# OWASP ASVS: A Comprehensive Overview

As software systems grow more complex, proving that they are secure has become as important as making them work. Many organizations rely on multiple frameworks to guide development and compliance, yet few provide a clear, testable way to measure application security itself. This is where the **OWASP Application Security Verification Standard (ASVS)** stands out.

ASVS defines what to verify in an application to demonstrate real assurance. It gives development and security teams a framework for building, testing, and maintaining secure software, supported by measurable requirements instead of abstract principles.

In this article, we'll explore what ASVS is, how version **5.0** modernizes the framework, and how it fits alongside frameworks like **OWASP SAMM**, **NIST 800-53**, and **CIS Controls**. You'll also see how tools such as **SAMMY** make ASVS implementation practical.

Table of Contents



## Key Takeaways

- **OWASP ASVS** gives a clear, shared set of security requirements so teams know exactly what to verify.
- The 17 chapters and three levels keep security structured, risk based and practically testable.
- Levels 1, 2 and 3 scale assurance from basic hygiene to high assurance for critical systems.

- ASVS fits into a broader security program, complementing frameworks that focus on process and governance.
- Using a platform like **SAMMY** to manage ASVS turns it from a static checklist into continuous, measurable improvement.

## What is OWASP ASVS?

The **OWASP Application Security Verification Standard (ASVS)** is a framework of security requirements that helps organizations develop, test, and maintain secure web applications. It defines *what* must be verified to achieve assurance.

ASVS provides a **community-maintained catalog of best practices**, giving teams a consistent reference instead of separate internal checklists. Each requirement is written as a testable statement, creating a shared language for developers, architects, and security testers.

The framework focuses on four principles:

1. **Application** – for web and API-based software.
2. **Security** – ensuring confidentiality, integrity, and availability.
3. **Verification** – enabling independent testing and review.
4. **Standard** – offering a repeatable baseline for benchmarking.

Unlike the **OWASP Top 10**, which lists common vulnerabilities, ASVS defines the controls that prevent them. It complements **OWASP SAMM**, which measures process maturity. Version **5.0**, released in May 2025, includes about **350 requirements across 17 chapters**, making application security measurable, comparable, and actionable.

## What is the Structure of OWASP ASVS?

The **structure of OWASP ASVS 5.0** organizes security verification into a clear hierarchy of **17 chapters**, covering all major aspects of web and API application security. The model is modular by design so that organizations can include only the parts that apply to their systems. For example, a pure API service can skip frontend or WebRTC requirements, while a user-facing application can include them.

Each requirement follows a **consistent reference format**:

v<version>-<chapter>.<section>.<requirement>

For example, *v5.0.0-2.1.3* refers to version 5.0.0, chapter 2, section 1, requirement 3.
This makes every requirement uniquely traceable across releases and simplifies mapping, reporting, and tool integration.

## The 17 Chapters of ASVS 5.0

| Chapter | Focus Area |
|---|---|
| V1: Encoding and Sanitization | Prevents injection flaws through output encoding, safe deserialization, and memory handling. |
| V2: Validation and Business Logic | Ensures robust input validation and protection of business workflows from logic abuse or automation. |
| V3: Web Frontend Security | Covers browser-side protections like headers, cookies, origin separation, and third-party resources. |
| V4: API and Web Service | Focuses on API security, including message validation, REST, GraphQL, and WebSocket controls. |
| V5: File Handling | Defines secure ways to upload, store, and serve files safely. |
| V6: Authentication | Covers identity proofing, password management, multi-factor authentication, and integration with identity providers. |
| V7: Session Management | Protects session handling, timeouts, reauthentication, and session invalidation. |
| V8: Authorization | Enforces access control at the object and function level and prevents privilege escalation. |
| V9: Self-Contained Tokens | Focuses on secure creation and validation of tokens such as JWTs. |
| V10: OAuth and OIDC | Addresses authorization flows, client registration, token exchange, and consent handling. |
| V11: Cryptography | Ensures secure encryption, key management, random number generation, and public key operations. |
| V12: Secure Communication | Verifies the use of TLS and secure channel configurations for all communications. |
| V13: Configuration | Covers secure configuration of frameworks, libraries, dependencies, and environment settings. |
| V14: Data Protection | Ensures proper handling of data at rest, in transit, and on the client side. |
| V15: Secure Coding and Architecture | Encourages defensive coding, safe use of dependencies, and resilient software design. |
| V16: Security Logging and Error Handling | Defines how to log security events, protect log data, and safely display errors. |

| | |
|---|---|
| **V17: WebRTC** | Specifies security controls for real-time communication, including media, signaling, and TURN usage. |

### Versioning and Traceability

ASVS follows a **Major.Minor.Patch** release scheme:

- **Major versions** can reorganize and renumber chapters and requirements.
- **Minor versions** can add or remove items while keeping numbering consistent.
- **Patch versions** typically fix or clarify text without functional changes.

This ensures accurate tracking between assessments and prevents confusion as the standard evolves.

### Documented and Implementation Requirements

Each chapter in ASVS contains two types of items:

- **Documented Security Decisions**, which capture architectural or design intent.
- **Implementation Requirements**, which define specific behaviors that can be tested or reviewed.

This dual structure encourages both *secure design* and *verifiable implementation*, bridging the gap between theory and practice.

This chapter-based structure allows teams to apply ASVS at any scale, from small web services to complex enterprise systems, while maintaining precise traceability and repeatable verification.

## What are the Three Verification Levels of OWASP ASVS?

The **three verification levels** in OWASP ASVS define how deeply an application should be reviewed and tested based on its risk profile, sensitivity, and intended use. Each level builds on the previous one, adding more detailed and demanding requirements. This makes ASVS scalable, so both small teams and high-assurance environments can use the same framework.

The levels are not certifications or compliance labels. They are practical benchmarks that help you choose how much verification effort is appropriate for a given application.

### The Three Verification Levels of OWASP ASVS

| Level | Purpose and Focus | Verification Effort | Typical Use Case |
|---|---|---|---|
| **Level 1 – Basic Verification** | Minimum baseline for all applications, even low-risk ones. Focuses on automatable and high- | Dynamic testing, SAST, or lightweight code review. Designed to be achieved | Marketing websites, internal tools, or |

| | | | |
|---|---|---|---|
| | impact controls to prevent common vulnerabilities such as injection, authentication issues, and misconfigurations. | quickly with minimal manual work. | prototypes exposed to the internet. |
| **Level 2 – Standard Verification** | Recommended default for most commercial and enterprise applications. Aims to ensure secure handling of sensitive data and complex business logic. | Combination of automated and manual testing, including code and documentation review. Requires active security practices in the SDLC. | Fintech platforms, healthcare portals, or SaaS products managing user data. |
| **Level 3 – Advanced Verification** | Highest assurance level for critical or high-risk environments where failure would have serious consequences. | Formal design reviews, detailed threat modeling, and extensive manual verification or penetration testing with full access to source and architecture. | Online banking, identity providers, or government and defense systems. |

## How the Levels Work Together

ASVS levels are cumulative, meaning Level 2 includes all Level 1 requirements, and Level 3 includes everything from Levels 1 and 2.

The OWASP community estimates that roughly 20 percent of requirements belong to Level 1, around 50 percent to Level 2, and the remaining 30 percent to Level 3. This distribution helps organizations plan resources according to their assurance goals.

The flexibility of ASVS means you can also mix levels per component. For instance, a public API might target Level 2, while the identity management service behind it aims for Level 3.

## Choosing the Right Level

Selecting the appropriate level should depend on three factors:

1. **Risk exposure** – how critical or sensitive the application's data and functionality are.
2. **User expectations** – the level of trust users place in the application.
3. **Regulatory requirements** – any industry or legal obligations for security assurance.

By aligning the verification level with business and technical risk, organizations can invest effort where it delivers the greatest return, rather than applying a one-size-fits-all approach.

## What's New in OWASP ASVS 5.0?

Version 5.0 of the **OWASP Application Security Verification Standard** is the most substantial update in the project's history. It refines the structure, expands coverage, and modernizes the language to make the framework more goal-oriented and practical for today's software environments.

**Refined Scope and Structure**
ASVS 5.0 clearly defines its focus on applications and APIs, leaving out infrastructure and mobile security. It now contains **17 chapters**, reorganized for clarity, and adds new coverage areas such as **Web Frontend Security (V3)**, **Self-Contained Tokens (V9)**, **OAuth and OIDC (V10)**, and **WebRTC (V17)** to reflect modern technologies.

**Documented Security Decisions**
A major conceptual change is the introduction of *Documented Security Decisions*. Each chapter starts with documentation requirements that capture how controls are applied and why. This strengthens accountability and bridges design and implementation.

**Goal-Oriented Requirements and Levels**
Requirements now emphasize *security outcomes* rather than technical prescriptions, making ASVS adaptable across architectures. Level 1 has been streamlined for quick adoption, Level 2 remains the standard default, and Level 3 retains its high-assurance focus.

**Mappings and Readability**
Outdated CWE and NIST 800-63 mappings were removed in favor of future alignment through the *Common Requirement Enumeration (CRE)*. Many requirements were rewritten for clarity, merging overlaps to make the standard easier to apply and reference.

ASVS 5.0 modernizes the framework while keeping its foundation intact, making application security verification clearer, more flexible, and more measurable.

## ASVS Use Cases and Benefits

The **OWASP Application Security Verification Standard (ASVS)** enables automated, measurable security. By integrating ASVS into daily workflows, teams shift from periodic, manual assessments to continuous, evidence based verification.

1. **Automating Security Requirements Testing**
   ASVS requirements can be turned into CI/CD checks. Many Level 1 and Level 2 controls are verifiable with static analysis, dependency scanning, and configuration review. According to ASVS expert Aram Hovsepyan in *"Security Requirements Driven Development: the best ROI for your SDLC,"* about **60 to 70 percent** of ASVS requirements can be automated, depending on architecture and coverage. This cuts recurring audit effort and keeps verification always on.
2. **Building a Traceable Assurance Framework**
   Each requirement has a unique identifier and a clear security objective. Mapping those to automated tests creates a chain from policy to evidence. **SAMMY** streamlines this by linking requirements to implemented controls, tracking status automatically, and centralizing proof. Teams gain low overhead traceability for audits and internal reviews.
3. **Enabling Continuous and Measurable Security**
   Automation creates a live feedback loop. Developers see which requirements pass, fail, or need manual validation, and security leaders see trends over time. Instead of static snapshots, organizations get a moving picture of maturity, supported by objective, repeatable test results.
4. **Reducing Time to Verification**
   Embedding ASVS checks in development shortens the gap between code changes and validation. Automated tests surface issues immediately, so fixes land before release. Delivery stays fast while assurance remains high.

ASVS automation turns verification into a proactive engineering practice, not a checklist. It aligns developers, testers, and security engineers around one measurable set of requirements. For a deeper dive into the automation approach, see "*Security Requirements Driven Development: the best ROI for your SDLC*".

## Common Challenges with ASVS

Adopting the **OWASP Application Security Verification Standard (ASVS)** delivers strong results but can feel demanding at first. Its 17 chapters and hundreds of requirements make it comprehensive, yet this scope can also seem daunting. The key to success is understanding where teams typically struggle and how to address those challenges early.

1. **Perceived Complexity**
   ASVS can look overwhelming, especially to smaller teams. The best approach is to start small, focusing on the chapters and levels that matter most. Many organizations begin with Level 1 for quick wins, then expand gradually. Platforms like SAMMY simplify this by highlighting priorities and tracking progress over time.
2. **Balancing Automation and Manual Verification**
   Not every requirement can be automated. While automation covers much of Level 1 and part of Level 2, higher levels require manual validation. The goal is balance: automate where possible, and reserve expert review for complex or high-risk areas. A simple verification matrix helps teams stay consistent.
3. **Cross-Team Alignment**
   ASVS adoption works best when development, QA, and security teams share a common understanding of what "secure" means. Embedding ASVS requirements into user stories and holding short cross-functional reviews encourages alignment and reduces confusion or duplication.
4. **Documentation Fatigue**
   ASVS 5.0 introduces documented security decisions for traceability. These can feel like extra work if treated as paperwork rather than part of the process. Making documentation part of the development flow, and storing it centrally, keeps it manageable and valuable.
5. **Compliance-Driven Thinking**
   ASVS is not a certification checklist. Its purpose is continuous improvement. Treat the three verification levels as milestones toward maturity, not pass-or-fail gates. This mindset builds long-term security capability rather than short-term compliance.

In short, successful ASVS adoption depends on gradual rollout, shared ownership, and a healthy mix of automation and expert review. Teams that tailor the framework to their reality achieve steady, lasting improvement.

## Comparing OWASP ASVS with Other Security Frameworks

ASVS is part of a broader security ecosystem. Most organizations already use frameworks that address different layers of assurance, from application controls to enterprise governance. **OWASP SAMM**, **NIST SP 800-53**, and the **CIS Controls** share the same goal of improving security but operate at different levels. SAMM focuses on maturing software security practices, while NIST and CIS define organization-wide safeguards that include application, infrastructure, and governance layers.

The table below compares these frameworks by purpose, structure, and scope, showing how they complement one another: ASVS defines *what to verify* in applications, SAMM defines *how to mature* security processes, and NIST and CIS connect those efforts to *governance and compliance*.

| Dimension | OWASP ASVS | OWASP SAMM | NIST SP 800-53 r5 | CIS Controls v8.1.2 |
|---|---|---|---|---|
| **Primary goal** | Define **what to verify** in web/app security through concrete requirements. | Define **how mature** your software security program/process is and **how to improve** it. | Provide a comprehensive **security and privacy control catalog** for systems and organizations. | Provide a prioritized set of **safeguards** for practical cyber defense. |
| **Scope focus** | Application-level verification requirements across topics like auth, file handling, etc. (L1–L3). | Organization and SDLC practices across governance, design, implementation, verification, operations. | Enterprise/system controls across 20 families, with base controls and enhancements. | Enterprise security practices spanning assets, identity, vulnerability mgmt, and more. |
| **Structure** | **Verification areas** (chapters) with **numbered requirements**, each mapped to **Level 1–3**. | **5 business functions → 15 practices → 2 streams/practice → activities at 3 maturity levels.** | **20 control families** (e.g., AC, IA, SC). Each has **base controls** plus **control enhancements**. | **18 Controls**, each with **Safeguards** and guidance; prioritized via **Implementation Groups**. |
| **Granularity** | Fine-grained, testable **app requirements**. | Program/process-level **capabilities** and improvement activities. | Very detailed **organizational/system controls** with tailoring and enhancements. | Actionable **practices and Safeguards**, less granular than 800-53 but very operational. |
| **Leveling / Prioritization** | **Level 1–3** verification depth by risk/assurance. | **Maturity Level 1–3** per stream/practice. | Tailor baselines using **control enhancements** and risk context. | **IG1–IG3** map to enterprise size/risk to phase adoption. |
| **Typical use** | Define acceptance criteria for secure features, guide | Assess and roadmap AppSec program maturity across the SDLC. | Build or assess a full control set for regulated or high-assurance environments. | Bootstrap or strengthen a pragmatic security program with |

| | | | | |
|---|---|---|---|---|
| | testing, and audits at app level. | | | prioritized safeguards. |
| **Best fit** | Product teams, AppSec engineers, pentesters validating **specific applications**. | Security leaders and teams improving **process maturity**. | GRC, security architects, compliance teams needing **comprehensive control coverage**. | IT/SecOps teams seeking **quick wins** and phased uplift across the estate. |

ASVS is the most **application-focused** of the frameworks, defining concrete security requirements that can be verified at the code, configuration, or design level. **SAMM** complements it by measuring how effectively those practices are implemented and improved across the software lifecycle. Together, they address both the *what* and the *how* of secure development.

**NIST SP 800-53** and the **CIS Controls** operate at a broader enterprise scope, covering infrastructure, identity, and governance. Many ASVS requirements align with NIST families such as Access Control (AC) and System Integrity (SI) or with CIS Controls like Secure Configuration and Application Security. These connections allow ASVS results to strengthen compliance and risk management efforts.

In practice, organizations use ASVS to reinforce the application security pillar within NIST or CIS-based programs, while SAMM measures how well those practices are embedded and continuously improved. Combined, they deliver both technical depth and organizational breadth, linking secure development with enterprise governance.

## How to Implement OWASP ASVS?

Implementing the **OWASP Application Security Verification Standard (ASVS)** is much easier with **SAMMY**, which turns the framework into a guided, measurable workflow. Instead of tracking progress in spreadsheets, SAMMY lets you manage ASVS assessments, evidence, and reporting directly within your development environment.

Start by defining a **scope**, a product, project, or team, then select ASVS as the framework to apply. SAMMY automatically loads all ASVS chapters and verification levels, allowing you to complete assessments, add documentation, and record *Documented Security Decisions*. As results are submitted, SAMMY generates **dashboards and reports** showing overall ASVS scores, comparisons across scopes, and progress toward your target posture.

The platform transforms ASVS from a static checklist into a continuous improvement process, giving you real-time visibility into security maturity and helping align teams around measurable goals.

## Conclusion

The **OWASP Application Security Verification Standard** has become the foundation for measurable, evidence-based software security. It defines a common language for developers, testers, and security leaders, helping teams move from reactive fixes to proactive assurance.

With **ASVS 5.0**, organizations gain a clearer, more flexible framework that aligns with modern architectures and enables automation. When implemented through **SAMMY**, ASVS becomes even more powerful, transforming verification from a static checklist into a continuous improvement process.

If you're ready to make security measurable across your applications, start by exploring how SAMMY supports ASVS 5.0 assessments, reporting, and automation. It's the fastest way to turn your security framework into visible, actionable progress.