Electronics and information technologies, 2026, 33, 71–86
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC 004.94

APPLICATION OF PENETRATION TESTING FOR ASSESSING THE INFORMATION SECURITY LEVEL OF WEB-ORIENTED INFORMATION SYSTEMS

Sergiy Sveleba[1]*, Ivan Katerynchuk[1], Ivan Kunyo[1], Oleh Krupych[1], Yaroslav Shmyhelskyy[1], Marta Dufanets[1], Natalia Sveleba[2], Lucjan Pelc[3], Volodymyr Brygilevych[3]

[1]Ivan Franko National University of Lviv,
107, Gen. Tarnavsky St., 79017 Lviv, Ukraine
[2]Private Higher Education Establishment "European University",
16B, Academician Vernadsky Boulevard, 03115 Kyiv, Ukraine
[3]State Academy of Applied Sciences in Jarosław
ul. Czarnieckiego 16, 37-500 Jarosław Poland

Sveleba, S., Katerynchuk, I., Kunyo, I., Krupych, O., Shmyhelskyy, Ya., Dufanets, M., Sveleba, N., Pelc, L., Brygilevych, V. (2026). Application of Penetration Testing for Assessing the Information Security Level of Web-Oriented Information Systems. *Electronics and Information Technologies*, 33, 71–86. https://doi.org/10.30970/eli.33.6

ABSTRACT

Background. The increasing role of web-oriented information systems in business, education, and public administration is accompanied by a growing number and complexity of cyber threats. Traditional security mechanisms do not always enable the identification of actual system weaknesses, which necessitates the application of practice-oriented methods for assessing the level of information security. In this context, Penetration Testing is considered an effective instrument for simulating the actions of a potential attacker in order to detect and validate exploitable vulnerabilities.

Materials and Methods. The study employs a risk-oriented approach in accordance with international standards ISO/IEC 27001 and ISO/IEC 27005, as well as the recommendations of OWASP and NIST SP 800-115. Penetration Testing is implemented as a structured, multi-stage process that includes information gathering, attack surface analysis, threat modeling, execution of non-invasive validation scenarios, and risk assessment. The practical component was conducted in a controlled test environment using Nmap, Burp Suite, and Wireshark, supplemented by custom-developed Python modules for automated analysis of HTTP security headers, TLS certificates, and exposed services.

Results and Discussion. The study identified several configuration-related weaknesses at the application level, including the absence of essential HTTP security headers and deficiencies in TLS certificate management. The obtained results were formalized in a structured findings register with quantitative risk evaluation based on the *Likelihood* × *Impact* model. The analysis demonstrated that even in the absence of critical exploitable vulnerabilities, configuration errors significantly increase the overall risk level and may create preconditions for more sophisticated attacks.

Conclusion. The findings confirm the effectiveness of Penetration Testing as a comprehensive instrument for assessing the information security of web-oriented systems. The proposed approach facilitates the transition from technical testing results to substantiated managerial decisions aimed at risk reduction and enhancement of the overall protection level of information resources.

Keywords: information security, penetration testing, vulnerabilities, risk assessment, web-oriented information systems.

INTRODUCTION

The digitalization of business processes and the rapid expansion of web-oriented information systems across various domains have led to increased requirements for ensuring information security [1]. At the same time, there is a steady growth in both the number and sophistication of cyberattacks aimed at compromising the confidentiality, integrity, and availability of information resources [2]. Traditional protection mechanisms, including antivirus solutions and firewalls, do not always enable the timely identification of actual weaknesses in information systems, particularly those related to configuration errors or improper implementation of security mechanisms [3]. In this context, Penetration Testing is regarded as an effective method for assessing the level of security by simulating the actions of a potential attacker, thereby enabling the transition from formal analysis to practical validation of vulnerability exploitability [4].

The principal threats to information security are traditionally classified according to the CIA triad model, which encompasses confidentiality, integrity, and availability of information [5]. Threats to confidentiality involve unauthorized access to data; threats to integrity relate to data modification or tampering; and threats to availability concern disruption or denial of service affecting information systems [6].

Configuration errors, the use of outdated or weak cryptographic mechanisms, and the absence or insufficient formalization of security policies are particularly critical factors [7]. The human factor also plays a significant role in risk formation, including the use of weak passwords, administrative errors, and the impact of social engineering techniques [8].

Penetration Testing is defined as a method for evaluating the level of information security through the simulation of real attacks on an information system [9]. This approach involves modeling the actions of a potential attacker in order to assess the system's resilience to relevant threats. Unlike purely automated vulnerability scanning, Penetration Testing enables the identification of weaknesses that can be realistically exploited and allows for the evaluation of the effectiveness of existing technical and organizational security controls [10].

A comparison between Vulnerability Scanning and Penetration Testing demonstrates that the latter is based on a combination of automated and manual analytical methods [11]. This integrated approach reduces the number of false positives and provides a more objective assessment of the security posture of an information system [12].

Information security assessment should be considered a sequential process encompassing the chain "protection – attack – vulnerability – risk – decision" [13]. Within this logical framework, Penetration Testing provides a comprehensive and practice-oriented evaluation of the security level and forms a substantiated basis for managerial decision-making aimed at enhancing the protection of information systems [14].

MATERIALS AND METHODS

Penetration Testing is a comprehensive method for assessing the security level of information systems based on the simulation of real-world attacks [15]. Unlike automated vulnerability scanning, this approach enables validation of the practical exploitability of identified weaknesses and allows for an assessment of their actual impact on system security [16].

The study relies on international standards ISO/IEC 27001 and ISO/IEC 27005, which define requirements for information security management systems and risk assessment processes [17-18]. The OWASP Testing Guide and OWASP Top 10 methodologies were applied for web application security analysis [19-20]. The Penetration Testing Execution Standard (PTES) was used to describe the full penetration testing lifecycle

[21], while NIST SP 800-115 provided guidance on the technical aspects of security testing [22]. The application of these standards ensured the relevance, completeness, and alignment of the threat model with contemporary information security conditions.

In this work, Penetration Testing was implemented as a structured, multi-stage process in accordance with OWASP, PTES, and NIST methodologies. The initial phase involved information gathering, including the identification of assets, domains, services, and open ports. This was followed by attack surface analysis to determine potential entry points and system weaknesses. Based on the developed threat model, attack scenarios were formulated and validated through the simulation of real exploit conditions. The final stages included risk assessment and the preparation of an analytical report containing recommendations for mitigating identified issues.

The experimental component of the study was aimed at practically validating the effectiveness of Penetration Testing as a tool for assessing the security level of web-oriented information systems. The research was conducted in a controlled test environment that excluded any impact on production infrastructure and complied with ethical and legal requirements. The tested information system employed a client–server architecture and provided user access to web resources via HTTP/HTTPS protocols. The server component was deployed on a standard web server with TLS support. The system architecture encompassed network, application, and data storage layers, enabling security evaluation across multiple levels.

Professional Penetration Testing tools were used in the experiment: Nmap for network reconnaissance and open port analysis [23], Burp Suite for web application testing and HTTP traffic inspection [24], and Wireshark for in-depth packet analysis [25]. All tools were integrated within the Kali Linux environment, ensuring experimental integrity and reproducibility [26].

The experimental environment was implemented using Python 3.11. The following libraries and modules were used in the developed scripts:

– requests 2.31.0 for HTTP communication and header analysis;
– ssl (Python standard library) for TLS certificate inspection;
– xml.etree.ElementTree for parsing Nmap XML output;
– json 2.0 for structured result serialization;
– python-docx 0.8.11 for automated generation of DOCX security reports.

All experiments were conducted within the Kali Linux 2024 environment.

The practical Penetration Testing methodology was developed in accordance with the concept of sequential multi-layered analysis and includes the following stages (**Fig. 1**):

1. Automated initialization of the testing process.
   Implemented through the central script run_pentest.py, which coordinates the execution of testing modules and ensures reproducibility.
2. Network reconnaissance and attack surface analysis.
   Conducted using Nmap with automated parsing of scan results by the script parse_nmap_xml.py.
3. Application-layer and security header analysis.
   The script check_security_headers.py verifies the presence and correctness of HTTP security headers.
4. Cryptographic protection assessment.
   The script tls_cert_check.py analyzes TLS certificates and encryption parameters.
5. Results formalization and checklist validation.
   Security requirements stored in checklists.yaml are used for compliance verification.
6. Automated report generation.
   The script generator_docx.py produces a structured report in DOCX format.

Figure 1 illustrates the sequential logic of conducting Penetration Testing as a structured process for assessing information security. At the Planning / Scope stage, the

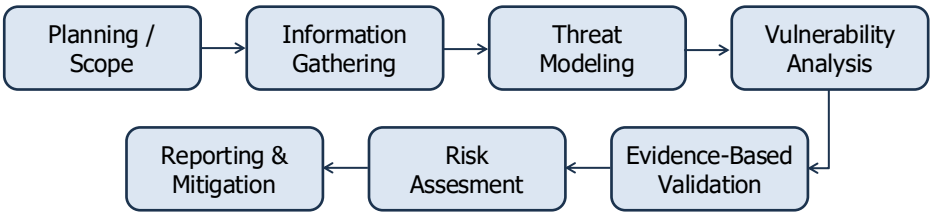


Fig. 1. Generalized model of the Penetration Testing methodology.

testing object, assessment boundaries, and legal considerations are defined. Information Gathering involves the collection of technical data regarding assets, services, and the attack surface. During the Threat Modeling phase, a threat model is constructed, identifying assets, potential actors, and attack scenarios. Vulnerability Analysis focuses on detecting configuration and technical weaknesses at various system levels. Evidence-Based Validation ensures confirmation of vulnerability exploitability without performing destructive actions. This is followed by Risk Assessment, where risks are evaluated using the *Likelihood* × *Impact* model. The final stage, Reporting & Mitigation, includes report generation and the development of recommendations for addressing identified issues.

The study applies the *Asset–Actor–Surface–Threat–Scenario* model, which provides a formalized and easily automated description of the testing process [27]. The identified assets include the web application, user accounts, confidential data, infrastructure, and network services, as well as the TLS communication channel. Actors are represented by an external attacker, a user with limited web interface access, and, where relevant, an internal user. The attack surface comprises the domain or URL, open ports and services, HTTP headers, and TLS certificates. The identified threats include traffic interception or manipulation, clickjacking, MIME sniffing, insecure configurations, outdated or misconfigured TLS settings, and unnecessarily exposed services. Scenarios are implemented as specific test cases that produce measurable results and are incorporated into the final report.

For analytical consistency, the implemented scenarios were aligned with the STRIDE model: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [28]. Within the non-invasive approach adopted in this study, the most relevant threat categories are Information Disclosure, Tampering, and Spoofing, as well as, to a limited extent, Denial of Service manifested through configuration weaknesses.

Python was selected for the practical implementation of the research due to its scientific and applied advantages. The choice of Python is justified by its rapid development capabilities, code readability, extensive libraries for network analysis, XML/JSON processing, and HTTP requests, broad automation potential for security-related tasks, integration with Penetration Testing tools [29], and suitability for DevSecOps implementations [30]. Thus, Python serves not only as an implementation tool but also as a universal platform for automating information security testing processes.

RESULTS AND DISCUSSION

Formalization of the Threat Model and the Testing Process

Within the scope of the study, the module threat_model.py was analyzed, as it implements a formalized description of assets, attack surface, threats, and testing scenarios. This module constitutes a core component of the test selection system and effectively functions as the central element of the Penetration Testing process. The SystemModel class is used to describe the testing object, including the URL or host, as well as known entry points. The Threat class formalizes threats according to the STRIDE framework, while the Scenario class represents an individual test case whose measurable outcome is used to generate a confirmed finding in the final report.

The project repository includes a set of typical testing modules, namely:

- *check_security_headers.py* – verification of the presence and correctness of HTTP security headers;
- *tls_cert_check.py* – retrieval and analysis of TLS certificates;
- *parse_nmap_xml.py* in conjunction with *nmap_output_sample.xml* – analysis of open ports and services based on XML-formatted scan results.

The module scenarios.py encapsulates individual functions into formalized testing scenarios. The core concept is not the unsystematic execution of all checks, but rather the selection of scenarios according to rules derived from the combination of the SystemModel, ThreatModel, and predefined constraints (execution time, testing boundaries, and non-invasive nature).

Within the structured repository, the central script run_pentest.py collects the results of HTTP header and TLS certificate checks. A key feature of the implementation is the automated selection of scenarios, the use of a unified output format (JSON), and the straightforward transformation of collected data into confirmed findings for subsequent reporting through the module *generator_docx.py*.

Thus, the study establishes a formalized threat model and implements a mechanism for scenario selection based on it. The Python-based implementation ensures formalization of assets, actors, and threats; automated selection of non-invasive scenarios; reproducible collection of results in a structured format; and a foundation for further risk analysis and reporting.

Process of Practical Penetration Testing

Practical Penetration Testing in the controlled test environment was implemented as a sequential validation pipeline comprising the following stages (**Fig. 2**):

1. Information Gathering – inventory of the attack surface and parameters of accessible components.
2. Vulnerability Scanning / Baseline Security Checks – identification of typical configuration weaknesses at the network, application, and cryptographic levels.
3. Exploitation – within this study, only safe validation of impact was performed, without destructive actions, with evidence recorded in the form of control artifacts.
4. Post-Exploitation Analysis – formalization of consequences, risk evaluation, preparation of recommendations, and generation of reporting documentation.

The fundamental objective of this approach is to ensure process reproducibility, whereby each stage produces its own artifacts in the form of logs, JSON data, or reports, enabling replication and verification of the research results.

The objective of the information gathering stage is to identify entry points (URL, host, ports, and services) and to obtain primary technical attributes that influence risk, including HTTP security headers, TLS certificate parameters, and information about open ports.

Data Collection Implementation.

The collection of HTTP security headers using the module *check_security_headers.py* enables the assessment of the status of fundamental web controls such as Content-Security-Policy, HSTS, X-Frame-Options, and others. The

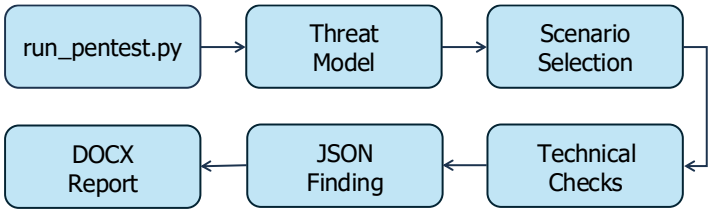


Fig. 2. Automated pipeline for the practical implementation of Penetration Testing.

absence or misconfiguration of these headers is considered an indicator of weak configuration and an insufficient level of system hardening.

The analysis of TLS settings is performed using the module *tls_cert_check.py*, which verifies certificate validity, expiration dates, and trust parameters. Incorrect TLS configurations increase the risk of traffic interception or manipulation and may reduce user trust.

Information on open ports is obtained by analyzing Nmap scan results in XML format using the module *parse_nmap_xml.py*. The presence of unnecessary services, even without active exploitation, is interpreted as an expansion of the attack surface and a potential source of risk.

Approach to Exploitation and Risk Assessment

The vulnerability scanning phase focuses on verifying typical classes of weaknesses, including configuration errors, absence of baseline security controls, and outdated TLS parameters, in accordance with OWASP recommendations and established hardening practices.

In the classical sense, the exploitation stage involves confirming the exploitability of vulnerability. In this study, a safe *evidence-based validation* approach is applied, whereby no destructive actions are performed, and no unauthorized access is attempted. Instead, weaknesses are validated based on configuration evidence and controlled testing artifacts. For example, the absence of the HSTS header or the presence of an exposed administrative port is treated as a confirmed indicator of elevated risk.

The post-exploitation stage is interpreted as an analytical phase that includes impact assessment, risk calculation, prioritization of issues, development of a mitigation plan, and preparation of reporting documentation.

Formalization of Results and Risk Model

Upon completion of Penetration Testing, a set of technical artifacts is generated; however, their practical value is realized only after analytical interpretation. For this purpose, the results obtained from various tools were normalized into a unified finding structure of the following form:

$$Finding = \{ID,\ Title,\ Category,\ Asset,\ Evidence,\ Impact,\ Likelihood,\ Risk,\ Recommendation,\ Priority\}.$$

For each finding, the reliability of confirmation, likelihood of exploitation, impact on confidentiality, integrity, and availability, as well as the exploitation context, are evaluated.

Risk assessment is conducted using the basic model:

$$Risk = Likelihood \times Impact,$$

where the values of $Likelihood$ and $Impact$ are determined using discrete scales according to the level of exposure and asset criticality [31]. This approach enables quantitative interpretation of results and supports the prioritization of information security improvement measures.

Extended Risk Assessment Model Incorporating Confidence

To enhance the accuracy of quantitative risk evaluation, an extended model was applied that incorporates the level of confidence in the testing results. The model is defined as:

$$Riskadj = (Likelihood \times Impact) \times Confidence,$$

where $Confidence \in \{0.5;\ 0.75;\ 1.0\}$ reflects the degree of validation of the identified weakness based on the obtained evidence.

Within the study, the $Impact$ value was determined according to asset criticality using a five-point scale (1-5), while the $Likelihood$ indicator was derived from the severity of the identified issue and its exposure level. This approach allows both the technical characteristics of the weakness and the context of its potential exploitation to be taken into account.

A comprehensive evaluation of Penetration Testing results was performed, including normalization of technical data, formation of a register of confirmed findings, quantitative risk assessment using the $Likelihood \times Impact$ model with the $Confidence$ coefficient, and development of a Mitigation Plan. Additionally, structural requirements for the final report were defined, ensuring its suitability for both technical specialists and managerial personnel.

The proposed approach facilitates the transition from purely technical security metrics to substantiated managerial decisions and aligns with contemporary risk-oriented information security management practices (**Fig. 3**). This figure illustrates the transformation of technical Penetration Testing results into a format suitable for managerial analysis.

At the initial stage, Technical Artifacts (Headers / TLS / Ports) are generated, representing primary data obtained from network, application, and cryptographic-level assessments. These data are subsequently structured into Normalized Findings, ensuring their standardized and unified representation.

The next step is Risk Scoring, within which quantitative risk evaluation is performed using the $Likelihood \times Impact$ model (with the optional inclusion of the Confidence coefficient where applicable).

The resulting data are aggregated into an Executive Summary, providing a concise overview of the overall risk level for management.

The final stage involves the development of a Mitigation Plan, which specifies concrete corrective actions, prioritization levels, and responsible stakeholders for reducing the identified risks.

Thus, the diagram demonstrates the logical transition from technical analysis to substantiated managerial decision-making in the field of information security.

The generated file *pentest_report.docx* contains a structured report that includes one confirmed finding, an example of which is presented in **Fig. 4**.

Structure and Content of the Reporting Documentation

The generated document is entitled “Penetration Testing Security Report,” which serves as the formal title and clearly defines the document as a report presenting the results of Penetration Testing.

Scope / Object

Section “1. Scope / Object” specifies the testing object as follows:

- Target URL: *https://example.com*
- Target Host: *example.com*

A clear definition of the object and testing boundaries is essential to ensure the methodological and legal validity of the report, as it enables unambiguous identification of the information system subjected to testing.

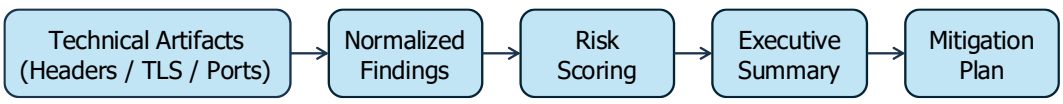
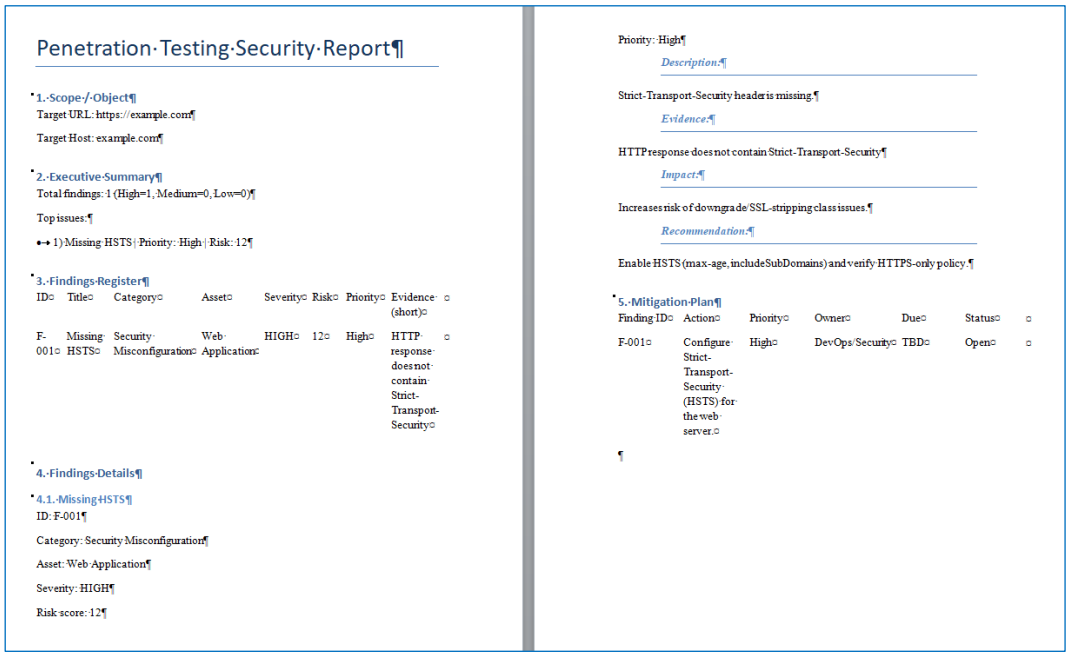


Fig. 3. Logical framework for transitioning from technical results to managerial decision-making.



Penetration·Testing·Security·Report¶

1.·Scope·/·Object¶
Target·URL:·https://example.com¶
Target·Host:·example.com¶

2.·Executive·Summary¶
Total·findings:·1·(High=1,·Medium=0,·Low=0)¶
Top·issues:¶
•→ 1)·Missing·HSTS·|·Priority:·High·|·Risk:·12¶

3.·Findings·Register¶

ID¤	Title¤	Category¤	Asset¤	Severity¤	Risk¤	Priority¤	Evidence·(short)¤	¤
F-001¤	Missing·HSTS¤	Security·Misconfiguration¤	Web·Application¤	HIGH¤	12¤	High¤	HTTP·response·does·not·contain·Strict-Transport-Security¤	¤

4.·Findings·Details¶
4.1.·Missing·HSTS¶
ID:·F-001¶
Category:·Security·Misconfiguration¶
Asset:·Web·Application¶
Severity:·HIGH¶
Risk·score:·12¶
Priority:·High¶
Description:¶
Strict-Transport-Security·header·is·missing.¶
Evidence:¶
HTTP·response·does·not·contain·Strict-Transport-Security¶
Impact:¶
Increases·risk·of·downgrade/SSL-stripping·class·issues.¶
Recommendation:¶
Enable·HSTS·(max-age,·includeSubDomains)·and·verify·HTTPS-only·policy.¶

5.·Mitigation·Plan¶

Finding·ID¤	Action¤	Priority¤	Owner¤	Due¤	Status¤	¤
F-001¤	Configure·Strict-Transport-Security·(HSTS)·for·the·web·server.¤	High¤	DevOps/Security¤	TBD¤	Open¤	¤

¶

Fig. 4. Structured report containing a single confirmed finding.

Executive Summary

Section “2. Executive Summary” provides a concise managerial overview of the testing results in the form of aggregated indicators. Specifically, the report states:

- total number of identified findings - 1;
- severity distribution: High - 1, Medium - 0, Low - 0;
- most critical issue - Missing HSTS with High priority and a risk score of 12.

This format allows for the rapid identification of high-priority issues requiring immediate remediation.

Findings Register

Section “3. Findings Register” presents a tabulated register of identified issues containing the following fields: identifier, title, category, asset, severity level, risk score, priority, and a brief description of evidence. The report documents one finding with the following characteristics:

- ID: F-001;
- Title: Missing HSTS;
- Category: Security Misconfiguration;
- Asset: Web Application;
- Severity: High;
- Risk: 12;
- Priority: High;
- Evidence: absence of the Strict-Transport-Security header in the HTTP server response.

The findings register functions as an inventory of identified issues and serves as a practical instrument for further analysis and decision-making.

Findings Details

Section “4. Findings Details” provides an extended description of the Missing HSTS finding, including a problem statement, supporting evidence, impact assessment, and mitigation recommendations. In particular, the absence of the Strict-Transport-Security

header increases the risk of downgrade and SSL-stripping attacks. The recommended mitigation measure includes enabling HSTS with appropriate parameters (max-age, includeSubDomains) and verifying the enforced HTTPS policy.

This section follows a clear analytical structure – problem – evidence – consequences – recommendations – and represents the most informative component of the report.

Mitigation Plan

Section “5. Mitigation Plan” outlines corrective measures for addressing the identified issue, specifying the finding identifier, recommended action, priority level, responsible parties, implementation timeline, and current status. For finding F-001, the status is defined as Open, with DevOps/Security specialists designated as responsible parties, and the implementation deadline to be determined.

The identified issue Missing HSTS is classified as an application-level configuration weakness. Although it does not result in immediate system compromise, it significantly increases the risk of HTTPS-related attacks under certain conditions. In the report, this weakness is categorized as Security Misconfiguration, assigned a High severity level and a risk score of 12, thereby justifying its high remediation priority.

Summary of Results and Automation of Reporting

As a result of the Penetration Testing conducted in the controlled environment, a formalized security report was generated, integrating validated technical findings with their analytical interpretation. The analytical processing of results enabled the transition from purely technical observations to quantitative risk evaluation using the Likelihood × Impact model and supported the prioritization of security measures.

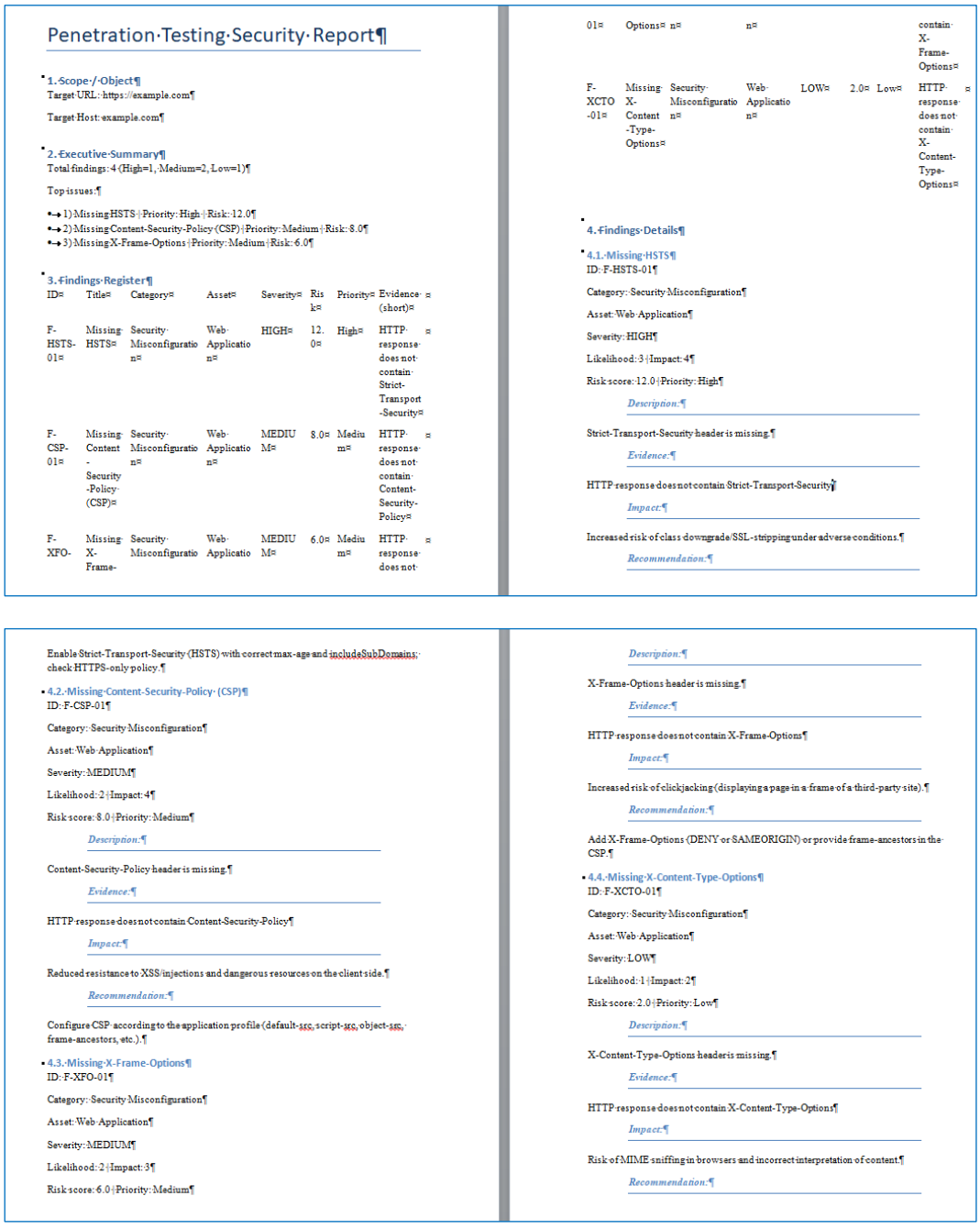
The study also implemented automated addition of multiple findings to the DOCX report. For this purpose, the generate_report() function was extended through the module auto_findings.py, which automatically generates findings based on the results of security header checks, TLS configuration analysis, and open port assessments, and transfers them to the report generation module.

As a result of the practical implementation of the Penetration Testing process, a DOCX report was automatically generated containing a structured set of technical and analytical results assessing the security level of the web application. The report was developed in accordance with a risk-oriented approach and incorporates all key components necessary for further information security management (**Fig. 5**).

In the Scope / Object section, the testing object – the web resource https://example.com and the corresponding host example.com - is clearly identified. Such identification ensures the formal correctness of the report, defines the responsibilities of the involved parties, and confirms the legitimacy of conducting Penetration Testing within the defined test environment.

The Findings Register section presents a tabulated register of all identified weaknesses, including the finding identifier, title, category, asset, severity level, risk score, and priority. All identified issues were classified as Security Misconfiguration and attributed to the application layer of the web application. This register functions as a centralized risk inventory tool and allows the report to serve not only as a technical document but also as a foundation for managerial analysis.

The Findings Details section provides an extended description of each identified finding using a unified analytical structure that includes a problem statement, supporting evidence, impact assessment, and remediation recommendations. In particular, the absence of HSTS is assessed as a high-risk issue due to the potential for SSL-stripping attacks; the absence of Content-Security-Policy reduces the application’s resilience to cross-site scripting and other client-side attacks; the lack of X-Frame-Options creates conditions for clickjacking attacks; and the absence of X-Content-Type-Options may lead to MIME-sniffing and incorrect content interpretation by browsers. For each weakness,



Penetration Testing Security Report

1. Scope / Object

Target URL: https://example.com

Target Host: example.com

2. Executive Summary

Total findings: 4 (High=1, Medium=2, Low=1)

Top issues:

- 1) Missing HSTS | Priority: High | Risk: 12.0
- 2) Missing Content-Security-Policy (CSP) | Priority: Medium | Risk: 8.0
- 3) Missing X-Frame-Options | Priority: Medium | Risk: 6.0

3. Findings Register

ID	Title	Category	Asset	Severity	Risk	Priority	Evidence (short)
F-HSTS-01	Missing HSTS	Security Misconfiguration	Web Application	HIGH	12.0	High	HTTP response does not contain Strict-Transport-Security
F-CSP-01	Missing Content-Security-Policy (CSP)	Security Misconfiguration	Web Application	MEDIUM	8.0	Medium	HTTP response does not contain Content-Security-Policy
F-XFO-01	Missing X-Frame-Options	Security Misconfiguration	Web Application	MEDIUM	6.0	Medium	HTTP response does not contain X-Frame-Options
F-XCTO-01	Missing X-Content-Type-Options	Security Misconfiguration	Web Application	LOW	2.0	Low	HTTP response does not contain X-Content-Type-Options

4. Findings Details

4.1. Missing HSTS

ID: F-HSTS-01

Category: Security Misconfiguration

Asset: Web Application

Severity: HIGH

Likelihood: 3 | Impact: 4

Risk score: 12.0 | Priority: High

Description:

Strict-Transport-Security header is missing.

Evidence:

HTTP response does not contain Strict-Transport-Security

Impact:

Increased risk of class downgrade/SSL-stripping under adverse conditions.

Recommendation:

Enable Strict-Transport-Security (HSTS) with correct max-age and includeSubDomains; check HTTPS-only policy.

4.2. Missing Content-Security-Policy (CSP)

ID: F-CSP-01

Category: Security Misconfiguration

Asset: Web Application

Severity: MEDIUM

Likelihood: 2 | Impact: 4

Risk score: 8.0 | Priority: Medium

Description:

Content-Security-Policy header is missing.

Evidence:

HTTP response does not contain Content-Security-Policy

Impact:

Reduced resistance to XSS/injections and dangerous resources on the client side.

Recommendation:

Configure CSP according to the application profile (default-src, script-src, object-src, frame-ancestors, etc.).

4.3. Missing X-Frame-Options

ID: F-XFO-01

Category: Security Misconfiguration

Asset: Web Application

Severity: MEDIUM

Likelihood: 2 | Impact: 3

Risk score: 6.0 | Priority: Medium

Description:

X-Frame-Options header is missing.

Evidence:

HTTP response does not contain X-Frame-Options

Impact:

Increased risk of clickjacking (displaying a page in a frame of a third-party site).

Recommendation:

Add X-Frame-Options (DENY or SAMEORIGIN) or provide frame-ancestors in the CSP.

4.4. Missing X-Content-Type-Options

ID: F-XCTO-01

Category: Security Misconfiguration

Asset: Web Application

Severity: LOW

Likelihood: 1 | Impact: 2

Risk score: 2.0 | Priority: Low

Description:

X-Content-Type-Options header is missing.

Evidence:

HTTP response does not contain X-Content-Type-Options

Impact:

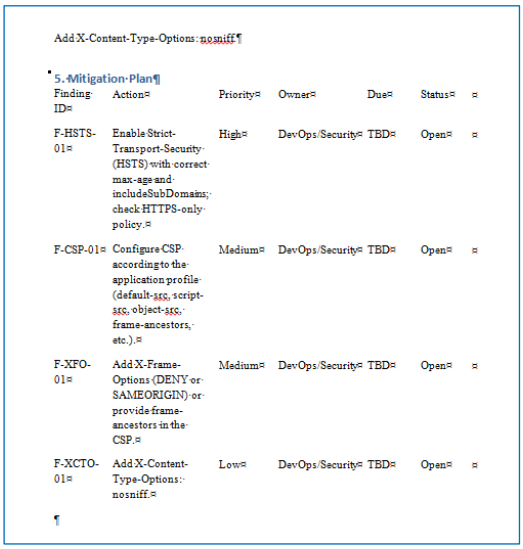
Risk of MIME sniffing in browsers and incorrect interpretation of content.

Recommendation:

Fig. 5 (beginning). Complete structured security report.

specific technical recommendations are provided in accordance with contemporary web application security best practices.

The final element of the report is the Mitigation Plan, which specifies, for each finding, the recommended actions, priority level, responsible parties, and current implementation status. The presence of such a plan demonstrates the transition from technical identification of issues to the practical implementation of measures aimed at improving the overall level of information security.



Add X-Content-Type-Options: nosniff ¶

5. Mitigation Plan ¶

Finding ID	Action	Priority	Owner	Due	Status	
F-HSTS-01	Enable Strict-Transport-Security (HSTS) with correct max-age and includeSubDomains; check HTTPS-only policy.	High	DevOps/Security	TBD	Open	
F-CSP-01	Configure CSP according to the application profile (default-src, script-src, object-src, frame-ancestors, etc.).	Medium	DevOps/Security	TBD	Open	
F-XFO-01	Add X-Frame-Options (DENY or SAMEORIGIN) or provide frame-ancestors in the CSP.	Medium	DevOps/Security	TBD	Open	
F-XCTO-01	Add X-Content-Type-Options: nosniff	Low	DevOps/Security	TBD	Open	

¶

Fig. 5 (ending)**.** Complete structured security report.

The generated file pentest_report.docx constitutes a comprehensive analytical document that integrates technical Penetration Testing results with quantitative risk assessment and managerial recommendations. This confirms the effectiveness of the applied approach and highlights the practical value of Penetration Testing as a tool for assessing and enhancing the information security of modern information systems.

Figure 6 presents a concise Penetration Testing report in Markdown format, reflecting the structure of the collected data, their interpretation, and the logical conclusions derived from the analysis. The file report.md consists of three main components: testing metadata (target, host, timestamp), technical results (security header and TLS certificate verification), and generalized conclusions.

```
# Pentest report
- ** Destination URL:** `N/A`
- ** Host / IP:** `N/A`
- ** Scan time:** `2026-01-23T23:58:02`
## Other data
### headers
- **status**:
    200
- **headers**:
    - **Content-Security-Policy**:
        None
    - **Strict-Transport-Security**:
        None
    - **X-Frame-Options**:
        None
    - **X-Content-Type-Options**:
        None
    - **Referrer-Policy**:
        None
### cert
- **subject**:
    - item #1:
        - item #1:
            - countryName
```

Fig. 6 (beginning). Penetration Testing report in Markdown format.

```
            - US
    - item #2:
        - item #1:
            - stateOrProvinceName
            - California
    - item #3:
        - item #1:
            - localityName
            - Los Angeles
    - item #4:
        - item #1:
            - organizationName
            - Internet Corporation for Assigned Names and Numbers
    - item #5:
        - item #1:
            - commonName
            - *.example.com
- **issuer**:
    - item #1:
        - item #1:
            - countryName
            - US
    - item #2:
        - item #1:
            - organizationName
            - DigiCert Inc
    - item #3:
        - item #1:
            - commonName
            - DigiCert Global G3 TLS ECC SHA384 2020 CA1
- **version**:
    3
- **serialNumber**:
    0AD893BAFA68B0B7FB7A404F06ECAF9A
- **notBefore**:
    Jan 15 00:00:00 2025 GMT
- **notAfter**:
    Jan 15 23:59:59 2026 GMT
- **subjectAltName**:
    - item #1:
        - DNS
        - *.example.com
    - item #2:
        - DNS
        - example.com
- **OCSP**:
    - http://ocsp.digicert.com
- **caIssuers**:
    - http://cacerts.digicert.com/DigiCertGlobalG3TLSECCSHA3842020CA1-2.crt
- **crlDistributionPoints**:
    - http://crl3.digicert.com/DigiCertGlobalG3TLSECCSHA3842020CA1-2.crl
    - http://crl4.digicert.com/DigiCertGlobalG3TLSECCSHA3842020CA1-2.crl
## Conclusions
Based on the obtained results, it is recommended to further analyze open
ports, TLS/SSL configurations, and potential vulnerabilities (if identified),
as well as to strengthen the web server’s security policies.
```

Fig. 6 (ending). Penetration Testing report in Markdown format.

The analysis of metadata indicates that the testing was conducted at a specified point in time using non-invasive methods for assessing web server configuration and the cryptographic parameters of the secured connection. The absence of specific URL and host values in the report is interpreted as a result of test environment anonymization or the omission of parameters during script execution, which does not affect the overall validity of the analysis.

The results of the HTTP security header assessment reveal the absence of fundamental application-layer protection mechanisms, including Strict-Transport-Security, Content-Security-Policy, X-Frame-Options, X-Content-Type-Options, and Referrer-Policy. The absence of these headers is classified as a configuration weakness within the category of Security Misconfiguration. Although it does not result in immediate system compromise, it significantly increases the overall level of information security risk.

Particular attention should be given to the TLS certificate analysis results. It was determined that the certificate, issued by the trusted certification authority DigiCert, had expired at the time of testing. An expired TLS certificate represents a critical operational risk, as it may lead to loss of client trust, potential browser access blocking, and reduced service availability. This issue directly affects the Availability component of the CIA model and indicates insufficient lifecycle management of cryptographic certificates.

CONCLUSION

The overall evaluation of the results leads to the conclusion that the identified issues are predominantly configuration-related and operational in nature and are not accompanied by active exploitation of vulnerabilities. At the same time, such weaknesses often constitute preconditions for more sophisticated attacks under real-world operating conditions. The application of a quantitative risk assessment model based on the combination of likelihood and impact made it possible to classify the expired TLS certificate and the absence of HSTS as priority issues requiring immediate remediation, whereas the remaining missing security headers may be addressed through planned security enhancement measures.

Thus, the results of the Penetration Testing confirm the effectiveness of this approach as a comprehensive instrument for information security assessment. The obtained data not only reflect the technical condition of the system but also provide a substantiated basis for managerial decision-making aimed at risk reduction and improvement of the overall protection level of information resources.

The source code is available in the GitHub repository at the following link: *incom2025/Secur_infor_syst*.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors received no financial support for the research, writing, and/or publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that they have no competing interests.

AUTHOR CONTRIBUTIONS

Conceptualization, [S. S.]; formal analysis, [S. S., I. Ka., Y. S.,]; investigation, [I. Ku.]; resources, [I. Ku.]; data curation, [M. D., N. S., L. P., V. B.], writing – original draft preparation, [S. S., I. Ka, I. Ku.]; writing – review and editing, [O. K., Y. S., M. D., N. S., L. P.], visualization, [O. K., Y. S., V. B.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. 3rd ed. Wiley, 2020. URL: https://www.wiley.com/en-us/Security+Engineering%3A+A+Guide+to+Building+Dependable+Distributed+Systems%2C+3rd+Edition-p-9781119642787.

[2] Behl A., Behl K. Cyberwar: The Next Threat to National Security and What to Do About It. Oxford University Press, 2017.

[3] Bishop M. *Computer Security: Art and Science*. 2nd ed. Addison-Wesley, 2019. URL: https://ptgmedia.pearsoncmg.com/images/9780321712332/samplepages/9780321712332_Sample.pdf

[4] OWASP Foundation. OWASP Web Security Testing Guide v4. 2023. URL: https://owasp.org/www-project-web-security-testing-guide/.

[5] European Union Agency for Cybersecurity (ENISA). *Good Practices for Security of IoT – Secure Development and Testing*. 2020. URL: https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot

[6] ISO/IEC 27001:2022. Information Security Management Systems - Requirements. ISO, 2022. URL: https://www.iso.org/standard/82875.html.

[7] ISO/IEC 27002:2022. *Information Security Controls*. ISO, 2022. URL: https://www.iso.org/standard/75652.html.

[8] Kizza J. M. *Guide to Computer Network Security*. 5th ed. Springer, 2020. https://doi.org/10.1007/978-3-030-38141-7.

[9] MITRE. MITRE ATT&CK®: Adversarial Tactics, Techniques, and Common Knowledge. 2023. URL: https://attack.mitre.org/

[10] CVE Program. Common Vulnerabilities and Exposures (CVE). MITRE Corporation, 2024. URL: https://www.cve.org/.

[11] NIST SP 800-53 Rev. 5. *Security and Privacy Controls for Information Systems and Organizations*. National Institute of Standards and Technology, Gaithersburg, 2020. https://doi.org/10.6028/NIST.SP.800-53r5.

[12] NIST SP 800-115. *Technical Guide to Information Security Testing and Assessment*. National Institute of Standards and Technology, Gaithersburg, 2008. URL: https://doi.org/10.6028/NIST.SP.800-115.

[13] OWASP Foundation. *OWASP Testing Guide v4*. 2023. URL: https://owasp.org/www-project-web-security-testing-guide/

[14] OWASP Foundation. OWASP Top 10 - Web Application Security Risks. 2021. URL: https://owasp.org/www-project-top-ten/.

[15] Scarfone, K., Mell, P. *Guide to Intrusion Detection and Prevention Systems (IDPS)*, (NIST SP 800-94), National Institute of Standards and Technology, Gaithersburg, 2007. https://doi.org/10.6028/NIST.SP.800-94.

[16] Nelson A., Rekhi S., Souppaya M., Scarfone K. Incident Response Recommendations and Considerations for Cybersecurity Risk Management. NIST SP 800-61 Rev.3. National Institute of Standards and Technology, 2025. https://doi.org/10.6028/NIST.SP.800-61r3.

[17] Stallings W. Network Security Essentials: Applications and Standards. 6th ed. Pearson, 2017. URL: https://www.pearson.com/en-us/subject-catalog/p/network-security-essentials-applications-and-standards/P200000003333

[18] Ross, R. (2012), *Guide for Conducting Risk Assessments*, (NIST SP 800-30 Rev. 1), National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.800-30r1.

[19] ENISA. Threat Landscape Report 2023. European Union Agency for Cybersecurity. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023.

[20] Whittaker J. A., Arbon J., Carollo J. *How Google Tests Software*. Addison-Wesley, 2012. URL: https://www.informit.com/store/how-google-tests-software-9780321803023

[21] OSSTMM Institute. Open Source Security Testing Methodology Manual (OSSTMM) v3. 2019. URL: https://www.isecom.org/OSSTMM.3.pdf.

[22] PTES. *Penetration Testing Execution Standard (PTES)*. 2020. URL: http://www.pentest-standard.org/.

[23] Shostack A. Threat Modeling: Designing for Security. Wiley, 2014. URL: https://www.wiley.com/en-us/Threat+Modeling%3A+Designing+for+Security-p-9781118809990.

[24] Singer, P. W., & Friedman, A. (2013). *Cybersecurity and Cyberwar: What Everyone Needs to Know*. Oxford University Press. URL: https://global.oup.com/academic/product/cybersecurity-and-cyberwar-9780199918096?q=Cybersecurity%20and%20Cyberwar:%20What%20Everyone%20Needs%20to%20Know&cc=ua&lang=en.

[25] Humble J., Farley D. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley, 2010. URL: https://www.informit.com/store/continuous-delivery-reliable-software-releases-through-9780321601919.

[26] Kim G., Behr K., Spafford G. *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*. IT Revolution Press, 2018. https://itrevolution.com/product/the-phoenix-project/.

[27] Nelson, A., Rekhi, S., Souppaya, M. and Scarfone, K. *Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile*. NIST SP 800-61 Rev. 3. National Institute of Standards and Technology, Gaithersburg, 2025. https://doi.org/10.6028/NIST.SP.800-61r3.

[28] CVE Program. Common Vulnerabilities and Exposures (CVE). MITRE Corporation, 2024. URL: https://www.cve.org/.

[29] Dempsey, K., Johnson, L., Scholl, M., Stine, K., Clay, A., Orebaugh, A., Chawla, N. and Johnston, R. (2011), *Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations*, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.800-137 .

[30] Zalewski M. *The Tangled Web: A Guide to Securing Modern Web Applications*. No Starch Press, 2012. https://nostarch.com/tangledweb.

[31] MITRE. *MITRE ATT&CK®: Adversarial Tactics, Techniques, and Common Knowledge*. 2023. URL: https://attack.mitre.org/.

ЗАСТОСУВАННЯ *PENETRATION TESTING* ДЛЯ ОЦІНЮВАННЯ РІВНЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ВЕБ-ОРІЄНТОВАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

***Сергій Свелеба*[1]*, *Іван Катеринчук*[1], *Іван Куньо*[1], *Олег Крупич*[1], *Ярослав Шмигельський*[1], *Марта Дуфанець*[1], *Наталя Свелеба*[2], *Люціан Пельц*[3], *Володимир Бригілевич*[3]**

[1]Львівський національний університет імені Івана Франка
вул. Ген. Тарнавського, 107, 79017 Львів, Україна
[2]Приватний вищий навчальний заклад «Європейський університет»,
бульвар Академіка Вернадського, 16 В, 03115 Київ, Україна
[3]Державна вища школа технологій та економіки в Ярославі
вул. Чарнецького, 16, 37-500 Ярослав, Польща

АНОТАЦІЯ

Вступ. Зростання ролі веб-орієнтованих інформаційних систем у бізнесі, освіті та державному управлінні супроводжується підвищенням кількості та складності кіберзагроз. Традиційні засоби захисту не завжди дозволяють виявити реальні слабкі місця систем, що обумовлює необхідність застосування практично орієнтованих методів оцінювання рівня інформаційної безпеки. У цьому контексті Penetration Testing розглядається як ефективний інструмент імітації дій потенційного зловмисника з метою виявлення та підтвердження експлуатованих вразливостей.

Матеріали та методи. У роботі використано ризик-орієнтований підхід відповідно до міжнародних стандартів ISO/IEC 27001, ISO/IEC 27005, рекомендацій OWASP та NIST SP 800-115. *Penetration Testing* реалізовано як поетапний процес, що включає збір інформації, аналіз поверхні атаки, формування моделі загроз, виконання неінвазивних сценаріїв перевірки та оцінку ризиків. Практичну частину виконано у тестовому середовищі з використанням інструментів Nmap, Burp Suite та Wireshark, а також власних Python-модулів для автоматизації аналізу HTTP-заголовків безпеки, TLS-сертифікатів і відкритих сервісів.

Результати. У ході дослідження виявлено низку конфігураційних слабкостей прикладного рівня, зокрема відсутність базових захисних HTTP-заголовків та недоліки в управлінні TLS-сертифікатами. Отримані результати формалізовано у вигляді реєстру знахідок із кількісною оцінкою ризиків за моделлю $Likelihood \times Impact$. Аналіз показав, що навіть за відсутності критичних експлуатаційних вразливостей конфігураційні помилки істотно підвищують загальний рівень ризику та можуть створювати передумови для складніших атак.

Висновки. Результати дослідження підтверджують ефективність *Penetration Testing* як інструменту комплексної оцінки інформаційної безпеки веб-орієнтованих систем. Запропонований підхід забезпечує перехід від технічних результатів тестування до обґрунтованих управлінських рішень, спрямованих на зниження ризиків та підвищення рівня захищеності інформаційних ресурсів.

***Ключові слова*:** інформаційна безпека, penetration testing, вразливості, оцінка ризиків, веб-інформаційні системи.

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
20 February, 2026	07 March, 2026	09 March, 2026	30 March, 2026