS STATIC APPLICATION SECURITY TESTING?

Despite the uniqueness, beauty, or traction generated by your application, it could still pose a liability without security. To stay safe, it is essential to be aware of the security risks associated with business applications.

We have all seen the business consequences when delicate information gets compromised. And while application security isn't the most exciting aspect of developing your business applications, it is essential to integrate this into your DevOps before it's too late.

The first step to securing your applications is recognizing that there are several security risks, even if you're not as large a company as Amazon. The size of your business does not diminish its vulnerability.

According to studies by 451 research, 64% of executives around the world and 74% of executives in the US believe that adhering to compliance requirements is an effective way to secure data. But if your organization only bases its data security on compliance standards, there could be gaps in protection that eventually lead to expensive data breaches.

And despite the fact that keeping source code secure is a priority for many businesses, it's common for developers to lack the security experience to avoid insecure programming patterns or know how to use secure APIs.

DevSecOps is pushing security "to the left" in the software development lifecycle. This raises the question of how we can ensure security compliance at an early stage of the process. That's where static application security testing (SAST) comes in.

SAST (Static Application System Testing) is a commonly used Application Security tool that scans an application's source code, binary code, assembly code, or byte code. It is a white- box testing tool that identifies the root cause of specific vulnerabilities. It also helps to remediate the underlying security issues.

SAST finds vulnerabilities in applications and identifies security issues before they become serious problems. It looks for known vulnerabilities in applications and approaches applications with new data or behavior to find weaknesses before they are exploited.

Static application security testing looks at the basic infrastructure of an application and creates attack scripts that simulate how an attacker might interact with the application in various ways. Both tools share the goal of discovering security problems before they happen.

WHY IS SAST IMPORTANT?

One thing that's important in any organization where applications are widely used is static analysis. Static analysis is a type of testing that helps to prevent or discover security loopholes in an application. It is often combined with other testing methods, such as source code analysis.

With the use of a SAST solution, an enterprise can detect security vulnerabilities in code before a cybercriminal has an opportunity to exploit the weakness. SAST is important from a pragmatic perspective, too. It helps developers find and address issues early in the SDLC so that there will be less disruption to the application after release.

Static testing helps developers build secure applications from day one so that they can save costs after release. Also, defending against such sophisticated attacks is not easy. The trend of sophisticated attacks targeting the application layer to breach networks has shown that companies need to update security measures. Application testing methodologies like SAST and DAST are essential for ensuring the integrity of applications.

A security breach even once can cause many problems. The right SAST will protect your network from attacks and damage to your reputation.

Companies should be using SAST in the QA or DevOps stage of the SDLC. Doing so will enable developers to code with automated suggestions to ensure there are no glaring security issues to address further down the line.

As you can see, SAST solutions and static testing offer a variety of benefits to businesses. Let us explore these a bit more in-depth:

- **Identify source code vulnerabilities:** SAST solutions can help you identify coding vulnerabilities in source code and find issues such as SQL injection and buffer overflows. These issues could disrupt the application's service, so catching them helps you minimize exposure to external risks. When you're using SAST tools to find flaws in your code, it means you can stay up-to-date on any potential vulnerabilities and quickly fix them before they lead to a costly compromise of your security.
- **Prompt and Early diagnostics:** Source code analysis tools give the ability to view code before it has been compiled, making it ideal for running diagnostics early in the SDLC. These solutions can look at the code before it is collected and guide developers on improvements well ahead. The early early-stage of development is the best time to test and fix bugs, as it's cheaper and more accessible than after the application has been deployed. In conclusion, if you find bugs early in the development cycle you can save money by fixing them before they make it into production.
- **Locate the exact segment of code affected:** SAST tools warn you of incorrect code, but they also pinpoint where the problem is. This makes it simpler to identify and correct the root problem. Highlighting enables users to quickly see the parts of a program that are most likely to contain errors. Coders can spend more time creating solutions and less time searching for them by selecting those problem areas.
- **No test cases required:** While DAST tools will need you to decide which parts of the code you want to test, SAST tools will automatically apply all of their testing rules, evaluating each project and year of programming experience to catch vulnerabilities that you didn't even know existed. SAST follows a set of machine learning algorithms trained on popular, open-source code repositories and implemented by SAST tool creators.
- **No execution required:** SAST works on the source code before the application runs. As a result, SAST scans don't need to analyze parts of the code that won't be executed.
- **Easy automation:** Many SAST tools don't need a graphical interface or depend on GUI interaction, which is much easier to automate than DAST tools.

WHAT DO SAST TOOLS DO?

The primary goal of a SAST tool is to eliminate vulnerabilities during development, so a developer doesn't have to fix vulnerabilities after the release. SAST tools automate scanning code and help developers find vulnerabilities early in the SDLC.

A SAST software platform shows developers real-time feedback as they code. For example, the software would scan for vulnerabilities, highlighting problematic code onscreen for the developer to address.

These tools also include more features for monitoring security issues, like dashboards and reports, which allow you to see which problems have already been remediated. This lets you quickly scan the threat level of your app and respond accordingly.

OTHER TYPES OF APPLICATIONS SYSTEM TESTING

DAST: DAST, fully known as Dynamic Application System Testing, is a black-box testing method that scans applications during runtime. It is applied at the later stages of the CI pipeline testing. DAST is an effective method for avoiding regressions and does not depend on a particular programming language.

IAST: IAST is short for Interactive Application Security Testing, and it is similar to DAST in that it focuses on the application's behavior in runtime. However, while DAST is based on Black-box testing only, IAST is based on a combination of black-box testing and the scanning and analysis of internal application flows.

The advantage of IAST is its ability to combine DAST-like findings and source code like SAST. The disadvantage of the IAST method is that it causes dependence on the IAST programming language, which can only be performed later in the CI pipeline.

SCA: Software composition analysis focuses on the third-party code dependencies used in the application. SCA is well suited for software and applications that are using multiple open- source libraries. Like IAST, SCA is programming language-dependent.

Each one takes a different approach to detect vulnerabilities.

For professional application security testers, there are two dominant methods for application security testing. And the standard procedure is to use either static application security testing or dynamic application security testing.

The most significant difference is that SAST occurs at the beginning of the SDLC, and DAST appears while an application is running. DAST solutions attempt to penetrate applications to discover vulnerabilities located outside of the code or inside third-party interfaces.

Security testing tools are designed to test the security of applications without needing to use the source code or a particular application framework. A penetration test finds the vulnerabilities like validation issues that cybercriminals are most likely to target (because most attackers won't have access to the source code).

Professional programmers and developers may need tools that can highlight more specific code problems. However, they are more likely to be expensive to maintain. You should weigh the pros and cons of each type of tool for your own company's needs.

HOW CAN RISK BE ASSESSED?

The OWASP (open web application security project) "is an open community dedicated to enhancing the capability of organizations to develop, purchase and maintain applications and API's that can be trusted." They watch for major threats and guide developers around the world. In providing this guidance, they released a list of 4 basic dimensions to consider when analyzing security risks for business applications. These four basic dimensions include;

- **Exploitability:** This rates the difficulty encountered by the hacker when trying to exploit your application. Is your application easy or challenging to exploit?
- **Prevalence:** Prevalence measures how often this attack vector is used.
- **Ease of detection:** As the name implies, it is the level of difficulty to recognize a threat to this attack vector.
- **Technical Impact:** And the technical impact is a measure of the damage that can be caused by the attack.

According to OWASP, The most prevalent security risks for applications are:

1. Injection flaws

An injection flaw occurs when a hacker sends untrusted data into a system by hijacking one of these commands. Injection flaws are common, easy to exploit, and can severely compromise your system.

A code injection vulnerability occurs when data received from a user is not validated correctly or sanitized before being processed by the application, allowing attackers to inject malicious code into the system.

Any code that accepts input from a user can potentially be exploited via input injection attacks. The solutions to injection flaws are

- Separate data from the web application logic.
- Implement restrictions to reduce data exposure in case of successful injection attacks.
- Update your applications and constantly look for bugs in your code or vulnerabilities.

2. Broken authentication:

Broken authentication happens when authentication is weak or damaged, thereby allowing hackers to assume someone's identity in the system. By pretending to be an authenticated user, they can wreak all kinds of havoc.

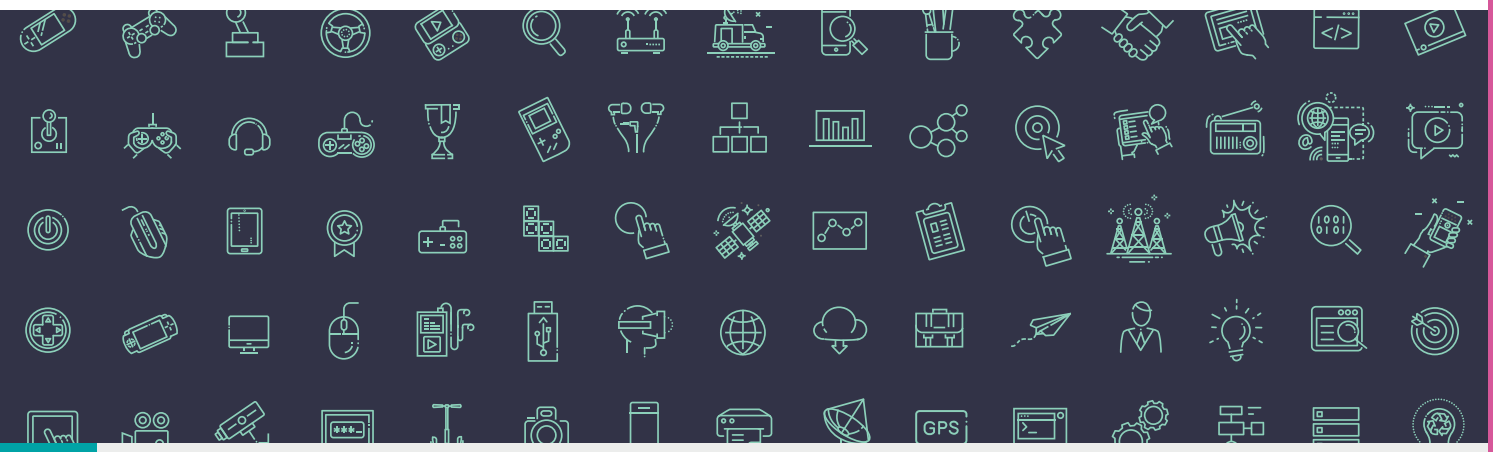
A broken authentication vulnerability can allow an attacker to take over any account they want in a system – or even worse – compromise the entire system. An authentication vulnerability usually refers to logic issues that occur on the application authentication's mechanism, like bad session management prone to username enumeration – which is when a malicious actor uses brute-force techniques to either guess or confirm valid users in a system.

One of the most common protective measures against broken authentication is to set up multi-factor authentication whenever possible.

3. Sensitive Data Exposure:

The most prevalent attacks in the last five years were related to the exposure of too much sensitive data. Applications that pass sensitive information via sessions, URLs, or poorly written code, raise the risk of this vulnerability. Some examples of sensitive data that requires protection are:

- Credentials
- Medical information
- Social Security Numbers
- Personally identifiable information (PII)
- Credit card numbers
- Other personal information



One solution for this attack is to keep an eye out for mismatched keys, indicating a difference between two redirect links. Other ways to prevent data exposure, according to OWASP, are:

- Classifying data processed, stored, or transmitted by an application.
- Identifying which data is sensitive according to privacy laws, business needs, or regulatory requirements.
- Applying controls according to the classification.
- Not storing sensitive data unnecessarily.
- Discarding sensitive data as soon as possible or using PCI DSS compliant tokenization or even truncation. Only retained data can be stolen.
- Making sure to encrypt every sensitive data at rest.

4. Broken access control:

An access control system refers to a set of rules designed to permit or deny users access to specific information or functions based on the information.

Broken access control can be exploited by a malicious person to bypass authorization and perform tasks as though they were a privileged user. A user could bypass authorization checks by modifying a URL parameter. For example, they could change out the 'account_id' with the ID of another user without any other verification.

Here are some examples of what we consider to be "access":

- Access to an administrative panel/hosting control
- Access to a server via SFTP/ FTP / SSH
- Access to other applications on your server
- Access to your database

A web application can use authorization tokens to set up access controls, and then tighten these controls.

5. Security misconfiguration:

A security misconfiguration is often the result of using default configurations or displaying excessively verbose errors because an application could reveal vulnerabilities in the application.

Misconfiguration can happen at any level of an application stack, including:

- Network services
- Custom code
- Platform
- Storage
- Web server
- Pre-installed virtual machines
- Application server
- Database
- Containers
- Frameworks

6. Insecure deserialization:

This attack involves a threat agent exploiting vulnerable applications that frequently serialize and deserialize data. Serialization and deserialization are both ways of taking an object from the application and converting it to a format that can be used for another purpose.

Serialization exploits like insecure deserialization attacks can lead to severe consequences like DDoS attacks and remote code execution breaches. While steps can be taken to try and monitor serialization and implement type checks, the only sure way to protect against insecure deserialization is by prohibiting the deserialization of data from untrusted sources.

7. XML external entities:

A web application that parses XML input can be exploited by a user entering an external entity, which allows the storage unit to be accessed directly by an attacker.

The default behavior of most XML parsers is vulnerable to XXE attacks. This is the main reason why developers need to be careful about ensuring their applications are not weak. According to the OWASP Top 10, the XML external entities (XXE) main attack vectors include the exploitation of:

- Vulnerable XML processors if hackers can upload XML or include hostile content in an XML document
- Vulnerable code
- Vulnerable dependencies
- Vulnerable integrations

8. Using components with known vulnerabilities:

Most web developers use components like libraries and frameworks in their web applications. These components are software components that help developers avoid redundant work and provide needed functionality; examples include front-end frameworks like React and smaller libraries that add sharing icons or a/b testing.

Some attackers look for weaknesses in these components that they can exploit. Some of the more common features are used on hundreds of thousands of website applications, so a hacker finding a security hole in one of these components could leave hundreds of thousands of sites vulnerable to attack or exploitation.

Component developers frequently release security patches and updates to cover up known vulnerabilities, but web application developers don't always have the patched versions of components running on their applications.

To avoid the risk of having only the outdated or vulnerable components running on your applications, developers should remove dormant components from their projects while ensuring that they receive components from a trusted and up-to-date source.

9. Insufficient logging and monitoring:

Many web applications do not take enough steps to detect data breaches. The average web developer response time for a breach is around 200 days after it has happened. This long period gives attackers a lot of time to cause damage before any form of response occurs.

However, OWASP recommends that web developers implement logging and monitoring and incident response plans to make sure that they are aware of attacks on their applications.

10. Cross-site scripting (XSS)

This security flaw allows attackers to insert malicious scripts into your website, using your website as a platform to spread the attack and compromise other websites. This allows the attacker to inject content into a website and modify how it is displayed, forcing the victim's browser to execute the code provided by the attacker while loading the page.

XSS affects about two-thirds of all applications.Generally, XSS vulnerabilities require some interaction by the user to be triggered. If an XSS vulnerability is not patched, it can be hazardous to any website.

HOW TO INTEGRATE SAST INTO DEVSECOPS

To efficiently use the SAST tool, you should

- Onboard each and every application. This is a one-time task to be performed by a security analyst, along with input from the development team.
- Scan and audit results. It is appropriate to ensure that you have all the source code and libraries before you begin the scan process.
- Review the results using the enterprise server or in the SAST IDE. During the audit process, decide what bugs will be fixed, what bugs are not high priorities, and the results that are false positives.
- Customize the rulesets. Having the knowledge of the application onboarded and the triaged results, you may want to customize the rulesets at this point.

After you have onboarded completely, you get the IDE plugins in the pre-commit checks to enable your developers to have access to the tool. Thus, they can find and fix issues as they occur.

Then, with the same set of rules, the client's top 10 can be run during the commit-time checks. A few examples of these checks are;

- SQL Injection
- Configuration review
- Session management
- Hard-coded credentials
- Resource leaks

The OWASP Top 10 are configured during build time. Some examples include

- Information leakage and error handling
- Insecure cryptographic storage
- Command injection
- Denial of service
- Path manipulation
- Weak encryption

Finally, the comprehensive rule sets are configured during test time. Some of the comprehensive rulesets are;

- XML injection
- DOM XSS
- XPath injection
- Open redirect
- Cookie injection
- XML external entity

The broader the set of rules you run, the longer it takes for the tool to complete the scan. That's one of the reasons to try to divide and conquer the rules you run at each phase of the DevSecOps pipeline.

Once these steps are completed, you will have good coverage of all your SAST rules. Depending on the language, technology, architecture, and framework you use, you will have to carefully configure your rules and also be willing to write custom rules.

HOW TO CHOOSE THE RIGHT STATIC APPLICATION SECURITY TESTING (SAST) TOOL FOR YOUR ORGANIZATION

The SAST market is full of offerings, often combined up with additional solutions, thus, making it a challenge to find the right fit for your organization.

The OWASP's list of criteria for selecting the right SAST tools can help companies narrow their options and choose the solution that better fits their AppSec strategies. These criteria include:

- **Language support:** If you use many different programming languages in your organization, make sure the SAST tool that you use offers complete coverage for those languages.
- **Vulnerabilities coverage:** An organization's SAST tool should cover at least all of OWASP's Top Ten web application security vulnerabilities.
- **Accuracy:** While there are false-positive and false-negative cases for SAST, it's crucial to find out how accurate your organization's tool is.
- **Compatibility:** Just like any other automated tool, the SAST tool you use must be supported by the frameworks you're already using so that it integrates easily into your SDLC.
- **IDE Integration:** After integrating a SAST tool into your IDE, you can spend less time bringing developers up to speed on integrating the security testing into their development workflow.

- **Easy Integration:** If you're using the SAST tools built into your IDE or a standalone solution that talks to your IDE, then your team may be able to integrate scanning into their existing DevOps practices more easily.
- **Scalability:** A SAST tool can scan on a small sample project and deliver similar results on larger projects. Your solution's cost can be directly affected by the scale it has to deal with. A good rule of thumb is to review how other similar solutions deal with scale.

THE CONS OF STATIC APPLICATION SECURITY TESTING

SAST increases security but at the cost of slowing down the development process. It is a high-cost approach that should be adopted only when the benefits are clear. These kinds of tradeoffs often involve a risk-reward calculation. In this case, higher developer productivity from skipping doing tedious code reviews is balanced against the disabling flaws that a static analysis tool like SAST could have found.

Unfortunately, the false positives and the need to further analyze SAST results are significant barriers to agility. Nowadays, software development teams are faced with many issues. As software development cycles become shorter, the risk of attacks to the application layer continuously increases, and development teams find it challenging to combine security and speed.

A company must find the right balance between finding all weaknesses and minimizing risk. If a company tries to find every flaw, there is more chance of breaking the product, but customers are at risk if there isn't enough time to find weaknesses. And that is the problem KIUWAN is solving.

KIUWAN: THE BEST SAST TOOL IN THE MARKET TODAY

Kiuwan is an all-in-one SAST solution that can be used in the software development process to speed up and reduce time commitments. Kiuwan is a global company that is providing solutions to the problem of application security platforms. Our tools are strategically designed to assist your team in identifying areas of vulnerabilities in your application code security.

With Kiuwan, we can create an action plan to deliver on the following:

- **Fast vulnerability detection:** Time is critical in application security testing, and so is the need for an easy and instant setup. With Kiuwan, you can scan and get the results immediately.
- **DevOps Approach to Code Security:** This is one of the significant benefits of Kiuwan because you can integrate Kiuwan with your CI/CD/DevOps pipeline. This will ensure that you can automate your security processes.
- **Flexible Licensing options:** Kiuwan offers various options to pick from: we offer one-time scans and continuous scanning. You can also choose between the SaaS model or the on-Premise model.
- Security developers can also create customized reports for their projects with our security automation tools; they can export them offline and track them in dashboards. Tracking all the security issues reported by our automated tools in an organized way can help developers find the bugs and fix them faster.

The tools that help you test your software are a critical part of DevSecOps processes. Automating them is an integral part of building a sustainable program, as it drives efficiency,consistency, and early detection.

Our tools implement automated testing, reporting, and security scanning, and manual QA/testing. They allow for a cost-effective, proactive, and secure DevOps process.

KIUWAN PRODUCTS

At Kiuwan, we have features and functionality for all stakeholders and every stage in the software development life cycle. Some of these products are

- Code Security (SAST): With our SAST tool, you can automatically scan your code to locate and remediate vulnerabilities. We are compliant with the most strict security compliance standards, and these include OWASP and CWE. The Kiuwan Code Security-SAST covers all essential languages and integrates directly with important DevOps tools across the software development life cycle (SDLC). Try a Kiuwan Demo.
- Insights (SCA): It's essential to manage your open source risk, and our insights tool helps limit risk from third-party components, ensure license compliance and remediate vulnerabilities. Additionally, you can automate policies throughout the SDLC.

Kiuwan Modules make it an all-in-one solution with

- **Code Analysis (QA):** With Kiuwan, you can identify defects in your code based on industry-standard characteristics such as maintainability, efficiency, portability, and reliability.
- **Lifecycle:** You can also audit your software deliveries from external and internal providers, define unique checkpoints and compare the modifications. Kiuwan integrates directly into your development environment.
- **Governance:** Lastly, you can manage your application portfolio across different languages; this enables you to understand your business risks. Additionally, our tools provide predictive analysis for objective decision-making.

OVER 18,000 USERS ACROSS THE GLOBE TRUST KIUWAN, AND YOU CAN FIND MORE INFORMATION ABOUT US HERE:

TRY KIUWAN CODE SECURITY FOR FREE

KIUWAN.COM/FREE

CONTACT US

CONTACT@KIUWAN.COM
LIVE CHAT: KIUWAN.COM

BECOME A PARTNER

PARTNERS@KIUWAN.COM

kiuwan