Application Security Testing (AST)

Buyer's Guide

GSA January 2023 Version 1.0

Table of Contents

1 Executive Summary 1
2 Purpose 2
3 Audience 3
4 Scope 3
5 What is AST? 4
6 Why is AST Important? 4
7 Automated Testing Versus Manual Testing 7
7.1 Automated AST 9
7.1.1 Static Application Security Testing (SAST) 10
7.1.2 Dynamic Application Security Testing (DAST) 11
7.1.3 Interactive Application Security Testing (IAST) 12
7.1.4 Mobile Application Security Testing (MAST) 13
7.1.5 Software Composition Analysis (SCA) 14
7.2 Manual AST 15
8 Red Team Application Security Exercises 16
9 Implementation Considerations 17
9.1 AST Program 17
9.2 AST Methodologies 17
9.3 Integrating AST Across the SDLC 18
9.4 Application Security Testing Best Practices 18
9.5 Implementation Challenges 19
10 Other Considerations 20
10.1 Reporting Requirements 20
10.2 AST Delivery Models 20
10.3 AST Delivery Platforms 21
11 Choosing the Right AST Tool 21
12 Selecting an Independent Third-Party Application Security Tester 23
13 Crowdsourced Security Services 26
14 AST Buyer's Guide Contact Information 27

Appendix A – Glossary 28
Appendix B – GSA-Offered Products, Services, and Solutions for AST 31
Appendix C – References 34
Appendix D – Generic Testing Services Checklist 35

Foreword

This guide is intended to assist an agency in acquiring Application Security Testing (AST) products, services, and solutions. In order to effectively eliminate, reduce, and mitigate the overall risks from the application attack surface, an agency should implement a dedicated AST Program as part of its overall Development, Security, and Operations (DevSecOps) process.

Within DevSecOps, it is understood that each agency must start the process of implementing AST from its perspective. And based on its current DevSecOps maturity, an agency will address the most critical and foundational aspects of application security to address its own unique needs.

There is no cookie-cutter solution for AST, and every agency has a different approach to vetting products, services, and solutions throughout the Software Development Life Cycle (SDLC). Finding a best practice approach for improving AST requires adopting a holistic view of the application risk landscape, including the specific access and deployment models used for the application, such as whether the application is deployed on-premises or in the cloud, and considering how critical the application is for continued operations.

The information provided in this guide can help identify a broad range of products, services, and solutions to help develop, implement, and mature an agency's AST Program strategy. The General Services Administration (GSA) Information Technology Category is available to answer any questions and provide subject matter expertise related to any aspect of this guide and other Information Technology (IT) needs.

1 Executive Summary

In response to incidents such as the Colonial Pipeline and Solar Winds attacks, on May 12, 2021, President Biden signed E.O. 14028, "Improving the Nation's Cybersecurity." The E.O. directs Federal agencies on advancing security measures that drastically reduce the risk of successful cyber attacks against the Federal government's digital infrastructure. On January 26, 2022, OMB released the "Federal zero trust architecture strategy" in OMB Memorandum M-22-09, "Moving the U.S. Government Toward Zero Trust Cybersecurity Principles," in support of E.O. 14028.

OMB Memorandum M-22-09 "Federal zero trust architecture strategy" describes five (5) complementary areas of effort (pillars): Identity, Devices, Networks, Applications and Workloads, and Data. Under the Applications and Workloads pillar, the memo outlines six (6) actions Federal agencies need to take to improve application security. Specifically, agencies must operate dedicated application security testing programs and use highly-qualified firms specializing in application security for independent third-party evaluation.

A June 2022 Forescout report[1] demonstrated that many applications are insecure by design due to the persistent absence of basic security controls. When prioritizing between security concerns and market pressures to deliver new and innovative products, application vendors often choose to support faster growth and enhanced user experience over security concerns. A reduced emphasis on security testing frequently leads to applications being susceptible to exploits by threats actors.

Despite various efforts by the Cybersecurity and Infrastructure Security Agency (CISA)[2] and the National Institute of Standards and Technology (NIST)[3] to bring greater attention to application vulnerabilities, insecure-by-design practices are still very much the norm. As such, security practitioners supporting a government agency should focus more on AST.

AST is often either overlooked by an agency due to budget limitations or minimized because a choice is made to focus more on traditional network security operations. However, an agency must realize that application hacking, not IT compromise, is the number one attack vector exploited by threat actors.[4]

---

[1] Dos Santos, Daniel. "OT:ICEFALL – How to Tackle a Decade of Insecure-by-Design Practices in OT." Forescout, 20 June 2022, https://www.forescout.com/resources/ot-icefall-report/
[2] Examples include CISA Build Security In initiative, CISA Shields UP initiative, publication of Known Exploited Vulnerabilities Catalog, and the National Cyber Awareness System (NCAS).
[3] NIST Cybersecurity Framework and various Special Publications such as SP 800-95, *Guide to Secure Web Services*.
[4] https://blog.shiftleft.io/threat-actors-focus-on-the-application-layer-do-you-3a74c714825b

According to the Verizon 2022 Data Breach Investigations Report[5], application hacking is the number one attack vector, involving approximately 70% of all incidents and breaches. Historically, a typical government agency would spend most of its cybersecurity budget on traditional network security technologies such as firewalls and intrusion detection systems. These technologies serve a purpose in developing a layered defense, but this approach does not address the root cause of the modern-day threat applications released with detectable and correctable vulnerabilities that are subject to exploitation.

This traditional budget allocation may have been appropriate if an agency relies on third-party software and services to secure their daily operations, but in this current era, where applications are now the main attack vector exploited by threat actors, the budget allocation for AST should receive greater priority in the budget to address the primary risk of compromise. As evidence of this priority shift, the Forrester Analytics: Application Security Solutions Forecast, 2020 To 2025 (Global), reported 63% of security decision-makers expect their application security budget to increase.[6]

AST must be part of an agency's layered approach to application security, and as such, an agency should prioritize AST in its budget to fund the necessary AST activities to develop and maintain applications securely and efficiently. This is especially important as agencies move towards zero trust architectures pursuant to Executive Order (E.O.) 14028 and Office of Management and Budget (OMB) Memorandum M-22-09, which envisions an increased number of applications being exposed to the Internet.

This buyer's guide has been developed by GSA in accordance with the responsibilities under OMB Memorandum M-22-09. GSA is responsible for developing rapid procurement vehicles for AST products, services, and solutions.

2 Purpose

The purpose of this buyer's guide is to assist customer agencies with acquiring AST products, services, and solutions that align with their responsibilities as mission, business, and system owners and operators.

This guide identifies key features and capabilities of AST products, services, and solutions agencies should consider during their AST evaluations to:

- Reduce the number of vulnerabilities in released applications

[5] https://www.verizon.com/business/resources/reports/dbir/
[6] https://www.forrester.com/report/forrester-analytics-application-security-solutions-forecast-2020-to-2025/RES176225

- Mitigate the potential impact of the exploitation of undetected or unaddressed vulnerabilities
- Identify and address the root causes of vulnerabilities to prevent future recurrences
- Provide greater insight into the agency's application security posture

Choosing an AST product, service, or solution is a critical, but challenging task, especially with the multitude of options available. AST is an essential part of any DevSecOps program. In keeping with Systems Security Engineering (SSE) principles[7], applications should be built secure from the start; efficiently maintained throughout the entire SDLC; and agency advocates, application system owners, and key stakeholders should be encouraged to take a more active role in AST.

This guide aims to provide basic knowledge of the AST offerings available on the GSA contract procurement schedule.

3 Audience

This buyer's guide is for mission and business owners, system owners, acquisition personnel, software architects, developers, testers, and cybersecurity professionals who seek to implement or improve their AST capability. Familiarity with DevSecOps and software quality concepts is recommended, along with a basic knowledge of SDLC models and methodologies.

4 Scope

There is an emerging conversation within the cybersecurity community surrounding DevSecOps and Security, Development, and Operations (SecDevOps) and what, if anything, defines and distinguishes one from the other. While the overall goal to produce more secure applications might be the same, the approaches are quite different in both practice and philosophy.

- **DevSecOps:** A software development methodology primarily concerned with integrating security processes into DevOps cycles while maintaining efficiency.
- **SecDevOps:** A software development methodology whose main priority is given to the security of the application.

Regardless of approach, application security is not about eliminating risks, but managing risks in a manner that protects both data and delivery schedules. Organizations ultimately find that by moving to DevSecOps and including security at each step of the SDLC, their applications become more stable, require less patching, and can be released on a faster cycle. DevSecOps is a business enabler, not an insurance policy. While the SecDevOps methodology might offer more protection,

[7] Systems Security Engineering (SSE) Project | CSRC (nist.gov) https://csrc.nist.gov/Projects/systems-security-engineering-project

the costs of that protection are significant. Obviously, there are reasons for each approach. When choosing between SecDevOps and DevSecOps approaches, decisions must be made carefully.

Throughout this buyer's guide, GSA has chosen to reference DevSecOps because it can be concluded this methodology is preferred as it includes AST across each step of the SDLC. However, this guide can also be useful to the SecDevOps approach.

5 What is AST?

AST is the process of testing, analyzing, and reporting on the security level of an application as it moves through the SDLC. AST makes applications more resistant to security threats by identifying security weaknesses and vulnerabilities in the source code. AST can be static, dynamic, or interactive, and it can be manual, automated, or a combination of both. Most organizations use a combination of several AST methodologies.

Additionally, the same AST methodologies applied to traditional information technology applications can also be applied to most software applications that interact with industrial control systems or operational technology (OT) systems. These OT-related applications include, for example, human-machine interface software, control application software, and data historians. However, broader product testing is required for complete OT systems.[8]

6 Why is AST Important?

AST is important because vulnerabilities in software applications are common. According to Forbes[9], it has been reported that 84% of security incidents happen at the application layer of the Open Systems Interconnection (OSI) model[10]. Therefore, it is essential to identify application security flaws early in the SDLC to ensure that vulnerabilities in the application are remediated or mitigated before they are exploited.

The attack surfaces of modern application architectures provide attackers a multitude of potential ways to exploit vulnerabilities and compromise systems. However, most attackers tend to exploit publicly known, recently disclosed, and often dated software vulnerabilities against broad target sets, including public and private sector organizations worldwide.

In response to these types of exploits, on an annual basis, various organizations track commonly reported, frequently identified, routinely exploited, and the most targeted security vulnerabilities.

---

[8] International Society of Automation, "ISASecure – Quick Start Guide: An Overview of ISASecure Certification." Available at: <0920-ISASecure-Certifications-Guide-FINAL.pdf >.
[9] Clark, Tim. "Most Cyber Attacks Occur from This Common Vulnerability." Forbes, 10 March 2015, https://www.forbes.com/sites/sap/2015/03/10/most-cyber-attacks-occur-from-this-common-vulnerability/?sh=3509dffb7454
[10] Cloudflare Article, "What is the OSI Model?," https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/

There are two organizations which specifically track the application vulnerabilities attackers are likely to exploit. These organizations are:

- Open Web Application Security Project (OWASP) Foundation
- The MITRE Corporation (MITRE)

The OWASP Top 10 list tracks the most critical web application security risks whereas MITRE's Common Weakness Enumeration (CWE) Top 25, sponsored by CISA, tracks the most dangerous software weaknesses. Both are demonstrative lists of the most common and impactful application security risks. These weaknesses are dangerous because they are often easy to find, exploit, and allow adversaries to completely take over a system, steal data, or prevent an application from working.

Table 1 details the OWASP Top Ten list of web application security risks for 2021.

Table 1 – OWASP Top 10 for 2021

Rank	Name	Description
1	Broken Access Control	Broken access control refers to vulnerabilities that enable attackers to elevate their own permissions or otherwise bypass access controls to gain access to data or systems they are not authorized to use.
2	Cryptographic Failures	Cryptographic failures refer to vulnerabilities caused by failures to apply cryptographic solutions to data protection. This includes improper use of obsolete cryptographic algorithms, improper implementation of cryptographic protocols, and other failures in using cryptographic controls.
3	Injection	Injection flaws enable attackers to submit hostile data to an application. This includes crafted data that incorporates malicious commands, redirects data to malicious web services, or reconfigures applications.
4	Insecure Design	Insecure design includes risks incurred because of system architecture or design flaws. These flaws relate to the way the application is designed, where an application relies on processes that are inherently insecure. Examples include architecting an application with an insecure authentication process or designing a website that does not protect against bots.
5	Security Misconfiguration	Security misconfiguration flaws occur when an application's security configuration enables attacks. These flaws involve changes related to applications filtering inbound packets, enabling a default user ID, password, or default user authorization.
6	Vulnerable and Outdated Components	Vulnerable and outdated components relate to an application's use of software components that are unpatched, out-of-date, or otherwise vulnerable. These components can be a part of the application platform, as in an unpatched version of the underlying OS or an unpatched program interpreter. They can also be part of the application itself as with old application programming interfaces or software libraries.

Table 1 – OWASP Top 10 for 2021		
Rank	**Name**	**Description**
7	Identification and Authentication Failures	Identification and authentication failures encompass authentication weaknesses, including flaws that enable credential stuffing and brute force attacks, or that lack support for multi-factor authentication and invalidation of expired or inactive user sessions.
8	Software and Data Integrity Failures	Software and data integrity failures cover vulnerabilities related to application code and infrastructure that fails to protect against violations of data and software integrity. For example, when software updates are delivered and installed automatically without a mechanism like a digital signature to ensure the updates are properly sourced.
9	Security Logging and Monitoring Failures	Security logging and monitoring failures include failures to monitor systems for all relevant events and maintain logs of these events to detect and respond to active attacks.
10	Server-Side Request Forgery	Server-side request forgery refers to flaws that occur when an application does not validate remote resources users provide. Attackers use these vulnerabilities to force applications to access malicious web destinations.

Table 2 shows the top 10 (of the 25) most dangerous software weaknesses items on the MITRE's CWE list.

Table 2 – MITRE's 2022 CWE Top 10 (of the 25) Most Dangerous Software Weaknesses		
Rank	**Name**	**Description**
1	Out-of-bounds Write	Software that improperly writes past a memory boundary can cause data corruption, system crash, or enable malicious code execution.
2	Cross-site Scripting	Improper neutralization of potentially harmful input during webpage automation enables attackers to hijack website users' connections.
3	SQL Injection	Software that does not properly neutralize potentially harmful elements of a SQL command. These flaws enable attacks against databases.
4	Improper Input Validation	Lack of validation or improper validation of input or data enables attackers to run malicious code on the system.
5	Out-of-bounds Read	Software that improperly reads past a memory boundary can cause a crash or expose sensitive system information that attackers can use in other exploits.
6	OS Command Injection	Software that constructs all or part of a command using externally influenced input from an upstream component but does not neutralize or incorrectly neutralizes special elements which could modify the intended command sent to a downstream component.
7	Use After Free	Software which references memory that had been freed can cause the program to crash or enable code execution.
8	Path Traversal (directory traversal)	Improper limitation of a pathname to a restricted directory.
9	Cross-Site Request Forgery	When a web app fails to validate that a user request was intentionally sent, it may expose data to attackers or enable remote malicious code execution.

Table 2 – MITRE's 2022 CWE Top 10 (of the 25) Most Dangerous Software Weaknesses		
Rank	**Name**	**Description**
10	Unrestricted Upload of File with Dangerous Type	Software that permits unrestricted file uploads opens the door for attackers to deliver malicious code for remote execution.

7 Automated Testing Versus Manual Testing

An organization's AST Program should employ both automated testing and manual testing. Manual testing is testing of the application where tests are executed manually by an application security tester. It is performed to discover bugs in applications under development that automated testing cannot scan for or to resolve automated testing's false positives. Automated testing is generally performed to identify the needles in the haystack that require manual examination and focused testing. In other words, manual analysis by itself is inefficient and automated AST by itself is incomplete. They need to be used together.

In manual AST, the tester checks all the essential features of the given application. In this process, the tester executes test cases and generates the test reports without the aid of any automated AST tool. It is a classical method of all testing types and helps find more complex and logical bugs in applications, generally conducted by an experienced tester to accomplish the application testing process requiring a substantial level of effort. Manual testing also often incorporates some test scripting for repetitive processes.

In contrast to automated AST, manual AST is very good for identifying vulnerabilities in the business logic, standards violations, and design flaws, especially when the code is technically secure but logically flawed. Such scenarios are unlikely to be detected by any automated AST.

A manual code review requires an expert code reviewer who is proficient in both the language and the frameworks used for the application. Full code review can be a slow, tedious, time-consuming process for the reviewer, especially given large code bases with many dependencies.

With automated AST, a tester utilizes tools or writes code/test scripts to automate test execution and uses the automation tools to develop the test scripts and validate the application. The goal is to complete test execution in less time than manual testing. Automated testing entirely relies on the pre-scripted test which runs automatically to compare the actual results with the expected results. This helps the tester to determine whether an application performs as expected. Automated testing allows a tester to execute repetitive tasks and regression tests without the intervention of a manual tester during test execution. Even though all processes are performed automatically, automation requires some manual effort to create initial test scripts.

Both methods have their strengths and weaknesses. In general, manual testing is slow and tedious, but its strength is that it is better suited to handle complex scenarios than automated testing. In comparison, automated testing requires heavy coding and maintenance, but it is much faster, enables high-volume testing, and covers many more test permutations.

Table 3 depicts the differences between the two testing methods.

Table 3 – Differences Between Automated Testing and Manual Testing

Aspects of Testing	Automated	Manual
Definition	▪ **Automated testing** uses automated software tools to execute test cases	▪ **Manual testing** is executed by a human tester and software
Processing Time	▪ Significantly **faster** than a manual approach	▪ **Slower**, very time-consuming, and uses more human resources
Exploratory Testing	▪ **No** - Does not allow random testing	▪ **Yes** - Exploratory testing is possible
Initial Investment	▪ **High**, but the return on investment (ROI) is greater in the long term	▪ **Low**, but ROI is less in the long term.
Reliability	▪ **Good** reliability as it is performed by tools and scripts and no testing fatigue	▪ **Lower** reliability due to human fallibility and testing fatigue
Accuracy	▪ Automated testing yields a **lower degree** of overall **accuracy** producing more false positives	▪ Manual testing yields a **higher degree** of overall **accuracy** producing less false positives
User Interface (UI) Change Impact	▪ **Significant** - Any UI change would require test scripts to be updated	▪ **Very Low** - Any UI change would not thwart execution of a manual tester
Investment	▪ Automated tools take less time and human effort by lesser experienced personnel, hence the investment is comparatively **low**	▪ Because manual AST is performed by a highly skilled security professional, it is more time consuming, and the investment is more **costly**
Cost-Effective	▪ Cost effective for **high** volume regression testing	▪ Cost effective for **low** volume regression testing
Test Report Visibility	▪ Test results are **accessible** to all stakeholders as results are stored internally via the automated system and can be automatically incorporated into issue and project tracking software	▪ Tests results are **not readily available** as results are usually recorded in external files such as a spreadsheet
Human Observation	▪ Offers low level of assurance of user-friendliness and positive customer experience since human observation is **not possible**	▪ Offers high level of assurance since human observation is **possible**, which is useful in offering a user-friendly system
Performance Testing	▪ **Well equipped** to handle performance tests such as load testing, stress testing, and spike testing because tests are executed compulsorily by the tool	▪ Performance Testing is **not feasible** manually
Parallel Execution	▪ **Yes** - Testing can be executed in parallel (and on different operating platforms) to reduce test execution time	▪ **Yes (limited)** - Manual tests can be executed in parallel, but limited to the available human resources which increases expenses
Batch Testing	▪ **Yes** - Batching or grouping multiple test scripts for execution is possible	▪ **No** - Multiple tests cannot be batched or grouped
Programming Knowledge	▪ **Little to no** programming knowledge is needed	▪ **Strong** programming knowledge is required for developing custom test scripts
Test Configuration and Setup	▪ **Less** complex	▪ **More** complex
Engagement Lengths	▪ More **accurate** and performance is not impacted by engagement lengths	▪ More **error prone** as engagement lengthens
Ideal Approach	▪ Useful when the **many different** test cases need to run **often**	▪ Useful when the test case only needs to run **once or twice**

Table 3 – Differences Between Automated Testing and Manual Testing		
Aspects of Testing	**Automated**	**Manual**
Build Verification Testing (BVT)	▪ More **quickly** performs BVT with **less** level of **effort** (LOE)	▪ More **difficult** and **time-consuming**
Deadlines	▪ Offers **zero to low risk** of skipping a predetermined test	▪ Offers a **higher risk** of skipping a predetermined test
Framework(s)	▪ **Can use** frameworks like data drive, keyword, and hybrid to accelerate the automation process	▪ **Does not use** frameworks but may use guidelines, checklists, and stringent processes to draft certain test cases
Test Case Documentation	▪ More easily **examined at a granular level** which makes the testing logic more understandable	▪ More difficult to understand the test case logic because the test case is **documented only at a conceptual level**
Test Design	▪ **Enforces** a test-driven development design process	▪ **Does not enforce** a test-driven development design process
DevOps	▪ Is an **integral** part of the DevOps Cycle	▪ **Defeats** the automated build principle of DevOps
When to Use	▪ Ideally suited for **regression testing, performance testing, load testing**, and highly repeatable functional testing	▪ Ideally suitable for **exploratory, usability, and ad hoc testing**, or when the test changes frequently

In the next sections, this guide will detail how automated AST and manual AST are implemented using various testing strategies.

7.1 Automated AST

Organizations have several options when it comes to application security products, but most will fall into one (1) of two (2) categories: Application Security "Testing" Tools, which analyze the state of the application security posture, and Application Security "Shielding" Tools, which defend and fortify applications to make breaches much more difficult to execute.

This buyer's guide focuses on different types of application security testing tools which can be delivered through vendor offerings such as products, services, and solutions.

Application security testing tools automate the testing of code. Application testing tools can be used during the development process, or they can be applied to released (production) code to identify potential issues. The examples discussed in this buyer's guide include the following:

- Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)
- Interactive Application Security Testing (IAST)
- Mobile Application Security Testing (MAST)
- Software Composition Analysis (SCA)

AST is a critical component of protecting data integrity and ensures software developers can identify and remediate application vulnerabilities early in the SDLC. Software developers rely on a variety of common automated AST tools to certify the application complies with a specific set

of security criteria. These approaches each have pros, cons, and cover a variety of different types of vulnerabilities. When integrated into the SDLC and combined with manual AST and penetration testing, they can support a comprehensive approach to evaluating application security as part of an organization's AST Program.

7.1.1 Static Application Security Testing (SAST)

Also known as static code analyzers and source code analysis tools, SAST tools are application security tools that detect security vulnerabilities within the source code of applications. The output of a SAST is a list of security vulnerabilities, including the type of vulnerability and the vulnerability location in the application's codebase.

Pros of SAST:

- Identifies and eliminates vulnerabilities in source, binary, or byte code
- Reviews static analysis scan results in real-time with access to recommendations and line-of-code navigation to find vulnerabilities faster and allow for collaborative auditing
- Fully integrates with the Integrated Developer Environment (IDE)
- Offers a broad coverage of programming languages and development platforms
- Is easy to implement and adopt

Cons of SAST:

- Not capable of identifying vulnerabilities in dynamic environments
- High risk of reporting false positives
- Code analyzers must be tuned and configured
- Lacks third-party component analysis
- Since the report is static, the results are quickly outdated
- Limited security coverage
- The scanning process can be lengthy which slows down the development process
- Not applicable for use in production environment

SAST tools cover a variety of vulnerabilities including, but not limited to, the following:

⦿ Application Misconfiguration	⦿ Insufficient Binary Protection
⦿ Credential/Session Prediction	⦿ Cross-Site Scripting
⦿ Directory Indexing	⦿ Injection Attacks
⦿ Insufficient Authorization/Authentication	⦿ Inter-Process Communication
⦿ Automatic Reference Counting	⦿ OS Commanding
⦿ Cross-Site Request Forgery	⦿ Insecure Cryptography
⦿ Information Leakage	⦿ SQL Injection

⦿ Insufficient Transport Layer Protection	⦿ Cryptographic Related Attacks

7.1.2 Dynamic Application Security Testing (DAST)

Also known as web scanners, DAST tools find security vulnerabilities in web applications while the application is running, verifying the security during run time by testing different attack types against the running application. DAST does not require access to the application's source code. The vulnerability assessment is conducted from the exterior, with no access to the application source code architecture, so DAST is considered a black-box assessment approach. DAST simulates controlled attacks on a running web application or service to identify exploitable vulnerabilities in a running environment.

Pros of DAST:

- Provides a comprehensive view of application security by focusing on exploitable components and covering all components (server, custom code, open source, and services)
- Integrates into the Development, Quality Assurance (QA), and Production environments to offer a continuous holistic view
- Enables a broader approach for managing portfolio risk (scanning thousands of applications)
- Tests functional applications, so unlike SAST, DAST is not language-constrained, and runtime and environment-related issues can be discovered

Cons of DAST:

- Does not find the exact location of a vulnerability in the code
- Security knowledge is needed to interpret reports
- Tests can be time-consuming
- Lack of zero-day vulnerability support
- Finds vulnerabilities late in the SDLC, or after the development cycle is complete
- Some DAST tools may not be well adapted to support DevSecOps

DAST tools cover a variety of vulnerabilities including, but not limited to, the following:

⦿ Application Misconfiguration	⦿ Mail Command Injection
⦿ Directory Indexing	⦿ Path Traversal
⦿ HTTP Response Smuggling	⦿ Routing Detour
⦿ Improper Input Handling	⦿ SSL Injection
⦿ Insufficient Transport Layer Protection	⦿ Injection
⦿ OS Commanding	⦿ Cross-Site Scripting

⦿ Remote File Inclusion	⦿ Format String Attack
⦿ SQL Injection	⦿ Improper File System Permissions
⦿ XML External Entities	⦿ Information Leakage
⦿ XQuery Injection	⦿ Null Byte Injection
⦿ Content Spoofing	⦿ Predictable Resource Location
⦿ Fingerprinting	⦿ Server Misconfiguration
⦿ HTTP Response Splitting	⦿ URL Redirector Abuse
⦿ Improper Output Handling	⦿ XPath Injection

7.1.3 Interactive Application Security Testing (IAST)

IAST tools analyze code for security vulnerabilities while the application is run by an automated test, human tester, or any activity "interacting" with the application's functionality. It searches for known vulnerabilities inside the application's functions by simulating the various scenarios in which a user runs or interacts with the application. The analysis is conducted from the inside of the application, which provides an ideal vantage point to perform security testing. More specifically, the implementation relies on an agent that injects functionality in certain points of the execution of the application.

Pros of IAST:

- Effectively pushes testing toward the early stages of software development (shifts testing left), so problems are caught earlier in the development cycle, reducing remediation costs and delays
- Provides detailed information (including lines of code) to help development and security teams triage test results
- Performs analysis from within applications and has access to application code, runtime control and dataflow information, memory and stack trace information, Hypertext Transfer Protocol (HTTP) requests and responses, and libraries, frameworks, and other components (via an SCA tool). This analysis allows developers to pinpoint the source of an identified vulnerability and fix it quickly
- Integrates easily into continuous integration (CI) and continuous development (CD) tools
- Offers a high degree of testing accuracy including low false negative rates (failing to detect risk that exists) and low false positive rates (reporting a risk which does not actually exist)
- Allows for earlier, less costly fixes
- Delivers both static and dynamic visibility
- Useful during all phases of the SDLC

Cons of IAST:

- Limited to the discovery of different flaw types in comparison to DAST and SAST
- Compatible with only major programming languages
- Contains non-blocking functionality, meaning that even when a risk is detected, the execution flow continues in the server
- IAST based scanners cannot operate on their own and almost always require an additional external testing component in the form of a DAST scanner

IAST tools cover a variety of vulnerabilities including, but not limited to, the following:

⦿ Command Injection	⦿ Cross-Site Scripting
⦿ Insecure Cookie	⦿ LDAP Injection
⦿ Path Traversal	⦿ SQL Injection
⦿ Trust Boundary Violation	⦿ Weak Encryption Algorithm
⦿ Weak Hash Algorithm	⦿ Weak Random Number
⦿ XPath Injection	

7.1.4 Mobile Application Security Testing (MAST)

MAST tools analyzes and identifies vulnerabilities in applications used with mobile platforms (e.g., iOS, Android, and Windows 10 Mobile) during or post development.

Pros of MAST:

- Identifies and remediates iOS, Android, and Windows Phone application risks
- Assesses and reports on mobile application security to executive management and other stakeholders
- Identifies critical information exposures attributed to mobile applications in the environment
- Evaluates the security posture of new mobile technologies in development

Cons of MAST:

- Identifies the most limited range of issues in comparison to other AST tools
- Requires expertise to execute properly and is more time-consuming
- Needs cover a multiplicity of mobile devices with different versions of each operating system (OS), capabilities, features, and limitations.

MAST tools cover a variety of vulnerabilities including, but not limited to, the following:

⦿ Configuration Settings ⦿ Binary Analysis ⦿ Anti-Analysis ⦿ Jailbreak/Root Detection ⦿ Inadequate Authentication/Authorization ⦿ Session Management ⦿ Malware applications on user's device exploiting other mobile applications	⦿ Cryptography ⦿ Data Handling ⦿ Unsafe Data Storage ⦿ Handling of Personal Information ⦿ Certificates ⦿ Malware applications on user's device exploiting other mobile applications

7.1.5 Software Composition Analysis (SCA)

SCA tools identify open-source software (OSS) in a codebase, for the purpose of risk management, security, and license compliance. Popular open-source software libraries often have public bug lists, which make this technique highly effective.

Pros of SCA:

- Provides visibility into risks that can be introduced by third-party and open-source components
- Reliably detects known open-source vulnerabilities that cannot be found by other methods
- Provides a full accounting of the open source and third-party components used in the application's Software Bill of Materials (SBOM)[11]
- Monitors for newly discovered vulnerabilities

Cons of SCA:

- Often generates lengthy lists of potential risks, including negligible risks and false positives which contribute to noise in the system and can delay remediation. Manual review of results is often required, which can consume valuable resources which should be spent on addressing true risks.
- No clear prioritization of risks.
- May not detect every third-party component in a scanned codebase.

SCA tools cover a variety of vulnerabilities including, but not limited to, the following:

⦿ Historically known vulnerabilities referred to entries (Common Vulnerabilities and	⦿ Open-Source Vulnerabilities

[11] Software Bill of Materials (SBOM). See the National Telecommunications and Information Administration (NTIA) resource page at https://www.ntia.gov/SBOM.

Exposures [CVEs]) in the National Vulnerability Database (NVD) • Vulnerabilities in package managers, manifest files, source code, binary files, container images, and more	• Vulnerabilities related to out-of-date or end-of-life components • Lists licenses and versions used to detect license/compliance violations

The AST tools listed above can be consolidated into a central management and coordination console for all testing tools, known as Application Security Testing Orchestration (ASTO).

These AST tools listed above can also be provided as Application Security Testing as a Service (ASTaaS). ASTaaS is the process of enlisting an external company to perform all application testing. ASTaaS usually combines static and dynamic security methods, including penetration testing and Application Programming Interface (API) evaluations.

7.2 Manual AST

Even with rapid improvements in automation technology, many elements still need human attention to verify or to accurately determine potential security vulnerabilities in an application. Some potential vulnerabilities such as business logic issues or cryptographic issues, require a human to verify the vulnerability. Therefore, organizations should incorporate manual security testing in addition to automated security testing.

Manual security testers often use a combination of handpicked security testing tools best suited to evaluate an application, which may include customized scripts and automated scanning tools. Advanced techniques to do security testing manually involve precise test cases such as checking user controls, evaluating the encryption capabilities, and thorough analysis to discover the nested vulnerabilities within an application.

Performing security testing manually does not imply organizations cannot use automation. Rather, testers can leverage automation technology to find patterns or other clues that might uncover valuable information about the application's vulnerabilities.

The primary goal of manual security testing is to discover weaknesses and potential vulnerabilities in an application that might not be easily revealed by automated security testing alone. Regardless of the number of automated testing tools one might use, it is critical to manually analyze software behavior to ensure its integrity, confidentiality, and availability principles are not being violated.

Organizations can perform security testing manually when any weakness in the application security needs a human judgment call. An array of manual security testing techniques exist that can help assess an organization's applications and systems to ensure it is secure.

Some of the most effective and efficient ways to perform security testing manually are as follows:

⦿ Monitor Access Control Management	⦿ Password Management
⦿ Dynamic Analysis (Penetration Testing)	⦿ Brute-Force Attacks
⦿ Static Analysis (Static Code Analysis)	⦿ SQL Injection
⦿ Check Server Access Controls	⦿ Cross-Site Scripting
⦿ Ingress/Egress/Entry Points	⦿ Fuzz Testing
⦿ Session Management	⦿ Load Testing

8 Red Team Application Security Exercises

With many agencies falling victim to nation state attacks, organizations need to make an extra effort to ensure the proper security controls are in place for ongoing application maintenance as part of a comprehensive AST Program. Many organizations assume that AST is sufficient to maintain or improve their application security postures maturity. While AST can highlight known weaknesses, Red Team application security exercises demonstrate to an organization how well their application defense capabilities would hold up against a real-world cyber-attack.

A Red Team application security exercise is the process of staging a hacker-style attack on an organization's application to detect and analyze security vulnerabilities that an attacker could exploit. The entire process of the Red Team application security exercise is focused on helping organizations get a better understanding of the application's security posture, its strengths, and resilience. Red teaming application security exercise services are usually reserved for organizations with mature security programs because of the cost and time needed for planning and execution.

Red Team application security exercises utilize a risk-based approach to manually identify critical application-centric security flaws within all in-scope applications. Red Team application security exercises combine the results from industry-leading automated tools with manual testing to enumerate and validate security vulnerabilities, configuration errors, and business logic flaws. In-depth manual expert analysis enables Red Teams to find what an AST often misses.

In contrast to automated or manual AST, Red Team application security exercises implies intensive human expert testing and skillful analysis performed by experienced and certified penetration testers. Red Team application security penetration testers have backgrounds in software development. They understand the common mistakes developers can make, so they go beyond merely trying to break an application and use their experience to find critical issues before they become a security crisis.

The ideal time to conduct Red Team application security exercises would be before a production release. However, schedule pressures often lead to developers deploying applications without putting them through the proper security testing. If organizations do not conduct Red Teaming

application security exercises, the organization might be unaware of potential vulnerabilities in the application ecosystem.

The market for Red Team application security exercises is growing exponentially with new vendors offering innovative solutions. Organizations should choose wisely and only trust the market leaders with their application security. Red Team application security exercises are not cheap; however, the outcome may be worth the investment if planned and executed correctly.

9 Implementation Considerations

There are several considerations an organization should contemplate when implementing an AST strategy. These considerations are discussed in the following subsections.

9.1 AST Program

Most organizations have more than one application. Some large enterprises have hundreds of applications in development and production. Each application is constantly updated to fix security issues, improve performance, and meet new customer demands. An essential part of the update process is to test the application for security issues. Most organizations use several different application software security testing tools to analyze their applications prior to release. AST is not simply about deploying tools and running tests. It is about aligning people, processes, and technology to address application security risks holistically. Therefore, organizations should implement a holistic AST Program.

An AST Program is an organizational process for continuously assessing and addressing the threat, vulnerability, and overall risk exposure of an organization's internal and external applications, as well as its underlying platforms. As damaging breaches continue to make headlines and government authorities bring regulatory pressure to bear on organizations, many of them are implementing AST Programs to gain better visibility into potential security issues across their application landscape and more effectively resolve any vulnerabilities they find before those applications go into production.

Five tips for an effective AST Program:

1. Address application security early in the SDLC
2. Build collaborative relationships
3. Choose the right AST tool(s)
4. Select an independent third-party evaluator specializing in application security
5. Evaluate the AST tool with a proof-of-concept and use cases

9.2 AST Methodologies

Although there are many methodologies for implementing AST, the OWASP Software Assurance Maturity Model (SAMM) often stands out as the method of choice. It is well equipped to help in

the assessment, formulation, and implementation of a software security strategy. And because it is technology and process agnostic, SAMM can be integrated into an existing SDLC and adapted to an agency's unique risk tolerance model as it currently exists or even as it changes over time.

The SAMM methodology can support an agency's efforts in evaluating its existing software security practices and help build a balanced, DevSecOps approach in well-defined iterations. It can also aid in demonstrating concrete improvements to a DevSecOps approach with quick wins that build toward long-term goals and define and measure security-related activities within the agency.

Other AST methodologies include:

1. Agile Security Testing
2. OWASP Security Testing Framework
3. Penetration Testing Methodologies and Standards (PTES)
4. Information System Security Assessment Framework (ISSAF)
5. Open-Source Security Testing Methodology Manual (OSSTMM)
6. Building Security in Maturity Model (BSIMM)

9.3 Integrating AST Across the SDLC

As awareness of the importance of AST has increased in recent years, more organizations have begun factoring security concerns earlier into the SDLC. In doing so, they can better mitigate potential risks, detect bugs sooner, identify user experience problems earlier, and lower the costs involved with remediating these issues later in the software development process. DevSecOps seeks to explicitly embed different types of AST into specific phases of the SDLC, as depicted in Figure 1 below.

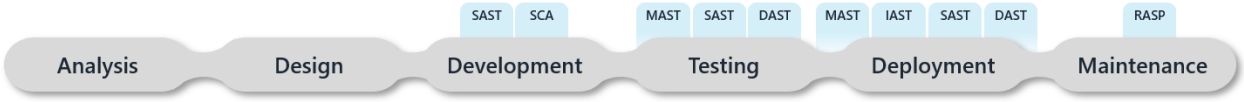

Figure 1- AST Along the SDLC

9.4 Application Security Testing Best Practices

As previously stated, AST should be initiated from the start of the SDLC and be adopted by the entire development and security operation team. The similarities in cloud, mobile, web, and desktop software development processes elicit the same AST best practices. Follow these best practices for efficient AST:

- **Shift Security Testing Left:** Organizations should emphasize the need to integrate security into every stage of the software development life cycle. AST tools can:

- Help developers understand security concerns and enforce security best practices at the development stage
- Educate developers on how to build applications that are secure by design
- Help testers identify security issues early before software ships to production
- Identify and block vulnerabilities in source code that is in production with an application security shielding tool such as Runtime Application Self-Protection (RASP)

- **Test Internal Interfaces, Not Just APIs, and UIs:** Organizations cannot focus AST on external threats only; attackers exploit weak authentication or vulnerabilities on internal systems, once already inside the security perimeter. AST should be leveraged to test and ensure the inputs, connections, and integrations are secure between internal systems.

- **Test Often:** It is essential to test critical systems as often as possible, prioritize issues focusing on business-critical systems and high-impact threats, and allocate resources to remediate them fast.

- **Third-Party Code Security:** Organizations should employ AST practices to any third-party code they use in their applications. Never "trust" that a component from a third party is secure, whether commercial or open source.

9.5 Implementation Challenges

Bad or suboptimal AST practices do not merely make organizations less secure; they drain resources, increase development time, and threaten a lack of compliance. Combined, this makes the application development process far more expensive and less secure. If vulnerabilities are not caught before release, the situation is far worse, potentially exposing government data. It is more efficient to invest in early and continuous AST, than to risk breaches and the potential loss of revenue. Organizations face many challenges in trying to improve their AST strategy. Among the most common AST challenges are:

- **Time to market pressures**

  Developers are typically under extreme time pressure from stakeholders. The result can be unreasonable deadlines, and when developers are rushed, security mistakes can occur. Even worse, unrealistic time demands may result in code-checking being shortened or even eliminated.

- **Misuse of unvalidated third-party code**

  Code reuse, such as open-source components, is not on its own a bad practice. It can offer many advantages such as better transparency, greater agility, and lower costs. However, using third-party code comes at a risk. In particular, it is hard to precisely determine how much quality-control is done (if any) prior to public release of the used open-source

component. Moreover, security holes often exist, and these holes could have been unintentional, made by a programmer years ago or, far worse, some intentional malware that a malicious actor planted with the hope that it would spread wildly. In short, code reuse is sometimes necessary, but it must be used cautiously.

- **Manual security activities**

  Historically, automated security tools did not meet expectations, which resulted in reduced adoption. As a result, some security executives prefer to rely on manual security activities. Lack of security automation means security validation is expensive, slow, and prone to errors.

10 Other Considerations

10.1 Reporting Requirements

Regardless of what AST framework, methodology, or tool is used, it is important to consider the regulatory reporting requirements. For example, to ensure agencies can receive vulnerability information from the general public, OMB issued OMB Memorandum M-20-32, "Improving Vulnerability Identification, Management, and Remediation," on September 2, 2020; and CISA published Binding Operational Directive (BOD) 20-01, "Develop and Publish a Vulnerability Disclosure Policy," on September 20, 2020. These authorities require agencies to publish security contact information, and a clear and welcoming Vulnerability Disclosure Policy (VDP).

OMB may, at any time, ask an agency to produce an application's most recent Security Assessment Report (SAR). The SAR must contain analysis prepared by more time-intensive, specialized, and application-specific methods.

10.2 AST Delivery Models

When selecting AST tools there are two (2) main variations of delivery models:

1. On-premises
2. Application Security Testing as a Service

When organizations select on-premises tools as an AST delivery model, they must install the tools, maintain them, train employees to operate them or hire experienced specialists, and be responsible for the results of the tests. Typically, the delivery model does not scale, is expensive, and requires skill and time to operate the AST tools.

ASTaaS does not require organizations to buy tools, install them, maintain them, learn how to use them, run them, or take responsibility for the accuracy of vulnerability detection or the latency between test request and results return. Instead, the independent Third-Party Application Security Tester (3PAST) handles these tasks on behalf of the enterprise. ASTaaS is an advanced delivery model that makes security transparent to development and operating specialists.

Automated AST tools are often provided as ASTaaS. ASTaaS is the process of enlisting an external company to perform all application testing. It usually combines static and dynamic security methods, including penetration testing and API evaluations.

10.3 AST Delivery Platforms

When selecting AST tools, there are four (4) common types of platforms. A platform is simply a computer or hardware device and/or associated OS, or a virtual environment, on which applications can be installed or run.

- **Desktop-based platform applications:** Run on the desktop OSs like MacOS, Windows 10 Pro, and Ubuntu, etc.
- **Mobile-based platform applications:** Run on Mobile OSs like Android OS, iOS, Blackberry OS, Windows Phone, etc.
- **Web-based platform applications:** Run on web servers like Apache, Microsoft Internet Information Server (IIS), Oracle Web Center etc.
- **Cloud-based platform applications:** Run on cloud computing resources, typically data-centric, on physical or virtually hosted servers such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure.

11 Choosing the Right AST Tool

Several AST tools and suites exist; however, it is important for organizations to take into consideration several finer aspects while making the decision. Moreover, organizations must first determine the delivery model to use. Organizations might plan to extend access beyond the core cybersecurity team to systems engineers or developers who may not be well-versed in the use of security products, which are complex tools, and difficult to understand and navigate. Once the organization decides who will use the tools, the feature list must be determined. There are many features and claimed support for these features. However, it is critical for organizations to conduct due diligence and research the vendor offering AST tools. In addition to the feature list, the credibility and reputation of the vendor are essential evaluation factors. Some of the most important and critical features organizations must expect within an AST tool are:

1. **Accuracy:** There is no security without accuracy. AST helps organizations identify vulnerabilities before potential discovery by attackers. The accuracy of these scans determines how well cybersecurity teams can use the results to find and fix the organization's highest-priority security and compliance issues.
2. **Quality and Speed:** Scanning for vulnerabilities is a real-time process that is highly time-sensitive; therefore, comprehending the reliability and promptness of the vulnerability tool is extremely critical to assure business continuity.

3. **User Experience:** The product should be seamless to navigate with easy interpretation capabilities, as organizations are heavily investing in vulnerability tools, which are enriched with multiple options to assist in detecting risks real-time with minimal complications.
4. **Compatibility:** The product's signature database needs to cover all the major OSs, applications, and infrastructure components to integrate easily with the existing systems. The tool's compatibility with legacy systems, modern software development tools, and web applications is important for a smooth transition as organizations might be initially reluctant to rely on tech-driven tools to assess and detect vulnerabilities in real-time.
5. **Support:** Along with the compatibility, the tool should support all of the advanced configurations required to run regular scans through diverse systems.
6. **Compliance:** The product should support all relevant compliance programs that apply to the specific government environment. It should effectively perform required scans and robust self-assessments.
7. **Prioritization:** The product should include a mix of manual configuration and automated prioritization that efficiently meet all business goals. As per the functionalities, the product needs to provide the required human-bot balance to match all customer expectations with the desired level of human control.
8. **Remediation Guidance:** The product should provide advanced remediation guidance to identify vulnerabilities. The assurance offered by the product and its advanced features should empower the testers to be totally guided to track down the vulnerabilities quickly and sort them out.
9. **Vendor Support:** The tool should offer robust support as a part of the contract to deliver the vendor's promised response time. The vendor's promise to provide extended support throughout will always remain a top factor before selecting or investing in any AST tool.
10. **Team Collaboration:** The right amount of collaboration across the team, backed by shared responsibilities are critical to assure the success of the AST tools. Without the team's collaboration and support, it is difficult to define the success of the AST tools, regardless of the product's advanced features.

If organizations can implement only one AST tool, the following guidelines can assist in determining which type of tool to choose:

- If the application is written in-house or there is access to the source code, a good starting point is to run a SAST and check for coding issues and adherence to coding standards. In fact, SAST is the most common starting point for initial code analysis.
- If the application is not written in-house or there is no access to the source code, DAST is the best choice.
- Whether there is access to the source code or not, if a lot of third-party and open-source components are known to be used in the application, then SCA tools are the best choice. Ideally, SCA tools are run alongside SAST and/or DAST tools, but if resources only allow

for implementation of one tool, SCA tools are imperative for applications with third-party components because they will check for vulnerabilities already widely known.

Table 4 – Comparison of AST Tools

CAPABILITIES	SAST	DAST	IAST	MAST	SCA	MANUAL
Flaws in Custom Web Applications	✓	✓	✓			✓
Flaws in Custom Non-Web Applications	✓		✓			✓
Flaws in Custom Mobile Applications	✓		✓	✓		✓
Known vulnerabilities in Open-Source Components					✓	✓
Behavioral Issues	✓		✓			✓
Configuration Errors		✓				✓
DOM-Based Cross-Site Scripting	✓	✓				✓
Business Logic Flaws						✓
Coverage of Full Application	✓	✓			✓	✓
Repeatable Process for Automation	✓	✓	✓		✓	
Scalable to All Corporate Applications	✓	✓	✓		✓	
Scan Speed	Seconds to Hours	Hours	Seconds to Minutes	Days	Seconds to Minutes	Days to Weeks
Cost	$$	$	$$$	$$$	$	$$$$

12 Selecting an Independent Third-Party Application Security Tester

An independent 3PAST specializes in AST services that give organizations insight into possible security weaknesses and attack vectors in their application environment. Being in such high demand, more 3PASTs are emerging, presenting organizations with a new challenge of selecting which service to use. Before an organization decides on a 3PAST, it is important to ask the right questions both of an organization and its needs, and of the vendor's processes, capabilities, reputation, and experience.

What kind of testing is needed?

Each 3PAST is different, with varying expertise and specialties. Before an organization decides on a 3PAST, it is important to have an idea of what kind of testing is needed. The organization will need to decide on the scope of testing and what area of the application requires assessment (e.g., network, web applications, or different devices). Organizations should consider the project type, determining whether a more focused AST exercise is required that will uncover and exploit weaknesses, or a more comprehensive AST exercise simulating an attack scenario aimed at

training a blue team which has an inside-out view of the organization. Items of discussion in determining potential 3PAST services include:

- **Scope of work:** Code, front-end, or user interface; middleware or processing logic; and/or data storage
- **Objective:** The reasons or purpose of the testing; the object of the testing is the work product to be tested
- **Project type:** Penetration test, vulnerability assessment, application security assessment
- **Testing techniques:** Black box penetration testing, gray box penetration testing, white box penetration testing
- **Testing approach:** Static analysis, dynamic analysis
- **Testing environment:** Test and development, staging, production, user acceptance testing (UAT), single-tenant, multi-tenant
- **Methodology:** Testing methodologies are the various strategies or approaches used to test an application
- **Reporting:** A combined summary of testing objectives, activities, and results

By having an idea of what the organization's requirements are, the organization will be able to ensure alignment with the AST service approach chosen. There is a lot of variation between how organizations approach testing and even how they define certain terms. When having discussions with different 3PASTs, it is vital that the 3PAST understands the organization's AST Program requirements and expectations. For example, the 3PAST should be aware of the chosen AST methodology and how it is integrated into the SDLC.

Does the 3PAST have the necessary skill sets?

Not all 3PASTs are created equally. Many focus on basic, routine tests performed with an AST tool, packaging it as a custom service. However, such tools can be used by the organization's own security team, so it is important to find 3PASTs who are experts in tailoring their tests for an organization's needs and goals, and who are able to advise an organization on the different testing options.

There are many ways to evaluate 3PAST skill sets, including educational background and/or industry specific certifications that demonstrate advanced knowledge and skills. Some of the most important certifications include Certified Web Application Security Tester (C-WAST), Certified Application Security Engineer (CASE), and Certified Software Security Tester (CSST). It is important to find a 3PAST team that keeps its skills and certifications current with continuing education and training. It is also important to inquire if the team members are given time to conduct independent research of new techniques, or if they attend industry leading training and conferences.

Is the 3PAST team experienced?

Testing teams are usually made up of two or three practitioners working together. In most cases, a senior consultant will lead the effort and be the primary contact. Typically, a senior consultant should have at least five years of experience, solid technical skills, ideally holding at least one industry certification, and the ability to deal with changing test conditions. This level of experience is necessary to deal with multiple types of environments and identify threats in a short time frame.

As for the other members of the 3PAST team, experience within the cybersecurity industry can be extremely broad. Having a team with experience in different areas, such as network infrastructure, software development, auditing, and assessment can be particularly useful**.**

Does the 3PAST organization have a past performance?

A significant factor in the Government's selection of 3PAST contractors is the contractors' history of past performance. It is one of the primary evaluation factors for many acquisitions, along with factors such as price, delivery, and quality. If a contractor has demonstrated poor performance on past contracts, it increases the likelihood of poor performance on future contracts. The currency and relevance of the past performance information, source of the information, context of the data, and general trends in a contractor's performance must be considered.

The primary source of past performance information upon which contracting officers can draw is the Federal performance information repository, which is known as the Contractor Performance Assessment Reporting System (CPARS). CPARS is used to input data on contractor performance. Once this "report card" data is entered in CPARS, it is made available for use in source selections.

Past Performance Information (PPI) regarding a contractor's actions under previous contracts and orders, also known as past performance, is an indicator of future performance and is one of the most relevant factors that a selection official should consider in awarding a contract.

Does the 3PAST have defined processes?

One of the best ways to evaluate a 3PAST service is by the quality of the procedures. AST cannot be completed on a whim by unknown parties without a plan in place. During the AST process, a 3PAST has access to sensitive information. Therefore, it is vital to know exactly who will be conducting the tests. Additionally, it is important to know how a 3PAST decides who to hire, the names and professional biographies of potential testers, and security clearance requirements. Once it is determined who will perform the testing, it is important to know how the testing will be conducted. Any 3PAST firm considered should provide a proposal that details:

- Scoping
- Project methodology

- Team selection
- Rules of engagement
- Reporting
- Handling of Personally Identifiable Information (PII) data management
- Escalation

Reporting is a particularly important piece of AST and can determine how beneficial a test is long term. The report should not only include a thorough AST of the results, but it also needs to provide clarity about how the organization's DevSecOps team can move forward with remediation. This includes providing remediation steps, tools, techniques used in the project, and a list that prioritizes the most urgent concerns. Comparing sample reports can show the differences in structure and potential details provided. Looking at other work the 3PAST team has done more comprehensively demonstrates the quality of its expertise, and evaluation and reporting capability.

The Importance of Choosing the Right 3PAST

Most commonly, 3PASTs are sought after to validate industry standards, practices, and regulatory requirements, including OWASP, as well as OWASP SAMM and OWASP SAMM Secure Build (SB), Computer Emergency Response Team (CERT), CWE, and International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) Technical Specification (TS) 17961, "Information technology – Programming languages, their environments and system software interfaces – C secure coding rules." A talented 3PAST can also help an organization with taking a proactive approach to application security to prevent a devastating breach. Ultimately, choosing the right 3PAST can provide new insights to bolster an organization's security, providing a safe and secure outside opinion as well as a fresh perspective.

Many well-established 3PAST vendors can assist an organization in performing AST. Alternatively, crowdsourcing security services can help an organization get immediate help when needed at an affordable price.

13 Crowdsourced Security Services

Like all forms of crowdsourcing, crowdsourced security unites a disparate set of individuals to work towards a common goal. Crowdsourced security supports today's key application attack surfaces. As organizations move to the cloud, the biggest concerns are web application front-ends and APIs, which may be deployed on Internet of Things (IoT) devices, mobile applications, or on-premises cloud. All of these assets can be evaluated for risk by crowdsourced security. Furthermore, a public crowd program can uncover risk in areas unknown to the security organization, such as shadow applications or exposed perimeter interfaces.

Using crowdsourced security lowers security costs and operational overhead due to the following:

- No agent software on applications or clients

- No software instrumentation to support
- No network devices or virtual appliances to install and manage
- Little to no operational waste caused by false positives or low-priority events

As security budgets come under increasing scrutiny, crowdsourcing becomes an obvious choice for simultaneously controlling costs while still aggressively protecting a business.

Examples of top Crowdsourced Security Platforms (CSSPs) are:

- **HackerOne:** Software as a service (SaaS) based platform that enables security researchers to find and report security holes to companies before critical vulnerabilities are exploited.
- **Synack:** Hacker powered security platform arms clients with hundreds of the world's most skilled, highly vetted ethical hackers who provide a truly adversarial perspective of clients' IT environments.
- **Bugcrowd:** SaaS based platform that provides rapid triage, and data-driven insights to multiple security use cases, keeping all digital assets secure and resilient throughout the SDLC.
- **Detectify:** SaaS based website security service that analyzes and monitors the security level of a website by applying a broad range of emulated hacker attacks.

14 AST Buyer's Guide Contact Information

In today's threat-riddled development landscape, AST plays a vital role in maintaining the overall security posture of an application, and more importantly, the confidentiality, integrity, and availability of the information system in which the application resides. When implemented effectively, AST offers stakeholders the ability to discover and address security vulnerabilities early and throughout the SDLC, which in turn offers greater assurance of a high-quality product which can protect a user's data and privacy.

To discuss AST requirements or business needs, the contact information for this AST Buyer's Guide is as follows:

- E-mail ITSecurityCM@gsa.gov for Customer Support concerning the AST Buyer's Guide.
- E-mail RMASS@gsa.gov for any AST Buyer's Guide comments, suggestions, and options.
- For GSA-offered products, services, and solutions, contact the respective acquisition support for the GSA Schedules identified in Appendix B of this AST Buyer's Guide.

Appendix A – Glossary

The following table provides the key terms used in this document as well as a definition or explanation of the terms.

Appendix A - Glossary	
Term	**Definition**
Application	A computer program that is designed for a particular purpose.
Application Programming Interface (API)	A set of rules that allows programmers to develop software for a particular operating system without having to be completely familiar with that operating system.
Application Security Shielding	A critical security measure that makes the application resistant to intrusion.
Application Security Testing (AST)	A strategy to assess application vulnerabilities that may compromise the confidentiality, integrity, and availability of critical or sensitive data while evaluating the effectiveness of controls implemented, to ensure the application and organizations are not prone to application-based risks and mitigate their potential impact.
Application Security Testing as a Service (ASTaaS)	Security testing services an organization pays a vendor to perform on applications.
Application Security Testing Orchestration (ASTO)	A dedicated application security pipeline that runs in parallel to the development or production pipeline. This customized AppSec pipeline automates security testing throughout the entire software development life cycle (SDLC) not just a few stages.
AST Program	An organizational process for continuously assessing and addressing the threat, vulnerability, and overall risk exposure of a company's internal and external applications, as well as its APIs.
Automated Testing	Automated testing (software test automation) is an approach to verifying code that makes use of special software tools that execute tests automatically and then compare actual test results with expected results.
Common Weakness Enumeration (CWE)	A category system for hardware and software weaknesses and vulnerabilities.
Crowdsourcing Security	Crowdsourced security methodologies invite a group of people (a crowd) to test an asset for vulnerabilities.
Cyber Attacks	An attempt by hackers to damage or destroy a computer network or system.
Cybersecurity	The ability to protect or defend the use of cyberspace from cyber attacks.
Development, Security, and Operations (DevSecOps)	A software development methodology primarily concerned with integrating security processes into DevOps cycles while maintaining efficiency.
Dynamic Application Security Testing (DAST)	Also known as web scanners, DAST tools find security vulnerabilities in web applications while the application is running, verifying the security during run time by testing different attack types against the running application.
Framework	A layered structure indicating what kind of programs can or should be built and how they would interrelate.
GSA Schedule	A long-term governmentwide contract with commercial companies that provide access to millions of commercial products and services at fair and reasonable prices to the government.
Impact	The harm that may be suffered when a threat compromises an information asset.
Independent Third Party	Testing of software by any individual/independent organization that is not directly or indirectly involved in the development of the software.
Interactive Application Security Testing (IAST)	Analyze code for security vulnerabilities while the application is run by an automated test, human tester, or any activity "interacting" with the application's functionality.

Appendix A - Glossary	
Term	**Definition**
Manual Testing	The process of manually testing software for defects. It requires a tester to play the role of an end user whereby they use most of the application's features to ensure correct behavior.
Methodology	A body of methods, rules, and postulates employed by a discipline; a particular procedure or set of procedures.
Mobile Application Security Testing (MAST)	Analyzes and identifies vulnerabilities in applications used with mobile platforms (e.g., iOS, Android, and Windows 10 Mobile) during or post development.
On-Premises	Is installed and runs on computers on the premises of the person or organization using the software, rather than at a remote facility such as a server farm or cloud.
Open Systems Interconnection (OSI) Model	A conceptual model that describes the universal standard of communication functions of a telecommunication system or computing system, without any regard to the system's underlying internal technology and specific protocol suites.
Open-Source Software (OSS)	Computer software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose. Open-source software may be developed in a collaborative public manner.
Penetration Testing	A penetration test, also known as a pen test, is a simulated cyber-attack against a computer system to check for exploitable vulnerabilities and exploits the vulnerabilities identified.
Platform	The computer architecture and equipment using a particular operating system.
Product	An item or a good that can be purchased and used by a consumer.
Risk	An estimation of the likelihood a threat will create an undesirable impact. In terms of this method, risk may be expressed as the product of likelihood and an impact.
Risk Landscape	An assessment of risks exposure of assets, will be based on a threat landscape (i.e., assume some threats), while taking into account impact and providing mitigation controls for the assumed threats.
Runtime Application Self-Protection (RASP)	A security technology that is built or linked into an application or application runtime environment and is capable of controlling application execution and detecting and preventing real-time attacks.
Security Controls	Safeguards or countermeasures to avoid, detect, counteract, or minimize security risks to physical property, information, computer systems, or other assets.
Security, Development, and Operations (SecDevOps)	A software development methodology whose main priority is given to the security of the application.
Service	Refers to a business serving as a resource to help and support clients in a certain area.
Shift Security Testing Left	Security measures implemented during the entire development life cycle, rather than at the end of the cycle.
Software Bill of Materials (SBOM)	A formal record containing the details and supply chain relationships of various components used in building software. Software developers and vendors often create products by assembling existing open source and commercial software components. The SBOM enumerates these components in a product.
Software Composition Analysis (SCA)	Tools that identify open-source software (OSS) in a codebase, for the purpose of risk management, security, and license compliance.
Software Development Life Cycle (SDLC)	An application of standard business practices to building software applications, typically divided into six to eight steps: Planning, Requirements, Design, Build, Document, Test, Deploy, and Maintain.
Solution	A solution is an offering that aims to solve a common or specific problem with the application of a product that is tailored to individual clients.

Appendix A - Glossary	
Term	**Definition**
Static Application Security Testing (SAST)	Also known as static code analyzers and source code analysis tools, SAST tools are application security tools that detect security vulnerabilities within the source code of applications.
Technology-Driven	Management philosophy that pushes for development of new goods or services based on a firm's technical abilities instead of proven demand (e.g., to make keys first and then look for locks to open). Practically every breakthrough innovation is based on a technology-driven orientation.
Third-Party Application Security Tester	An independent external application security tester that performs a thorough evaluation of an organization's application security.
Threat Landscape	A collection of threats in a particular domain or context, with information on identified vulnerable assets, threats, risks, threat actors, and observed trends.
User Interfaces (UIs)	Software designed to allow a computer user to interact with the operating system of a machine or system (such as by selecting presented options or entering text commands).
Vulnerability	Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.

Appendix B – GSA-Offered Products, Services, and Solutions for AST

This table lists the General Services Administration (GSA) Schedules for Application Security Testing (AST) related products, services, and solutions.

GSA-Offered Products, Services, and Solutions for AST		
Product, Service, or Solution	**Description**	**GSA Solution**
Secure Development Platforms	Operation systems or operating environments in which software can be developed and run where the security has been incorporated into every phase of the software development life cycle (SDLC). Example: Application Security Testing Orchestration (ASTO)	Second Generation Information Technology (2GIT) Blanket Purchase Agreements (BPAs) • To search for individual products, see GSAAdvantage! GSA eLibrary Special Item Number (SIN) 511210 Software Licenses • To search for individual products, see GSAAdvantage!
Code Scanning Tools	Tools used to analyze source code or compiled versions of code to help find security flaws. Example: Static Application Security Testing (SAST)	2GIT BPAs • To search for individual products, see GSAAdvantage! GSA eLibrary SIN 511210 Software Licenses • To search for individual products, see GSAAdvantage!
Application Testing Tools	Software applications using scripts, tools, or any test automation frameworks in order to identify errors in an application in the development process. Examples: Dynamic Application Security Testing (DAST), Mobile Application Security Testing (MAST)	2GIT BPAs • To search for individual products, see GSAAdvantage! GSA eLibrary SIN 511210 Software Licenses • To search for individual products, see GSAAdvantage! Enterprise Infrastructure Solutions (EIS) leveraging Platform as a Service (PaaS) • To search for individual products, see GSAAdvantage!

GSA-Offered Products, Services, and Solutions for AST		
Product, Service, or Solution	**Description**	**GSA Solution**
Application Shielding Tools	Tools that protect applications from reverse engineering, tampering, and other threats. Examples: Runtime Application Self-Protection (RASP), Web Application Firewall (WAF)	2GIT BPAs • To search for individual products, see GSAAdvantage! GSA eLibrary SIN 511210 Software Licenses • To search for individual products, see GSAAdvantage! GSA eLibrary SIN 517312 Wireless Mobility Solutions (Mobile Threat Protection Subcategory) • To search for individual products, see GSAAdvantage!
Security as a Service	Outsourced service used to manage an organization's cybersecurity, such as using an antivirus software over the Internet. Examples: Application Security Testing as a Service (ASTaaS), Crowdsourced Security	2GIT BPAs • To search individual products, see GSAAdvantage! EIS leveraging the Managed Security Service (MSS), PaaS, and Software as a Service (SaaS) • To search individual products, see GSAAdvantage! Highly Adaptive Cybersecurity Services (HACS): GSA eLibrary SIN 54151HACS GSA eLibrary SIN 511210 Software Licenses • To search individual products, see GSAAdvantage! GSA eLibrary SIN 541990IPS Data Breach Response and Identity Protection • To search individual products, see GSAAdvantage! GSA eLibrary SIN 518210C Cloud Computing and Cloud Related IT Professional Services GSA eLibrary IT Professional Services SIN 54151S • IT Backup and Security Services Subcategory

GSA-Offered Products, Services, and Solutions for AST		
Product, Service, or Solution	**Description**	**GSA Solution**
		• Information Assurance Subcategory Alliant 2 Governmentwide Acquisition Contract (GWAC) 8(a) STARS III GWAC VETS II GWAC
Independent Third-Party Application Security Tester	Independent testing by an organization that was not involved in the design and implementation of the application being tested and is not intended as the eventual user of that object. Examples: Application Penetration Testing, SAST, DAST, MAST, Interactive Application Security Testing (IAST)	GSA eLibrary SIN 54151HACS • Risk and Vulnerability Assessment (RVA) Subcategory • Penetration Testing Subcategory GSA eLibrary IT Professional Services SIN 54151S • IT Backup and Security Services Subcategory • Information Assurance Subcategory Alliant 2 GWAC 8(a) STARS III GWAC VETS II GWAC GSA eLibrary SIN 518210C Cloud Computing and Cloud Related IT Professional Services

Appendix C – References

This buyer's guide was developed in accordance with the following references:

- **Executive Order (E.O.) 14028**, "Improving the Nation's Cybersecurity," 12 May 2021.
- **Office of Management and Budget (OMB) Memorandum M-22-09,** "Moving the U.S. Government Toward Zero Trust Cybersecurity Principles," 26 January 2022.
- **OMB Memorandum M-22-18,** "Enhancing the Security of the Software Supply Chain through Secure Software Development Practices," 14 September 2022
- **OMB Memorandum M-21-30**, "Protecting Critical Software Through Enhanced Security Measures," 10 August 2021.
- **National Institute of Standards and Technology (NIST) Special Publication (SP) 800-218**, *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*, February 2022.
- **NIST SP 800-161 Rev.1**, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*, May 2022.
- **NIST SP 800-163 Rev.1**, *Vetting the Security of Mobile Applications*, April 2019.
- **NIST SP 800-115,** *Technical Guide to Information Security Testing and Assessment*, September 2008.
- **NIST SP 500-269,** *Software Assurance Tools: Web Application Security Scanner Functional Specification Version 1.0*, February 2008.
- **NIST SP 500-268**, *Source Code Security Analysis Tool Function Specification Version 1.1*, February 2011.
- **NIST SP 500-270,** *Source Code Security Analysis Tool Test Plan Version 1.1*, July 2011.
- **NIST Internal Report (IR) 8397**, *Guidelines on Minimum Standards for Developer Verification of Software*, October 2021.
- **NIST IR 8018**, *Public Safety Mobile Application Security Requirements Workshop Summary*, January 2015.
- **NIST IR 8135,** *Identifying and Categorizing Data Types for Public Safety Mobile Applications: Workshop Report*, May 2016.
- NIST Software Assurance Metrics and Tool Evaluation (SAMATE) Project

Appendix D – Generic Testing Services Checklist

Tester:	
Customer:	
App Name:	

Status Key
Not Tested (N/A)
In Progress
Follow Up
Complete

1 Information Gathering	Status	Date	Testing Notes
1.1 Spiders, Robots and Crawlers (OWASP-IG-001)			
1.2 Search Engine Discovery/Reconnaissance (OWASP-IG-002)			
1.3 Identify application entry points (OWASP-IG-003)			
1.4 Testing for Web Application Fingerprint (OWASP-IG-004)			
1.5 Application Discovery (OWASP-IG-005)			
1.6 Analysis of Error Codes (OWASP-IG-006)			

2 Configuration Management Testing	Status	Date	Testing Notes
2.1 SSL/TLS Testing (SSL Version, Algorithms, Key length, Digital Cert. Validity) (OWASP-CM-001)			
2.2 DB Listener Testing (OWASP-CM-002)			
2.3 Infrastructure Configuration Management Testing (OWASP-CM-003)			
2.4 Application Configuration Management Testing (OWASP-CM-004)			
2.5 Testing for File Extensions Handling (OWASP-CM-005)			
2.6 Old, Backup and Unreferenced Files (OWASP-CM-006)			
2.7 Infrastructure and Application Admin Interfaces (OWASP-CM-007)			
2.8 Testing for HTTP Methods and XST (OWASP-CM-008)			

3 Authentication Testing	**Status**	Date	Testing Notes
3.1 Credentials transport over an encrypted channel (OWASP-AT-001)	☐		
3.2 Testing for user enumeration (OWASP-AT-002)	☐		
3.3 Testing for Guessable (Dictionary) User Account (OWASP-AT-003)	☐		
3.4 Brute Force Testing (OWASP-AT-004)	☐		
3.5 Testing for bypassing authentication schema (OWASP-AT-005)	☐		
3.6 Testing for vulnerable remember password and pwd reset (OWASP-AT-006)	☐		
3.7 Testing for Logout and Browser Cache Management (OWASP-AT-007)	☐		
3.8 Testing for CAPTCHA (OWASP-AT-008)	☐		
3.9 Testing Multiple Factors Authentication (OWASP-AT-009)	☐		
3.10 Testing for Race Conditions (OWASP-AT-010)	☐		

4 Session Management Testing	**Status**	Date	Testing Notes
4.1 Testing for Session Management Schema (OWASP-SM-001)	☐		
4.2 Testing for Cookies attributes (OWASP-SM-002)	☐		
4.3 Testing for Session Fixation (OWASP-SM-003)	☐		
4.4 Testing for Exposed Session Variables (OWASP-SM-004)	☐		
4.5 Testing for CSRF (OWASP-SM-005)	☐		

5 Authorization testing	**Status**	Date	Testing Notes
5.1 Testing for path traversal (OWASP-AZ-001)			
5.2 Testing for bypassing authorization schema (OWASP-AZ-002)			
5.3 Testing for Privilege Escalation (OWASP-AZ-003)			

6 Business Logic Testing (OWASP-BL-001)	**Status**	Date	Testing Notes
6.1 Testing for cross-user data separation			
6.2 Testing for privilege escalation			
6.3 Testing for forceful browsing			

7 Data Validation Testing	**Status**	Date	Testing Notes
7.1 Testing for Reflected Cross Site Scripting (OWASP-DV-001)			
7.2 Testing for Stored Cross Site Scripting (OWASP-DV-002)			
7.3 Testing for DOM based Cross Site Scripting (OWASP-DV-003)			
7.4 Testing for Cross Site Flashing (OWASP-DV-004)			
7.5 Testing for SQL Injection (OWASP-DV-005)			
7.5.1 Oracle Testing			
7.5.2 MySQL Testing			
7.5.3 SQL Server Testing			
7.5.4 MS Access Testing			
7.5.5 Testing PostgreSQL (from OWASP BSP)			
7.6 Testing for LDAP Injection (OWASP-DV-006)			
7.7 Testing for ORM Injection (OWASP-DV-007)			
7.8 Testing for XML Injection (OWASP-DV-008)			
7.9 Testing for SSI Injection (OWASP-DV-009)			
7.10 Testing for XPath Injection (OWASP-DV-010)			
7.11 IMAP/SMTP Injection (OWASP-DV-011)			
7.12 Testing for Code Injection (OWASP-DV-012)			
7.13 Testing for Command Injection (OWASP-DV-013)			
7.14 Testing for Buffer overflow (OWASP-DV-014)			
7.14.1 Testing for Heap overflow			
7.14.2 Testing for Stack overflow			
7.14.3 Testing for Format string			
7.15 Testing for incubated vulnerabilities (OWASP-DV-015)			
7.16 Testing for HTTP Splitting/Smuggling (OWASP-DV-016)			

8 Web Services Testing	Status	Date	Testing Notes
8.1 WS Information Gathering (OWASP-WS-001)			
8.2 Testing WSDL (OWASP-WS-002)			
8.3 XML Structural Testing (OWASP-WS-003)			
8.4 XML Content-level Testing (OWASP-WS-004)			
8.5 HTTP GET parameters/REST Testing (OWASP-WS-005)			
8.6 Naughty SOAP attachments (OWASP-WS-006)			
8.7 Replay Testing (OWASP-WS-007)			

9 AJAX Testing	Status	Date	Testing Notes
9.1 AJAX Vulnerabilities (OWASP-AJ-001)			