Научная статья

МРНТИ 004.8:004.4:37.013 **https://doi.org/ 10.55956/GVNI2510**

С.А. Агабек
Магистрант,
Таразский университет имени М.Х. Дулати
Тараз, Казахстан
sultanbek.agabek@gmail.com
М.А.Ахметжанов
физика-математика ғылымдарының кандидаты
М.Х.Дулати атындағы Тараз университеті
Тараз қ., Қазақстан
ma.akhmetzhanov@dulaty.kz
Ж.К. Тасжурекова
Техника ғылымдарының кандидаты
М.Х. Дулати атындағы Тараз университеті
Тараз қ., Қазақстан
zhk.taszhurekova@dulaty.kz
Ж.Б. Садирмекова
PhD, ассоциированный профессор кафедры
“Информационные системы”
Таразский университет имени М.Х. Дулати
Тараз, Казахстан

ИССЛЕДОВАНИЕ УГРОЗ И МЕТОДОВ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ WEB-ПРИЛОЖЕНИЙ ПРИ НЕСАНКЦИОНИРОВАННЫХ ПРОНИКНОВЕНИЯХ

Аннотация. *В статье рассматриваются типичные угрозы, с которыми сталкиваются Web-приложения, и предложен практический подход к оценке их защищённости. Методика включает последовательные этапы — от разведки и инвентаризации сервисов до тестирования с ручной проверкой ключевых наблюдений. Показано, что надёжная защита Web-приложений достигается за счёт совокупности мер: грамотной настройки, чёткого контроля доступа, безопасного программирования и регулярной проверки системы. Практическая часть подкреплена инструментальными результатами, позволяющими фиксировать уязвимости и отслеживать эффективность улучшений.*

Ключевые слова*: Web-приложения, кибербезопасность, уязвимости, несанкционированный доступ, тестирование безопасности, OWASP, DevSecOps.*

Введение. Web-приложения – неотъемлемая часть современной цифровой инфраструктуры. Они обеспечивают работу онлайн-сервисов в госсекторе, бизнесе и повседневной жизни пользователей. Обычно такие системы включают клиентскую часть, серверную логику, API, базы данных и интеграции с внешними сервисами. Однако чем

сложнее архитектура, тем больше потенциальных уязвимостей. Ошибки в настройке, слабый контроль доступа или недостатки при обработке входных данных могут привести к серьёзным последствиям: от компрометации учётных записей до утечек данных и сбоев в работе.

Особенность современных атак – их поэтапность: от сбора информации и анализа инфраструктуры до поиска уязвимостей и закрепления в системе. Поэтому защита Web-приложений требует комплексного подхода: многоуровневых мер и регулярного пересмотра безопасности.

Цель статьи – систематизировать основные угрозы, связанные с несанкционированным доступом, и представить методику практической оценки защищённости Web-приложений на основе реальных сценариев.

Материалы и методы исследования

1. Анализ архитектуры Web-приложений. Изучены клиент-серверные модели, SPA-приложения, микросервисы и API-ориентированные системы.

2. Систематизация типов угроз и уязвимостей. Использованы официальные классификации уязвимостей Web-приложений, включая OWASP Top 10 [4].

3. Анализ методик атак. Рассмотрены основные этапы несанкционированного проникновения: разведка, сканирование, подтверждение уязвимостей и постэксплуатационные действия (в рамках этичных ограничений).

4. Исследование технологий защиты. Рассмотрены WAF, IDS/IPS, криптографические методы, DevSecOps-подходы и модели авторизации [1, 2].

5. Сравнительный анализ методов тестирования. Оценены возможности автоматизированных сканеров и преимущества ручного тестирования [3].

Практическая часть исследования строилась поэтапно – с акцентом на последовательность действий и реальную применимость методики. Всё начиналось с пассивной разведки (OSINT), в рамках которой я собрал инфраструктурные данные о домене и оценил, как система выглядит со стороны для потенциального злоумышленника (см. рис. 1–2). Затем перешёл к активной инвентаризации сервисов: определил, какие порты открыты, какие технологии используются и какие версии сервисов доступны извне (см. рис. 3). После этого выполнил перечисление ресурсов приложения – задача состояла в том, чтобы найти дополнительные точки входа: скрытые директории, тестовые разделы и резервные файлы (см. рис. 4). Дальнейший аудит конфигураций позволил выявить типовые ошибки, вроде отсутствия заголовков безопасности или устаревших компонентов (см. рис. 5). На финальном этапе я вручную перепроверил ключевые находки, анализируя HTTP(S)-трафик и структуру ответов сервера, чтобы убедиться в реальности угроз и исключить ложные срабатывания (см. рис. 6).

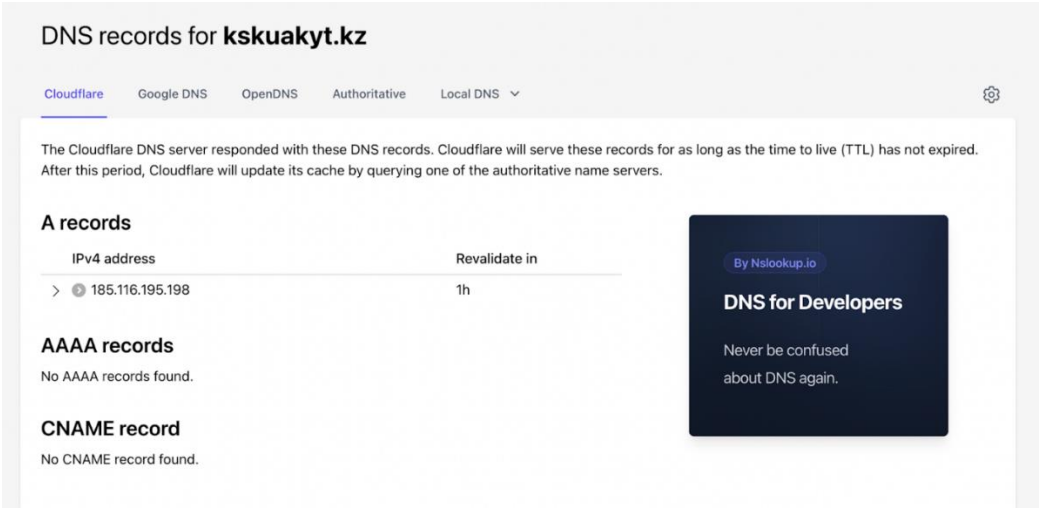

Рис. 1. Пассивная разведка: анализ DNS-записей и инфраструктурных признаков домена

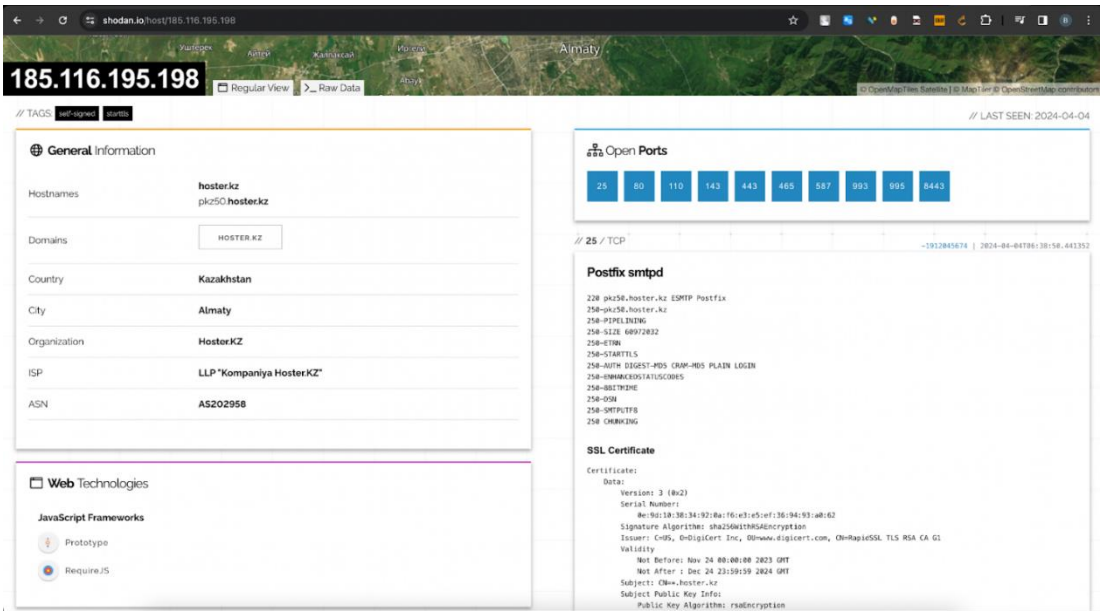

Рис. 2. Пассивная разведка: внешняя видимость сервисов по данным OSINT-источников

Результаты исследования

1) Угрозы по уровням архитектуры Web-приложений

Если опираться на общепринятые классификации [4], то можно выделить несколько уровней, на которых чаще всего проявляются риски для Web-приложений. В практической работе с безопасностью такие угрозы удобно рассматривать по слоям архитектуры:

- **Клиентский уровень** – сюда относятся такие уязвимости, как XSS, неправильные настройки CORS, отсутствие важных атрибутов у cookie, проблемы с безопасностью локального хранилища и атаки типа clickjacking.
- **Серверный уровень** – это уже ошибки в конфигурации веб-сервера, использование устаревших компонентов, слабые механизмы аутентификации и управления

сессиями, а также уязвимости, связанные с несанкционированной обработкой входных данных.

- **API-уровень** – здесь часто встречаются недостатки контроля доступа, отсутствие лимитов на частоту запросов, чересчур подробные сообщения об ошибках (что облегчает работу злоумышленнику), а также ошибки в авторизации и работе с токенами.
- **Инфраструктурный и облачный уровень** – это риски, связанные с управлением доступом отсутствием сетевой сегментации, неправильно настроенными контейнерами, публичными хранилищами или открытыми административными панелями.

Таблица 1

Основные угрозы и меры защиты

Уровень	Типичные угрозы	Возможные последствия	Методы защиты
Клиент	XSS, CORS, cookies	Кража токенов, подмена интерфейса	CSP, secure/httponly cookies
Сервер	misconfig, уязвимые версии	Утечки, компрометация	Hardening, обновления, WAF
API	ошибки контроля доступа	Эскалация прав	OAuth2/RBAC, rate-limit, логирование
Инфра	слабый IAM/RBAC	Масштабные утечки	IAM, сегментация, мониторинг

2) Активная разведка: доступные сервисы и инвентаризация

После пассивного сбора данных выполняется инвентаризация доступных сетевых сервисов и их характеристик. Этот этап важен, поскольку “лишние” сервисы и устаревшие версии повышают вероятность компрометации и усложняют контроль периметра. Результаты инвентаризации открытых сервисов и их характеристик представлены на рис. 3.

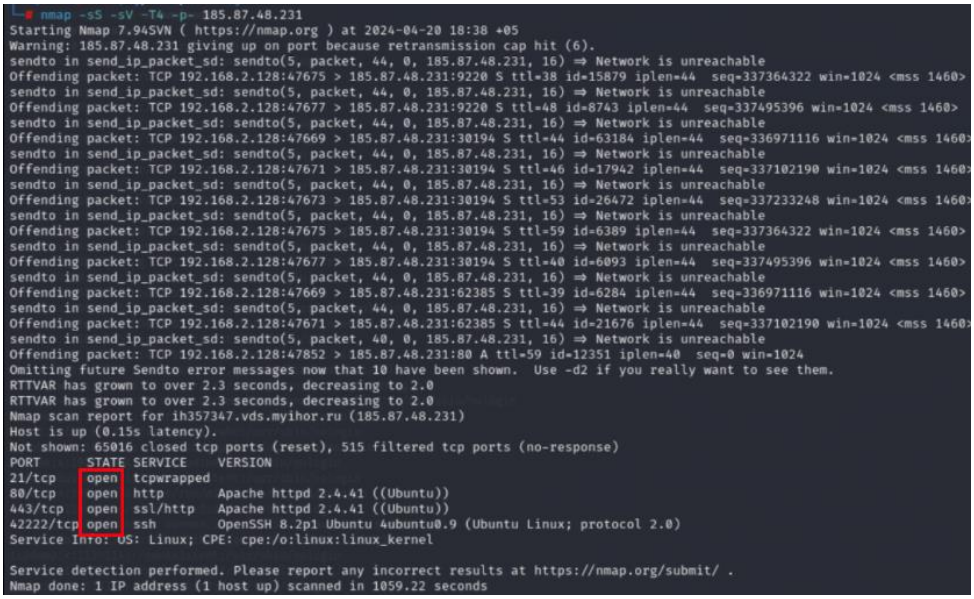

Рис. 3. Инвентаризация доступных сервисов и версий

3) Перечисление ресурсов приложения

Даже при корректной внешней конфигурации риски могут быть связаны с недокументированными ресурсами: тестовыми разделами, резервными файлами, служебными директориями. Поэтому отдельным этапом проводится перечисление путей (directories/files) для выявления дополнительных точек входа. Пример результатов перечисления директорий и ресурсов приложения приведён на рис. 4.

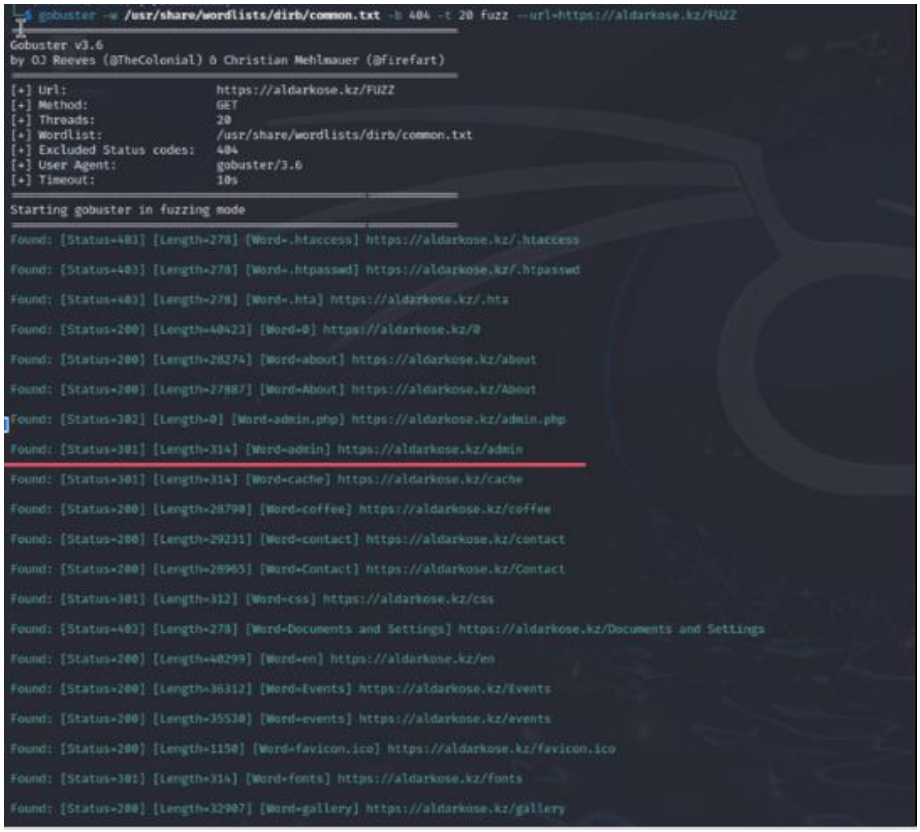

Рис. 4. Перечисление ресурсов Web-приложения: обнаруженные пути/директории

4) Первичный аудит конфигурации веб-уровня

Отдельную группу рисков составляют ошибки конфигурации веб-сервера и связанных компонентов. Такие проблемы часто устраняются настройками и обновлениями, что делает их “быстрыми победами” в повышении устойчивости. Иллюстрация результатов первичного аудита показана на рис. 5.

5) Ручная верификация: анализ HTTP(S)-трафика

Автоматизированные результаты и первичные предупреждения требуют ручной проверки, поскольку без контекста приложения невозможно корректно оценить влияние и критичность. В соответствии с практиками веб-тестирования, автоматизация применяется для покрытия типовых проблем, а ручная верификация — для подтверждения реального риска и исключения ложных срабатываний [3]. Пример ручной верификации через анализ HTTP(S)-трафика и

структуры ресурса приведён на рис. 6.

```
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          185.87.48.231
+ Target Hostname:    aldarkose.kz
+ Target Port:        443
---------------------------------------------------------------------------
+ SSL Info:        Subject:  /CN=aldarkose.kz
                   Altnames: aldarkose.kz
                   Ciphers:  AEAD-AES256-GCM-SHA384
                   Issuer:   /C=US/O=Let's Encrypt/CN=R3
+ Start Time:         2024-04-23 22:19:37 (GMT5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.41 (Ubuntu)
+ /: The site uses TLS and the Strict-Transport-Security HTTP header is not defined. See: https://developer.mozilla.org/en-US/doc
s/Web/HTTP/Headers/Strict-Transport-Security
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a differe
nt fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-heade
r/
+ /: Cookie PHPSESSID created without the secure flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.41 appears to be outdated (current is at least 2.4.58). Apache 2.2.34 is the EOL for the 2.x branch.
+ /: The Content-Encoding header is set to "deflate" which may mean that the server is vulnerable to the BREACH attack. See: http
://breachattack.com/
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: DEBUG HTTP verb may show server debugging information. See: https://docs.microsoft.com/en-us/visualstudio/debugger/how-to-en
able-debugging-for-aspnet-applications?view=vs-2017
+ /css/: Directory indexing found.
+ /css/: This might be interesting.
+ /job/: This might be interesting.
+ /library/: Directory indexing found.
+ /library/: This might be interesting.
+ /order/: This might be interesting.
+ /images/: Directory indexing found.
+ /kz/: This might be interesting: potential country code (Kazakhstan).
+ /ru/: This might be interesting: potential country code (Russian Federation).
+ /api.php?t_path_core=http://cirt.net/public/rfiinc.txt?&cmd=id: Retrieved access-control-allow-origin header: *.
+ /login.php: Admin login page/section found.
+ 7961 requests: 0 error(s) and 19 item(s) reported on remote host
```

Рис. 5. Первичный аудит веб-сервера и конфигурационные риски

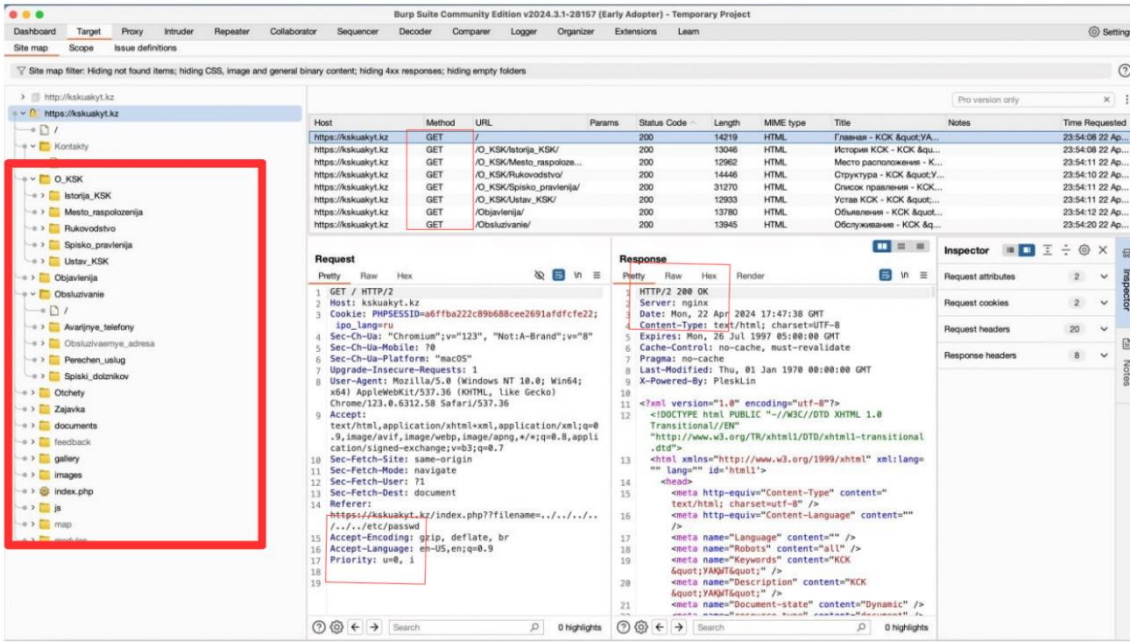

Рис. 6. Анализ HTTP(S)-трафика и структуры ресурсов приложения при верификации наблюдений

Обсуждение. Результаты показывают, что основные риски в Web-приложениях чаще всего связаны с ошибками конфигурации, устаревшими компонентами и недостаточным контролем доступа. Автоматические инструменты хорошо справляются с обнаружением типовых уязвимостей (см. рис. 3-5), но чтобы точно оценить критичность и исключить ложные срабатывания, необходима ручная проверка в контексте самого приложения (см. рис. 6) [3].

Для повышения защищённости стоит использовать многоуровневую

защиту: WAF, мониторинг, а также меры, встроенные в процесс разработки – например, DevSecOps [1, 2]. Особенно важны: регулярные обновления, корректные заголовки и cookie-атрибуты, строгая авторизация, ограничение запросов и хорошая система логирования.

Заключение. В работе представлена прикладная методика оценки защищённости Web-приложений – от пассивной разведки и инвентаризации до анализа конфигураций и ручной проверки уязвимостей.

Ключевые выводы:

1. Угрозы могут возникать на любом уровне архитектуры – от клиента до инфраструктуры.
2. Чаще всего причины проблем – в ошибках настроек и слабом контроле доступа.
3. Автоматизация помогает найти базовые уязвимости, но ручная проверка всё равно необходима.
4. Для надёжной защиты важно внедрять многоуровневые меры и регулярно оценивать безопасность как часть разработки (DevSecOps).

Список литературы

1. Web Application Security: Exploitation and Countermeasures for Modern Web Applications. O'Reilly Media, 2020
2. Bug Bounty Bootcamp: The Guide to Finding and Reporting Web Vulnerabilities. No Starch Press, 2021.
3. Ловушка для багов. Полевое руководство по веб-хакингу. Издательство «Питер», 2020.
4. Synopsys. What is the OWASP Top 10? Detailed understanding and context of the OWASP Top 10. URL: https://www.synopsys.com/glossary/what-is-owasp-top-10.html

С.Ә. Ағабек[1], М.А. Ахметжанов[1], Ж.К. Тасжурекова[1], Ж.Б. Садирмекова[1]

[1]Таразский университет имени М.Х. Дулати, Тараз, Қазақстан

Рұқсатсыз ену кезінде веб-қосымшаның киберқауіпсіздік технологиясын зерттеу, талдау және тестілеу

Аңдатпа. *Бұл мақалада Web-қосымшалардың қауіпсіздігіне төнетін негізгі қатерлер және рұқсатсыз ену тәуекелдерін төмендетуге арналған практикалық бағалау тәсілі қарастырылады. Зерттеу әдістемесі пассивті барлау (OSINT), қолжетімді сервистерді түгендеу, жасырын ресурстарды анықтау (enumeration), бастапқы конфигурациялық аудит және маңызды бақылауларды HTTP(S)-трафик арқылы қолмен верификациялау кезеңдерін қамтиды. Нәтижелер Web-жүйелердің тұрақтылығы бір ғана қорғаныс механизмімен емес, қауіпсіз конфигурация, қолжетімділікті басқару, қауіпсіз әзірлеу және тұрақты бағалау сияқты көпдеңгейлі шаралардың жиынтығымен қамтамасыз етілетінін көрсетеді. Мақалада алынған артефактілер (сканерлеу нәтижелері, есептік материалдар) нәтижелерді қайталап тексеруге және түзету шараларының тиімділігін бағалауға мүмкіндік береді.*

S.Ə. Agabek[1], M.A. Akhmetzhanov[1], Zh.K. Taszhurekova[1], Zh.B. Sadirmekova[1]

[1]Taraz University named after M.Kh. Dulati, Taraz, Kazakhstan

Research, analysis and testing of cybersecurity technology in a Web applications in case of unauthorized intrusions

Abstract. *This article examines common security threats to web applications and presents a practical approach to assessing resilience against unauthorized access. The methodology follows a structured workflow including passive reconnaissance (OSINT), service inventory, resource enumeration, baseline configuration auditing, and manual verification of key observations through HTTP(S) traffic analysis. The results highlight that web application security cannot be achieved by a single control; instead, it requires a layered set of measures such as secure configuration, robust access control, secure development practices, and continuous assessment. The paper also emphasizes the value of evidence-based artifacts (scan outputs and reporting materials) to ensure repeatability and to evaluate the effectiveness of remediation actions.*

Поступило в редакцию 27.04.2026
Принято в печать 1.07.2026

Ссылка на статью: *