Global Research journal of Natural Science & Technology (GRJNST)

Volume: 04 - Issue 2 (2026), 2076
ISSN P: 2790-7643 ISSN E: 2790-7651
www.grjnst.net
https://doi.org/10.53762/grjnst.04.02.27

AuthStateBench: A Standards-Aligned Benchmark for Stateful Authorization and Authentication Workflows

Received: 01 April 2026. Accepted: 23 April 2026. Published: 29 April 2026

Muhammad Shahzad Khadim (Corresponding Author)
Kohat University of Science and Technology, Kohat, Pakistan
ytshahzad257@gmail.com
Syed Mufassir Shah
Kohat University of Science and Technology, Kohat, Pakistan
mufassirshah3@gmail.com
Zubair Khan
International Islamic University Islamabad, Pakistan
zubairafridi2312@gmail.com

Abstract:

Authentication and authorization weaknesses in modern web applications rarely arise as isolated request-level defects. They often depend on role changes, session lifecycle conditions, object ownership boundaries, API authorization rules, and business workflow ordering. Existing vulnerability benchmarks and scanner evaluations remain valuable, but they often represent weaknesses as code-level or input-output defects and therefore underrepresent semantic failures such as IDOR/BOLA, function-level authorization bypass, stale-session reuse, tenant-boundary violation, privilege transition errors, and workflow bypass. This article introduces AuthStateBench, a standards-aligned benchmark design for modeling stateful authorization and authentication workflow vulnerabilities in web applications and APIs. The study uses a structured literature-based and standards-mapping methodology that draws on access-control testing research, stateful web testing, web logic flaw analysis, scanner-evaluation studies, vulnerability benchmark literature, AI-assisted vulnerability-analysis work, and major security guidance including OWASP Top 10, OWASP API Security Top 10, OWASP ASVS, OWASP WSTG, NIST SSDF, MITRE CWE, CISA Secure by Design, OAuth 2.0 security guidance, OpenID Connect, and software-assurance benchmark resources. AuthStateBench contributes a four-dimensional state model built around role state, session state, object-ownership state, and workflow state; a scenario taxonomy; a benchmark scenario template; standards-mapping logic; formal scenario and coverage equations; and comparison criteria for manual, scanner-assisted, AI-assisted, and standards-based assessment. The article does not claim empirical detection accuracy, tool execution, live-system testing, or dataset results. Instead, it provides a reproducible design artifact and validation roadmap for future controlled implementation and comparative evaluation.

Keywords: *Web application security; broken access control; authentication workflow; authorization testing; benchmark design; stateful security testing*

I. Introduction

1.1 Background

Modern web applications rarely fail through a single isolated defect. Their security posture emerges from the interaction of identity providers, session stores, API gateways, access-control middleware, object-level permission checks, business workflows, audit mechanisms, and recovery logic. A request that appears harmless in isolation may become dangerous when it is executed with the wrong role, a stale session, an unowned object identifier, or a skipped workflow step. This stateful character is especially visible in authorization and authentication flaws. A user may be authenticated but still unauthorized to perform a function; an expired token may be accepted after logout; an object identifier may expose another user's record; or a workflow endpoint may allow an operation before the required approval state has been reached.

The current application-security landscape confirms the importance of this problem. OWASP Top 10:2025 identifies Broken Access Control as the leading web application security risk, and OWASP API Security Top 10:2023 places Broken Object Level Authorization at the first position for API security risk [1]-[4]. OWASP ASVS 5.0.0 and the OWASP Web Security Testing Guide provide deeper verification and testing guidance for authentication, session management, access control, API behavior, and business logic [5], [6]. NIST SSDF and CISA Secure by Design guidance frame these weaknesses as secure-development and product-responsibility concerns rather than isolated testing events [8], [19], [20].

Despite the availability of these standards, there remains a benchmark-design problem. Existing vulnerability benchmarks such as OWASP Benchmark, NIST SARD, and Juliet provide valuable tool-evaluation support, but their structure is stronger for code-level and input-driven weakness classes than for multi-step authorization and authentication failures that depend on role, session, object, and workflow state [7], [9]-[11]. This gap matters because scanners, AI-assisted testing systems, manual testers, and secure-development teams need comparable scenario definitions before meaningful evaluation can occur.

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27

1.2 Problem Statement

The central problem addressed in this article is that stateful authorization and authentication workflow vulnerabilities are difficult to benchmark using isolated HTTP requests, generic scanner signatures, or code snippets detached from application state. A conventional vulnerability test case often asks whether a specific payload triggers a specific response. In contrast, a stateful authorization flaw may require two or more accounts, at least one protected object, a known workflow state, a token lifecycle condition, and an expected policy decision. Without documenting these conditions, two studies may appear to evaluate the same vulnerability category while actually testing different security properties.

This creates three practical weaknesses in the research landscape. First, scanner evaluations can overrepresent input-driven flaws while underrepresenting semantic authorization failures. Second, AI-assisted testing studies may claim progress without showing whether role, session, object ownership, and workflow preconditions were modeled. Third, secure-development guidance may remain difficult to operationalize because standards requirements are not translated into benchmark scenario templates. The problem is therefore not a lack of standards or individual testing techniques; the problem is the absence of a structured benchmark design that connects standards, state dimensions, scenario categories, and evaluation criteria.

1.3 Research Gap

Prior work has examined web application security, automated vulnerability detection, access-control analysis, business logic flaws, scanner performance, and software-vulnerability benchmarks [26]-[40]. However, the literature remains fragmented when the target weakness depends on a combination of authenticated identity, authorization policy, object ownership, token lifecycle, request ordering, and business-state transition. A stronger benchmark design is needed to make such weaknesses reproducible and comparable across testing approaches.

The precise research gap is as follows: existing web security research lacks a standards-aligned benchmark design that systematically models stateful authorization and authentication workflow vulnerabilities using role, session, object-ownership, and workflow-state dimensions. This gap prevents consistent comparison of manual testing, scanner-assisted testing, AI-assisted testing, and standards-based review for flaws such as

IDOR/BOLA, privilege escalation, session misuse, role confusion, and workflow bypass.

1.4 Aim and Objectives

The aim of this article is to design a standards-aligned benchmark model for classifying, structuring, and evaluating stateful authorization and authentication workflow vulnerabilities in modern web applications.

The objectives are: (1) to review literature on broken access control, authentication workflow flaws, stateful web security testing, benchmark-based evaluation, scanner limitations, and AI-assisted vulnerability analysis; (2) to identify recurring patterns involving role misuse, object ownership, session state, workflow bypass, token lifecycle errors, and privilege transitions; (3) to map these patterns to OWASP Top 10:2025, OWASP API Security Top 10, OWASP ASVS, OWASP WSTG, NIST SSDF, CISA Secure by Design guidance, and MITRE CWE; (4) to develop a benchmark scenario taxonomy for stateful authorization and authentication workflow weaknesses; (5) to define evaluation criteria for manual testing, scanner-assisted testing, AI-assisted testing, and standards-based review; and (6) to propose a benchmark documentation template and future validation roadmap.

1.5 Research Questions

RQ1. What stateful authorization and authentication workflow vulnerabilities are most frequently discussed in recent web application security literature?

RQ2. How can role, session, object ownership, and workflow state be used to classify authentication and access-control weaknesses?

RQ3. How can OWASP Top 10:2025, OWASP API Security Top 10, OWASP ASVS, OWASP WSTG, NIST SSDF, CISA Secure by Design guidance, and MITRE CWE be mapped to benchmark scenarios for stateful web security testing?

RQ4. What benchmark scenario categories are needed to represent IDOR/BOLA, privilege escalation, session misuse, workflow bypass, role confusion, and token lifecycle weaknesses?

RQ5. What evaluation criteria can compare manual, scanner-assisted, AI-assisted, and standards-based testing approaches without reporting unsupported empirical results?

RQ6. What limitations exist in current web security benchmarks for evaluating access-control and authentication workflow flaws?

RQ7. What future validation pathway is required to make AuthStateBench suitable for empirical cybersecurity research?

1.6 Scope

The scope of AuthStateBench is benchmark design, scenario structuring, standards mapping, evaluation logic, and future validation planning. The article focuses on web applications and web APIs where authorization and authentication outcomes depend on state. The benchmark design is intentionally abstract: it does not require unauthorized testing of real systems, private datasets, exploit demonstrations, scanner output, screenshots, or fabricated tool results. It is suitable for later implementation in controlled vulnerable applications, teaching laboratories, controlled research environments, or expert-review exercises.

The scope excludes malware analysis, network intrusion detection, blockchain security, IoT firmware testing, and generic AI-in-cybersecurity discussions unless they directly inform benchmark-design principles. The article also excludes claims of detection accuracy because no tool execution or empirical implementation is reported.

1.7 Contributions

This article makes five concrete contributions. First, it proposes AuthStateBench, a benchmark-design artifact for stateful authorization and authentication workflow weaknesses in web applications and APIs. Second, it defines a four-dimensional state model that captures role state, session state, object-ownership state, and workflow state. Third, it introduces a scenario taxonomy and editable documentation template for repeatable benchmark construction. Fourth, it adds a standards-mapping layer that connects scenario classes to OWASP, ASVS, WSTG, NIST SSDF, CISA Secure by Design, MITRE CWE, OAuth, and OpenID guidance. Fifth, it defines evidence-based comparison criteria for manual testing, scanner-assisted testing, AI-assisted testing, and standards-based review without inventing unsupported empirical results.

The contribution differs from a generic OWASP survey because it does not merely summarize risk categories. It translates recurring access-control and authentication failure patterns into reusable scenario classes. It also differs from a scanner-evaluation

paper because it does not claim performance measurements. Instead, it prepares a benchmark structure that can later support such measurements transparently.

1.8 Structure of the Paper

Section 2 reviews key concepts, recent studies, standards, and limitations of existing work. Section 3 presents AuthStateBench and its conceptual components. Section 4 explains the structured literature-based and standards-mapping methodology. Section 5 presents analytical findings, benchmark outputs, standards alignment, and comparison with existing approaches. Section 6 discusses implications, limitations, and future research. Section 7 concludes the article.

Fig. 1 summarizes the article's logic in a roadmap style: evidence is gathered, gaps are synthesized, the state model is defined, benchmark artifacts are produced, and validation is reserved for controlled future work.

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27

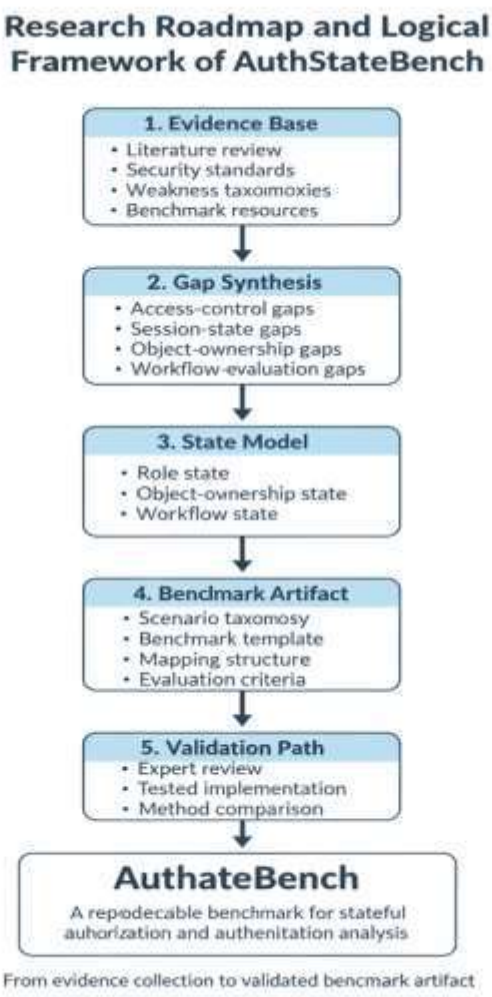

Fig. 1. Research roadmap and logical framework for AuthStateBench.

2. Literature Review

2.1 Key Concepts

Authentication verifies the identity of an entity, while authorization determines whether that entity may access a resource or perform a function. Session management maintains continuity between authenticated interactions, often through cookies, bearer tokens, refresh tokens, or server-side session identifiers. Object-level authorization checks whether a user may access a specific object instance, not merely whether the user belongs to a broad role. Workflow authorization checks whether an action is permitted at a

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27

particular stage of a business process. These concepts are separated analytically but often fail together in real applications.

A benchmark for stateful authorization and authentication weaknesses must therefore model more than a vulnerability label. For example, IDOR/BOLA is not simply "changing an ID." It is a failure to enforce ownership or tenant isolation when a user-controlled identifier points to a protected object [3], [4], [16]-[18]. Similarly, session misuse may include insufficient session expiration, reuse of tokens after logout, fixation, privilege transition failures, or incomplete invalidation after account changes [15], [21]-[23]. Workflow bypass refers to reaching an endpoint or state transition without satisfying preconditions such as payment, approval, verification, or reauthentication.

2.2 Review of Recent Studies

Research on access-control vulnerability detection has developed through static analysis, black-box testing, role-differential analysis, model inference, and workflow-based reasoning. Sun et al. proposed static detection of access-control vulnerabilities by inferring role-based access assumptions from code [29]. Li and Xue introduced BLOCK, a black-box approach for detecting state violation attacks by observing normal behavior and identifying invariant violations [30]. Felmetsger et al. highlighted that logic vulnerabilities receive less attention than classic input-validation flaws, even though they can cause serious security failures [31]. Pellegrino and Balzarotti examined black-box detection of logic flaws using behavioral patterns extracted from interactions [32].

More recent work continues to show that authorization flaws are difficult to evaluate without stateful context. Rennhard et al. presented an approach to automatically detect HTTP GET request-based access-control vulnerabilities [26]. Zhong et al. surveyed prevention and detection of access-control vulnerabilities in web applications and emphasized roles, permissions, resources, and business logic [27]. BACScan addressed black-box detection of broken access-control vulnerabilities and reinforced the need to consider multiple users and permissions [28]. SWaTEval proposed an evaluation framework for stateful web application testing, and ProFuzzBench showed the value of explicit benchmarks for stateful protocol fuzzing even outside web authorization [35], [36].

Benchmarking literature also motivates AuthStateBench. OWASP Benchmark and NIST SARD support evaluation of vulnerability detection tools, while Juliet and newer SARD

resources provide curated software-assurance test cases [7], [9]-[11]. However, benchmark critiques have noted that benchmark structure can shape tool behavior and may not always represent real semantic vulnerabilities [38]-[40]. AI-assisted vulnerability detection further increases the need for clearly defined evaluation tasks because LLM-based systems may appear effective on code-level benchmarks while struggling with multi-step security specifications, contextual authorization rules, and workflow semantics [41]-[46].

2.3 Existing Standards, Frameworks, and Models

OWASP Top 10:2025 and OWASP API Security Top 10:2023 provide risk taxonomies that place broken access control, object-level authorization, broken authentication, and function-level authorization among major application-security concerns [1]-[4]. OWASP ASVS 5.0.0 provides a verification standard for web application technical controls and is particularly relevant because it includes requirements for authentication, session management, access control, API behavior, error handling, logging, and business logic [5]. OWASP WSTG complements ASVS by describing testing activities and reporting expectations [6].

NIST SSDF describes secure software development practices for mitigating software vulnerability risk, while CISA Secure by Design guidance frames secure defaults, product accountability, and reduction of customer security burden as core software manufacturer responsibilities [8], [19], [20]. MITRE CWE provides weakness families that can map benchmark scenarios to well-known categories such as improper access control, improper authentication, missing authentication for critical function, insufficient session expiration, authorization bypass through user-controlled key, missing authorization, and incorrect authorization [12]-[18]. OAuth 2.0 security best current practice and OpenID Connect specifications help ground authentication and token lifecycle scenarios in real identity protocols [21]-[23].

2.4 Limitations of Existing Work

Existing work has four main limitations for this article's purpose. First, many benchmarks emphasize source-code-level flaws or input-driven vulnerabilities, which are important but do not fully capture stateful authorization semantics. Second, scanner-comparison studies often evaluate whether tools detect known classes without documenting the role, session, object, and workflow state required to reproduce authorization failures. Third, access-control studies vary in their threat models and

assumptions, making comparison difficult across manual, automated, and AI-assisted approaches. Fourth, standards provide requirements and guidance but do not always translate them into benchmark-ready scenario templates.

These limitations do not reduce the value of existing standards or benchmarks. Rather, they identify a missing layer between standards and tool evaluation: a scenario-design model that specifies preconditions, actors, state, expected secure behavior, insecure behavior, evidence requirements, and mapping to standards. AuthStateBench is intended to provide that layer.

2.5 Summary of Research Gap

The literature shows strong interest in web application security, access-control analysis, scanner evaluation, software-assurance datasets, and AI-assisted vulnerability detection. It also shows a persistent gap: stateful authorization and authentication failures require benchmark scenarios that encode role, session, object ownership, and workflow conditions. Without such encoding, researchers and practitioners risk comparing tools and methods on unclear or incomplete assumptions. AuthStateBench responds to this gap by proposing a structured benchmark design rather than claiming empirical results.

Table 1. Literature search strategy and source categories.

Source category	Examples	Purpose in the review
Academic literature	IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, Wiley, Taylor & Francis, Google Scholar discovery	Identify peer-reviewed studies on access-control testing, stateful web testing, logic flaws, benchmarks, scanners, and AI-assisted vulnerability analysis.
Security standards and guidance	OWASP Top 10:2025, OWASP API Security Top 10, OWASP ASVS 5.0.0, OWASP WSTG, NIST SSDF, CISA Secure by Design	Ground scenario categories in recognized application-security and secure-development expectations.
Weakness taxonomies	MITRE CWE families for	Map benchmark scenarios

	access control, authentication, authorization, session expiration, and user-controlled keys	to common weakness identifiers and improve traceability.
Benchmark resources	OWASP Benchmark, NIST SARD, Juliet, SARD documentation, ProFuzzBench, SWaTEval	Compare existing benchmark assumptions with the proposed stateful scenario design.
Identity and risk specifications	OAuth 2.0 Security BCP, OpenID Connect, CVSS, EPSS	Support authentication, token lifecycle, session, and risk-evidence interpretation.

Table 2. Inclusion and exclusion criteria.

Criterion type	Included	Excluded
Topic relevance	Web application security, API security, authentication, authorization, session management, workflow abuse, benchmark design, scanner evaluation, AI-assisted testing	Generic cybersecurity with no web application relevance; malware-only, blockchain-only, IoT-only, or network-only studies.
Method relevance	Studies proposing, evaluating, surveying, or systematizing testing methods, standards, benchmarks, scanners, taxonomies, or frameworks	Papers that mention tools without explaining vulnerability modeling or test design.
Source quality	Peer-reviewed papers, official standards, recognized cybersecurity guidance, and well-established benchmark	Unverifiable blogs, marketing pages, unsupported claims, and sources without sufficient technical relevance.

	resources	
Time period	Primarily 2020-2026, with older foundational work retained where it shaped access-control or benchmark research	Older material without continuing relevance or citation value.
Integrity boundary	Sources compatible with a literature-based design article without fabricated results	Studies requiring private datasets, unauthorized testing, or non-reproducible company-only evidence.

Table 3. Literature comparison on stateful web security testing.

Research stream	Representative sources	Strength	Limitation addressed by AuthStateBench
Static access-control analysis	Sun et al. [29]	Can infer access-control assumptions from source code and detect role-related weaknesses.	Not suitable when source code is unavailable; benchmark scenarios still need stateful documentation.
Black-box state or logic testing	BLOCK [30], Pellegrino and Balzarotti [32], Li et al. [34]	Models behavior from interactions and can address logic or state violations.	Different studies use different assumptions; AuthStateBench standardizes role-session-object-workflow dimensions.
Parameter tampering and IDOR/BOLA analysis	NoTamper [33], Rennhard et al. [26], BACScan [28]	Targets object identifiers, request parameters, and access-control	Object ownership and victim/attacker role conditions need explicit benchmark

		violations.	representation.
General vulnerability benchmarks	OWASP Benchmark [7], SARD and Juliet [9]-[11]	Useful for repeatable tool evaluation and known test cases.	Primarily stronger for code-level or input-driven weaknesses than multi-user workflow semantics.
AI-assisted vulnerability detection	Xu et al. [41], Far et al. [42], CVE-Bench [44], Cybench [45]	Highlights growing role of AI in security testing and exploitation reasoning.	Requires clearer task specifications to evaluate semantic authorization and authentication workflow behavior.

3. Proposed Framework / Benchmark / Model

3.1 Conceptual Basis

AuthStateBench is built on the premise that authorization and authentication vulnerabilities are not adequately represented by a single request or payload. They must be represented as stateful security-policy failures. The benchmark therefore uses four state dimensions: role state, session state, object-ownership state, and workflow state. Role state describes the identity and privilege level of the actor. Session state describes token validity, freshness, login/logout condition, reauthentication status, and privilege-transition effects. Object-ownership state describes whether the target resource is owned by, shared with, hidden from, or unrelated to the actor. Workflow state describes whether the requested action occurs in the expected business sequence.

The benchmark design treats a vulnerability scenario as a controlled policy test. Each scenario begins with a documented precondition, an attacker role, an optional victim or target role, a protected object, a session condition, a workflow condition, an action, expected secure behavior, observed insecure behavior in a vulnerable implementation, and a standards/CWE mapping. This approach allows future implementation without relying on private systems or unauthorized exploitation.

To make the design explicit, each benchmark scenario is represented as a stateful policy-test tuple rather than a single vulnerable request:

$$B_s = \langle R_s, S_s, O_s, W_s, A_s, P_s, E_s \rangle \quad (1)$$

where R_s is role state, S_s is session state, O_s is object-ownership state, W_s is workflow state, A_s is the attempted action, P_s is the expected policy decision, and E_s is the required evidence record.

Scenario coverage can later be computed as:

$$Coverage = |C_{tested} \cap C_{required}| / |C_{required}| \quad (2)$$

A future method-comparison study may score evidence quality using a weighted criterion model:

$$Score_m = \Sigma_{(k=1..n)} w_k x_{m,k} \quad (3)$$

where x_m,k denotes whether method m satisfies criterion k, and w_k allows future researchers to prioritize scenario recognition, precondition handling, evidence quality, false-positive control, and reproducibility.

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27

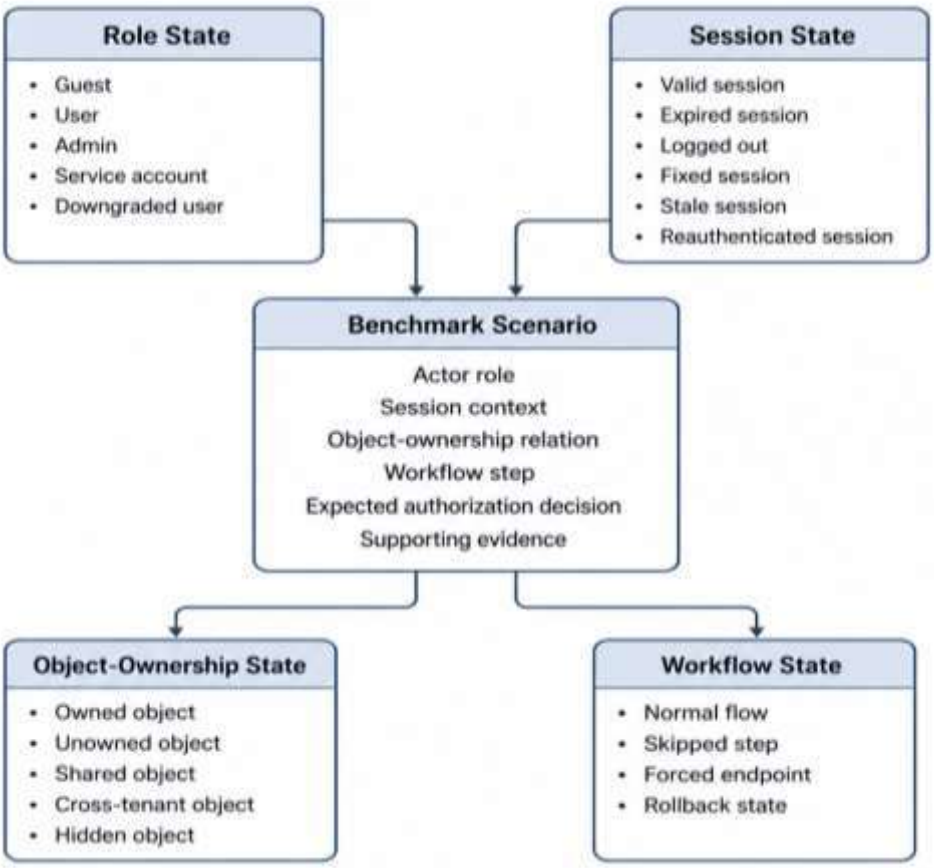

Fig. 2. Four-dimensional state model used to construct AuthStateBench scenarios.

3.2 Main Components

AuthStateBench contains five components. The first component is the scenario taxonomy, which groups benchmark cases into recurring classes such as object-level authorization failure, function-level authorization failure, privilege escalation, session lifecycle failure, workflow bypass, role-confusion failure, tenant-isolation failure, and reauthentication failure. The second component is the state matrix, which combines role, session, object, and workflow conditions. The third component is the standards-mapping layer, which connects scenarios to OWASP, NIST, CISA, MITRE, OAuth, and OpenID guidance. The fourth component is the scenario documentation template. The fifth component is the evaluation criteria set, which supports future comparison of manual testing, scanner-assisted testing, AI-assisted testing, and standards-based review.

These components are designed to be modular. A researcher can use the taxonomy to classify scenarios, the template to document cases, the standards mapping to justify relevance, and the evaluation criteria to compare methods. A practitioner can use the same structure for training, secure-code review, and controlled laboratory exercises.

Fig. 3 operationalizes the benchmark construction sequence. The process starts from a security policy, converts it into allowed and denied state pairs, records evidence, maps the scenario to standards, and then allows a future evaluator to compare testing methods.

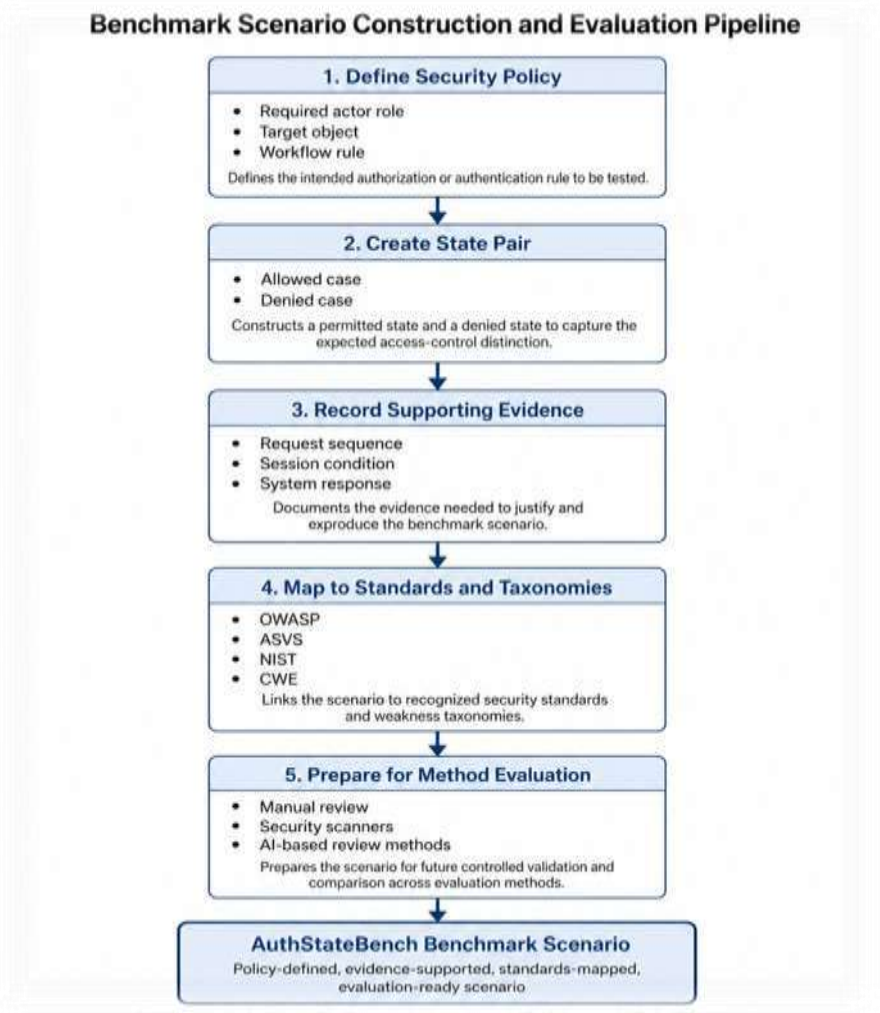

Fig. 3. Benchmark scenario construction and evaluation pipeline.

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27

3.3 Standards or Literature Mapping

The standards-mapping layer is essential because it prevents the benchmark from becoming an arbitrary list of invented scenarios. Broken access control maps to OWASP A01:2025 and to API-level risks such as BOLA and BFLA [2]-[4]. Authentication and session-management scenarios map to ASVS requirements, OAuth 2.0 security guidance, OpenID Connect, and CWE categories for improper authentication, missing authentication, insufficient session expiration, and token misuse [5], [13]-[15], [21]-[23]. Secure-development and vulnerability-management relevance maps to NIST SSDF, CISA Secure by Design, CVSS, EPSS, and benchmark literature [8], [19], [20], [24], [25], [37]-[40].

3.4 Evaluation Logic

AuthStateBench does not present accuracy, precision, recall, F1-score, exploit success, scanner results, or AI-agent performance. Instead, it defines how future studies can evaluate such outcomes responsibly. A future evaluation can compare whether a method identifies the correct scenario class, recognizes the required preconditions, distinguishes authentication from authorization, determines object ownership, checks workflow order, explains evidence, and maps findings to standards. This design avoids false empirical claims while still providing a concrete foundation for empirical work.

3.5 Justification

Stateful authorization and authentication weaknesses require scenario-based benchmark design because isolated request testing is insufficient. For example, a GET request for /invoice/124 may be secure for the owner and insecure for a different user. A POST request that approves a transaction may be correct after review and insecure before review. A session token may be valid before logout and insecure if accepted afterward. These cases cannot be benchmarked by request shape alone; they require documented state. AuthStateBench makes this state explicit and therefore improves reproducibility, comparability, and standards alignment.

Table 4. AuthStateBench scenario taxonomy.

Scenario class	Core failure	Typical state dimensions	Example secure expectation
Object-level authorization failure	Actor accesses an object that belongs	Role state + object ownership + session	The server checks ownership or tenant

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27

	to another user, tenant, or role.	validity	boundary for every object access.
Function-level authorization failure	Actor reaches a function outside permitted role or permission scope.	Role state + workflow state	The server enforces function permissions independent of hidden UI controls.
Privilege escalation	Actor gains elevated capability by manipulating role, token, endpoint, or transition.	Role state + session state + workflow state	Privilege transitions require server-side authorization and reauthentication where appropriate.
Session lifecycle failure	Expired, logged-out, fixed, or stale tokens remain usable.	Session state + role state	Tokens are invalidated and refreshed according to security requirements.
Workflow bypass	Actor skips or reorders required process steps.	Workflow state + role state + object ownership	Business operations require all preconditions and state transitions.
Role-confusion failure	Application confuses guest, user, privileged user, admin, or downgraded role.	Role state + session state	Server-side policy resolves role correctly after login, logout, downgrade, or account changes.
Tenant-isolation failure	Actor crosses organization, workspace, or tenant boundary.	Object ownership + role state	Tenant boundary is enforced for every resource and function.
Reauthentication failure	Sensitive action proceeds without	Session state + workflow state	High-risk operations require

	fresh authentication or step-up verification.		fresh identity assurance or equivalent control.

Table 5. Role-session-object-workflow state matrix.

Dimension	Representative states	Security question	Failure indicator
Role state	Guest, registered user, privileged user, admin, downgraded user, service account	Is the actor allowed to perform this function?	Function succeeds for a role outside intended permission scope.
Session state	Valid, expired, reused, fixed, logged out, token changed, privilege changed, reauthenticated	Is the session state acceptable for the requested action?	Action succeeds with stale, invalid, fixed, or insufficiently fresh session context.
Object-ownership state	Owned object, unowned object, shared object, hidden object, tenant-specific object	Does the actor have rights over this specific object instance?	Object data or action succeeds across ownership or tenant boundary.
Workflow state	Normal sequence, skipped step, repeated step, forced endpoint, post-approval state, rollback state	Has the process reached the required business state?	Endpoint allows action before required preconditions or after invalid transition.

Table 6. OWASP Top 10 / ASVS / NIST SSDF / CWE mapping table.

Benchmark class	OWASP mapping	ASVS / WSTG mapping	NIST / CISA mapping	CWE mapping
Object-level authorization	OWASP A01:2025;	Access control and API testing	SSDF verification and	CWE-284, CWE-862,

failure	API1:2023 BOLA	requirements	vulnerability response; Secure by Design default protection	CWE-863, CWE-639
Function-level authorization failure	OWASP A01:2025; API5:2023 BFLA	Server-side authorization verification; business logic testing	Secure design review and threat modeling	CWE-862, CWE-863
Authentication workflow failure	OWASP authentication-related risks; API2:2023 Broken Authentication	Authentication and identity verification requirements	SSDF secure design and verification	CWE-287, CWE-306
Session lifecycle failure	Broken access control and authentication-adjacent risk	Session management verification; logout and timeout testing	Secure default session behavior	CWE-613, CWE-287
Workflow bypass	Business logic abuse; broken access control	Business logic and workflow testing	Threat modeling and secure requirements	CWE-840, CWE-863, CWE-862
Tenant-isolation failure	Broken access control; API object-level access control	Access control and data isolation verification	Secure architecture and product safety	CWE-284, CWE-862, CWE-863
Reauthentication failure	Broken access control; authentication	Fresh authentication for sensitive	Identity assurance and secure defaults	CWE-287, CWE-306

	control weakness	actions		

Table 7. Benchmark scenario template.

Field	Description
Scenario ID	Unique identifier such as ASB-OBJ-001 or ASB-SES-003.
Scenario class	Taxonomy category, such as object-level authorization failure or session lifecycle failure.
Attacker role	The role from which the unauthorized action is attempted.
Victim/target role	The role or account owning the target resource, if applicable.
Object ownership condition	Owned, unowned, shared, hidden, cross-tenant, or system-owned object.
Session condition	Valid, expired, logged out, token refreshed, privilege changed, stale, fixed, or reauthenticated.
Workflow precondition	Normal sequence, skipped stage, repeated stage, forced endpoint, pre-approval, post-approval, or rollback.
Attack action	Abstract action attempted in the controlled benchmark scenario.
Expected secure behavior	Policy decision that should occur in a secure implementation.
Insecure behavior	Failure condition that marks the scenario vulnerable in an intentionally vulnerable implementation.
Standards mapping	OWASP, ASVS, WSTG, NIST, CISA, MITRE CWE, OAuth/OIDC mapping as applicable.
Evidence requirement	What future testers must record: request sequence, session state, role pair, object ID

	relation, response, and policy rationale.

4. Methodology

4.1 Research Design

The research design combined structured literature review, standards mapping, conceptual synthesis, and benchmark design. The study was not conducted as an empirical tool-evaluation experiment. No scanner was executed, no vulnerable laboratory was deployed, no live target was tested, and no private dataset was analyzed. The method instead used literature and standards to derive a benchmark design that can later support implementation and evaluation.

This design is appropriate because the contribution is a scenario-construction model. A benchmark-design article must first define what counts as a scenario, what security property is being tested, what state must be recorded, and how relevance is mapped to standards before performance claims can be evaluated.

4.2 Search Strategy / Data Source Strategy

The search strategy used combinations of terms such as "broken access control web application testing," "authorization vulnerability benchmark," "authentication workflow vulnerability," "stateful web application security testing," "IDOR BOLA benchmark," "role-based access control web vulnerability," "workflow bypass web security," "session management vulnerability testing," "OWASP ASVS access control requirements," "web vulnerability benchmark evaluation," and "AI-assisted vulnerability detection benchmark." Searches prioritized peer-reviewed databases and official standards sources. Google Scholar was used for discovery, while preference was given to publisher pages, official project pages, government guidance, RFCs, and standards pages where available.

The source base included academic literature on access-control analysis, stateful web testing, web logic flaws, scanner evaluation, benchmarks, and AI-assisted vulnerability detection [26]-[46]. It also included standards and guidance from OWASP, NIST, MITRE, CISA, FIRST, OAuth, OpenID, and ISO/IEC [1]-[25], [47]-[49].

4.3 Inclusion and Exclusion Criteria

Sources were included when they focused on web application security, API security, authentication, authorization, session management, workflow abuse, benchmark design,

vulnerability detection, scanner evaluation, or secure-development guidance. Foundational older sources were retained when they introduced important concepts or methods for access-control vulnerability detection or web logic testing. Sources were excluded when they focused only on generic cybersecurity, malware, blockchain, IoT, or network intrusion without web application relevance, or when they made unsupported claims about automation replacing human security testing.

4.4 Screening or Selection Process

The screening process followed a transparent review approach rather than a fully quantified PRISMA systematic review. Because exact search counts, duplicate counts, and exclusion counts were not recorded in a formal review registry, this article does not claim a completed PRISMA study. Instead, sources were screened by title, abstract, technical relevance, standards relevance, and contribution to scenario modeling. Selected sources were then coded according to vulnerability type, testing approach, state dimension, benchmark relevance, and standards applicability.

4.5 Coding and Synthesis Method

Thematic synthesis grouped the literature into four state dimensions: role state, session state, object-ownership state, and workflow state. Role state captured guest, user, privileged user, admin, downgraded user, and service-account contexts. Session state captured valid, expired, reused, fixed, logged-out, token-changed, and reauthenticated contexts. Object state captured owned, unowned, shared, hidden, tenant-specific, and system-owned objects. Workflow state captured normal sequence, skipped step, repeated step, forced endpoint, pre-approval, post-approval, and rollback conditions.

Fig. 4 illustrates how literature, standards, and weakness taxonomies were synthesized into the benchmark outputs. The figure also makes clear that the article is a design study rather than a tool-execution experiment.

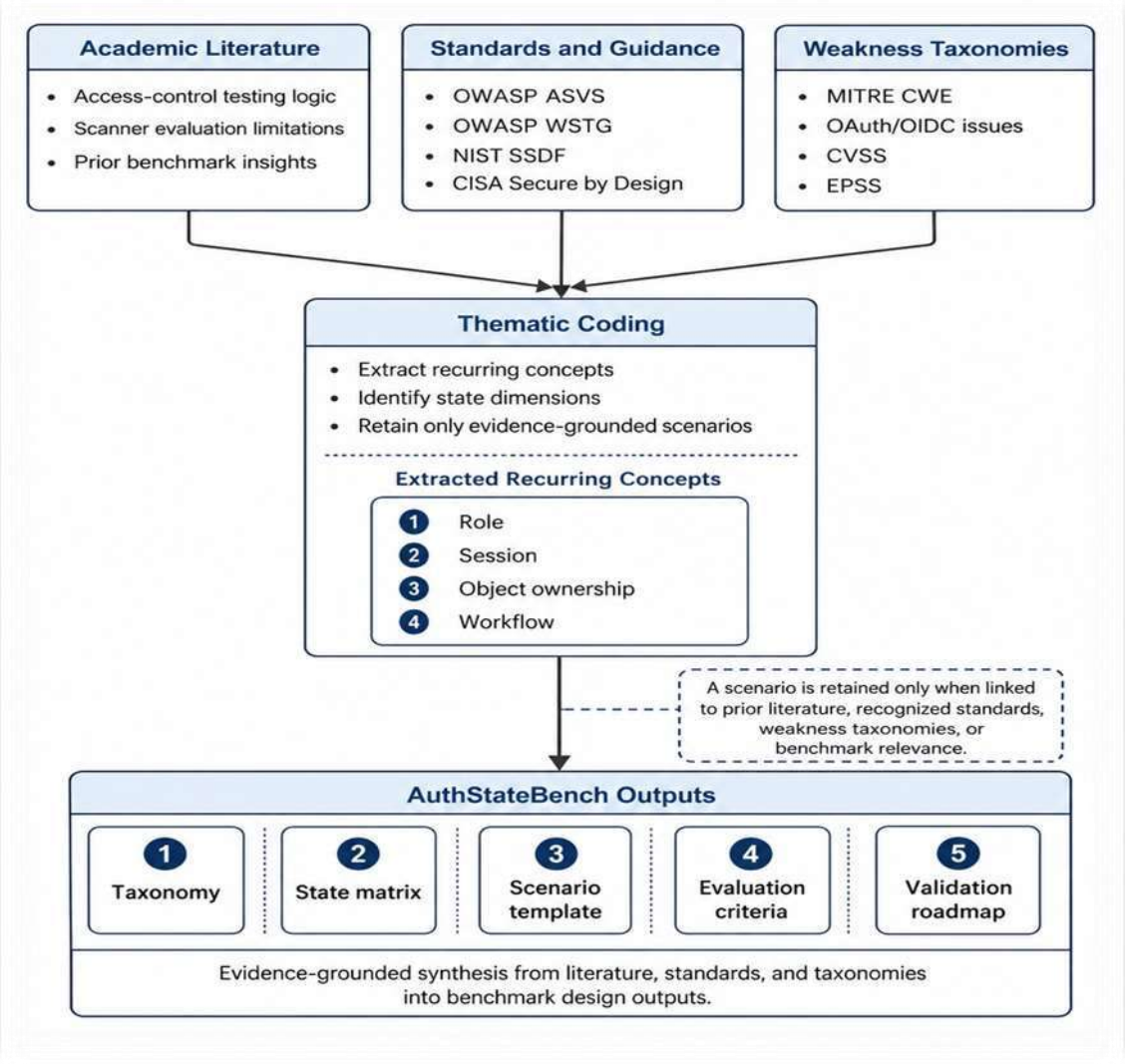

Fig. 4. Literature and standards synthesis process used to derive AuthStateBench.

Algorithm 1. AuthStateBench scenario construction procedure.

Input: Candidate weakness pattern, relevant standard clauses, state dimensions, and expected security policy.

1. Identify the protected action and target object.
2. Define the legitimate role, unauthorized role, session condition, and workflow precondition.
3. Specify the expected secure decision and vulnerable behavior to be represented in a

controlled implementation.

4. Map the scenario to OWASP, ASVS/WSTG, NIST/CISA, MITRE CWE, and identity-protocol guidance where applicable.

5. Record evidence requirements: actor role, object relation, session state, request sequence, response, and policy rationale.

Output: A benchmark-ready scenario record that can later be implemented and evaluated in a controlled testbed.

After coding, recurring vulnerability patterns were converted into benchmark scenario classes. Each scenario class was checked against relevant standards and weakness taxonomies to verify that it corresponded to recognized security concerns rather than arbitrary examples.

4.6 Comparison Criteria

Existing approaches were compared using standards alignment, explicit state modeling, reproducibility, evaluation readiness, evidence requirements, ability to support manual testing, ability to support scanner-assisted testing, ability to support AI-assisted testing, and suitability for secure-development education. These criteria were selected because a benchmark design must be useful across research and practice, not only within a single tool category.

4.7 Validity and Reliability

Validity was supported through triangulation across peer-reviewed literature, official standards, weakness taxonomies, and benchmark resources. Reliability was supported by using explicit scenario fields and state dimensions rather than narrative-only descriptions. The main validity limitation is that the benchmark has not yet been implemented as runnable vulnerable applications or evaluated by independent experts. A recommended next step is expert review by application-security researchers, professional penetration testers, and secure software engineers.

4.8 Ethical Considerations

The study does not involve human subjects, real user data, live-system testing, unauthorized access, exploit deployment, screenshots, tool execution, or private vulnerability findings. Scenarios are described abstractly and are intended for controlled educational or research environments. Any future implementation should use

intentionally vulnerable applications, local testbeds, explicit permission, and responsible disclosure principles where applicable.

Table 8. Evaluation criteria for manual, scanner-assisted, and AI-assisted testing.

Criterion	Manual testing	Scanner-assisted testing	AI-assisted testing	Standards-based review
Scenario recognition	Tester identifies role, session, object, and workflow conditions.	Tool must discover or be configured with relevant state.	Model must infer or be provided with scenario state.	Reviewer checks whether requirement coverage matches scenario.
Precondition handling	Strong when testers control accounts and workflows.	Often limited without authenticated crawling and multi-user support.	Variable; depends on prompts, context windows, and tool integration.	Strong for policy completeness but not a runtime proof.
Evidence quality	Request sequence, account roles, object relation, and response evidence.	Automated traces plus manual confirmation.	Explanation must cite observed evidence, not only plausible reasoning.	Traceable checklist and requirement mapping.
False-positive control	Human judgment can validate business context.	May flag request differences without understanding policy.	May hallucinate policy unless constrained by evidence.	Can miss runtime behavior if review is document-only.
Reproducibility	Requires documented steps and	Requires stable crawler state and	Requires fixed prompts, logs, and scenario	Requires versioned standards and

	accounts.	login/session handling.	context.	mapping rationale.
Best use	Deep semantic testing and business-logic reasoning.	Broad coverage and repeatable baseline scanning.	Assisted reasoning, test planning, and evidence summarization.	Requirements traceability and secure-development governance.

5. Results and Analytical Outputs

5.1 Thematic Findings

The synthesis produced four thematic findings. First, access-control failures are semantic vulnerabilities: whether a request is secure depends on who sends it, what object is targeted, what session state exists, and what workflow state applies. Second, automated scanners can provide valuable coverage but may struggle with multi-user authorization, object ownership, and business process semantics unless state and credentials are explicitly modeled. Third, standards provide strong control expectations, but additional benchmark documentation is needed to turn those expectations into reproducible test cases. Fourth, AI-assisted testing can support security reasoning, but it requires structured task definitions and evidence constraints to avoid plausible but unsupported conclusions.

5.2 Gap Mapping

The gap mapping shows that existing resources support parts of the problem but not the complete stateful benchmark requirement. OWASP and NIST standards define what secure behavior should look like. MITRE CWE defines weakness families. OWASP Benchmark and SARD provide tool-evaluation resources. Access-control research proposes detection approaches. AI-security research highlights emerging automated reasoning capabilities. AuthStateBench integrates these strands by representing each benchmark scenario as a stateful policy test with traceable standards mapping.

5.3 Framework or Benchmark Outputs

The primary benchmark outputs are the scenario taxonomy, the state matrix, the standards mapping, and the scenario template presented in Tables 4-7. These outputs allow a future benchmark implementation to include representative scenario families rather than isolated examples. For instance, an object-level authorization scenario can be

instantiated with different combinations of roles, tenants, session states, and object-sharing rules. A workflow bypass scenario can be instantiated with skipped approval, repeated confirmation, forced endpoint access, or post-rollback actions. This modular design supports expansion while preserving consistent documentation.

5.4 Standards Alignment

AuthStateBench aligns with standards at three levels. At the risk-taxonomy level, it maps to OWASP Top 10:2025 and OWASP API Security Top 10:2023 [1]-[4]. At the verification level, it maps to OWASP ASVS 5.0.0 and WSTG testing concerns [5], [6]. At the secure-development and weakness-classification level, it maps to NIST SSDF, CISA Secure by Design guidance, and MITRE CWE categories [8], [12]-[20]. Identity-protocol references such as OAuth 2.0 Security BCP and OpenID Connect support token, authentication, and session lifecycle scenarios [21]-[23].

5.5 Comparison with Existing Approaches

Compared with generic OWASP reviews, AuthStateBench adds scenario-level reproducibility. Compared with scanner evaluations, it documents state conditions that tools must handle or be given. Compared with code-level vulnerability datasets, it emphasizes policy semantics and multi-step workflows. Compared with AI-security benchmarks, it provides domain-specific scenario templates for authorization and authentication workflow reasoning. It therefore does not replace existing benchmarks or standards; it complements them by filling a stateful scenario-design gap.

Table 9. Research gap matrix.

Existing resource	What it provides	Gap for stateful authorization/authentication	AuthStateBench response
OWASP Top 10 / API Top 10	Risk categories for web and API security.	Risk category does not by itself specify role, session, object, and workflow preconditions.	Converts risk categories into scenario classes.
OWASP ASVS / WSTG	Verification and testing guidance.	Requirements need benchmark-ready scenario fields for comparison studies.	Maps scenarios to requirements and evidence expectations.

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27

NIST SSDF / CISA Secure by Design	Secure-development governance and product-security principles.	Guidance is broad and not a vulnerability test-case benchmark.	Links benchmark scenarios to secure-development and validation activities.
MITRE CWE	Weakness family taxonomy.	CWE IDs alone do not define multi-user workflow conditions.	Uses CWE as traceability, not as the entire scenario definition.
OWASP Benchmark / SARD / Juliet	Repeatable test cases for tool evaluation.	Less focused on multi-step authorization and authentication workflow semantics.	Adds stateful scenario design for auth/access-control cases.
AI vulnerability benchmarks	Evaluate AI or agentic security capabilities.	May not isolate authorization workflow reasoning from exploit execution.	Defines structured tasks for future AI-assisted testing comparison.

Table 10. Future validation and research roadmap.

Stage	Purpose	Recommended activity	Expected output
Stage 1: standards alignment review	Check whether mappings are accurate and complete.	Compare each scenario class with OWASP, ASVS, WSTG, NIST, CISA, MITRE, OAuth, and OpenID sources.	Validated mapping table and revised scenario definitions.
Stage 2: expert review	Assess practical realism and clarity.	Invite 3-5 application-security academics, penetration testers, or secure software	Expert feedback matrix and updated taxonomy.

		engineers.	
Stage 3: controlled implementation	Turn abstract scenarios into local vulnerable applications.	Implement representative scenarios in a testbed with documented accounts and workflows.	Runnable benchmark prototype and ground-truth labels.
Stage 4: method comparison	Evaluate manual, scanner-assisted, AI-assisted, and standards-based approaches.	Run controlled tests without live targets and record evidence quality.	Comparative evaluation results with transparent limitations.
Stage 5: public release	Enable reuse and replication.	Publish documentation, templates, scenario definitions, and implementation notes.	Versioned public benchmark artifact.

6. Discussion

6.1 Key Insights

The first key insight is that stateful authorization and authentication testing must be framed as policy-state verification rather than payload detection. A payload-centered view is useful for injection and many input-driven flaws, but it is insufficient for understanding whether an authenticated user should access a resource, whether a tenant boundary should apply, or whether a workflow state authorizes an operation. The second insight is that standards mapping improves benchmark legitimacy, but standards alone do not create benchmark reproducibility. The third insight is that manual, scanner-assisted, and AI-assisted approaches should not be treated as interchangeable. Each has different strengths and evidence requirements.

6.2 Cybersecurity Implications

AuthStateBench has implications for application-security research and cyber defense. For researchers, it provides a way to define comparable tasks for access-control and authentication workflow testing. For tool builders, it clarifies what state information a scanner or AI-assisted system must handle. For defenders, it encourages evidence-based assessment of whether authorization and session controls enforce server-side policy across realistic business states. For educators, it offers a structured way to teach junior penetration testers why authentication, authorization, session management, and workflow logic must be tested together.

6.3 Practical and Policy Implications

In practice, the benchmark design can support secure development lifecycle activities. Requirements engineers can use the state matrix to document authorization rules. Developers can use scenario templates to write tests for object ownership and workflow preconditions. Penetration testers can use the taxonomy to structure manual testing evidence. Security managers can map findings to standards for reporting and remediation prioritization. Policy teams can use the structure to connect secure-by-design expectations with concrete verification evidence.

6.4 Limitations

The article has clear limitations. AuthStateBench is a design artifact, not a completed empirical benchmark implementation. No tool results are reported, and no claim is made about detection accuracy. The taxonomy may require refinement after expert review and controlled implementation. Some standards references may evolve, so mappings should be versioned. The current design focuses on web applications and APIs and may not directly apply to mobile, IoT, blockchain, or low-level protocol systems without adaptation.

6.5 Future Research Roadmap

Future work should proceed in five steps. First, an independent standards-alignment review should verify scenario mappings. Second, expert review should assess practical realism and coverage. Third, selected scenarios should be implemented in controlled vulnerable web applications with versioned documentation. Fourth, manual, scanner-assisted, AI-assisted, and standards-based methods should be compared using explicit evidence criteria. Fifth, the benchmark should be released as a public artifact with

scenario definitions, implementation notes, ground-truth labels, and clear ethical-use guidance.

Fig. 5 converts the future-work discussion into a validation roadmap. This separation between design and empirical validation is important because the current article does not claim scanner accuracy or AI-agent performance.

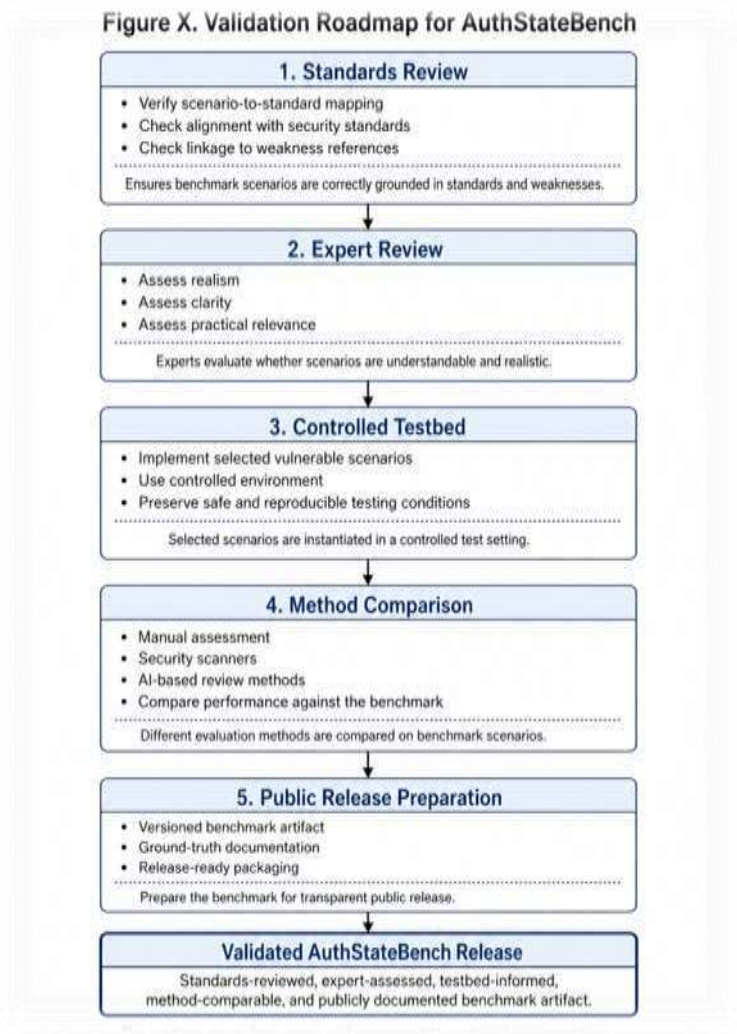

Fig. 5. Future validation roadmap for turning AuthStateBench into an empirical benchmark.

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27

7. Conclusion

Stateful authorization and authentication workflow weaknesses remain difficult to benchmark because their security depends on role, session, object ownership, and workflow conditions. Existing standards and benchmarks provide valuable foundations, but they do not fully solve the problem of scenario-level reproducibility for semantic access-control and authentication failures. AuthStateBench addresses this gap by proposing a standards-aligned benchmark design that includes a four-dimensional state model, scenario taxonomy, standards mapping, scenario template, evaluation criteria, and validation roadmap.

The article deliberately avoids unsupported empirical claims. It does not report scanner results, AI-agent performance, exploit success, or real-system testing. Its contribution is a design framework that future researchers and practitioners can implement and evaluate transparently. By making stateful conditions explicit, AuthStateBench can improve comparability in web security testing, support secure-development education, and provide a foundation for more rigorous evaluation of manual, scanner-assisted, AI-assisted, and standards-based approaches to authorization and authentication workflow security.

Declarations

Funding

No funding was received for this work.

Competing Interests

The author declares no competing interests.

Ethics Approval

Not applicable. This article is a literature-based and standards-aligned benchmark-design study and does not involve human participants, personal data, live-system testing, or animal subjects.

Consent for Publication

Not applicable.

Data Availability

No empirical dataset was generated or analyzed. The article is based on publicly available literature, standards, and guidance sources cited in the reference list. Future benchmark scenarios should be released as versioned documentation if implemented.

Author Contributions

Muhammad Shahzad Khadim conceptualized the study, designed the methodology, developed the benchmark model, and wrote the manuscript. Syed Mufassir Shah contributed to literature review, data organization, and manuscript editing. Zubair Khan assisted in validation design, formatting, and final proofreading.

Acknowledgements

The author acknowledges the public work of OWASP, NIST, MITRE, CISA, FIRST, IETF, OpenID Foundation, and the academic security research community whose standards and studies informed this conceptual benchmark design.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work, the author used ChatGPT by OpenAI to assist with language refinement, formatting improvement, clarity enhancement, and manuscript organization. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the content of the submitted manuscript.

References

[1] OWASP Foundation. OWASP Top Ten Web Application Security Risks 2025. OWASP, 2025. Accessed 27 April 2026.

[2] OWASP Foundation. A01:2025 - Broken Access Control. OWASP Top 10:2025. Accessed 27 April 2026.

[3] OWASP Foundation. OWASP API Security Top 10 2023. OWASP, 2023. Accessed 27 April 2026.

[4] OWASP Foundation. API1:2023 - Broken Object Level Authorization. OWASP API Security Top 10 2023. Accessed 27 April 2026.

[5] OWASP Foundation. OWASP Application Security Verification Standard 5.0.0. OWASP, 2025. Accessed 27 April 2026.

[6] OWASP Foundation. OWASP Web Security Testing Guide, latest release. OWASP. Accessed 27 April 2026.

[7] OWASP Foundation. OWASP Benchmark Project. OWASP. Accessed 27 April 2026.

[8] M. Souppaya, K. Scarfone, and D. Dodson. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. NIST SP 800-218, 2022.

[9] NIST SAMATE. Software Assurance Reference Dataset (SARD). National Institute of Standards and Technology. Accessed 27 April 2026.

[10] P. E. Black. The Software Assurance Reference Dataset. NIST Internal Report 8561, 2025.

[11] F. E. Boland Jr. and P. E. Black. The Juliet 1.1 C/C++ and Java Test Suite. National Institute of Standards and Technology, 2012.

[12] MITRE. CWE-284: Improper Access Control. Common Weakness Enumeration. Accessed 27 April 2026.

[13] MITRE. CWE-287: Improper Authentication. Common Weakness Enumeration. Accessed 27 April 2026.

[14] MITRE. CWE-306: Missing Authentication for Critical Function. Common Weakness Enumeration. Accessed 27 April 2026.

[15] MITRE. CWE-613: Insufficient Session Expiration. Common Weakness Enumeration. Accessed 27 April 2026.

[16] MITRE. CWE-639: Authorization Bypass Through User-Controlled Key. Common Weakness Enumeration. Accessed 27 April 2026.

[17] MITRE. CWE-862: Missing Authorization. Common Weakness Enumeration. Accessed 27 April 2026.

[18] MITRE. CWE-863: Incorrect Authorization. Common Weakness Enumeration. Accessed 27 April 2026.

[19] CISA. Secure by Design. Cybersecurity and Infrastructure Security Agency. Accessed 27 April 2026.

[20] CISA and international partners. Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Secure by Design Software, 2023.

[21] T. Lodderstedt, J. Bradley, A. Labunets, and D. Fett. Best Current Practice for OAuth 2.0 Security. RFC 9700, BCP 240, RFC Editor, 2025. doi:10.17487/RFC9700.

[22] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. OpenID Connect Core 1.0 incorporating errata set 2. OpenID Foundation. Accessed 27 April 2026.

[23] OpenID Foundation. OpenID Connect Session Management 1.0. Final specification, 2022.

[24] FIRST. Common Vulnerability Scoring System Version 4.0: Specification Document. Forum of Incident Response and Security Teams. Accessed 27 April 2026.

[25] FIRST. Exploit Prediction Scoring System (EPSS). Forum of Incident Response and Security Teams. Accessed 27 April 2026.

[26] M. Rennhard, et al. Automating the Detection of Access Control Vulnerabilities in Web Applications. SN Computer Science, 2022.

[27] L. Zhong, et al. A Survey of Prevent and Detect Access Control Vulnerabilities in Web Applications. arXiv preprint arXiv:2304.10600, 2023.

[28] F. Liu, et al. BACScan: Automatic Black-Box Detection of Broken Access-Control Vulnerabilities. ACM CCS, 2025.

[29] F. Sun, L. Xu, and Z. Su. Static Detection of Access Control Vulnerabilities in Web Applications. USENIX Security Symposium, 2011.

[30] X. Li and Y. Xue. BLOCK: A Black-box Approach for Detection of State Violation Attacks Towards Web Applications. ACSAC, 2011.

[31] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna. Toward Automated Detection of Logic Vulnerabilities in Web Applications. USENIX Security Symposium, 2010.

[32] G. Pellegrino and D. Balzarotti. Toward Black-Box Detection of Logic Flaws in Web Applications. Network and Distributed System Security Symposium, 2014.

[33] P. P. S. Bisht, T. Hinrichs, N. Skrupsky, R. Bobrowicz, and V. N. Venkatakrishnan. NoTamper: Automatic Blackbox Detection of Parameter Tampering Opportunities in Web Applications. ACM CCS, 2010.

[34] X. Li, Y. Xue, and M. Chu. Automated Black-Box Detection of Access Control Vulnerabilities in Web Applications. SACMAT, 2014.

[35] A. Borcherding, et al. SWaTEval: An Evaluation Framework for Stateful Web Application Testing. International Conference on Web Information Systems and Technologies, 2023.

[36] R. Natella, et al. ProFuzzBench: A Benchmark for Stateful Protocol Fuzzing. ACM ISSTA, 2021.

[37] E. Fong, V. Okun, and R. Gaucher. Web Application Scanners: Definitions and Functions. NIST SAMATE, 2007.

[38] P. Nunes, J. Fonseca, and M. Vieira. Benchmarking Static Analysis Tools for Web Security. IEEE Transactions on Reliability, 2018.

[39] M. Miltenberger, et al. Benchmarking the Benchmarks. ACM, 2023.

[40] N. Risse, et al. On Benchmarking in Machine Learning for Vulnerability Detection. ISSTA, 2025.

[41] H. Xu, S. Wang, N. Li, K. Wang, Y. Zhao, K. Chen, T. Yu, Y. Liu, and H. Wang. Large Language Models for Cyber Security: A Systematic Literature Review. arXiv:2405.04760, 2024.

[42] S. M. Taghavi Far, et al. Large Language Models for Software Vulnerability Detection. International Journal of Information Security, 2025.

[43] Y. Chen, et al. A Survey of Large Language Models for Cyber Threat Detection. Computers & Security, 2024.

[44] Y. Zhu, A. Kellermann, D. Bowman, P. Li, A. Gupta, A. Danda, R. Fang, C. Jensen, E. Ihli, J. Benn, et al. CVE-Bench: A Benchmark for AI Agents' Ability to Exploit Real-World Web Application Vulnerabilities. ICML, 2025.

[45] R. Fang, et al. Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models. arXiv:2408.08926, 2024.

[46] M. Malkawi and R. Alhajj. AI-Powered Vulnerability Detection and Patch Management in Cybersecurity: A Systematic Review of Techniques, Challenges, and Emerging Trends. Machine Learning and Knowledge Extraction, vol. 8, no. 1, Article 19, 2026. doi:10.3390/make8010019.

[47] ISO/IEC. ISO/IEC 27034-1:2011, Information technology - Security techniques - Application security - Part 1: Overview and concepts. International Organization for Standardization, 2011.

[48] ISO/IEC/IEEE. ISO/IEC/IEEE 29119-1:2022, Software and systems engineering - Software testing - Part 1: General concepts. International Organization for Standardization, 2022.

[49] NIST. Digital Identity Guidelines: Authentication and Lifecycle Management, SP 800-63B. National Institute of Standards and Technology, latest available revision accessed 27 April 2026.

GRJNST, Volume: 04 - Issue 2 (2026) / ISSN P: 2790-7643
Article ID: 2076
https://doi.org/10.53762/grjnst.04.02.27