The No BS Guide to Static Application Security Testing (SAST)

A WhiteHat Security Best Practice Guide

Executive Summary

While board level executives understand the concepts and terms used in Network Security or Perimeter Security, Application Security, as a concept and discipline, is not quite firmly defined. AppSec is much different than other security domains, so applying standard methods from them don't necessarily address the unique challenges that AppSec can bring. An effective solution relies heavily on process diligence in combination with ongoing training and developer guidance within the development and security organizations.

We don't have to look very far to see that applications have been under full-blown frontal assaults. There have been numerous data breaches recently – Equifax, Sonic, and Whole Foods, among many others – most of which breached an AppSec vulnerability. Network security is everywhere. We have all been hyperaware of securing the perimeter and having our firewalls on high alert at all times. Now, application vulnerabilities are being exploited and it's time to do something about it.

It is our intention to give you the right information to educate your peers, colleagues, and executives about initiating a successful application security program, including:

- Make application security visible to security and development organizations.
- Provide guidance for building and managing application security processes.
- Measure and manage application security risks and processes.
- Prioritize vulnerability remediation based on risk exposure to the business.
- Institute application security training for developers and managers.
- Assure compliance of applications with security regulations for privacy, data protection and information security.

Application Security Terms and Concepts

In application security terms, what exactly do we mean by "application"? Everyone understands two basic categories – desktop applications and mobile applications. Some examples include, Notepad, Chat Client, Firefox, and Chrome (Desktop applications), and Waze, Kindle, and Candy Crush (Mobile applications).

What do we mean by "web application"? You often use a desktop application (a browser) to get to a web application like Gmail, or Dropbox, or logging into your bank account, or your healthcare provider's website. Websites usually have multiple applications within them.

This is a Website. It is full of apps.

Large organizations can use over a thousand applications within their environment. Some can be in the tens of thousands. Many of these are on their website, or touch their website in some way.

There are divided responsibilities for application security. Responsibility to patch the client (web browser) is on the user (or if at work, the IT department). The responsibility to secure the web application lies with the organization who owns the application. This is where application security comes in.

The Tools of Application Security

APPLICATION SCANNING
This can be done in production, or on source code. Finding the vulnerabilities and prioritizing them by criticality of both impact and asset is the key. This is the first step for an AppSec program.

WEB APPLICATION FIREWALLS (WAF):
A WAF is a specific kind of firewall, or a firewall with special capabilities that limits access to the computer's operating system by specific applications, such as limiting the execution of files, or handling of data. It can sense attacks.

RASP (RUN-TIME APPLICATION SELF-PROTECTION):
Built or linked into an application or the application environment, which controls application executions. It can sense attacks.

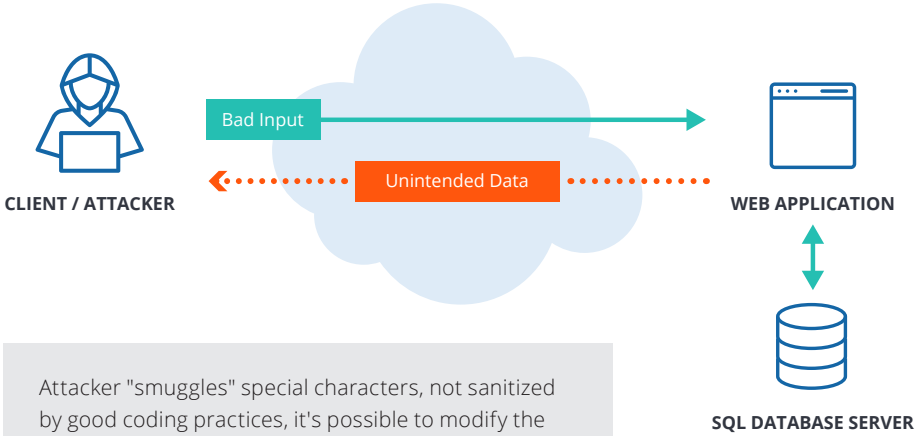
DATA-LOSS PREVENTION SOLUTIONS:
DLP usually refers to software products that let an administrator control which data an end user can transfer, move, or download.

ENCRYPTION:
This means scrambling data at rest and in motion, ideally including login information within a system.

VIRTUAL PATCHING:
As you know, a patch is a quick repair job on part of a program. Typically, a patch is developed and distributed as a replacement for, or insertion into, compiled code. A virtual patch mitigates the risk of a vulnerability by combining application scanning results (a list of vulnerabilities) with a WAF to create a temporary band aid protecting the application from exploits. This is a great thing for zero day threats, new releases, and legacy systems that cannot be patched without a lot of down time, or third-party applications running on your website.

The challenge is often that once a virtual patch is created, an organization (specifically, their development team) feels a reduced motivation to produce a permanent fix. A virtual patch doesn't eliminate defects in an application, and cannot address all possible methods or locations in which an exploit can be launched.

Consider the SQL Injection



Attacker "smuggles" special characters, not sanitized by good coding practices, it's possible to modify the SQL queries, their logic, and the applications behavior.

SQL injections are one of the most common AppSec vulnerabilities, because they are also one of the easiest coding mistakes to make. Perhaps it is because most programmers design and build their applications based on the intended behavior of a customer or user, not a hypothetical attacker. It is a mindset in development that is slowly changing to the more appropriately suspicious, but there is a gap in education and what secure coding looks like, and that still to be addressed.

What is Static Application Security Testing? (SAST)

Let's take the OWASP definition: Static application security testing (SAST) tools are designed to analyze source code, as well as compiled and/or uncompiled code, in order to find security vulnerabilities. They scale well, can be run on lots of software, and can be run repeatedly (as with nightly builds or continuous integration). SAST is especially useful for automatically finding critical software flaws such as Cross Site Scripting (XSS) and SQL Injection flaws. Output tends to be good for developers, as it highlights the precise source files, line numbers, and even subsections of lines that are affected.

See the following for more info:

WhiteHat Sentinel Source
WhiteHat Software Composition Analysis

The Truth About False Positives

Here at WhiteHat Security, we receive a lot of questions about what constitutes an ideal static analysis (SAST) solution, the importance of depth of coverage, and some causes of false positives – how they come up, why they happen, and what can be done about them. The solutions architects and security engineers from our Threat Research Center have even been called onsite to companies who need help deciphering reports of false positives.

What you're about to read may be surprising, but we feel it necessary to clear up some confusion regarding source code scanning, language support and how to handle false positives.

While there are exceptions, the fact is, in order to rapidly develop support for 20+ languages, some companies have built assumptions into their scanners, and most of the time, those assumptions are wrong. The output of the scanner and vulnerabilities may look legitimate, but they're not – in our experience in reviewing reports from other companies offering application security technology, when some of the code doesn't make sense to their scanner, or when code or dependencies are missing, those scanners assume that the code is vulnerable. This creates a large number of false positives that resource-stretched security and development teams often struggle to verify as false positives.

Companies should look for scanning technology that is not programmed with assumptions, and ideally, a solution that is a combination of technology and managed services to scale their application security program. At WhiteHat, our research and development teams take the time to thoroughly review all supported languages, frameworks, and libraries to encode their security properties into our scanner. This ensures that Sentinel Source reviews each part of the code quickly, efficiently, with accurate

results. Often, algorithms are built on a simple UML diagram convergence model, in which the correct location to fix a vulnerability is assumed to be where all of a vulnerability's data-flows come together.

UML (Unified Modeling Language) is a general purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system. However, this approach is only applicable in the simplest cases. Often in enterprise software, the location highlighted by this method will not only be the wrong location to fix the vulnerability, but it will break the application functionality.

What's the Correct Way to Handle This?

WhiteHat's Sentinel Source uses patented Positional Analysis Technology that identifies the correct location to fix a vulnerability without affecting application functionality and when combined with our patented directed remediation technology, it generates a patch file that immediately secures the code.

It's also important to question claims that "no configuration is needed". This ignores the reality that source code scans need to be scoped according to the architecture and data boundaries of the application or platform that is being assessed. Scanning a repository or archive of code without taking these important factors into account can lead to false positives, incorrect risk and threat assessments as well as a false sense of confidence in the coverage of the scanner due to the assumption that the scans must be working correctly, if they are producing a large number of findings.

Sentinel Source leverages the expertise of WhiteHat's Threat Research Center who review all scan configurations to ensure that the scan is setup to accurately reflect the architecture and data boundaries of the application or platform being scanned. The Threat Research Center will make recommendations and assist in improving scan configuration to achieve the best coverage and most efficient scanning.

What to Look for in a SAST Solution

SAST Vulnerability Remediation

Time-constrained developers want to fix software bugs quickly to meet tight schedules. The three main options for integrated speed vulnerability remediation are open-source point solutions, vendor point solutions, or a managed service. There are compelling reasons to evaluate using a SAST platform like WhiteHat Sentinel Source, instead of using a point solution to run SAST scans.

In our experience, some developers are unqualified to validate the results such open source or unverified vendor point solutions provide, or to evaluate the real risk of the findings. Often, they will "curve fit" their solutions through trial and error changes, producing "fixes" that will fool the scanner they use, but not solve the underlying issue. Developers should only commit fixes that they know will resolve the issue, not guess and hope the scanner no longer finds the issue.

Access to Security Experts is Paramount

If the company you're considering limits access to security experts or only works with a long list of external service providers, question whether or not your organization can reach your goals of managing an application security program. Often times, these companies' offerings will get hugely expensive when you need assistance, which may or may not be feasible for your security budget. The total cost of both the technology and managed services should be simple and transparent. This makes for a much better and predictable experience.

Software Composition Analysis

It's estimated that between 70 to 90 percent of source code is composed of open source components. If we take this into consideration, doesn't it make sense that SCA should be included in a SAST solution? Yet, most companies do not offer SCA within their SAST, and in our view, this is backwards. We've all experienced times when there are numerous tools stacked on top of each other, that don't always integrate, and it can be hugely disruptive to the process and to the results obtained. WhiteHat's Sentinel Source comes with integrated SCA, at no additional charge.

To learn more about WhiteHat's Sentinel Source and Software Compositon Analysis, please contact your account manager or contact WhiteHat directly at sales_dev@whitehatsec.com.