(Libyan Journal of Contemporary Academic Studies) المجلة الليبية للدراسات الأكاديمية المعاصرة
LJCAS, eISSN: 5970-3005
Volume 4, Issue 2, 2026, Pages: 27-54

Comparative Evaluation of OWASP ZAP, Burp Suite Community Edition, and Acunetix for Detecting Web Application Vulnerabilities

Mohsen Ibrahim Mohamed [1*], Esmaeil M. Albakoosh [2]
[1] Department of Information Technology, Higher Institute of Engineering Technology-Bani Walid, Bani Walid, Libya
[2] General Department, Higher Institute of Medical Sciences and Technologies-Bani Walid, Bani Walid, Libya
*Corresponding author: mohsenalahmerr@gmail.com

تقييم مقارن لـ OWASP ZAP وBurp Suite Community Edition وAcunetix لاكتشاف ثغرات تطبيقات الويب

محسن إبراهيم محمد [1*]، إسماعيل محمد البكوش [2]
[1] قسم تقنية المعلومات، المعهد العالي للتقنيات الهندسية - بني وليد، بني وليد، ليبيا
[2] القسم العام، المعهد العالي للتقنيات الطبية، بني وليد، ليبيا

Received: 25-04-2026; **Accepted:** 20-07-2026; **Published:** 05-08-2026

Abstract:
Dynamic Application Security Testing (DAST) is often evaluated by simply counting scanner alerts, even though alert volume conflates crawler reachability, reporting redundancy, vulnerability judgment quality, and false positives. This Stage 1 registered report presents a controlled, reproducible experiment comparing ZAP (formerly OWASP ZAP), Burp Suite Community Edition, and an academic or trial deployment of Acunetix, across the DVWA and OWASP Juice Shop environments. The design evaluates SQL injection, stored and reflected cross-site scripting, command injection, cross-site request forgery, authentication and access control failures, security misconfiguration, sensitive data exposure, path traversal, local and remote file inclusion, weak cookies, missing security headers, open redirects, and server information disclosure; coverage is further extended to map onto the OWASP Top 10:2021. Positive test units and matching negative controls are established before scanner reports are opened.
Two detection modes separate end-to-end crawl effectiveness from reachability-conditioned detection. Ten randomized iterations per scanner, per application, per mode provide timing and resource observations, while double-blind adjudication determines true and false findings. Pre-registered metrics include detection rate, precision, recall, false positive rate, false discovery rate, overall accuracy, F1, OWASP coverage, execution time, CPU and memory consumption, CVSS distribution, severity agreement, remediation utility, report quality, setup effort, usability, learning burden, automation, CI/CD integration, and export support. Paired tests, non-parametric alternatives, confidence intervals, multiple-comparison correction, and effect sizes are also pre-specified.
The capability gap is treated explicitly as a structural difference: Community Edition includes neither Burp Scanner nor automated crawling, so its self-reported DAST results are structurally inapplicable, not zero. Burp Community is therefore evaluated on a separate track explicitly labeled as analyst-assisted and educational in nature. This protocol contributes a falsifiable benchmark, a closed ground-truth model, a weighted, profile-sensitive selection framework, and a complete reproduction package, designed to prevent artificial or incomparable scanner rankings and to support an evidence-based Stage 2 article.

Keywords: Acunetix, Burp Suite Community Edition, DAST, dynamic application security testing, false positives, , reproducible experiments, web vulnerability scanner.

الملخص :
غالبًا ما يُقيَّم اختبار الأمن الديناميكي للتطبيقات (DAST) بعدّ تنبيهات الماسح فقط، رغم أن هذا العدد يخلط بين قابلية الوصول عبر الزاحف، وتكرار الإبلاغ، وجودة الحكم على الثغرات، والإيجابيات الكاذبة. يعرض هذا التقرير المسجَّل من المرحلة الأولى تجربة مضبوطة وقابلة لإعادة الإنتاج تقارن ZAP (المعروف سابقًا باسمOWASP ZAP ، و Burp Suite Community Edition، ونشرًا أكاديميًا أو تجريبيًا لـAcunetix ، على بيئتي DVWA و OWASP Juice Shop.يغطي التصميم حقنSQL ، والبرمجة النصية عبر المواقع (المخزَّنة والمنعكسة)، وحقن الأوامر، وCSRF،

المجلة الليبية للدراسات الأكاديمية المعاصرة (Libyan Journal of Contemporary Academic Studies)
LJCAS, eISSN: 5970-3005
Volume 4, Issue 2, 2026, Pages: 27-54

وإخفاقات المصادقة والتحكم في الوصول، وسوء التهيئة الأمنية، وكشف البيانات الحساسة، واجتياز المسارات، وLFI/RFI، والكوكيز الضعيفة، ورؤوس الأمان المفقودة، وإعادة التوجيه المفتوحة، وكشف معلومات الخادم، مع تغطية إضافية وفق OWASP Top 10:2021.تُثبَّت وحدات اختبار إيجابية وضوابط سلبية مطابقة قبل فتح تقارير الماسح.
يفصل نمطا الاكتشاف بين فعالية الزحف من طرف إلى طرف والكشف المشروط بقابلية الوصول. توفر عشر تكرارات عشوائية لكل ماسح، ولكل تطبيق، ولكل نمط، بيانات زمنية وموارد، بينما يضمن التحكيم المزدوج المُعمّى تمييز النتائج الحقيقية من الكاذبة. تُحدَّد سلفًا مقاييس الدقة، والاسترجاع، ومعدل الإيجابيات الكاذبة، ونسبة الاكتشافات الخاطئة، وF1، والتغطية وفقOWASP ، وزمن التنفيذ، واستهلاك الموارد، وتوزيعCVSS ، والمنفعة العلاجية، وجودة التقرير، وسهولة الاستخدام، والتكامل معCI/CD ، إلى جانب الاختبارات الإحصائية المزدوجة وتصحيح تعدد المقارنات.
تُعامَل الفجوة في القدرات كاختلاف بنيوي: Community Edition تفتقر إلى Burp Scanner والزحف التلقائي، فنتائجها الذاتية غير منطبقة بنيويًا وليست صفرًا، وتُقيَّم ضمن مسار منفصل مدعوم بالمحلّل. يقدّم البروتوكول معيارًا قابلًا للتكذيب، ونموذج حقيقة أرضية مغلق، وإطار اختيار حسّاس للملف، وحزمة إعادة إنتاج كاملة، تمهيدًا لمقال لاحق قائم على الأدلة في المرحلة الثانية.

الكلمات المفتاحية: اختبار أمني ديناميكي للتطبيقات (DAST)أكيونيتيكس، بيرب سويت – الإصدار المجتمعي، ماسح ثغرات الويب، إيجابيات كاذبة(تنبيهات خاطئة تُظهر ثغرات غير موجودة).

1. INTRODUCTION

1.1 Background and Motivation

Web vulnerability scanners occupy an awkward position between software testing and penetration testing. They can exercise a large HTTP surface repeatedly, but their reports are not direct observations of source-code defects. A reported issue is produced by a chain of contingent operations: a crawler must reach a state, the scanner must identify an input location, a payload must survive transformations, an oracle must distinguish vulnerable from benign behavior, and a reporting layer must consolidate evidence. A missed state is therefore observationally indistinguishable from a failed detector unless reachability is measured separately. Conversely, counting every URL instance of a missing header can make a scanner appear productive while contributing little additional security information.

The OWASP Top 10:2021 remains a widely used risk taxonomy [1], while the Web Security Testing Guide (WSTG) and Application Security Verification Standard (ASVS) provide substantially finer testing and control detail [2], [3]. NIST SP 800-115 frames technical testing as a planned process rather than an ungoverned tool run [4], the Secure Software Development Framework places testing evidence within a broader development lifecycle [5], and NIST SP 800-53 situates assessment within a system of security controls [6]. These sources agree on a practical point: automated DAST is useful, but it cannot by itself establish the absence of authorization, business-logic, logging, or design failures.

Prior scanner comparisons have reported materially different winners and error rates [29]–[42]. Some variation is legitimate because products and targets change. Much of it, however, follows from avoidable design choices: a vulnerable-only target provides no defensible true-negative denominator; duplicate alerts are treated as independent vulnerabilities; the route, parameter, role, and vulnerability class are not defined as a test unit; scanners receive different authenticated surfaces; one report is treated as a statistical sample; or commercial and free editions with different capability boundaries are ranked as if they implemented the same operation. Earlier black-box studies already identified crawling, session maintenance, and vulnerability confirmation as limiting factors [47], [48]. Modern single-page applications amplify these factors [43], [45], [46].

1.2 Study Scope and Tool Editions

This study focuses on three products that are prominent in practice and education: ZAP 2.17.0, Burp Suite Community Edition 2026.4.3, and an Acunetix academic/on-premises deployment on the 26.6.1 release line. ZAP offers passive and active rules, traditional and browser-driven exploration, an API, and an Automation Framework [18], [19]. Acunetix offers automated crawling, full scan profiles, centralized findings, reporting, and API-based integration [23]–[25]. Burp Community is intentionally an “essential manual toolkit” containing the proxy, history, Repeater, Decoder, Sequencer, Comparer, and a demonstration Intruder [20]. PortSwigger explicitly reserves the web vulnerability scanner and automatic crawling for Professional or DAST editions [20], [21]. That fact is not a minor license detail; it changes the estimand.

Accordingly, the protocol distinguishes two questions. The autonomous track compares products that can perform a native automated DAST scan. Burp Community is marked N/A in that track. The analyst-assisted track evaluates how the three products support a controlled educational security-testing task without relabelling manual discoveries as autonomous scanner findings. If a future investigator requires a three-way automated engine

comparison, Burp Professional or Burp DAST must replace Community Edition before preregistration is frozen; changing editions after outcome inspection is prohibited.

1.3 Testbeds and Contributions

The selected testbeds are DVWA 2.5 and OWASP Juice Shop 20.1.1. DVWA offers compact PHP endpoints with selectable security levels and explicit classic vulnerabilities, making route- and parameter-level validation practical [17]. Juice Shop is a current Node.js/Express/Angular single-page application with API calls, roles, state transitions, and challenges spanning the OWASP Top 10 [15], [16]. Their architectural contrast is deliberate: DVWA tests conventional form-oriented PHP behavior, while Juice Shop tests modern client-side discovery and authenticated APIs. Neither is asserted to represent all production software.

The intended contributions are sixfold:

a comprehensive, edition-aware comparison of three widely used web security products;
a reproducible benchmark that separates discovery failure from detector failure;
a frozen, independently validated ground truth containing positive units and matched negative controls;
a weighted evaluation framework that combines security effectiveness, efficiency, report quality, usability, and operational fit without concealing structural N/A values;
a decision-support matrix for academic, penetration-testing, developer, and small-enterprise contexts; and
practical execution, adjudication, analysis, and replication procedures suitable for a subsequent confirmatory article.

The present manuscript is not a literature review and does not answer the six research questions with invented values. It is a Stage 1 protocol: methods and analysis decisions are fixed while outcomes remain unknown.

2. RELATED WORK

2.1 Scanner Effectiveness and Benchmarking

Black-box scanner evaluation predates the current products. Fonseca, Vieira, and Madeira compared SQL injection and XSS testing behavior [52]; Vieira, Antunes, and Madeira examined scanner use against web services [51]; and Bau et al. decomposed automated black-box testing into crawling, attack generation, and analysis [48]. Doupé, Cova, and Vigna demonstrated that scanners of the period missed vulnerabilities because they could not maintain application state, navigate workflows, or generate suitable inputs [47]. Makino and Klyuev later used DVWA and related benchmarks [49], while Mburano and Si employed the OWASP Benchmark [50]. These studies established that alert count alone is not an adequate measure of effectiveness.

Recent comparative work remains heterogeneous. Alazmi and De Leon found inconsistent target selection, configuration, metrics, and reporting across scanner studies [29]. Amankwah et al. compared commercial and open-source scanners using DVWA and WebGoat and emphasized precision and recall [34]. Albahar, Alansari, and Jurcut compared penetration-testing tools under a scoring framework [30]. Al Anhar and Suryanto studied modern Node.js targets [37], Zukran and Siraj compared SQL injection and XSS detection [38], and Ibrahim and Rosli focused on SQL injection [39]. Cruz, Almeida, and Oliveira examined open-source vulnerability-assessment solutions [40], while Kunda and Alsmadi considered the evolution of application modules and open-source testing tools [41]. Qadir et al. broadened the target set and mapped findings to OWASP and CWE categories [33]. These studies are valuable evidence, but their conclusions cannot be transferred mechanically to different editions, versions, authentication states, or definitions of a vulnerability instance.

Several papers advocate multi-tool or hybrid testing. Tudela et al. combined static, dynamic, and interactive analysis [36]; Shahid et al. compared web-security parameters and scanner behavior [35]; and Abdulghaffar et al. used multiple vulnerability scanners [56]. Those designs answer whether tool ensembles expose more reportable issues, not whether any single scanner has higher precision against a frozen truth set. The present protocol therefore preserves both tool-native findings and a common adjudicated layer.

2.2 Modern Crawling, State, and Detection Oracles

Crawler effectiveness is a first-order confounder. jÄk used dynamic analysis to expose JavaScript-generated navigation and inputs [46]. Black Widow modelled state and data dependencies to improve black-box exploration [45]. Scanner++ synchronized attack intent across scanners and reported gains in reached pages and discovered defects, illustrating that an unchanged vulnerability oracle may appear stronger when supplied with a better surface [43]. ScannerScope, although designed to detect unwanted scanning, further showed that scanner products exhibit characteristic request patterns [44]. RESTler demonstrated a related principle for stateful REST APIs: valid sequences and producer-consumer dependencies materially affect reachable behavior [63].

The protocol addresses this confounder with paired discovery modes. In root-only mode, crawler and detector are assessed end to end. In seed-controlled mode, each automated scanner receives an equivalent authenticated route/request corpus; the resulting recall estimates detector effectiveness conditional on reachability. The difference between the modes is reported rather than hidden.

Scanner research also extends beyond product comparisons. Attack-grammar planning [53], model-based security testing [54], business-layer race-condition testing [55], metamorphic security testing [58], runtime-derived checks [62], and black-box side-channel analysis [61] demonstrate that different vulnerability properties require different oracles. Comprehensive assessment frameworks [57] and recent pragmatic syntheses [59] reinforce why no one DAST engine should be interpreted as a complete verification method.

2.3 Error, Validation, and Human Use

A false positive is not merely an unwanted alert; it consumes validation time and can distort prioritization. Sacher analysed false-positive attribution in security tooling [60], and the broader scanner literature repeatedly identifies uncertain or non-reproducible alerts as a practical limitation [29], [32], [34]. The denominator also matters. FP/(TP+FP) is the false-discovery proportion and equals one minus precision; FP/(FP+TN) is the false-positive rate. Vulnerable-only applications do not supply TNs, so studies that call FP/(all alerts) an FPR are using a different quantity.

Remediation usefulness and education suitability add human judgement. NIST and OWASP guidance require reproducible evidence and actionable correction [2], [4], but scanner studies often stop at issue counts. This protocol uses independent ordinal ratings, inter-rater agreement, the System Usability Scale, task success, configuration time, and NASA-TLX workload [71], [74]. The analyst-assisted track is counterbalanced to reduce learning and carry-over bias.

3. RESEARCH GAP, QUESTIONS, AND PRE-REGISTERED CLAIMS

3.1 Research Gap

No available study resolves all of the following within one version-pinned design: (1) classic PHP and modern SPA/API testbeds; (2) a closed-world positive and negative truth set; (3) duplicate-resistant route/parameter/class units; (4) a decomposition of crawling and detection; (5) repeated timing and resource observations; (6) native severity and independently assigned CVSS; (7) remediation and report-quality scoring; (8) a usability and education track; and (9) an explicit treatment of Community Edition's absent scanner. The gap is methodological as much as empirical. A reliable comparison must define what each product is capable of doing before ranking what it reports.

3.2 Research Questions

TABLE I. RESEARCH QUESTIONS, ESTIMANDS, AND PRIMARY OUTCOMES

RQ	Question	Primary estimand	Capability track
RQ1	Which scanner detects the highest number of vulnerabilities?	Unique validated TP test units; micro and macro recall	Autonomous: ZAP vs. Acunetix; analyst-assisted: all three, separately
RQ2	Which scanner has the lowest false-positive rate?	FP/(FP+TN) on matched controls; false-discovery proportion reported separately	Same separation as RQ1
RQ3	Which scanner provides the best OWASP Top 10:2021 coverage?	Equal-weight macro recall over A01–A10 plus binary breadth	Autonomous; educational track secondary
RQ4	Which scanner completes scans fastest?	T1−T0 with right-censoring at 60 min; setup time separate	Autonomous-capable products only
RQ5	Which scanner provides the most useful remediation guidance?	Blinded six-dimension score, 0–24, normalized to 0–100	Native automated reports; Burp Community manual notes labelled separately
RQ6	Which scanner is most appropriate for academic environments and penetration-testing education?	Profile-weighted evidence, SUS, workload, task success, learning curve, transparency, cost, and deployability	Counterbalanced participant study

3.3 Null Hypotheses and Decision Rules

For RQ1–RQ5, the confirmatory null is equality of the relevant paired distribution between comparable products. For RQ6, the null is equality of within-participant usability and educational-utility scores. No directional winner is preregistered. Statistical significance requires a multiplicity-adjusted two-sided p-value below .05; practical significance additionally requires a confidence interval and an effect size that exclude a negligible region defined before analysis. A statistically detectable difference does not establish overall superiority if the product fails a hard deployment gate or if the rank changes under modest weight perturbation.

Structural absence is not a poor measured value. Burp Community's autonomous outcomes are N/A because the operation is unavailable [20], [21]. Unknown values are also not imputed. Only a feature that is demonstrably absent may receive zero in a profile that requires that feature.

4. METHODOLOGY

4.1 Study Design and Experimental Factors

The investigation is a randomized, blocked, repeated-measures comparative experiment, following established software-engineering experimentation principles [64]. The automated factors are scanner (ZAP or Acunetix), application (DVWA or Juice Shop), and discovery mode (root-only or seed-controlled). Repetition is a block: within each application and mode, both scanners are run from the same frozen target state in a randomized order. Ten measured blocks follow one unmeasured warm-up. The core automated design therefore contains 2 scanners × 2 applications × 2 modes × 10 repetitions = 80 measured scans. This count excludes invalidated runs, which are rerun from the same block under the preregistered exclusion rule and retained in an audit log.

Burp Community is not inserted into this factorial design because it cannot launch a native scanner or automatic crawl. Its educational value is examined in a distinct counterbalanced study in which participants use each product as supplied. This separation prevents analyst skill from becoming an unreported component of one “scanner” but not the others.

Planning, environment setup, configuration, scanning, validation, data collection, analysis, interpretation, and replication are executed in that order. Figure 1 records the artifact chain; Fig. 2 adds quality gates. Scanner identities are visible during operation but masked as Tool X, Y, or Z during finding and remediation adjudication.

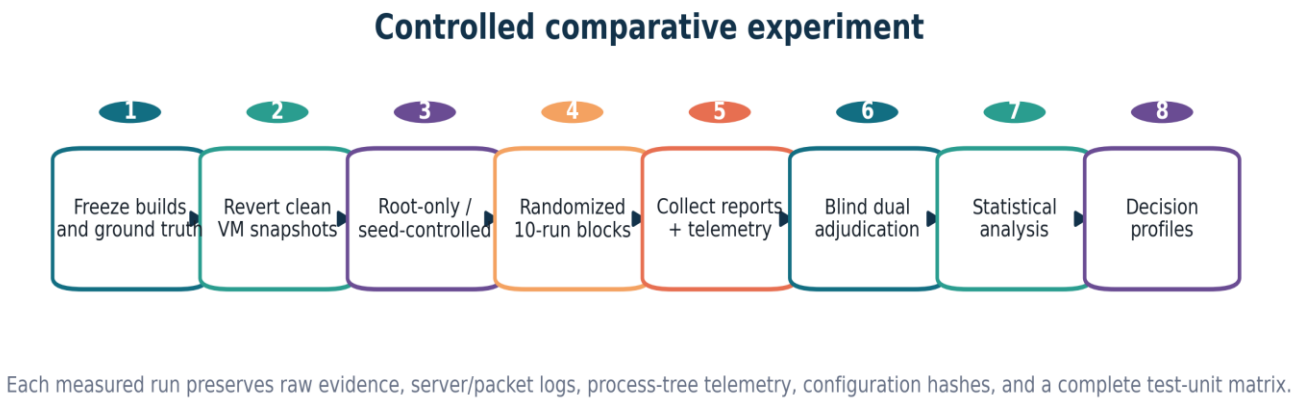

***Fig. 1**. Controlled experimental workflow.*

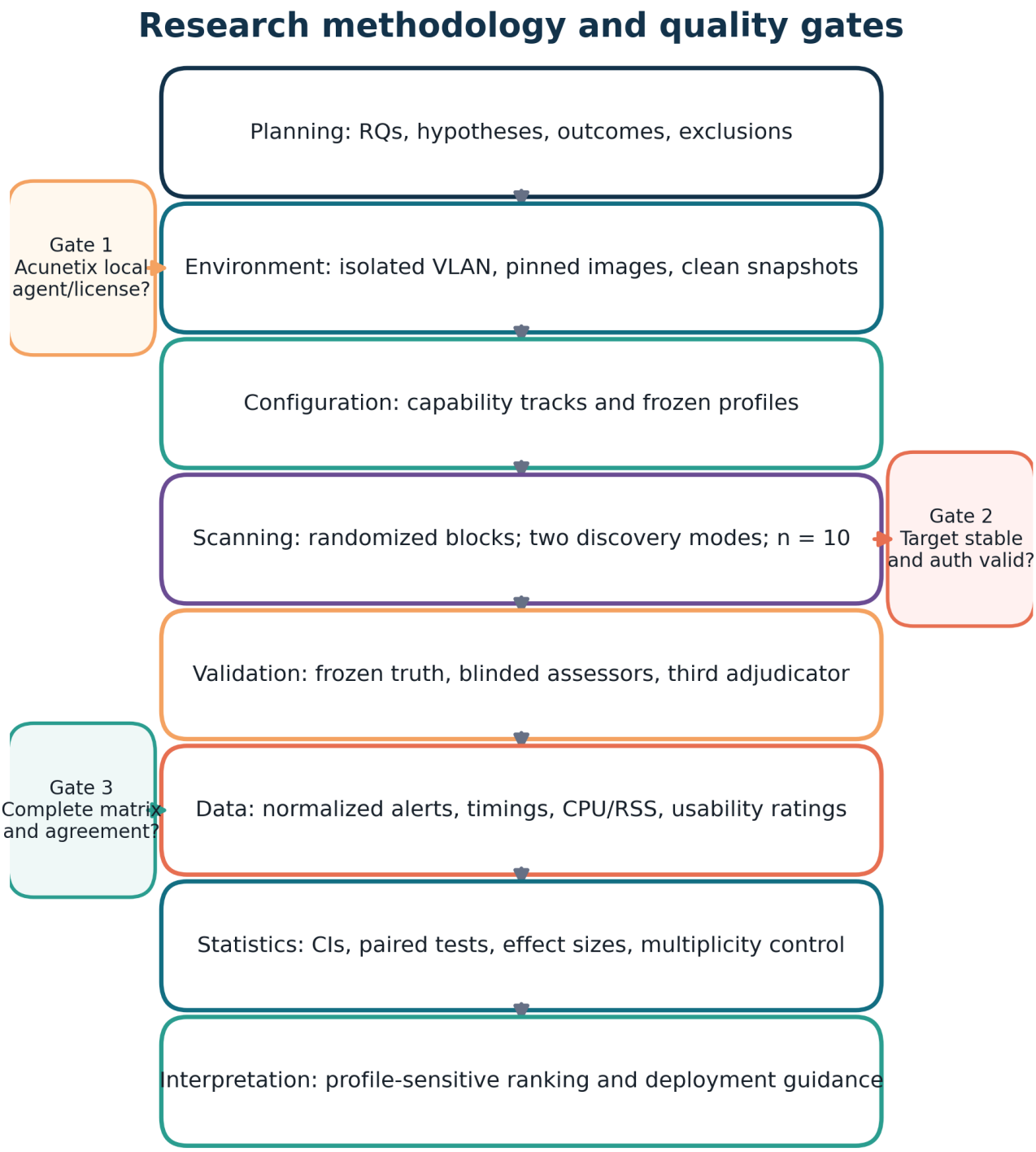

***Fig. 2**. Research methodology and pre-analysis quality gates.*

4.2 Tool Capability Boundary

The comparison is edition-specific. ZAP 2.17.0 is frozen with its installed add-on manifest; the release introduced core performance changes, alert de-duplication, systemic-alert handling, and additional scan insights [19]. Burp Suite Community 2026.4.3 is frozen by installer checksum [22]. It supplies manual tools but not Burp Scanner, project files, automated crawling, or professional report generation [20], [21]. Acunetix must be an academic/on-premises deployment or a local scanning agent whose engine, build, vulnerability database, and policy can be fixed. A cloud-only trial may be used for detection research, but it is ineligible for local CPU and memory comparisons; combining client-side telemetry for two tools with provider-side compute for the third would invalidate RQ4's resource component.

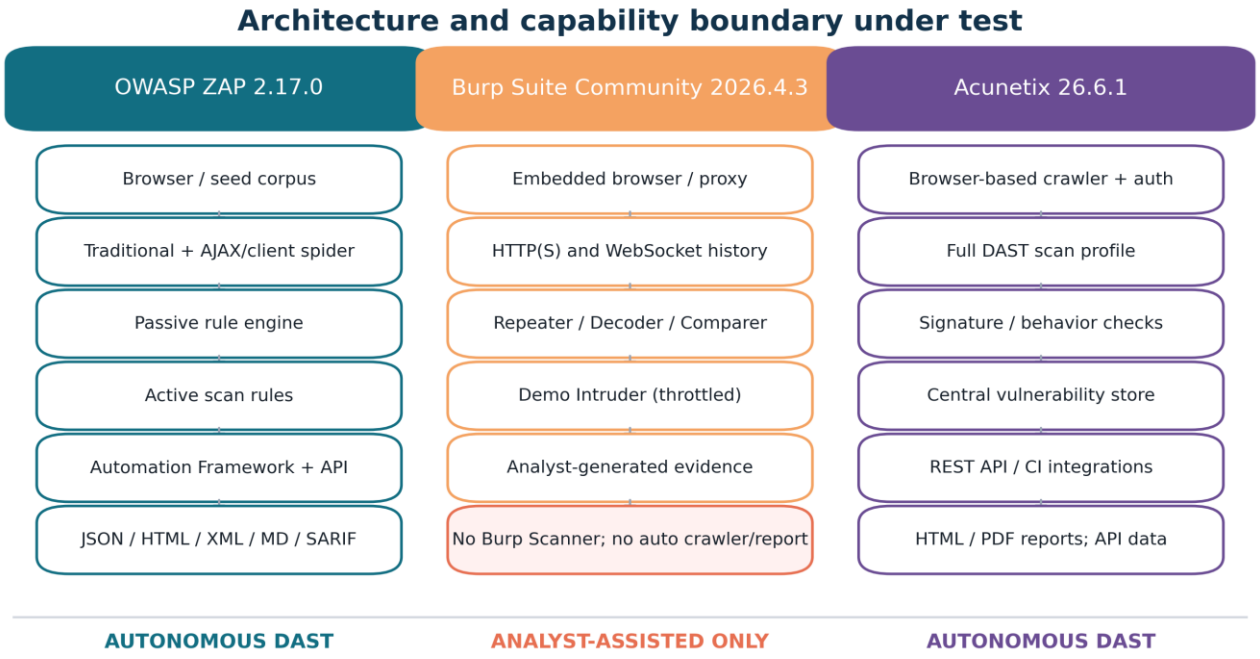

Fig. 3*. Scanner architecture and edition-specific capability boundary.*

TABLE II. PRE-EXPERIMENT FEATURE AND CAPABILITY COMPARISON

Capability	ZAP 2.17.0	Burp Community 2026.4.3	Acunetix academic/trial
Intercepting proxy/manual replay	Yes	Yes; principal strength	Product-dependent auxiliary workflow
Automatic crawler	Traditional plus AJAX/client exploration	No in Community Edition	Yes
Passive analysis	Yes	Traffic inspection, but no Burp Scanner issue engine	Yes within scan workflow
Active automated DAST	Yes	No	Yes
Authenticated scanning	Context/session configuration	Manual session operation	Form/basic/NTLM/OAuth options depend on edition
Headless/API automation	Automation Framework and REST API	No native Community scan API	REST API/local agent where licensed
CI/CD integration	Official automation and machine-readable reports	Not a native CE scanning capability	API-based integrations [24]
Native vulnerability report	HTML, JSON, XML, Markdown, SARIF templates	No automated vulnerability report	HTML/PDF templates; API data [23]
License/cost model	Open source	Free proprietary edition	Time- or institution-limited commercial license
Autonomous-track eligibility	Yes	No—structural N/A	Yes only with reproducible engine and local telemetry

The table records documented capability, not an empirical performance result. Feature availability is rechecked immediately before the version freeze, and any discrepancy is reported rather than silently updated.

4.3 Testbed Selection and Exposure Profiles

DVWA and Juice Shop were selected for complementary construct coverage. DVWA 2.5 exposes small, inspectable PHP/MySQL exercises for injection, file inclusion, CSRF, weak session identifiers, and multiple forms of XSS. Its low, medium, high, and impossible modes facilitate matched positive and defensive controls, although "impossible" is not accepted as ground truth without manual verification. Juice Shop 20.1.1 contains a JavaScript SPA, REST endpoints, object identifiers, authentication state, and business-logic challenges [15], [16]. It tests whether a scanner can move beyond static anchor extraction and retain authenticated API context.

Only laboratory-owned instances are scanned. The target VLAN has no route to the public Internet during measured runs. DVWA's remote-file-inclusion unit points to a controlled HTTP service on the same isolated VLAN; no external callback or third-party system is contacted. For out-of-band tests, a local sink records DNS and HTTP interactions. The canonical exposure is HTTP to preserve the applications' supplied behavior. A secondary HTTPS profile may be added before ground-truth freeze for HSTS and Secure-cookie controls, but it must apply identically to every scanner and must not insert protective response headers at a reverse proxy.

The testbeds are reset before each measured run. DVWA's database is rebuilt, its security level and PHP inclusion settings are applied from an immutable configuration, and the application container is recreated. Juice Shop's data volume is discarded and reseeded. Readiness checks verify expected status codes, authentication, challenge state, server clock, and response hashes for five sentinel routes.

4.4 Vulnerability Scope and Ground-Truth Units

A test unit is the tuple (application build, exposure profile, route, HTTP method, parameter or object, authentication role, vulnerability class). The tuple prevents a scanner from gaining credit multiple times for the same systemic issue while preserving genuinely distinct vulnerable inputs. For site-wide headers, the unit is the controlling response policy plus a small fixed set of representative routes; the number of crawled pages cannot inflate the score.

Ground truth is created before any evaluated scanner report is opened. Two assessors independently examine project documentation, source code, server logs, and a manual proof. Each positive unit requires a reproducible exploit or security-property violation and a SHA-256 hash of redacted evidence. Each negative unit is a matched route or configuration for which the same attack family is ineffective under the stated oracle. Disagreements are resolved by a third assessor. The manifest is then locked, hashed, and time-stamped. Scanner output cannot add a confirmatory test unit after unblinding. Novel, out-of-universe findings are preserved and judged in an exploratory annex.

TABLE III. PRIMARY VULNERABILITY CLASSES AND VALIDATION ORACLES

Vulnerability class	Principal testbed(s)	CWE	Primary OWASP 2021 category	Positive oracle	Matched negative control
SQL injection	Both	CWE-89	A03 Injection	Boolean/time/error or data-change proof tied to parameter	Parameterized/validated twin request
Stored XSS	Both	CWE-79	A03	Unique marker executes after persistence in clean browser	Encoded or sanitized storage path
Reflected XSS	Both	CWE-79	A03	Marker executes in reflected response context	Contextually encoded twin input
OS command injection	DVWA	CWE-78	A03	Local canary file or deterministic command output	Allow-listed/escaped route

Vulnerability class	Principal testbed(s)	CWE	Primary OWASP 2021 category	Positive oracle	Matched negative control
CSRF	DVWA	CWE-352	A01 Broken Access Control	Cross-origin state change without valid anti-CSRF condition	Token/origin-protected state change
Broken authentication	Both	CWE-287	A07 Identification and Authentication Failures	Bypass, credential-policy, or session proof	Correctly enforced authentication path
Security misconfiguration	Both	CWE-16/CWE-693	A05 Security Misconfiguration	Configuration-specific observable weakness	Hardened paired configuration
Sensitive-data exposure	Juice Shop; transport profile	CWE-200/CWE-319	A02 Cryptographic Failures, context-specific	Unauthorized or cleartext disclosure	Authorized/encrypted or redacted control
Directory traversal	Both where present	CWE-22	A01	Read outside intended directory	Canonicalized allow-listed path
Remote file inclusion	DVWA	CWE-98	A03	Controlled remote include/callback proof	Remote wrappers disabled/allow list
Local file inclusion	DVWA	CWE-98 or CWE-22	A03 or A01, by root cause	Include/read controlled local file	Fixed identifier-to-file mapping
Weak cookies	Both	CWE-614/CWE-1004	A05	Missing Secure/HttpOnly/SameSite or predictable session property	Hardened cookie on matched profile
Missing security headers	Both	CWE-693	A05	Frozen header-policy assertion fails	Header-complete representative route
Open redirect	Both where present	CWE-601	A01	Server/client navigation to untrusted local sink	Origin/allow-list enforcement
Server information disclosure	Both	CWE-200	A05	Version, path, stack, or framework detail in response	Suppressed disclosure control
Broken access control	Both, primarily Juice Shop	CWE-862/CWE-863	A01	Unauthorized read/write under role matrix	Same object/action correctly denied

The mandatory list contains both stored and reflected XSS as separate classes. The ground-truth file also records DOM-based XSS when present, but it is exploratory unless added before the freeze. File inclusion is mapped according to mechanism: PHP include/require control is A03; unauthorized path traversal is A01. Sensitive exposure is not automatically labelled A02—only failures of protection or cryptography use that category; authorization-driven disclosure remains A01. This context-sensitive mapping avoids forcing every alert title into a single category.

4.5 Completing OWASP Top 10:2021 Coverage

The mandatory classes do not naturally instantiate all ten OWASP categories. To answer RQ3 without treating absent categories as scanner failures, a supplementary stratum is frozen from Juice Shop or an equivalent local fixture before scanning. It includes at least one positive and one matched negative unit for A04 Insecure Design, A06 Vulnerable and Outdated Components, A08 Software and Data Integrity Failures, A09 Security Logging and Monitoring Failures, and A10 Server-Side Request Forgery. These units may be difficult or impossible for black-box DAST; that is a valid result if the reachable property and oracle are documented.

TABLE IV. OWASP TOP 10:2021 MAPPING AND COVERAGE DENOMINATORS

Category	Benchmark classes	Inclusion rule for denominator
A01 Broken Access Control	CSRF, traversal, open redirect, object/function authorization	Only frozen route-role-object units
A02 Cryptographic Failures	Cleartext transport/storage, weak protection of sensitive data	Independently verified protection failure
A03 Injection	SQLi, stored/reflected XSS, command injection, PHP file inclusion	Parameter- and context-specific units
A04 Insecure Design	Business-rule/design fixture	Design property with explicit misuse case
A05 Security Misconfiguration	Headers, cookies, debug/banner/path disclosure	Deduplicated policy/configuration units
A06 Vulnerable and Outdated Components	Known vulnerable component fixture	Exact package/version and authoritative advisory
A07 Identification and Authentication Failures	Authentication/session/password-policy units	Authentication state and role fixed
A08 Software and Data Integrity Failures	Unsigned/untrusted update, deserialization, or integrity fixture	Integrity boundary and tampering proof fixed
A09 Security Logging and Monitoring Failures	Missing event/audit fixture	Server-side event and expected log assertion fixed
A10 Server-Side Request Forgery	Local callback-sink unit	Sink interaction uniquely attributable to request

CVE identifiers [12], CISA's Known Exploited Vulnerabilities catalog [11], and the CWE taxonomy [7], [8] are used only where applicable. A deliberately vulnerable exercise without a CVE is not assigned one. The component fixture records an exact version and authoritative advisory; catalog membership is never inferred from a product name alone.

4.6 Discovery Modes

Root-only mode supplies the canonical root URL and credentials permitted by the product. The scanner must discover links, forms, API requests, state transitions, and parameters using its normal mechanisms. Seed-controlled mode begins from an identical, authenticated request corpus captured by a version-pinned Playwright script. The corpus contains routes and request shapes, not exploit payloads. Product-specific import or replay steps are documented and hashed. A scanner that cannot ingest a request type is recorded as such.

The two modes answer different questions. Root-only recall is ecological end-to-end effectiveness. Seed-controlled recall is conditional detector effectiveness. The endpoint-discovery ratio is the fraction of frozen positive-unit routes observed in the scanner's traffic before active payloads. The difference between conditional and root-only recall estimates the loss associated with reachability and session handling.

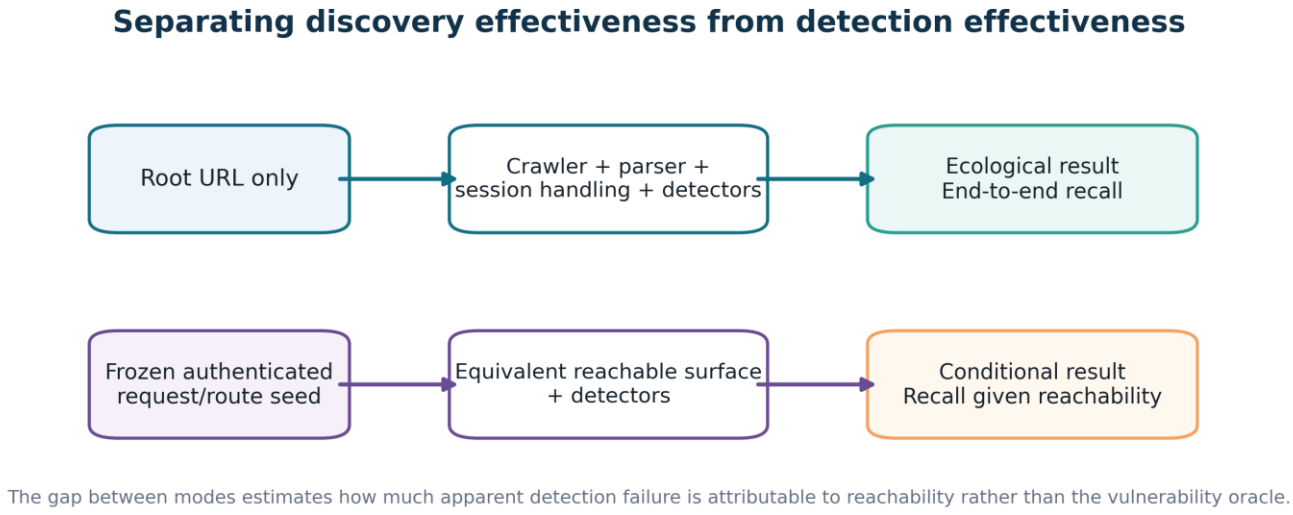

***Fig. 4**. Decomposition of discovery and detector effectiveness.*

4.7 Scanner Configuration

No scanner is tuned after outcomes are inspected. ZAP uses the Automation Framework with traditional and AJAX/client exploration, passive-scan completion, and the Default active policy at Medium strength and Low alert threshold. Four threads per host are allowed, anti-CSRF handling remains enabled, and the full add-on manifest is archived. The JSON-plus and HTML-plus reports are exported without severity filtering. The repository includes a YAML template, but authentication contexts and exclusions must be completed and frozen for each application.

Acunetix uses a non-incremental Full Scan profile, authenticated where supported. The exact profile UUID, crawler settings, concurrency, excluded logout/reset routes, agent build, Chromium build, and vulnerability database date are exported through the UI/API. “Smart,” incremental, or prior-knowledge scans are not mixed with full scans. A failed pre-scan authentication check invalidates the run; current release notes document explicit pre-scan validation in the 26.6.1 line [25].

Burp Community uses a fresh temporary project/session for each participant task. The embedded browser is proxied, scope is restricted to the target, and the supplied checklist permits Proxy history, Repeater, Decoder, Comparer, Sequencer, and demonstration Intruder. No Professional trial, Dastardly container, extension that implements an automated scanner, or external exploit tool is introduced. Findings are analyst-authored evidence and remain labelled as such.

4.8 Scanning, Monitoring, and Exclusion Rules

After one warm-up, every measured automated run follows this sequence: revert scanner and target snapshots; recreate application state; verify sentinel routes and credentials; start packet, server, cgroup, and process-tree monitors; wait 60 s for baseline; start the scanner; record T0 at the first in-scope scanner request; stop at native completion or 60 min; wait for passive queues and report serialization; record T1; export all artifacts; and compute hashes before reset.

Scanner CPU includes the complete process tree and any browser launched by the scanner. Cgroup v2 supplies CPU seconds and memory.current; one-second process samples supply mean and peak normalized CPU and resident memory. Target telemetry is collected separately. A run is invalid if authentication was not maintained, the target returned more than 1% HTTP 5xx responses, target CPU exceeded 85% for 30 consecutive seconds, the scanner crashed, the monitor lost more than 5 s of samples, or the configuration hash differed. Timeouts are valid right-censored observations, not failures to be silently rerun. Invalid runs remain in scan_runs.csv with a reason and are repeated from the same block.

4.9 Finding Normalization and Blinded Validation

Raw reports are immutable. A parser creates a normalized finding with native rule identifier, title, severity, confidence, URL, method, parameter, request and response hashes, evidence, CWE/OWASP labels, CVSS text, remediation text, references, and export source. Duplicate normalization occurs at the frozen test-unit level, never by title alone. An alert can map to at most one unit per vulnerability class; one finding may have multiple instances, but repeated instances of the same unit count once.

For every measured run, the dataset contains a complete row for every ground-truth unit. Positive detected units are TP, positive undetected units are FN, negative detected units are FP, and negative undetected units are TN. Out-of-universe findings are reviewed separately because assigning them post hoc to the closed denominator would leak scanner output into ground-truth construction.

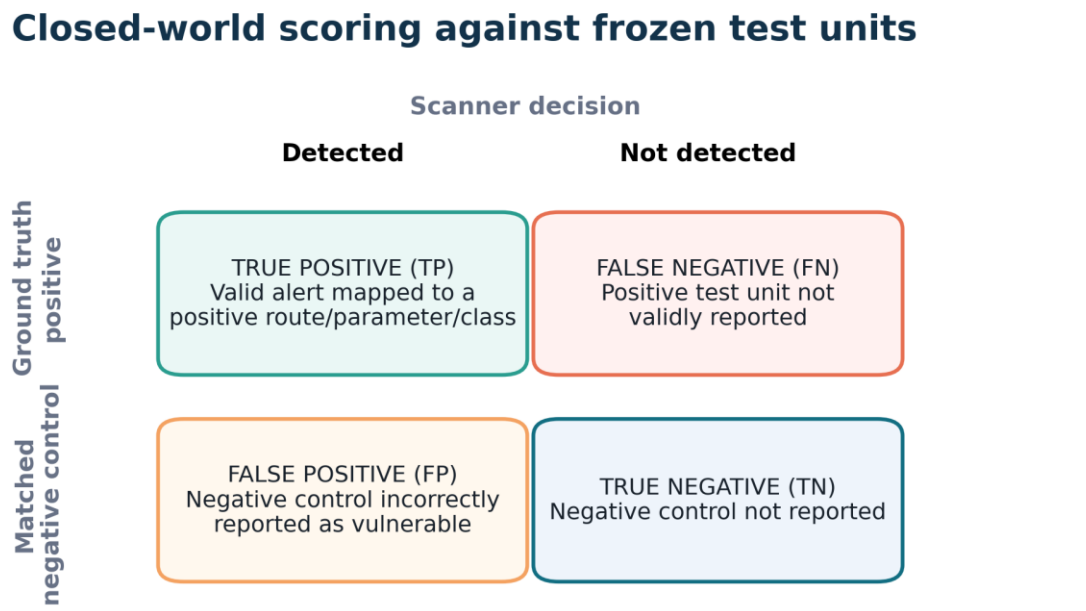

Fig. 5*. Confusion-matrix construction from frozen positive and negative test units.*

4.10 Remediation, Report Quality, and Usability

For each distinct valid vulnerability class reported by a product, assessors receive a scanner-masked excerpt containing the finding evidence and remediation. Six dimensions are scored 0–4: root-cause explanation, finding-specific evidence, actionability, code/configuration specificity, standards mapping, and retest guidance. The sum is normalized to 0–100. Technical errors cap the affected dimension at 1. Reports are additionally rated 1–7 for traceability, evidence reproducibility, de-duplication, severity rationale, developer readability, executive readability, and export fidelity.

The user study targets 30 complete participants after attrition, stratified equally between students/novices and participants with practical web-testing experience. Recruitment should begin with 32 participants. A three-period Latin-square sequence counterbalances the tools. Participants complete equivalent scope, authentication, traffic-inspection, testing, validation, and reporting tasks on disposable instances. Primary outcomes are task success, critical errors, configuration time, completion time, SUS, NASA-TLX, perceived learning curve, report quality, and educational utility. Training materials and task scripts are identical in length and disclose the edition boundary. Institutional ethics approval, informed consent, and anonymous identifiers are mandatory before recruitment.

4.11 Replication Procedure and Artifact Governance

The accompanying package contains empty data schemas, the experiment configuration, a ZAP plan template, a runbook, the adjudication rubric, vector and raster method figures, and an analysis script. Scanner installers and licensed Acunetix components are not redistributed. Every runtime version, OCI image, configuration, raw report, normalized dataset, and analysis output is hashed with SHA-256. Secrets are replaced with non-reversible

placeholders before publication; request and response hashes preserve traceability. The repository tag and archive DOI are added only after the Stage 2 package is deposited.

5. EXPERIMENTAL ENVIRONMENT

5.1 Reference Hardware and Virtualization

The reference implementation is prescribed rather than claimed as already executed. The Stage 2 article must replace "target" values with command-derived runtime values if they differ. A dedicated Ubuntu 24.04.3 LTS host uses an AMD Ryzen 9 5900X (12 physical cores/24 threads), 64 GiB DDR4 RAM, and an NVMe SSD. KVM/QEMU supplies three guests. Scanner and target vCPUs are pinned to disjoint physical cores; simultaneous multithreading siblings are not split between them. CPU frequency scaling is fixed to performance mode, host background services are minimized, and no other workload runs during blocks.

TABLE V. REFERENCE EXPERIMENTAL ENVIRONMENT TO BE FROZEN BEFORE DATA COLLECTION

Component	Prespecified configuration	Evidence recorded at freeze
Host OS	Ubuntu 24.04.3 LTS, 64-bit	uname -a, lsb_release -a, package manifest
Host CPU/RAM/storage	AMD Ryzen 9 5900X; 64 GiB; NVMe	lscpu, dmidecode, lsblk, governor state
Hypervisor	KVM/QEMU 8.2.x; libvirt	package versions and domain XML
Scanner VM	Ubuntu 24.04.3; 4 pinned vCPU; 8 GiB; 60 GiB disk	VM XML and snapshot hash
Target VM	Ubuntu 24.04.3; 4 pinned vCPU; 8 GiB; 60 GiB disk	VM XML and snapshot hash
Monitor VM	Ubuntu 24.04.3; 2 vCPU; 4 GiB; capture disk	VM XML and capture-tool versions
DVWA	v2.5, commit a96943dc...	Git commit, source archive hash, OCI digest
PHP	Target 8.3.32	php -v inside target image
Apache HTTP Server	Target 2.4.68	apachectl -v; module list
MySQL	Target 8.0.46	SELECT VERSION(); server config hash
Juice Shop	v20.1.1, commit f915bddd...	Git commit and OCI digest
Node.js	24.x supplied by pinned Juice Shop image	node --version inside container
ZAP	2.17.0 stable	core version, Java, add-on manifest, config hash
Burp Suite Community	2026.4.3 stable	installer SHA-256 and embedded Java/browser build
Acunetix	Academic/local-agent 26.6.1 line	product build, agent, VDB date, profile export
Seed browser	Chromium 145 build line; exact patch frozen	executable hash and --version
Monitoring	cgroup v2, pidstat, sar, tcpdump	tool versions and sampling configuration

Apache 2.4.68, PHP 8.3.32, and MySQL 8.0.46 are the intended reference patches [26]–[28]. DVWA's upstream container is based on a floating php:8-apache image and MariaDB; the experiment therefore builds a dedicated immutable target image rather than using latest. The exact Apache/PHP/MySQL combination must pass all

sentinel tests before the truth set is frozen. If a compatibility change is required, it is made before scanner execution and disclosed as a protocol amendment.

5.2 Network and Browser Configuration

The host-only VLAN is 192.168.56.0/24: scanner 192.168.56.10, target 192.168.56.20, and monitor 192.168.56.30. An nftables policy denies egress during measured runs. Local DNS resolves dvwa.test and juice.test. NetEm fixes one-way latency at approximately 1 ms with no induced loss; measured latency and packet loss are logged. A separate management interface is enabled only for updates and license activation, then disconnected before the freeze. Host time is the sole NTP source.

The standardized seed corpus is generated with the pinned Chromium build at a fixed 1366×768 viewport, English locale, UTC time zone, disabled cache, and a new profile. The corpus records HAR, WebSocket metadata, route list, cookies after authentication, and a replay script. Scanner-launched browsers may differ; their versions and process trees are recorded as part of the product.

5.3 Environment Validity Checks

Before every block, five assertions must pass: application build and configuration hashes match; the database is freshly seeded; credentials and roles behave as expected; the local callback sink is empty and reachable; and sentinel-response hashes match the exposure profile. At the end of a run, server logs are searched for unhandled exceptions, target saturation, and requests outside the allowed subnet. These checks distinguish scanner limitations from testbed failure.

6. EVALUATION METRICS AND WEIGHTED FRAMEWORK

6.1 Confusion-Matrix Metrics

Let TP, FP, FN, and TN denote unique test units per scanner and run. The primary equations are:

Detection Rate = Recall = TP / (TP + FN).
Precision = TP / (TP + FP).
False\ Positive\ Rate = FP / (FP + TN).
False\ Discovery\ Proportion = FP / (TP + FP) = 1 - Precision.
Accuracy = (TP + TN) / (TP + TN + FP + FN).
F1 = 2(Precision × Recall)/(Precision + Recall).

Micro values pool units; macro recall first computes recall per vulnerability class and then averages classes. The study reports both because micro recall can be dominated by classes with more units. Precision is undefined when a product reports no findings and is displayed as N/A, not 1. Accuracy is interpreted only with the deliberately balanced or documented negative-control set; it is not presented without class prevalence. Precision–recall graphics are interpreted with their underlying prevalence and confusion counts rather than as context-free geometry [72].

Wilson score intervals are used for binomial proportions [73]. Ten repeated runs quantify run-to-run variability, but they do not create ten independent vulnerabilities. The inferential unit is matched by application, mode, block, and—where appropriate—ground-truth test unit.

TABLE VI. METRICS, UNITS, AND INTERPRETATION

Metric	Unit/scale	Direction	Primary caveat
Unique TP and detection rate	Test units; proportion	Higher	Duplicate alerts collapsed
Precision	Proportion	Higher	Uses adjudicated FP findings mapped to controls
FPR	Proportion	Lower	Requires matched negative units
FN	Test units	Lower	Distinguish unreachable from reached-but-missed

Accuracy/F1	Proportion	Higher	Report with class balance and components
Endpoint discovery ratio	Proportion	Higher	Measured before active payload attribution
OWASP coverage score	0–100	Higher	Equal category weights; absent categories not silently dropped
Scan time/setup time	Seconds	Lower	Timeouts right-censored; setup reported separately
CPU	Mean/peak % and CPU seconds	Lower for equal outcomes	Complete scanner process tree
Memory	Mean/peak MiB	Lower for equal outcomes	JVM/browser/container children included
Risk/severity agreement	Weighted κ, MAE	Higher κ/lower MAE	Native labels compared with independent CVSS bins
Remediation utility	0–24 and 0–100	Higher	Blinded independent raters
Usability	SUS 0–100; task outcomes	Higher	Counterbalanced within-participant design
Workload	NASA-TLX 0–100	Lower	Report subscales and composite
Configuration/learning	Minutes, errors, 1–7 rating	Lower minutes/errors	Prior experience included as covariate
Automation/CI/export	Evidence rubric 0–4	Higher	Edition-specific; structural absence explicit

6.2 OWASP Coverage

For scanner s and OWASP category k, category recall is the fraction of positive frozen units in k detected by s. With equal weights w_k = 0.1, the graded coverage score is:
Coverage_s = 100 × Σ_{k=1}^{10} w_k Recall_{s,k}.
A secondary breadth score is 10 times the number of categories with at least one TP. Breadth is deliberately secondary because detecting one easy header issue should not imply comprehensive A05 coverage. The radar chart displays category recall, not raw alert volume.

6.3 Timing and Resource Metrics

Scanning time is T1−T0. T0 is the first in-scope scanner request; T1 occurs only when active work and passive queues are complete and the final report can be serialized. Authentication/setup time is measured from opening the product to the readiness check and is not included in scan time. If a scan reaches 3600 s, its duration is right-censored. Median, interquartile range, restricted mean completion time, and completion proportion are then more informative than replacing every timeout with exactly 3600 s and applying an ordinary mean comparison.
CPU is reported as process-tree CPU seconds, one-second mean and peak core percentage, and normalized percentage of the four allocated vCPUs. Memory is mean and peak resident/cgroup memory in MiB. Network requests, response classes, and bytes are diagnostic variables. Target-side CPU and error rate are validity checks rather than scanner efficiency scores.

6.4 Risk Classification and CVSS

The native severity, confidence, CVSS vector, CWE, and OWASP label are preserved unchanged. Separately, two assessors assign a CVSS v4.0 base vector to each positive ground-truth unit using the fixed laboratory scenario [9]. CVSS v3.x and v4.0 vectors are not numerically converted. Comparisons use the canonical v4.0 qualitative bins: None 0, Low 0.1–3.9, Medium 4.0–6.9, High 7.0–8.9, and Critical 9.0–10.0. Severity distributions count validated TP units. Quadratic weighted κ, exact agreement, and mean absolute ordinal error compare native categories with the independent bins.

6.5 Usability, Reports, Automation, and Operational Fit

Ease of configuration combines setup minutes, critical errors, and a 1–7 rating. Learning curve combines the change in task time across periods, help requests, and a 1–7 perceived-burden rating. Automation is scored 0–4 for deterministic headless execution, policy-as-code, API control, failure codes, and reproducible state. CI/CD integration is scored 0–4 using documented and executed pipeline evidence, not marketing claims. Export support is scored for machine-readable schema, evidence completeness, stable identifiers, and human-readable output. Report quality and remediation guidance use the blinded rubrics in Section 4.10. These operational and usability measures are quality characteristics rather than substitutes for security effectiveness; their separation is consistent with the product-quality perspective in ISO/IEC 25010 [14]. Compliance-oriented exports are checked where advertised, including requirements relevant to PCI DSS [13].

6.6 Weighted Evaluation Framework

The general profile assigns 40% to detection quality, 15% to efficiency, 15% to reporting/remediation, 15% to usability, and 15% to operational fit. Detection quality contains macro recall, precision, FPR, OWASP coverage, and severity agreement; efficiency contains completion time, CPU, and memory. Raw values are normalized against preregistered anchors, not the sample minimum and maximum, to avoid rank reversal when a tool is added. An education profile uses 25% detection, 10% efficiency, 10% reporting/remediation, 25% usability, 10% operational fit, and 20% pedagogical transparency. Pedagogical transparency measures whether students can observe, modify, replay, and explain HTTP exchanges. Separate penetration-testing and SME profiles may be added before data collection, with all weights summing to one. Weight sensitivity varies each non-zero weight by ±20%, renormalizes, and reports rank acceptability across 10,000 fixed-seed draws.

A missing observation prevents a complete profile score. A structurally absent capability may score zero only where the profile treats that capability as required. Thus Burp Community can be eligible for an education profile centred on manual HTTP reasoning while remaining ineligible for an autonomous CI/CD profile.

7. STATISTICAL ANALYSIS PLAN

7.1 Descriptive Statistics and Confidence Intervals

For every scanner, application, mode, and track, the analysis reports n, mean, median, standard deviation, interquartile range, minimum, maximum, and a 95% confidence interval. Proportion intervals use Wilson scores; continuous outcomes use t-based mean intervals and a fixed-seed 10,000-resample bootstrap for medians and skewed contrasts. Raw block-level points accompany box or violin plots.

7.2 Paired Comparisons

Paired differences are examined with Q–Q plots, the Shapiro–Wilk test [66], and influence diagnostics. When the paired-difference distribution is approximately symmetric and free from severe outliers, the paired t-test is reported [76]. Otherwise, the Wilcoxon signed-rank test with a documented zero method is primary [68]. Both are retained in the analysis table for transparency, but the assumption-appropriate test governs the confirmatory statement.

For binary detection of the same test units by two autonomous scanners, exact McNemar tests are more appropriate than a t-test on alert counts. Cochran's Q is used only in a comparable three-tool binary task, such as the analyst-assisted educational track. Remediation and ordinal ratings use Friedman tests followed by paired Wilcoxon comparisons. A one-factor repeated-measures ANOVA is used for three-tool within-participant continuous outcomes only when residual and sphericity assumptions are tenable; Greenhouse–Geisser correction is applied when needed. Otherwise, Friedman inference is primary. Scanner × application interactions are estimated with a mixed model using block/participant as a random intercept when the data support it.

7.3 Effect Sizes, Multiplicity, and Agreement

Paired continuous contrasts report mean/median difference, 95% CI, Cohen's d_z, and rank-biserial correlation [65]. Repeated-measures ANOVA reports partial η^2; Friedman reports Kendall's W. Binary paired comparisons report matched odds ratios and risk differences. All planned pairwise tests within an RQ family use Holm's sequential correction [75]. Exact adjusted p-values are reported; “p = 0.000” is prohibited. The Benjamini–Hochberg procedure is reserved for explicitly exploratory families [67].

Assessor reliability is evaluated before adjudication. Nominal finding labels use Cohen's κ [69]; ordinal rubric dimensions use weighted κ; composite remediation/report scores use a two-way absolute-agreement intraclass

correlation [70]. Low agreement triggers rubric clarification and a sensitivity analysis using each assessor's original values; it does not justify deleting inconvenient ratings.

TABLE VII. PRE-SPECIFIED INFERENTIAL ANALYSES

Outcome/RQ	Pairing/block	Primary test	Effect size	Multiplicity family
TP detection per unit (RQ1)	Same unit/app/mode	Exact McNemar; Cochran Q only if 3 comparable tools	Risk difference, matched OR	RQ1 pairwise
FPR on controls (RQ2)	Same negative units	Exact McNemar; bootstrap difference	Risk difference	RQ2
OWASP category recall (RQ3)	Same category/app	Paired t or Wilcoxon on category recalls	d_z or rank-biserial	RQ3
Scan time (RQ4)	Same block/app/mode	Paired t/Wilcoxon; censored method if needed	d_z/rank-biserial; restricted mean difference	RQ4
CPU and memory	Same block/app/mode	Paired t/Wilcoxon	d_z/rank-biserial	Efficiency secondary
Remediation score (RQ5)	Same class/evidence stratum	Wilcoxon/Friedman	Rank-biserial/Kendall W	RQ5
SUS/task outcomes (RQ6)	Same participant	RM-ANOVA or Friedman; paired post hoc	partial η^2/Kendall W	RQ6
Severity category	Same validated TP unit	Weighted κ and ordinal error	κ with CI; MAE	Descriptive/agreement

7.4 Statistical Significance and Practical Interpretation

The experiment does not interpret significance in isolation. A faster scanner with materially lower recall is not "better" under a detection-first profile; a one-point SUS difference is not educationally decisive; and a significant p-value based on duplicated URL alerts is invalid. The Stage 2 discussion must report absolute differences, uncertainty, effect size, raw distributions, exclusions, and the relevant capability boundary. Confirmatory and exploratory analyses are visibly separated.

8. PLANNED RESULTS AND REPORTING

8.1 Reporting Principle

No empirical values appear in this section because the protocol-first option was selected. The following tables are locked reporting shells. Dashes mean "not yet observed," not zero.

8.2 Locked Reporting Tables

TABLE VIII. DETECTED VULNERABILITIES—STAGE 2 REPORTING SHELL

Vulnerability class	Ground-truth positive units	ZAP TP/FN	Burp CE analyst-assisted TP/FN	Acunetix TP/FN	Notes on reachability
SQL injection	—	—	—	—	—
Stored XSS	—	—	—	—	—
Reflected XSS	—	—	—	—	—
Command injection	—	—	—	—	—
CSRF	—	—	—	—	—
Authentication/access control	—	—	—	—	—
Misconfiguration/disclosure	—	—	—	—	—
Traversal/file inclusion	—	—	—	—	—
Other Top 10 extension units	—	—	—	—	—

TABLE IX. PERFORMANCE METRICS—STAGE 2 REPORTING SHELL

Tool/track	TP	FP	FN	TN	Precision	Recall	FPR	Accuracy	F1	Coverage	Time, min	Peak RSS, MiB
ZAP autonomous	—	—	—	—	—	—	—	—	—	—	—	—
Burp CE autonomous	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Burp CE analyst-assisted	—	—	—	—	—	—	—	—	—	—	task time	—
Acunetix autonomous	—	—	—	—	—	—	—	—	—	—	—	—

TABLE X. STATISTICAL RESULTS—STAGE 2 REPORTING SHELL

RQ/contrast	n pairs	Mean/median difference	95% CI	Paired t p	Wilcoxon p	Adjusted p	Effect size	Decision
RQ1 detection	—	—	—	as applicable	as applicable	—	—	—
RQ2 FPR	—	—	—	as applicable	as applicable	—	—	—
RQ3 coverage	—	—	—	—	—	—	—	—
RQ4 time	—	—	—	—	—	—	—	—
RQ5 remediation	—	—	—	—	—	—	—	—
RQ6 usability	—	—	—	—	—	—	—	—

8.3 Planned Outcome Figures

The analysis script generates the requested Stage 2 figures only when empirical CSV rows are present: OWASP coverage radar, detection/accuracy bars, false-positive comparison, execution-time comparison, risk-severity distribution, precision–recall plot, confusion matrices, and profile-specific overall rankings. This safeguard prevents an illustrative chart from being mistaken for evidence. The raster outputs are 300 dpi and the method figures are also supplied as SVG.

TABLE XI. OUTCOME FIGURE GENERATION PLAN

Planned figure	Input	Statistical annotation
OWASP coverage radar	Unit detections by A01–A10	Category recall and CI in companion table
Detection accuracy bar	Run-level recall, precision, accuracy	95% CI; no raw alert counts
False-positive comparison	Matched negative controls	FPR and false-discovery proportion
Execution-time comparison	Completed/censored run log	Raw points, median, completion proportion
Risk-severity distribution	Validated TP plus independent CVSS v4	Canonical severity bins
Precision vs. recall	Run-level confusion matrices	Mean marker plus individual blocks
Confusion matrix	Frozen unit matrix	Counts and normalized rates
Overall performance ranking	Complete normalized rubric	Profile weights and sensitivity

9. DISCUSSION AND RESULT-INTERPRETATION LOGIC

9.1 Why Scanners May Differ

Differences will first be decomposed into surface discovery and conditional detection. A high root-only/seed-controlled gap indicates that authentication, state, client-side events, or route discovery—not the attack oracle—limited performance. This distinction is especially important for Juice Shop. A small gap combined with low conditional recall points instead to payload generation, context modelling, response normalization, or conservative confirmation thresholds. Scanner++ and modern crawling research show why page and request coverage can change apparent detection effectiveness without changing the underlying oracle [43], [45], [46].

Injection classes are generally more amenable to DAST because scanners can compare syntax, error, Boolean, delay, reflection, or execution effects. Stored XSS requires a scanner to return to a later rendering context, while DOM XSS may require browser execution. Authorization failures require role and object comparisons that a single-session crawl cannot infer reliably. CSRF detection can confuse absence of a token with exploitability if SameSite cookies or origin checks are effective. Logging failures require server-side observation and are not ordinarily demonstrable from an HTTP response. These mechanisms, not brand reputation, will anchor the Stage 2 explanation.

9.2 Expected Sources of False Positives

Anticipated false-positive mechanisms include generic error pages that mimic SQL/database errors, unstable response bodies interpreted as Boolean differentials, network delay mistaken for time-based injection, reflection without executable context treated as XSS, a missing token treated as exploitable CSRF, redirects that remain origin-restricted, file-path strings disclosed without unauthorized read, and missing-header alerts repeated across pages. WAF-like delay is not present in the baseline, but target saturation can create the same symptom; the resource exclusion rule addresses it. Each FP discussion must cite the reproduced request and control response.

The distinction between an invalid alert and a duplicate is preserved. Ten identical missing-header instances may be accurate but represent one systemic control. Duplicate suppression introduced in ZAP 2.17.0 [19] may therefore

reduce alert count without reducing unit-level recall. Raw count and security coverage can move in opposite directions.

9.3 Tool-Specific Limitations to Examine

ZAP's strengths are openness, automation, inspectable rules, and broad export, but its results may depend heavily on add-ons, authentication context, AJAX/client exploration, and alert threshold. A published configuration without the add-on manifest is not reproducible. Acunetix may supply mature crawling, centralized governance, and polished remediation, but trial scope, opaque detection logic, product updates, and license-controlled APIs can impede replication. Cloud execution prevents valid local resource comparison. Burp Community offers exceptional visibility into HTTP requests and supports deliberate student reasoning, but it has no native automated scanner, no automatic crawler, and no automated vulnerability report in the evaluated edition [20], [21]. Treating manual Repeater findings as if a scanner autonomously produced them would be misleading.

TABLE XII. ADVANTAGES, LIMITATIONS, AND INVALID INFERENCES

Tool	Capability-based advantages	Material limitations in this study	Inference that must not be made
ZAP	Open source; active/passive rules; automation; API; multiple exports	Add-on/config dependence; modern-app/auth reachability; resource-intensive browser exploration	More alert instances necessarily mean more vulnerabilities
Burp Community	Strong proxy, Repeater, comparison, and pedagogical request visibility; no acquisition cost	No Burp Scanner or automatic crawler/report; throttled demo Intruder; analyst-dependent	Community results estimate Burp Professional Scanner performance
Acunetix	Automated crawler/DAST; centralized findings; report and API integration	Licensed/opaque engine; version/VDB drift; cloud resource opacity; trial restrictions	A cloud trial's local CPU/RAM is comparable with local tools

9.4 Decision Framework

Selection is a constrained decision, not a universal league table. An institution requiring a headless CI gate must first exclude products without a reproducible scanner API. A course teaching HTTP structure may prioritize transparent interception and replay. A consultancy may require authenticated crawling, evidence-rich reports, issue governance, and vendor support. Only products passing the hard gate are scored under that profile. Rank stability is reported under weight variation.

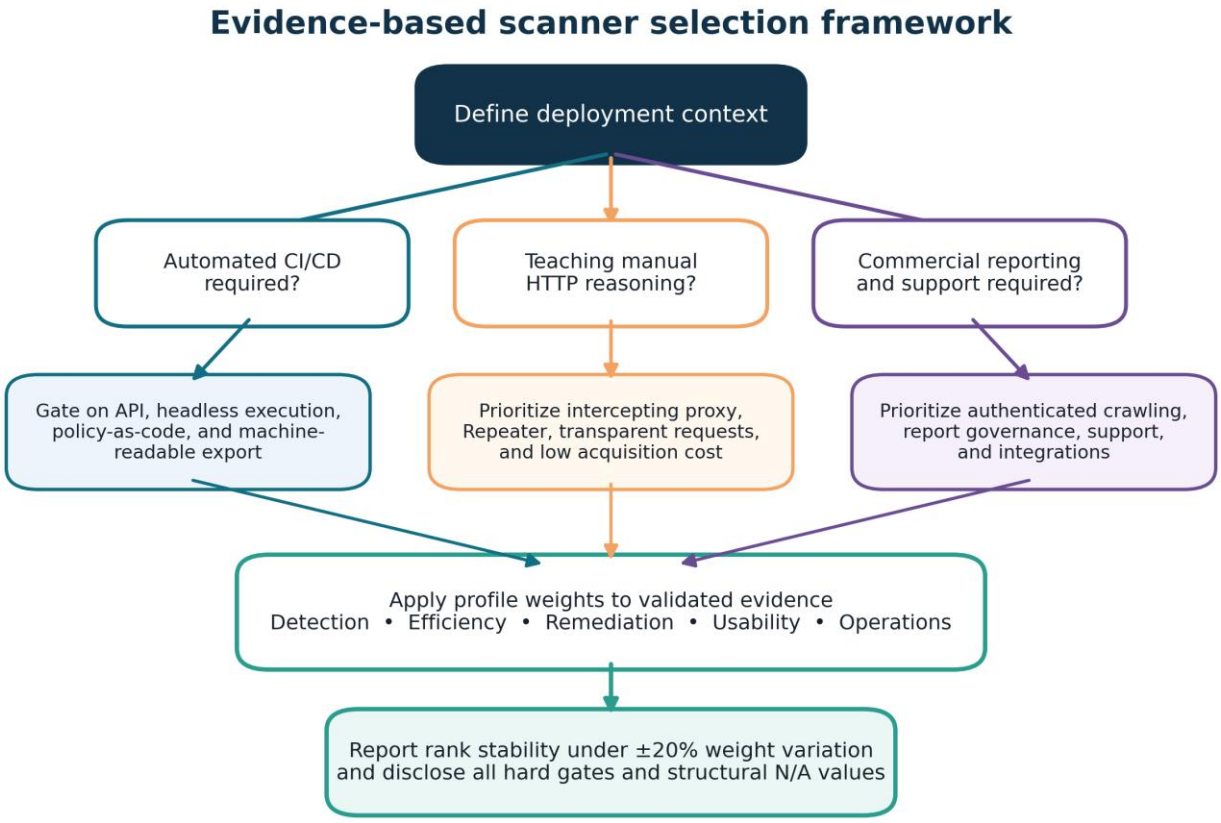

***Fig. 6**. Evidence-based, profile-sensitive scanner selection framework.*

10. THREATS TO VALIDITY

10.1 Construct Validity

The principal construct threat is equating alerts with vulnerabilities. Frozen test units, duplicate collapse, manual proofs, and matched controls reduce this risk. OWASP categories are broad and overlapping; context-sensitive CWE-to-OWASP mapping and a published mapping table make judgement visible. FPR depends on the chosen negative controls, so their composition and balance are reported. Independent CVSS scoring reduces dependence on vendor severity, but CVSS itself represents a scenario and not organization-specific business impact.

Remediation and usability are multi-dimensional human constructs. Established SUS and NASA-TLX instruments, objective task outcomes, masked report excerpts, and agreement statistics improve measurement. The custom remediation rubric is published in full; its content validity must be reviewed by at least two external application-security experts before recruitment.

10.2 Internal Validity

Authentication failure, cache state, mutable databases, scan-induced application changes, network variance, scanner updates, and target saturation could cause false differences. Snapshot reversion, deterministic seeding, sentinel hashes, isolated networking, randomized block order, one unmeasured warm-up, frozen versions, and target telemetry address these threats. Scanner operators cannot be blinded, but adjudicators are. Configuration decisions are frozen before reports are inspected.

The learning study faces carry-over. Latin-square ordering, equivalent but not identical target instances, standardized training, washout/reset periods, and prior-experience measurement reduce it. Participants cannot be blinded to interfaces, and brand attitudes remain possible.

10.3 External Validity

Two deliberately vulnerable applications do not represent the diversity of production frameworks, WAFs, federated identity, GraphQL, WebSockets, rate limits, microservices, or custom business workflows. DVWA and Juice Shop are also well-known; a scanner may contain target-specific signatures. The architecture contrast improves external validity relative to one target, but generalization remains analytical rather than statistical. Version pinning makes the experiment reproducible while limiting claims to those builds.

10.4 Conclusion and Statistical Validity

Ten repetitions provide useful timing and resource distributions but do not multiply the number of independent defect instances. Pairing and unit-level tests avoid pseudo-replication. Small samples reduce normality-test power; plots, robust intervals, non-parametric checks, and effect sizes are therefore emphasized. Multiple comparisons are Holm-corrected. Timeouts require censored analysis. Any protocol deviation, missing block, low assessor agreement, or post-freeze tool update is reported with a sensitivity analysis.

11. PRACTICAL IMPLICATIONS AND CONDITIONAL RECOMMENDATIONS

The following recommendations are based on documented capability and secure-testing principles, not on unobserved winners.

11.1 Recommendations for Developers and DevSecOps Teams

use DAST as one control in a layered programme. Gate builds on reproducible, validated findings rather than all raw alerts; retain request/response evidence; baseline known issues; and route logical authorization, design, and logging requirements to tests that can observe those properties. NIST SSDF and CISA Secure by Design favor repeatable engineering controls and default-safe behavior [5], [10], while patch-management planning governs vulnerable-component findings [77]. A product without headless execution cannot serve as the sole CI scanner.

11.2 Recommendations for Penetration Testers

combine an automated crawler/scanner with an intercepting proxy and manual role/workflow analysis. Use the seed-controlled findings to distinguish detector gaps from navigation gaps. Never report a missing token, reflected marker, or version banner as an exploitable high-risk defect without an oracle and a reproducer. Preserve native evidence and record the exact scanner build.

11.3 Recommendations for Universities

use ZAP when students need a no-cost, automatable DAST pipeline and access to rule/report internals; use Burp Community when the learning objective is manual HTTP interception, request editing, sequencing, and validation; and use Acunetix when an academic license supports enterprise-style scan governance, authenticated crawling, and report interpretation. These tools are complementary teaching instruments. Burp Community should not be advertised to students as containing Burp Scanner.

11.4 Recommendations for Small and Medium-Sized Enterprises

apply hard gates in this order: authorization to scan; target and authentication support; deployability inside the network; acquisition and maintenance cost; actionable evidence and false-positive burden; CI/report integration; and staff skill. A free tool with adequate automation may outperform an unused commercial subscription, while a managed commercial tool may justify cost where staff time and governance dominate. Pilot against an internal truth set before procurement.

11.5 Context-Specific Selection Matrix

TABLE XIII. TOOL SELECTION MATRIX BEFORE EMPIRICAL SCORING

Context	Hard requirement	Candidate emphasis	Exclusion trigger
CI/CD security gate	Headless repeatability, API/exit policy, machine-readable export	ZAP or licensed Acunetix subject to measured evidence	Burp Community lacks native scan operation
Introductory web-security laboratory	Low cost, visible traffic, safe manual replay	Burp Community and ZAP in complementary exercises	Any Internet-facing vulnerable target
Advanced DAST course	Authenticated crawl, automation, ground-truth validation	ZAP; Acunetix if academic license permits artifact access	Opaque cloud run without evidence export
Penetration-testing workflow	Automated coverage plus manual verification	Automated scanner plus intercepting proxy	Scanner-only assertion of business-logic completeness
SME procurement	Staff time, governance, support, total cost	Profile-weighted pilot	Ranking based only on raw alert count

12. LIMITATIONS OF THE PROTOCOL

The protocol cannot create an automated Burp Community result because the edition does not implement that function. This is an informative limitation of the requested comparison, not a missing experiment. The fixed 60-min cap may censor complex scans. The custom negative controls improve FPR validity but may not reproduce every benign pattern encountered in production. Building DVWA on a current PHP/Apache/MySQL stack can change incidental banners and parsing behavior relative to older studies. The human study requires ethics approval and adequate recruitment. Licensed Acunetix artifacts may not be fully redistributable. Finally, a Stage 1 manuscript cannot provide the requested scanner winner, p-values, or performance charts until measured evidence exists.

13. FUTURE WORK

After confirmatory execution, future replications should add WebGoat, OWASP Benchmark, API-specific fixtures, GraphQL, WebSockets, multi-factor authentication, WAF-mediated targets, and patched real applications. A parallel experiment using Burp Professional would permit a genuine three-engine autonomous DAST comparison while preserving the present Community-edition education question. Longitudinal replications could quantify version drift and rule/VDB updates. Mutation-based ground truth, code coverage, and server-side taint traces could further separate crawler, payload, and oracle failures. Cost-effectiveness could be expressed as analyst minutes per validated high-severity unit rather than license price alone.

14. CONCLUSION

This Stage 1 article specifies a complete experimental comparison without pretending that unexecuted scans are evidence. The design uses DVWA and Juice Shop, current pinned products, matched positive and negative test units, two discovery modes, ten randomized repetitions, process-tree telemetry, blinded adjudication, independent CVSS scoring, a counterbalanced education study, and a pre-specified statistical plan. It also corrects a central category error in the proposed comparison: Burp Suite Community Edition is a manual toolkit and does not include Burp Scanner. The resulting autonomous and analyst-assisted tracks answer different but useful questions.

Once populated with authentic reports and monitoring logs, the protocol will answer which autonomous scanner detects more validated units, which produces fewer false positives, which covers more of OWASP Top 10:2021, which completes faster, which provides stronger remediation, and which product best fits defined educational and operational profiles. Until then, all winners, confidence intervals, p-values, and outcome figures remain intentionally absent.

Compliance with ethical standards

Disclosure of conflict of interest

The authors declare that they have no conflict of interest.

REFERENCES

[1] OWASP Foundation, "OWASP Top 10:2021," 2021. [Online]. Available: https://owasp.org/Top10/. Accessed: Jul. 15, 2026.
[2] OWASP Foundation, "Web Security Testing Guide, version 4.2," 2023. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/. Accessed: Jul. 15, 2026.
[3] OWASP Foundation, "Application Security Verification Standard 4.0.3," 2021. [Online]. Available: https://owasp.org/www-project-application-security-verification-standard/. Accessed: Jul. 15, 2026.
[4] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, Technical Guide to Information Security Testing and Assessment, NIST SP 800-115. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2008, doi: 10.6028/NIST.SP.800-115.
[5] M. Souppaya, K. Scarfone, and D. Dodson, Secure Software Development Framework (SSDF) Version 1.1, NIST SP 800-218. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2022, doi: 10.6028/NIST.SP.800-218.
[6] Joint Task Force, Security and Privacy Controls for Information Systems and Organizations, NIST SP 800-53 Rev. 5. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2020, doi: 10.6028/NIST.SP.800-53r5.
[7] MITRE, "Common Weakness Enumeration (CWE)," 2026. [Online]. Available: https://cwe.mitre.org/. Accessed: Jul. 15, 2026.
[8] MITRE, "2024 CWE Top 25 Most Dangerous Software Weaknesses," 2024. [Online]. Available: https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html. Accessed: Jul. 15, 2026.
[9] Forum of Incident Response and Security Teams (FIRST), "Common Vulnerability Scoring System version 4.0: Specification Document," 2023. [Online]. Available: https://www.first.org/cvss/v4.0/specification-document. Accessed: Jul. 15, 2026.
[10] Cybersecurity and Infrastructure Security Agency, "Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Secure by Design Software," 2023. [Online]. Available: https://www.cisa.gov/resources-tools/resources/secure-by-design. Accessed: Jul. 15, 2026.
[11] Cybersecurity and Infrastructure Security Agency, "Known Exploited Vulnerabilities Catalog," 2026. [Online]. Available: https://www.cisa.gov/known-exploited-vulnerabilities-catalog. Accessed: Jul. 15, 2026.
[12] CVE Program, "CVE: Common Vulnerabilities and Exposures," 2026. [Online]. Available: https://www.cve.org/. Accessed: Jul. 15, 2026.
[13] PCI Security Standards Council, "Payment Card Industry Data Security Standard, version 4.0.1," 2024. [Online]. Available: https://www.pcisecuritystandards.org/document_library/. Accessed: Jul. 15, 2026.
[14] International Organization for Standardization, ISO/IEC 25010:2023, Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Product Quality Model, Geneva, Switzerland, 2023.
[15] OWASP Juice Shop Project, "OWASP Juice Shop v20.1.1," 2026. [Online]. Available: https://github.com/juice-shop/juice-shop/releases/tag/v20.1.1. Accessed: Jul. 15, 2026.
[16] B. Kimminich et al., "Pwning OWASP Juice Shop: Official Companion Guide," 2026. [Online]. Available: https://pwning.owasp-juice.shop/companion-guide/latest/. Accessed: Jul. 15, 2026.
[17] R. Dawes and contributors, "Damn Vulnerable Web Application, release 2.5," 2025. [Online]. Available: https://github.com/digininja/DVWA/releases/tag/2.5. Accessed: Jul. 15, 2026.
[18] ZAP Project, "ZAP API and active/passive scanning documentation," 2026. [Online]. Available: https://www.zaproxy.org/docs/api/. Accessed: Jul. 15, 2026.
[19] ZAP Project, "ZAP 2.17.0," 2025. [Online]. Available: https://www.zaproxy.org/blog/2025-12-15-zap-2-17-0/. Accessed: Jul. 15, 2026.
[20] PortSwigger Ltd., "Burp Suite Community Edition," 2026. [Online]. Available: https://portswigger.net/burp/communitydownload. Accessed: Jul. 15, 2026.
[21] PortSwigger Ltd., "Running your first scan with Burp Suite Professional," 2026. [Online]. Available: https://portswigger.net/burp/documentation/desktop/getting-started/running-your-first-scan. Accessed: Jul. 15, 2026.
[22] PortSwigger Ltd., "Professional / Community 2026.4.3," 2026. [Online]. Available: https://portswigger.net/burp/releases/professional-community-2026-4-3. Accessed: Jul. 15, 2026.
[23] Acunetix, "How to Generate Reports," 2026. [Online]. Available: https://www.acunetix.com/support/docs/wvs/generating-reports/. Accessed: Jul. 15, 2026.

[24] Acunetix, "Integrating Acunetix with GitLab for CI/CD," 2025. [Online]. Available: https://www.acunetix.com/support/docs/integrating-acunetix-with-gitlab-for-ci-cd/. Accessed: Jul. 15, 2026.

[25] Acunetix, "Acunetix 360 On-Demand v26.6.1 release notes," 2026. [Online]. Available: https://www.acunetix.com/changelogs/acunetix-360-on-demand/v26-6-1-16-june-2026/. Accessed: Jul. 15, 2026.

[26] Apache Software Foundation, "Apache HTTP Server 2.4.68," 2026. [Online]. Available: https://httpd.apache.org/download.cgi. Accessed: Jul. 15, 2026.

[27] PHP Group, "Supported PHP versions and releases," 2026. [Online]. Available: https://www.php.net/supported-versions.php. Accessed: Jul. 15, 2026.

[28] Oracle, "MySQL 8.0.46 release notes," 2026. [Online]. Available: https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-46.html. Accessed: Jul. 15, 2026.

[29] S. Alazmi, and D. C. De Leon, "A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners," IEEE Access, vol. 10, pp. 33200-33219, 2022, doi: 10.1109/ACCESS.2022.3161522.

[30] M. Albahar, D. Alansari, and A. Jurcut, "An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities," Electronics, vol. 11, no. 19, pp. 2991, 2022, doi: 10.3390/electronics11192991.

[31] E. A. Altulaihan, A. Alismail, and M. Frikha, "A Survey on Web Application Penetration Testing," Electronics, vol. 12, no. 5, pp. 1229, 2023, doi: 10.3390/electronics12051229.

[32] M. Aydos, Ç. D. Aldan, E. Coşkun, and A. Soydan, "Security testing of web applications: A systematic mapping of the literature," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 9, pp. 6775-6792, 2022, doi: 10.1016/j.jksuci.2021.09.018.

[33] S. Qadir, E. Waheed, A. Khanum, and S. Jehan, "Comparative evaluation of approaches & tools for effective security testing of Web applications," PeerJ Computer Science, vol. 11, pp. e2821, 2025, doi: 10.7717/peerj-cs.2821.

[34] R. Amankwah, J. Chen, P. K. Kudjo, and D. Towey, "An empirical comparison of commercial and open-source web vulnerability scanners," Software: Practice and Experience, vol. 50, no. 9, pp. 1842-1857, 2020, doi: 10.1002/spe.2870.

[35] J. Shahid, M. K. Hameed, I. T. Javed, K. N. Qureshi, M. Ali, and N. Crespi, "A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions," Applied Sciences, vol. 12, no. 8, pp. 4077, 2022, doi: 10.3390/app12084077.

[36] F. Mateo Tudela, J. R. Bermejo Higuera, J. Bermejo Higuera, J. A. Sicilia Montalvo, and M. I. Argyros, "On Combining Static, Dynamic and Interactive Analysis Security Testing Tools to Improve OWASP Top Ten Security Vulnerability Detection in Web Applications," Applied Sciences, vol. 10, no. 24, pp. 9119, 2020, doi: 10.3390/app10249119.

[37] A. Al Anhar, and Y. Suryanto, "Evaluation of Web Application Vulnerability Scanner for Modern Web Application," in 2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST), 2021, pp. 200-204, doi: 10.1109/icaicst53116.2021.9497831.

[38] B. Zukran, and M. M. Siraj, "Performance Comparison on SQL Injection and XSS Detection using Open Source Vulnerability Scanners," in 2021 International Conference on Data Science and Its Applications (ICoDSA), 2021, pp. 61-65, doi: 10.1109/icodsa53588.2021.9617484.

[39] R. Y. Ibrahim, and M. M. Rosli, "Evaluation of Web Application Vulnerability Scanners using SQL Injection Attacks," in 2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2023, pp. 1-6, doi: 10.1109/icraie59459.2023.10468295.

[40] D. B. Cruz, J. R. Almeida, and J. L. Oliveira, "Open Source Solutions for Vulnerability Assessment: A Comparative Analysis," IEEE Access, vol. 11, pp. 100234-100255, 2023, doi: 10.1109/ACCESS.2023.3315595.

[41] M. A. Kunda, and I. Alsmadi, "Practical web security testing: Evolution of web application modules and open source testing tools," in 2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), 2022, pp. 152-155, doi: 10.1109/idsta55301.2022.9923130.

[42] P. Lachkov, L. Tawalbeh, and S. Bhatt, "Vulnerability assessment for applications security through penetration simulation and testing," Journal of Web Engineering, vol. 21, no. 7, pp. 2187–2208, 2022, doi: 10.13052/jwe1540-9589.2178.

[43] Z. Yin, Y. Xu, F. Ma, H. Gao, L. Qiao, and Y. Jiang, "Scanner++: Enhanced Vulnerability Detection of Web Applications with Attack Intent Synchronization," ACM Transactions on Software Engineering and Methodology, vol. 32, no. 1, pp. 1-30, 2023, doi: 10.1145/3517036.

[44] X. Li, B. Amin Azad, A. Rahmati, and N. Nikiforakis, "Scan Me If You Can: Understanding and Detecting Unwanted Vulnerability Scanning," in Proceedings of the ACM Web Conference 2023, 2023, pp. 2284-2294, doi: 10.1145/3543507.3583394.

[45] B. Eriksson, G. Pellegrino, and A. Sabelfeld, "Black Widow: Blackbox Data-driven Web Scanning," in 2021 IEEE Symposium on Security and Privacy (SP), 2021, pp. 1125-1142, doi: 10.1109/sp40001.2021.00022.

[46] G. Pellegrino, C. Tschürtz, E. Bodden, and C. Rossow, "jÄk: Using Dynamic Analysis to Crawl and Test Modern Web Applications," in Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 295-316, doi: 10.1007/978-3-319-26362-5_14.

[47] A. Doupé, M. Cova, and G. Vigna, "Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners," in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 111-131, doi: 10.1007/978-3-642-14215-4_7.

[48] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the Art: Automated Black-Box Web Application Vulnerability Testing," in 2010 IEEE Symposium on Security and Privacy, 2010, pp. 332-345, doi: 10.1109/SP.2010.27.

[49] Y. Makino, and V. Klyuev, "Evaluation of web vulnerability scanners," in 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015, pp. 399-402, doi: 10.1109/idaacs.2015.7340766.

[50] B. Mburano, and W. Si, "Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark," in 2018 26th International Conference on Systems Engineering (ICSEng), 2018, pp. 1-6, doi: 10.1109/icseng.2018.8638176.

[51] M. Vieira, N. Antunes, and H. Madeira, "Using web security scanners to detect vulnerabilities in web services," in 2009 IEEE/IFIP International Conference on Dependable Systems & Networks, 2009, pp. 566-571, doi: 10.1109/dsn.2009.5270294.

[52] J. Fonseca, M. Vieira, and H. Madeira, "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks," in 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007), 2007, pp. 365-372, doi: 10.1109/prdc.2007.55.

[53] J. Bozic, and F. Wotawa, "Planning-based security testing of web applications with attack grammars," Software Quality Journal, vol. 28, no. 1, pp. 307-334, 2020, doi: 10.1007/s11219-019-09469-y.

[54] M. Peroli, F. De Meo, L. Viganò, and D. Guardini, "MobSTer: A model-based security testing framework for web applications," Software Testing, Verification and Reliability, vol. 28, no. 8, pp. e1685, 2018, doi: 10.1002/stvr.1685.

[55] M. Alidoosti, A. Nowroozi, and A. Nickabadi, "Semantic web Racer: Dynamic security testing of the web application against race condition in the business layer," Expert Systems with Applications, vol. 195, pp. 116569, 2022, doi: 10.1016/j.eswa.2022.116569.

[56] K. Abdulghaffar, N. Elmrabit, and M. Yousefi, "Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners," Computers, vol. 12, no. 11, pp. 235, 2023, doi: 10.3390/computers12110235.

[57] P. S. S. K. Gandikota, D. Valluri, S. B. Mundru, G. K. Yanala, and S. Sushaini, "Web Application Security through Comprehensive Vulnerability Assessment," Procedia Computer Science, vol. 230, pp. 168-182, 2023, doi: 10.1016/j.procs.2023.12.072.

[58] N. B. Chaleshtari, F. Pastore, A. Goknil, and L. C. Briand, "Metamorphic testing for web system security," IEEE Transactions on Software Engineering, vol. 49, no. 6, pp. 3430–3471, 2023, doi: 10.1109/TSE.2023.3256322.

[59] C. C. Aladi, "Web Application Security: A Pragmatic Exposé," Digital Threats: Research and Practice, vol. 5, no. 2, pp. 1-9, 2024, doi: 10.1145/3644394.

[60] D. Sacher, "Fingerpointing False Positives," Digital Threats: Research and Practice, vol. 1, no. 1, pp. 1-7, 2020, doi: 10.1145/3370084.

[61] P. Chapman, and D. Evans, "Automated black-box detection of side-channel vulnerabilities in web applications," in Proceedings of the 18th ACM conference on Computer and communications security, 2011, pp. 263-274, doi: 10.1145/2046707.2046737.

[62] A. L. S. Pupo, J. Nicolay, and E. G. Boix, "Deriving Static Security Testing from Runtime Security Protection for Web Applications," The Art, Science, and Engineering of Programming, vol. 6, no. 1, pp. 1, 2021, doi: 10.22152/programming-journal.org/2022/6/1.

[63] V. Atlidakis, P. Godefroid, and M. Polishchuk, "RESTler: Stateful REST API Fuzzing," in 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019, pp. 748-758, doi: 10.1109/ICSE.2019.00083.

[64] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in Software Engineering. Springer Berlin Heidelberg, 2012, doi: 10.1007/978-3-642-29044-2.

[65] D. Lakens, "Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs," Frontiers in Psychology, vol. 4, 2013, doi: 10.3389/fpsyg.2013.00863.

[66] S. S. SHAPIRO, and M. B. WILK, "An analysis of variance test for normality (complete samples)," Biometrika, vol. 52, no. 3-4, pp. 591-611, 1965, doi: 10.1093/biomet/52.3-4.591.

[67] Y. Benjamini, and Y. Hochberg, "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing," Journal of the Royal Statistical Society Series B: Statistical Methodology, vol. 57, no. 1, pp. 289-300, 1995, doi: 10.1111/j.2517-6161.1995.tb02031.x.

[68] F. Wilcoxon, "Individual Comparisons by Ranking Methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80, 1945, doi: 10.2307/3001968.
[69] J. Cohen, "A Coefficient of Agreement for Nominal Scales," Educational and Psychological Measurement, vol. 20, no. 1, pp. 37-46, 1960, doi: 10.1177/001316446002000104.
[70] P. E. Shrout, and J. L. Fleiss, "Intraclass correlations: Uses in assessing rater reliability.," Psychological Bulletin, vol. 86, no. 2, pp. 420-428, 1979, doi: 10.1037/0033-2909.86.2.420.
[71] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research," in Human Mental Workload, P. A. Hancock and N. Meshkati, Eds. Amsterdam, The Netherlands: North-Holland, 1988, pp. 139–183, doi: 10.1016/S0166-4115(08)62386-9.
[72] T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861-874, 2006, doi: 10.1016/j.patrec.2005.10.010.
[73] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," Journal of the American Statistical Association, vol. 22, no. 158, pp. 209–212, 1927, doi: 10.1080/01621459.1927.10502953.
[74] J. Brooke, "SUS: A quick and dirty usability scale," in Usability Evaluation in Industry, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, Eds. London, U.K.: Taylor & Francis, 1996, pp. 189–194.
[75] S. Holm, "A simple sequentially rejective multiple test procedure," Scandinavian Journal of Statistics, vol. 6, no. 2, pp. 65–70, 1979.
[76] Student, "The probable error of a mean," Biometrika, vol. 6, no. 1, pp. 1–25, 1908, doi: 10.2307/2331554.
[77] M. Souppaya and K. Scarfone, Guide to Enterprise Patch Management Planning: Preventive Maintenance for an Organization's Technology, NIST SP 800-40 Rev. 4. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2022, doi: 10.6028/NIST.SP.800-40r4.

APPENDIX A. REPLICATION CHECKLIST

Freeze tool editions, builds, add-ons/VDB, application commits, runtime versions, browser, VM definitions, and network policy.
Validate and hash positive and matched negative test units before opening scanner reports.
Complete authentication and sentinel checks; archive configuration hashes.
Execute one warm-up and ten randomized measured blocks per automated scanner/application/mode.
Preserve raw reports, process-tree telemetry, target logs, packet metadata, seed corpus, and hashes.
Populate a complete run × test-unit matrix; retain out-of-universe findings separately.
Conduct independent masked adjudication and publish agreement before consensus.
Run the supplied analysis script without editing the frozen analysis functions.
Insert observed tables and generated figures; mark all N/A values and deviations.
Deposit redacted data, code, environment manifests, and an immutable archive identifier.