INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY FUTURISTIC DEVELOPMENT

URL Access & License Testing in Modern Web Application Firewalls

John Komarthi
San Jose, CA, USA

* Corresponding Author: **John Komarthi**

Article Info

P-ISSN: 3051-3618
E-ISSN: 3051-3626
Impact Factor (RSIF): 8.31
Volume: 07
Issue: 01
Received: 13-01-2026
Accepted: 11-02-2026
Published: 10-03-2026
Page No: 67-73

Abstract
The role of Modern web application firewalls (WAFs) in protecting web applications is very crucial. They filter malicious traffic and enforce access policies. It's a well-known fact that web applications encounter high risk and destruction if they don't have firewalls. Therefore, this paper aims to explain software license enforcement testing and Uniform Resource Locators (URLs) access controls that are useful for cybersecurity professionals. The key insights include how WAF handles URL-based access controls and license enforcement mechanisms. Additionally, it underlines the importance of core methodologies involved and case studies with a vendor-neutral and research-driven focus. The discussion comprises mitigating or bypassing procedures of WAFs in risk scenarios. This paper also examines the techniques of common license enforcement and testing processes to check how robust they are. Furthermore, a novel holistic testing framework is proposed to ensure that the URL access constraints and licensing restrictions are enforced properly. This framework is unified, and it is a blend of traditional penetration testing and WAF systematic evaluation. In the end, the paper discusses the real-world examples, threat models, and future work to improve the effectiveness of WAF and integrate testing strategies for industry best practices like OWASP, NIST, and MITRE. Real-world examples include a major cloud breach caused by WAF bypass, an authentication bypass vulnerability that affects license-restricted features, etc.

DOI: https://doi.org/10.54660/IJMFD.2026.7.1.67-73

Keywords: Authorization, Broken access control, License Enforcement, Software Licensing, Testing, URL Filtering, Web Application Firewall (WAF)

Introduction

Attackers primarily target web applications for broken access. Broken access control consistently ranks as the most critical vulnerability class. In the 2025 Top 10 OWASP list, broken access control occupied the first place. Among the 100% of tested applications, all showed some types of access control weaknesses [17]. Forceful browsing is one such access control weakness that is commonly exploited by guessing or directly navigating to unauthorized URLs and parameter tampering to elevate privileges [10]. These attacks allow malicious users to access restricted pages or functions. Often, it leads to privilege escalation or unauthorized data exposure. Thus, as a frontline defense, modern organizations rely on executing Web Application Firewalls to reduce such threats. So, WAFs are the first line of security for most of the enterprises that have public-facing apps [2]. By filtering HTTP requests, WAF detects injection attacks, OWASP Top 10 threats, and cross-site scripting. Nevertheless, if the WAF is misconfigured or license checks and access controls of underlying applications are broken, relying on WAFs is a false sense of protection [2], [11].

URL Access Control and License Enforcement Mechanisms are the two aspects of web security. These are important to protect data and sensitive features of the software. URL Access Control makes sure that certain websites are not reached by unauthorized users. Based on software license entitlements, license enforcement mechanisms restrict functionality, protect revenue, and prevent abuse of premium features.

Web Application Firewalls (WAFs)

WAF filters, monitors, and blocks HTTP(S) traffic that comes and goes from a web application between servers and clients, thus WAF is a security solution [2]. WAF puts some rules to incoming requests to recognize the attack patterns and enforces security policies. Signature-based filtering, anomaly detection, and customized rules for specific URLs are the ordinary detection techniques of WAF [2]. Signature-based filtering technique matches known malicious payload patterns. Likewise, anomaly detection flags deviations from normal traffic profiles. WAFs can block common attacks like SQL injection and Cross-site scripting (XSS), rate-limit abusive behaviors, etc. These common attacks can be blocked by WAFs through inspection of HTTP methods, URLs, headers, and body content. In front of web servers and API endpoints, they are often deployed as cloud services or appliances. Frequently, modern WAFs are doubled as Content Delivery Networks (CDNs) or reverse proxies and introduce configuration complexities [12]. Despite their strengths, WAFs are not a single solution for all security issues, but they can defend against known attackers. Whereas they cannot fix the logical flaws in applications [11]. Attackers regularly develop WAF evasion techniques like encoding, payload obfuscation, or using alternate HTTP methods to slip malicious requests past filters [2]. Moreover, architectural issues or misconfigurations allow attackers to bypass WAFs totally. This is the reason industry best practices like OWASP's "Defense in Depth" and NIST guidelines advocate using WAFs as the primary layer of security, accompanied by rigorous testing, secure coding, and monitoring [11].

Broken Access Control

Broken access control leads to data tampering, unauthorized data exposure, or complete account takeover, making it critical for web security [17]. Authorization or access control ensures users perform actions or view information that is permitted to them in secured designs. Yet, failures are not uncommon in secured designs. Sometimes they are severe too. According to the OWASP's data, in recent years, 94% of the applications tested had some kind of access control weaknesses [21]. These weaknesses constitute role privilege escalation, missing authorization checks on sensitive endpoints, and Insecure Direct Object References (IDOR), where identifiers undergo manipulation to access the data that is not permitted [17]. Forced or forceful browsing is the most common attack variant, also called direct URL access. Attackers manually or with the help of tools access hidden or unlinked URLs and finally reach confidential pages or admin interfaces, if they are not protected properly [20, 4]. Forced browsing is successful when the application enforces login or role checks on navigation links from the client side and not on the target pages [4]. Other tactics to alter URL parameters or HTTP methods to detour controls, like sending HEAD requests while GET is blocked.

License Enforcement

Separate from security roles, many enterprise applications carry through license-based access control. License enforcement mechanisms ensure purchases and permission levels. As per subscriptions or purchased licenses, usage levels and features are restricted. For instance, a basic user license doesn't allow access to premium features or APIs, or limit the number of transactions. Importantly, license enforcement is a kind of business logic access control tied to privileges rather than user identity. If the controls break, it directly shows an impact on fairness and revenue. Breaking of controls also leads to unfairly unlocked paid features for free i.e., making it an open resource. In the classic sense, license enforcement flaws may not breach integrity or confidentiality, but they allow authorized use of software resources. In some cases, it can be exploited for more entitlements within any application. Thus, testing license enforcement is a crucial part of a comprehensive security review, particularly for B2B software and on-premise products. License keys or tokens that are validated at local servers, checking feature flags at runtime, and usage monitors that stop functionality when the limit exceeds are some of the common license enforcement approaches [5], [6]. Cryptographically signed license tokens and server-side checks for every privileged action are robust implementations. Likewise, weak implementations may not validate authorization on each request, and they hide UI elements for unpaid parties but ignore backend checks by trusting user-side checks. Therefore, weak license enforcement makes exploitation easy for the attackers. They exploit by forging or replaying license tokens, client-side flags, or using back-end data for direct tampering to lift license restrictions [6].

Industry Standards and Guidance

OWASP's guidelines, NIST's security controls, and MITRE ATT&CK framework are the established security principles. OWASP's guidelines, like the OWASP Authorization Cheat Sheet, allow deny by default. It didn't provide any access unless clearly granted. It also carries out checks server-side for every request [15]. NIST's security controls advocate least privilege and server-side access enforcement. This is due to important requirements for federal systems, and mapping to controls in the Access Control family (AC-3, AC-6, etc.). The MITRE ATT&CK framework lists ordinary competitive behaviors; particularly, exploitation of Public-Facing Application (T1190), where vulnerable web services are targeted by attackers as a primary access vector [8]. Often, it involves activities like probing for misconfigurations, flaws in access control, faults in authentication, and bypassing WAF protections for unauthorized entry [8].

URL Access Control in WAFs

Modern WAFs are able to implement some aspects of URL access control, such as acting as a guard to complete the application's own authorization logic. Whitelisting/blacklisting of URLs is a basic feature of modern WAFs. Administrators configure WAF rules to allow requests to a certain set of URL types and HTTP methods, blocking the remaining by default. This is the "deny by default" principle. At the network edge, implementation of the deny by default principle can prevent access to unlisted i.e., potentially sensitive or unintended endpoints. For instance, for any user, a WAF is set to allow GET and POST on /public/* pathways, whereas for /admin/* paths, it requires an authenticated session cookie and drops requests where the criteria are not met. Many WAFs of corporate-level provide forceful browsing protection in unconventional ways. Usually, this process involves monitoring for the number of

URLs attacked by speedy and continuous access attempts. This reveals that the attacker is trying to recount the resources saved. The Citrix NetScaler, a cloud computing company, produces WAF documentation. This documentation explains that WAF recognizes aggressive attempts by attackers to reach multiple URLs. WAF detects this forceful browsing through URL Protection checks and blocks them [3]. WAFs also either temporarily block or rate-limit the clients that provoke multiple URL attacks through forced browsing signatures. Moreover, WAF maintains signature sets for regular administrative or sensitive paths (e.g.,/admin, /config, /database), so if the requests are not from the normal traffic, WAFs deny or alert them [12].

Another transparent ability of WAF is login URL enforcement and session tracking. WAF detects when there is an authentic user, because WAFs like F5 BIG-IP ASM/Advanced WAF permit login page configuration and session cookies. Then, WAF can organize the protection in a way that it is accessible only after a successful login. If an unauthorized client attempts to access a tagged page that requires authentication, the WAF can respond as a coarse access control gate with an HTTP 401/403 error. Through designated login URLs, cookies, or headers, the detection and protection feature of WAFs is effective in the application's authentication process [22], [23]. WAFs help to stop unexpected exposures, for example, by mistake, if a developer leaves an admin endpoint unprotected, the login process might catch the access attempts. However, it's not a complete foolproof method because the application uses complex or conscious authorization logic.

Method-based Access Rules

WAF policies allow administrators to specify some allowed HTTP methods for certain URL patterns. It reduces some access control bypasses where unconventional methods elude application checks. For example, on a resource, an attacker uses a PUT or HEAD request to check if the server allows the action without authorization when an application only executes authorization checks for GET and POST requests. Some frameworks process HEAD instead of GET but skip some checks [1]. WAF has been configured in a way that restricts the unusual requests so that it thwarts bypass requests towards the edge. Practically, with front-end server configuration, organizations restrict dangerous methods like TRACE or CONNECT and limit PUT/DELETE to certain API endpoints that are needed.

Virtual Patching

WAFs can do virtual patching for known access control vulnerabilities. The common emergency response of WAFs is restricting access [10]. An example situation is accessing an admin URL without login. In such a case, either the WAF restricts access or blocks the URL until the developers fix it.

License Enforcement Mechanisms

In web applications, License enforcement of software ensures that users' installations are according to the terms of use. For example, limiting the features as per the given license tier or preventing usage beyond an allotted quota. Traditionally, while not a security control, access control issues are seen as a subset of license enforcement failures. These failures are also called business logic vulnerabilities. A breach of license enforcement allows resource consumption or unauthorized feature access. This may have security implications. E.g., using a cheap license to access admin-level functionality indirectly permits increased advantages in the application. This scenario gives a serious business impact at a minimum when the software abuse is enabled.

Activation Codes / License Keys: Users have to use the given alphanumeric key to input into the application for offline (using an embedded algorithm) validation or for the license server's verification. This may unlock features or time slots. Online verification or cryptographically signed keys lessen the forging or guessing of keys. From the client's view, store verification or predictable keys come under weak security designs that allow attackers to reverse-engineer and deliver valid keys.

License Files: License files may be in the form of JSON or XML types. License files are loaded by the application to enforce the encoded policy. A license file might be provided with encoded entitlements like the number of users, expiry date, features enabled, etc. To prevent tampering, the vendor should sign and verify digitally and detect the modification; if not, the attackers may add other features or extend the expiration, etc. To detect the attempts of the attackers cryptographic signature is required.

Online License Servers: Many modern solutions use a cloud service or centralized license server as a modern solution. To get the license token or to check out a usage slot, the application uses user credentials or instance ID, which allows dynamic control like revoking licenses, monitoring usage. The deployment requires a positive response from the server before enabling features. Here, the security considerations are using secure channels, authenticating the application to the server, and handling failures gracefully. A fail-safe is a network issue that disables features if the license cannot be confirmed; if not, attackers might block the license call to bypass it. Responses should be signed, time-stamped, and maybe of one-time-use when attackers attempt man-in-the-middle (MITM) or replay old license responses [6].

Feature Toggles and Checks: Confidentially, applications have conditional logic like: if (license.allows ("AdvancedReport")){showReportUI();} else { hide or error;}. The hidden UI element is not just secure enforcement, but server-side functions or endpoint features that check the license before implementation. Relying on the client/browser to disable or hide buttons for unlicensed features is a classic mistake that exposes the endpoint on the server. Hidden endpoints and the server are invoked with tools like a crafted HTTP request or Postman. Attackers don't verify the license on the call but act. This situation is equal to a missing level access control function, except based on licensing rather than user roles. On every relevant request, license verification must be done on the server-side. This is analogous to how authorization is being done for each request to avoid license verification [15], [6].

Usage Enforcement: Certain usage limits, like data storage volume, API calls per day, and number of concurrent users, are permitted by some licenses. When limits are exceeded, execution needs to track usage counts and deny service via the license server or counters/timers in the application. A security challenge ensures the user cannot manipulate or reset

the counters in client-side enforcement scenarios. For example, if usage data is stored on the client-side in an insecure business process, an attacker can alter or potentially raise or reset their usage to avoid enforcement.

Attacks on License Mechanisms

Attackers, like sophisticated pirates or intruders with insider access, have developed various techniques to bypass them:

Client-Side Patching: To omit license checks, attackers patch binary code in mobile software or desktop software. It translates to intercepting and modifying JavaScript in the browser or API responses in web applications. If any portion of the license check is done on the client side, it's open to manipulation.

Impeding License Calls: By intercepting network calls to license servers with tools like Burp Suite, attackers replay or modify responses successfully. In case of an insecure protocol, the server will be faked. In the case of HTTPS, even to compromise a client machine, an attacker might install a fake certificate to snoop. Replay can be reduced by making sure license responses are signed and including a timestamp/nonce [6].

Exploiting License Endpoints: License enforcement also has vulnerabilities. A real-life, recent example is CVE-2023-46805. Ivanti Connect Secure, a VPN appliance, was an authentication bypass for CVE-2023-46805 in the web interface, involving a license enforcement web API [24, 14]. Attackers directly call an endpoint of license or authentication flows without proper authentication to access restricted resources, including admin functions. In that way, an attacker sidesteps the normal login. This example gives an idea about how a defect in the license management element becomes a security hole.

Another appropriate example is CVE-2025-40819 in Siemens SINEMA Remote Connect. It permitted database access to an attacker to modify a table and detour license limitations on the industrial control system software [7]. In such cases, low-privileged users or a malicious insider with DB write access increase their privileges because the application couldn't validate license info from the DB. The privileges permit paid features like maintenance and admin rights, etc. [7]. While this needed internal access, it represents poor design, like license checks. So, the data that can be altered directly without any verification can't be trusted.

Flag Tampering Feature: Attackers definitely tamper if feature flags for licenses are stored in client-controlled parameters or cookies. In cases where flipping a Boolean parameter from true to false in a web request permitted a feature to indicate license status, the server-side code trusted that parameter. Importantly, this is a vulnerability of broken access control. This means the server is determining license permission, but not the client.

The research shows that license enforcement testing is equal to access control testing. It identifies the protected actions and attempts to execute them under prohibited conditions. The double approach of testing the rules of both URL access and license is reflected in a unified framework. In reality, WAFs do not know about the built-in software licensing states.

If the pattern is known, WAF can block the invalid license keys in requests, but without integration, it cannot decide who is a 'licensed' user. WAFs can still help by protecting the interfaces that handle licensing by adding another layer of defense when the license enforcement fails.

Testing Techniques (Access+License)

Effective testing of the Web Application Firewalls (WAFs) and the associated controls need blending of the traditional web penetration testing with the WAF-specific evaluation. The objective is twofold: to verify that the unauthorized URL access is consistently blocked and to ensure license enforcement cannot be bypassed through crafted inputs or request sequences.

Testing URL access controls

Authorization testing follows established OWASP Web Security Testing Guide principles. Testers generally first identify sensitive URLs and functions and then attempt access under varied roles and conditions.

Role matrix testing

Test accounts with differing privilege levels, such as unauthenticated, regular, user, and admin, are created. With the help of each context, an attempt to access the URLs that are meant for other roles is made, for instance, by browsing directly to the admin-only endpoints as a normal user. Improper controls manifest as successful responses or inconsistent error handling (HTTP 200 instead of 403 or the login redirect). As the WAF enforces the login centrally, unauthenticated requests can be intercepted before reaching the application; this behavior has to be verified through responses and logs.

Forced browsing and enumeration

Automated discovery tools such as Burp, OWASP ZAP Forced Browse, ffuf, DirBuster, and gobuster are used to uncover the unlinked URLs or any hidden endpoints. The response codes and the content differences are the key indicators; a 200 OK on an unexpected URL suggests exposure. For strict WAF configurations, most of the enumeration attempts have to be blocked and should return uniform errors. Distinctive block pages or the status codes, such as 406, help the fingerprint enforcement. Any URL that behaves differently needs further investigation.

HTTP method tampering

The endpoints are generally tested with the help of alternate HTTP methods such as HEAD, POST, PUT, DELETE, and OPTIONS. The historical framework issues have enabled auth bypass on less common methods. The method override headers, such as X-HTTP-Method-Override, are tested to identify any naive WAF filtering that inspects only the visible method. A robust setup generally restricts the methods per endpoint and normalizes the overrides.

Parameter and path manipulation

Testers generally attempt the IDOR scenarios by modifying the identifiers in the URLs or parameters to access other users' resources. While this is largely an application-level issue, some WAFs detect anomalous access patterns and can rate-limit or block them. Any unexpected behavior is consistent with 'access denied' or 'not found' response.

Encoding and normalization bypass
URL encoding tricks such as double encoding tricks, encoded slashes, null bytes, and mixed case paths are used to probe discrepancies between WAF inspection and backend routing. WAFs which are properly configured normalize or block such requests and confirm the corresponding logs. They deliberately trigger known WAF signatures and help validate the detection while ensuring that legitimate traffic is unaffected.

Threat Models
External attacker unauthenticated: Generally, any internet-based adversary tries for unauthenticated access to the sensitive functionality via scanning, forced browsing, or misconfigurations. The techniques align with the MITR ATT&CK T1190. One of the key risks is the architectural WAF bypass and the direct access to the origin servers. Previous research has shown that the backend exposure enables complete WAF evasion, due to which testing includes the verification that all the traffic paths are properly restricted.

Authenticated user escalation: A legitimate and low-privileged user attempts to access the high-privilege features or the paid functionality. To set those methods, such as IDOR, forceful browsing, and invoking the endpoints, are enabled in the higher license tiers. This framework is extremely useful in the case of SaaS products with tiered licensing.

External attacker unauthenticated: Generally, an internet-based adversary tries for unauthenticated access to the sensitive functionality via scanning, forced browsing, or misconfigurations. The techniques align with the MITR ATT&CK T1190. One of the key risks is the architectural WAF bypass and direct access to the origin servers. Previous research has shown that the backend exposure enables complete WAF evasion, due to which testing includes the verification that all the traffic paths are properly restricted.

Authenticated user escalation: A legitimate and low-privileged user attempts to access the high-privilege features or the paid functionality. To set those methods, such as IDOR, forceful browsing, and invoking the endpoints, are enabled in the higher license tiers. This framework is extremely useful in the case of SaaS products with tiered licensing.

Advanced Persistent Threat: This adversary may possess partial internal knowledge or have partial internal access, using which manipulation of the license data is done, and the trust relationship is abused. The full WAF administrative compromise will not happen, but the model underscores the need to secure the licensing and administrative endpoints with the same rigor as the core application logic.

License abuse and piracy: The attackers aim to use the software beyond the licensed terms through reusing the keys, spoofing the license servers, or through manipulating the enforcement logic, while WAFs will not be able to prevent all forms of license theft, they can help with the detection through identifying any anomalous usage patterns such as a single key being used from multiple locations.

WAF-specific evasion and disruption: Attackers may attempt to overwhelm the WAF or fingerprint it, thus exploiting the known product-specific bypasses or the misconfigurations. Through testing that includes WAF fingerprinting and targeted evasion attempts, this threat is found, and this highlights the importance of vendor-aware tuning without over-reliance on the defaults.

Case Studies
Capital One breach (2019) WAF bypass via SSRF
During the Capital One incident, the attacker exploited the SSRF vulnerability, which the WAF failed to block, and because of this, the attacker has been granted access to the cloud metadata services and stole credentials. While the root flaw lies in the application logic, the WAF was intended to mitigate the control, and it did not detect the malicious request pattern. This case demonstrates that the WAFs cannot be the sole defense and need to be tested against non-traditional attack paths such as SSRF. Strong outbound filtering and explicit metadata protections could have prevented the escalation.

Ivanti VPN license portal flaw(2023) Auth Bypass
A critical authentication bypass has occurred in Ivanti Connect Secure, which allowed unauthenticated access to the restricted functionality via poorly connected endpoints. This was reportedly related to the licensing and configurations. Hidden or forgotten URLs lack proper authorization and turn into a full compromise vector, and were exploited by advanced threat actors. This case reinforces the importance of treating the licensing and administrative interfaces as high-risk attack surfaces and having them explicitly included in access control testing. Numerous incidents occur from broken access control, IDOR flaws, or weak license enforcement. While these breaches are not always publicized as breaches, such failures represent a systemic security design weakness.

Future Directions
The long-standing security principle is that access control failures can rarely be attributed to a single layer. While the WAFs demonstrably improve the resilience through filtering the malformed or overtly malicious requests, they remain highly dependent on the configuration quality, architectural placement, and continuous validation. The findings confirm the industry observations that WAFs are best treated as adaptive compensating controls, not as primary authorization mechanisms.

Vendor nuances and detection models
Even though this work remains vendor-neutral, the testing has revealed the meaningful differences in the detection philosophy. Any signature-heavy rule sets, such as strict CRS style deployments, tend to block more edge case bypass attempts but also require extensive tuning in order to manage false positives. Conversely, the cloud native WAFs that are optimized for compatibility are more permissive, and they require custom rules in order to address the encoding and method override edge cases. Future research can formally compare the detection efficacy across the rule paradigms (signature-based vs anomaly-based), specifically for authorization abuse and license misuse, which are underrepresented in the current benchmarks.

Machine-learning-based WAFs

The emerging WAFs are increasingly relying on ML to profile normal traffic and to flag any anomalies. While this is promising, such systems raise some questions. Can these models reliably detect the low and slow access control violations that resemble the legitimate user behavior, such as gradual IDOR enumeration or selective invocation of the premium-only endpoints? The attackers may deliberately train themselves to mimic normal traffic patterns, which reduces the anomaly signal. Rigorous adversarial testing of the ML-driven WAFs under these conditions is a clear area of future empirical study.

API-centric and microservices architectures

Modern systems are increasingly relying on APIs and microservices. In these environments, the authorization is fragmented across the API gateways, backend services, and service meshes, and often it is enforced via OAuth scopes or JWT claims. WAFs are offering API-aware features such as schema validation and rate enforcement; their effectiveness against broken object-level or function-level authorization will remain unclear. Extending the framework to incorporate OWASP API security top 10 testing methodologies, scope abuse, and cross-service license enforcement would significantly broaden the applicability.

DevSecOps and continuous validation

The future work focuses on operationalizing the framework, which includes the development of CI/CD integrations that automatically execute the authenticated access control tests, WAF evasion, and forced browsing attempts, probing builds, or staging deployments. Tooling that will produce the deterministic pass/fail signals for authorization regressions that would materially improve the adoption and reduce the reliance on periodic manual assessments.

License enforcement as a security signal

The licensing models are evolving toward feature flags, tenant-specific entitlements, usage-based billing, and enforcement logic that becomes both more complex and more security relevant. Behavioral license monitoring is another promising direction, where correlating the license entitlements with observed endpoint usage and flagging the inconsistencies in real-time. Whether it may be WAFs, or API gateways, or application layer analytics that are best suited for this role remains an open question.

Limitations

Scope and architecture constraints

Framework assumes that a web application that is properly fronted by a WAF does not deeply analyze the scenarios involving the compromised WAF management planes, upstream identity provider failures, or insider access to the rule configuration. While these risks are real, they fall outside the scope of the access control testing and into the broader governance and infrastructure security.

Generalization across vendors

Even after being vendor agnostic, WAF behavior varies significantly across the products and deployments. These techniques are effective against one WAF may fail against another because of the differences in the normalization, rule execution order, or decoding. The security practitioners have to adapt the framework to their company-specific WAF implementation and threat landscape.

Zero trust and identity-aware controls

At every request boundary, Zero Trust Architecture introduces a rise of identity-aware enforcement. In this type of model, WAFs inspect identity headers or JWTs. These models also enforce applications where policy decisions are traditionally handled. Though powerful, the convergence increases configuration complexity and blurs responsibility borderlines. In addition, identity claims for authorization and license semantics should be carefully mapped.

Ethical and Operational Considerations

Organizations should make sure of authorized testing, such as WAF-evasion testing and aggressive access control. Indiscriminate performance of authorized testing disrupts production systems as they are scoped and ideally conducted. Particularly while assessing third-party software, license enforcement testing should respect legal and contractual boundaries.

Conclusion

In web applications, weak license enforcement and broken access controls are the most critical weaknesses. To overcome these weaknesses, intelligent WAF configuration and robust application design are needed.

Enforce Defense in Depth: Single-layer security, like just a WAF or an application, is always unreliable. Server-side authorization checks for every request are very helpful [15]. Use of WAF strengthens and backstops the checks, such as illegitimate attempts and locks unused paths to lessen the attack surface [10], [3].

Proactive and Continuous Testing: To test high-risk access points and licensed features regularly, a structured approach is needed. To ensure that WAF configurations and apps are properly validated against current threats, incorporate known attack patterns such as OWASP Top 10, MITRE ATT&CK, and newly disclosed exploits into testing cycles [8], [10].

Stay Aligned with Standards: OWASP supplies multiple resources on securing web apps and testing authorization [10], [15]. NIST guidelines reinforce some principles, like deny-by-default and least privilege. They reflect on WAF policies and app codes [9]. This alignment improves the security posture of security, facilitates compliance, and communication with stakeholders.

Real-World Awareness: The Capital One breach and Ivanti vulnerability cases explain how subtle overlooked endpoints or misconfigurations lead to major incidents [12], [14]. Comprehensive coverage of all access vectors, like licensing and administrative interfaces, is not negotiable.

Holistic Framework: A holistic framework with elements like development, security, and operations collaboration should be used by the organizations to map and to know the whereabouts of access control, either in code vs. in WAF. Organizations use this unified framework to avoid gaps and assumptions and offer a design from deployment to monitoring.

In 2025, the cybersecurity landscape is beyond demand. Defense should include application logic and WAF rules as complementary parts of a defense strategy because attackers always search for the weakest link or a loophole across the total system. In conclusion, modern WAFs are powerful tools in the security system to block attacks like SQL injections and XSS. While Modern WAFs are not penetrable shields, they reduce dynamic browsing and other web exploits [3], [2] and contribute to access control enforcement. Organizations should stop using the "set and forget" approach to WAFs. Alternatively, using a four-step continuous verification with configure, test, monitor, and refine is helpful to make sure both app and WAF controls block the threats. In this way, both WAF and application security standards safeguard the web applications from license misuse and unauthorized access. So it protects sensitive data, functionality, and preserves software vendors' income. Eventually, to use the application, license controls and robust access provide trust for users, administrators, and the business.

References

1. OWASP Foundation. A01:2025 – Broken Access Control. OWASP Top 10 (2025) [Internet]. 2025 [cited 2026 Apr 28]. Available from: https://owasp.org
2. Nagaraj K. Web application firewall (WAF) bypass techniques that work in 2025 [Internet]. InfosecMatrix on Medium; 2025 Jul [cited 2026 Apr 28]. Available from: https://medium.com
3. Cloud Software Group. The URL protection checks (Citrix NetScaler WAF documentation) [Internet]. 2025 Sep [cited 2026 Apr 28]. Available from: https://docs.netscaler.com
4. MITRE. CAPEC-87: Forceful browsing. Common Attack Pattern Enumeration and Classification v3.9 [Internet]. 2023 [cited 2026 Apr 28]. Available from: https://capec.mitre.org
5. Schneider Electric. License enforcement and information. EcoStruxure Building Operation Help [Internet]. 2025 Oct [cited 2026 Apr 28]. Available from: https://ecostruxure-building-help.se.com
6. Robert A. Licensing model security review: finding and fixing the cracks [Internet]. Hoop.dev blog; 2025 Oct [cited 2026 Apr 28]. Available from: https://hoop.dev
7. Siemens ProductCERT. SSA-626856: Multiple vulnerabilities in SINEMA Remote Connect Server < V3.2 SP4 [Internet]. 2025 Dec [cited 2026 Apr 28]. Available from: https://cert-portal.siemens.com
8. MITRE ATT&CK. T1190 – Exploit public-facing application. ATT&CK for Enterprise v12 [Internet]. 2025 Oct [cited 2026 Apr 28]. Available from: https://attack.mitre.org
9. Scarfone K, Hoffman P. Guidelines on firewalls and firewall policy. NIST SP 800-41 Rev. 1 [Internet]. 2009 [cited 2026 Apr 28]. Available from: https://nvlpubs.nist.gov
10. OWASP Foundation. OWASP web security testing guide v4: Section 4.5 authorization testing [Internet]. 2014 [cited 2026 Apr 28]. Available from: https://owasp.org
11. Cavkusic S. The illusion of security: why relying solely on WAF is a bad practice [Internet]. Bright Security Blog; 2025 Feb [cited 2026 Apr 28]. Available from: https://brightsec.com
12. Zafran Research Team. BreakingWAF: widespread WAF bypass impacts nearly 40% of Fortune 100 [Internet]. 2024 Dec [cited 2026 Apr 28]. Available from: https://zafran.io
13. U.S. CISA. Known exploited vulnerabilities catalog – CVE-2023-46805 [Internet]. 2024 Jan [cited 2026 Apr 28]. Available from: https://tenable.com
14. SentinelLabs. CVE-2023-46805: Ivanti Connect Secure auth bypass flaw (analysis) [Internet]. 2025 Nov [cited 2026 Apr 28]. Available from: https://sentinelone.com
15. OWASP Foundation. Authorization cheat sheet: enforce least privilege and deny by default [Internet]. 2022 [cited 2026 Apr 28]. Available from: https://cheatsheetseries.owasp.org
16. OWASP Foundation. OWASP proactive controls: C1 define access control [Internet]. 2018 [cited 2026 Apr 28].
17. OWASP Foundation. OWASP ASVS 4.0 – V8 access control verification requirements [Internet]. 2019 [cited 2026 Apr 28].
18. NIST. SP 800-53 Rev. 5: security and privacy controls [Internet]. 2020 Sep [cited 2026 Apr 28].
19. CWE. CWE-863: incorrect authorization [Internet]. MITRE; 2019 [cited 2026 Apr 28].
20. PortSwigger. Web security academy lab: forced browsing to admin pages [Internet]. 2021 [cited 2026 Apr 28].
21. Acunetix. Broken access control [Internet]. [cited 2026 Apr 28]. Available from: https://www.acunetix.com/blog/web-security-zone/broken-access-control/
22. F5 Networks. ASM implementations: login enforcement and URL configuration [Internet]. [cited 2026 Apr 28]. Available from: https://techdocs.f5.com
23. F5 Community. ASM unauthenticated URLs with login enforcement [Internet]. [cited 2026 Apr 28]. Available from: https://community.f5.com
24. Vicarius. Two zero-day critical vulnerabilities in Ivanti (CVE-2023-48605 and CVE-2024-21887/21889) [Internet]. [cited 2026 Apr 28]. Available from: https://www.vicarius.io
25. Fastly. The WAF efficacy framework: measuring effectiveness of your WAF [Internet]. [cited 2026 Apr 28]. Available from: https://www.fastly.com
26. BugBase. Top 10 ways to bypass WAF [Internet]. [cited 2026 Apr 28]. Available from: https://bugbase.ai

How to Cite This Article

Komarthi J. URL access & license testing in modern web application firewalls. Int J Multidiscip Futur Dev. 2026;7(1):67–73. doi:10.54660/IJMFD.2026.7.1.67-73