В.Т.ЛЭ[1], С.М.БОРОВИКОВ[2], А.В.БУДНИК[3]

ПОДХОД К ОЦЕНКЕ ОЖИДАЕМОЙ НАДЕЖНОСТИ ПРИКЛАДНЫХ КОМПЬЮТЕРНЫХ ПРОГРАММ ДЛЯ СИСТЕМ ТЕЛЕКОММУНИКАЦИЙ

[1]Институт технологий, г. Ханой, Вьетнам, научный сотрудник, магистр техники и технологии
[2]Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники», г. Минск, Республика Беларусь, доцент, кандидат технических наук, доцент
[3]Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники», г. Минск, Республика Беларусь, декан факультета, кандидат технических наук, доцент

Современные системы телекоммуникаций фактически представляют собой компьютерные информационные системы, которые постоянно совершенствуются и наделяются все новыми и новыми функциями и обеспечивают пользователей (клиентов) все более широкими возможностями. При этом успех в использовании систем телекоммуникаций во многом определяется надежностью прикладного программного обеспечения, используемого в составе систем. По литературным данным [1] на долю программного обеспечения может приходиться до 40 процентов отказов компьютерных информационных систем. Поэтому оценке и обеспечению эксплуатационной надежности прикладных компьютерных программ для систем телекоммуникаций должно уделяться особое внимание.

Оценка ожидаемой эксплуатационной надежности прикладной компьютерной программы до начала ее проектирования, включающего разработку алгоритма будущей программы и написание кода на языке программирования, является полезной и ответственной процедурой, так как позволяет сориентироваться в принципиальной возможности за отводимое бюджетное время создать прикладную компьютерную программу, отвечающую требованию надежности. А это, в свою очередь, определит целесообразность проектирования новой или модернизации действующей телекоммуникационной системы. Кроме того, сведения об ожидаемой эксплуатационной надежности будущей прикладной компьютерной программы позволят определиться в оправданности вкладывания заказчиками систем телекоммуникаций дальнейших финансовых средств на разработку прикладной компьютерной программы.

Оценка ожидаемой надежности прикладных компьютерных программ авторами рассматривалась в [2, 3]. В частности, в работе [2] были определены основные принципы и подходы к оценке ожидаемой надежности прикладных компьютерных программ, а в [3] на конкретном примере показана оценка ожидаемой надежности прикладной компьютерной программы для определенной области ее применения.

Одним из важнейших этапов оценки ожидаемой надежности будущей прикладной компьютерной программы является определение прогнозного значения начальной плотности отказов, иначе – плотности проявления ошибок в компьютерной программе (обозначим эту плотность через F_0). В работах [2, 3] в качестве основы определения F_0 использована «Римская модель» RL-92-52, предложенная в [4]. Эта модель в качестве составляющих компонентов определения F_0 включает метрику D, учитывающую особенность проектной организации, опыт и квалификацию программистов разрабатывающей прикладную компьютерную программу, а также метрику S, принимающую во внимание основные характеристики самой будущей компьютерной программы. Причем в работе [2] для метрики S рекомендуется учитывать следующие ее характеристики: сложность, новизну, средства разработки, степень использования стандартных модулей.

Для повышения достоверности определения прогнозной оценки начальной плотности ошибок F_0 и, как следствие, проектной эксплуатационной надежности будущей прикладной компьютерной программы авторами предлагается дополнить математическое выражение определения F_0 метрикой (обозначим ее через U), учитывающей, не принятые во внимание, но гипотетически влияющие на F_0, факторы, связанные с организацией, разрабатывающей программу, программистами и характеристиками будущей прикладной компьютерной программы. Метрику U будем называть метрикой полноты учета факторов, влияющих на начальную плотность отказов будущей прикладной компьютерной программы. Модель определения F_0 с учетом метрики U предложена в виде [3]

$$F_0 = F_Б \cdot D \cdot S \cdot (2 - U), \text{ ошибок / KLOC}, \quad (1)$$

где $F_Б$ – метрика области применения прикладной компьютерной программы, представляющая собой среднюю по сфере производственной деятельности (отрасли) плотность отказов, ее называют базовой плотностью отказов [4], размерность $F_Б$ – «ошибок / KLOC»; запись KLOC означает тысячу исполняемых строк программного кода.

Отметим, что значение базовой плотности отказов при определении прогнозного значения начальной плотности отказов F_0 для прикладных компьютерных программ, разрабатываемых для телекоммуникаций, составляет $F_Б$ = 7,8 ошибок / KLOC [2, 4].

Метрика U модели (1) является безразмерной и может быть определена методом экспертной оценки с учетом условия

$$0 < U \leq 1. \quad (2)$$

Причем значение $U = 1$ должно соответствовать случаю учета практически всех факторов, определяющих значение начальной плотности отказов F_0, т. е. фактически качество написания программного кода будущей прикладной компьютерной программы, а $U < 1$ – случаю, когда какая-то часть факторов, гипотетически влияющих на начальную надежность будущей компьютерной программы, не принята во внимание в метриках D и S.

Определив по выражению (1) начальную плотность отказов F_0, и зная прогнозное число строк программного кода (в KLOC), по модели, приводимой в [2], можно оценить начальное значение интенсивности отказов компьютерной программы (λ_0, размерность 1/ч), а далее с учетом планируемого бюджетного времени, отводимого на процедуру тестирования, найти проектное значение эксплуатационной интенсивности отказов ($\lambda_{экс}$, размерность 1/ч) компьютерной программы и другие характеристики ее надежности [3], в том числе, вероятность безотказной работы компьютерной программы в течение интересующего календарного времени τ, обозначена как $P(\tau)$:

$$P(\tau) = (P_1)^{\eta \cdot \tau} = \left[\exp\left(-\lambda_{экс} \cdot t_1\right)\right]^{\eta \cdot \tau}, \quad (3)$$

где P_1 – вероятность безотказной обработки прикладной компьютерной программой одного усредненного набора исходных данных при функционировании системы телекоммуникаций; η – среднее прогнозное число обращений в час к прикладной компьютерной программе в течение календарного периода τ; t_1 – среднее процессорное время (в часах) обработки прикладной компьютерной программой одного усредненного набора данных.

По вопросам оценки ожидаемой проектной надежности планируемых к разработке прикладных компьютерных программ можно обращаться в учебно-исследовательскую лабораторию «Надежность электронных систем» кафедры проектирования информационно-компьютерных систем Белорусского государственного университета информатики и радиоэлектроники (ауд. 37-1 корп.).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Borovikov, S. M. Assessment of expected reliability of applied software for computer-based information systems / S. M. Borovikov [et al.] // Informatics. – 2021. – Vol. 18, no. 1. – P. 84-95 (in Russian). https://doi.org/10.37661/1816-0301-2021-18-1-84-95.

2. Боровиков, С. М. Надежность разрабатываемых прикладных компьютерных программ для информационных систем = Reliability of developed application computer programs of information systems / С. М. Боровиков, А. В. Будник, Ван Там Лэ // Проблемы инфокоммуникаций : научный журнал. – 2025. – № 1. – С. 25-34. doi: 10.61726/6446.2025.12.87.001

3. Практика оценки ожидаемой надежности прикладных компьютерных программ для информационных систем = Practice of assessing the expected reliability of applied computer programs for information systems / Ван Там Лэ, С. М. Боровиков, А. В. Будник, С. К. Дик // Big Data и анализ высокого уровня = Big Data and Advanced Analytics : сб. науч. статей XII Международ. научно-практ. конф., Республика Беларусь, Минск, 23 апреля 2026 года, в 2 ч. Ч. 1. – Минск : Белорусский государственный университет информатики и радиоэлектроники, 2026. – С. 229-237.

4. McCall, J. A. Software reliability, measurement, and testing guidebook for software reliability measurement and testing / J. A. McCall [et al.] // Final Technical Report, Vol II (of two) ; RL-TR-92-52. – Science Applications International Corp., Research Triangle Institute : Rome Laboratory, NY 13441-5700, 1992. – 256 p.

B.B.BABAYEV

CURRENT PROBLEMS OF WEB RESOURCE SECURITY: A CONTEMPORARY ANALYSIS

Institute of Telecommunications and Informatics of Turkmenistan, Ashgabat, Turkmenistan

This paper examines the evolving landscape of web resource security in the contemporary digital era. Despite advancements in cryptographic protocols and security frameworks, web applications remain primary targets for malicious actors. This study analyzes prominent architectural and runtime vulnerabilities, including injection flaws, authentication breakdowns, supply chain risks in open-source dependencies, and the rising threat of automated botnets. By evaluating current countermeasures alongside their limitations, this paper outlines critical trajectories for future research and defense mechanisms. The global migration toward cloud computing, microservices, and decentralized web architectures has transformed web resources into the operational backbone of modern commerce, governance, and social interaction. However, this rapid digital expansion has outpaced conventional security paradigms. Web resources–encompassing web applications, Application Programming Interfaces (APIs), and cloud-hosted data repositories–are fundamentally exposed to the public internet by design.

While legacy perimeter defenses such as standard network firewalls were sufficient in the past, modern threats target the application logic and structural components of web platforms. The primary objective of this paper is to investigate the most critical contemporary vulnerabilities plaguing web resources, evaluate why traditional countermeasures are failing, and discuss the academic and practical paradigms required to secure the next generation of web systems.

Core Taxonomy of Contemporary Web Vulnerabilities

Advanced Injection Flaws and Input Validation Failures

Despite decades of awareness, injection vulnerabilities—most notably Structured Query Language Injection (SQLi) and Cross-Site Scripting (XSS)—persist as dominant threats. Modern web frameworks have integrated parameterized queries and auto-escaping contexts to mitigate these risks. However, the complexity of modern web ecosystems introduces new vectors:

– NoSQL and Graph Database Injections: The proliferation of non-relational databases has led to unique injection styles where attackers manipulate execution logic through object serialization or query language structures.

– Server-Side Request Forgery (SSRF): Driven by cloud-native designs and microservice communication patterns, SSRF allows attackers to force a server-side application to make HTTP requests to arbitrary domains, often exposing metadata services or internal networks.

– Identification, Authentication, and Session Failures

– As single-page applications (SPAs) and decentralized user tracking become standard, session management and identity verification have grown incredibly complex.

– Broken Object-Level Authorization (BOLA): Frequently appearing at the top of API security risks, BOLA occurs when an application exposes endpoints that handle object identifiers, creating an expanded attack surface where users can access other individuals' private records by manipulating variables within request strings.

– Credential Stuffing and MFA Bypass: Automated tools leverage massive breaches of historical credentials to execute high-velocity credential-stuffing attacks. Furthermore, contemporary adversarial tactics have evolved to bypass Multi-Factor Authentication (MFA) using reverse-proxy phishing kits (e.g., Evilginx), making session hijacking highly prevalent.

Emerging Paradigm Threats

Modern web development heavily relies on third-party libraries, content management systems (CMS), and framework packages managed via tools like npm, PyPI, and Maven.

– Transitive Dependencies: A single web application may rely on thousands of nested, underlying open-source packages. Attackers exploit this blind spot through dependency confusion or by directly compromising the developer accounts of maintainers to inject malicious payloads into updates.

– Lagging Patch Management: Organizations frequently fail to inventory or audit their internal dependency graphs, allowing widely known vulnerabilities to remain exploitable in active web environments for years.

– Automated Client-Side Exploitation and Botnets

– The traditional server-focused security model often overlooks the client environment.

– Magecart and Formjacking: Attackers compromise third-party scripts (such as analytics or chat widgets) loaded directly into user browsers to harvest confidential inputs, such as credit card information, directly from the DOM before it ever reaches the secure server infrastructure.

– DDoS and Layer-7 Exploitation: Distributed Denial of Service (DDoS) threats have evolved from simple network-layer flooding to highly sophisticated, distributed Layer-7 (Application Layer) request floods. These mimic real human behavior, making them exceptionally difficult to detect and neutralize without disrupting legitimate traffic.

Threat Vector	Primary Impact	Contemporary Catalyst	Primary Defense Paradigm
BOLA / IDOR	Unauthorized Data Access	Proliferation of Microservices / APIs	Strict Attribute-Based Access Control (ABAC)
Supply Chain Risks	Remote Code Execution (RCE)	Over-reliance on Open-Source Ecosystems	Automated Software Bill of Materials (SBOM) auditing
SSRF	Internal Network Exposure	Cloud Metadata Services Adoption	Strict Input Allowlisting & Network Isolation
Layer-7 DDoS	Resource Exhaustion / Outages	Low-cost Global Botnets / Cloud Proxies	AI-driven Behavioral Rate-Limiting

Conclusion and Future Directions

Web resource security can no longer rely on a reactive approach or simple perimeter defense. As web topologies expand to incorporate edge computing and complex cloud dependencies, security architectures must transition toward a comprehensive Zero Trust Philosophy, wherein no internal or external request is trusted implicitly. Future scientific exploration must focus heavily on the integration of contextual, artificial intelligence-driven behavior analysis capable of isolating runtime anomalies in real time. Ultimately, reducing the global attack surface requires a cultural shift toward standardizing robust software engineering patterns, forcing security to be treated as a core operational feature rather than a late-stage add-on.

REFERENCES

1. Singh, M. M., Adzman, K. A. A. K., Hassan, R. Near Field Communication (NFC) Technology Security Vulnerabilities and Countermeasures // International Journal of Engineering & Technology. – 2018. – No 7. – Pp. 298–305.

2. Kavya, S., Pavithra, K., Rajaram, S., Vahini, M., Harini, N. Vulnerability Analysis And Security System For NFC-Enabled Mobile Phones // International Journal of Scientific & Technology Research. – 2014. – Vol. 3. – No. 6. – Pp. 207–2010.