Research of Vulnerability Scanners of Web Applications of Intelligent Transport Systems

Mikhalevich Igor Feodosevich, Fedorenko Bogdan Nikolaevich, Shelamov Maxim Dmitrievich

Russian University of Transport (MIIT), Moscow, Russia

Contribution on the State of the Art

https://doi.org/10.7251/JIT2202127F UDC: 621.86/.87:65.011.56

Abstract: The intellectualization of transport systems is accompanied by the widespread use of web applications. The paper presents a system of criteria for evaluating the effectiveness of vulnerability scanners for web applications of intelligent transport systems, the features of the functioning of which impose additional requirements for the secure development of applications used in critical information infrastructure and systems interacting with it. A study was made of the most famous web application vulnerability scanners.

Keywords: Attack, computer attack, critical information infrastructure, information security, information security threat, intelligent transport system, vulnerability scanner, vulnerability web application.

Introduction

The construction of intelligent transport systems (hereinafter referred to as ITS) is based on the V2X concept [1, 2]. This concept provides that each intelligent vehicle interacts with other vehicles and transport infrastructure, as well as with any other objects, the functioning of which can affect transport security. On the other hand, ITS, as a kind of transport systems, are objects of critical information infrastructure (hereinafter referred to as CII) [3, 4]. The security of ITS (CII) significantly depends on the information security of software (hereinafter referred to as software).

Due to these factors, when developing and operating applications, ITS (CII) owners are required to take measures to ensure the information security of software at all stages of its life cycle, including following the rules for secure development, distribution, operation, technical support and decommissioning of software, and to utilize trusted software development tools, as well as trusted security controls for software used at ITS facilities (CII) [5, 6].

The intellectualization of transport systems is accompanied by the widespread use of web applications. This is due, in particular, to the possibility of their development for cross-platform use and the relatively weak requirements for the computing power of devices and their memory. Web applications are also characterized by comparative simplicity and high speed of development, which, along with advantages, entails the risk of deficiencies in architecture, code, support, and others.

Let us clarify that a web application is a program executed by a web server that responds to dynamic web page requests via the HTTP protocol [7].

For ITS web applications, a wide range of threats are relevant, which can manifest themselves in the physical, network and wireless areas of information security and create diverse vectors of computer attacks on ITS (Fig. 1 [8]). In the physical domain, threats to ITS web applications arise due to the possibility of direct access to devices embedded in vehicles and transport infrastructure facilities. Violations of the information security of web applications can be caused by the possibility of carrying out attacks on vehicles, transport infrastructure objects

and devices of other objects located in close proximity to the attacked ITS, via wireless accessed network. ITS web application vulnerabilities can also be exploited remotely via networks.

Successful attacks on web applications can violate the security of information of ITS owners, vehicles and other ITS objects, lead to violations of confidentiality, integrity, accessed onness, authenticity, non-traceability, accountability and other important properties of information.

Software vulnerability scanners are an important information security tool. The effectiveness of vulnerability scanners (their performance, accuracy of vulnerability detection, and other properties) significantly depends on the completeness of the databases used, the volumes of which can be quite significant. Under these conditions, it is advisable to adapt the algorithms of the scanners for specific types of software or use specialized scanners. A web application vulnerability scanner is an automated security program that looks for software vulnerabilities in web applications [7]. These scanners are widely available both in the form of free software and in proprietary (commercial, paid) versions.

Figure 1. *Information security areas of ITS*

Figure 2. *ITS information security threat landscape*

Given the peculiarities of the architecture of web applications, as well as the intensive growth of threats in this particular area of information technology, this paper considers web application vulnerability scanners (WAVS, web application vulnerability scanners).

WEB APPLICATION ARCHITECTURE

A web application consists of a set of scripts that reside on a web server and interact with databases or other dynamic content sources. Using the infrastructure of the Internet, web applications allow service providers and customers to share and manipulate information regardless of platform.

The web application has a distributed n-tier architecture. Typically, there is a client (web browser), a web server, an application server (or multiple application servers), and a database server (data storage system). Figure 1 shows a simplified view of the web application. A firewall may operate between the web client and the web server.

Figure 3. *Layered web application architecture*

CLASSIFICATION OF WEB APPLICATION VULNERABILITIES

The growing number of web site vulnerabilities has prompted many organizations to take a critical look at the security quality of their web applications and has led to the formation, among other things, of large international communities seeking to improve the security of web applications. These com-

munities have proposed several classifications for threats, vulnerabilities, and attacks on web applications. These communities include OWASP (Open Web Application Security Project) and WASC (Web Application Security Consortium).

Classification of vulnerabilities according to WASC

The Web Application Security Consortium (WASC) identifies 49 types of threats classified according to two basic criteria: weaknesses (the cause of the vulnerability) and attacks (types of attacks) [9]. This classification is presented in Table 1.

Table 1. *Classification of threats according to WASC*

Threat	
Attack	**Weakness**
Abuse of functionality	Incorrect application configuration
Brute force	Catalog indexing
Buffer overflow	Incorrect file system permissions
Substitution of information	
Session/credential prediction	Incorrect input handling
Cross Site Scripting (XSS)	Incorrect output handling
Cross-site request forgery	Information leak
Denial of Service	Insecure Indexing
Fingerprinting	Insufficient protection against automation
String format error	
Smuggling HTTP responses	Insufficient authentication
Splitting HTTP responses	Insufficient authorization
Smuggling HTTP Requests	Insufficient password recovery
Splitting HTTP Requests	Incomplete process check
Integer overflows	Insufficient session expiration
LDAP injection	Insufficient transport layer protection
Implementing mail commands	
Zero bit injection	Incorrect server configuration
OS management	
Detour path	
Resource location prediction	
Remote File Inclusion (RFI)	
Route detour	
Session fixation	
SOAP array abuse	
SSI injections	
SQL injection	
URL redirect abuse	
XPath Injection	
XML Attribute Extension	
External XML Objects	
Extending XML Objects	
XML injection	
XQuery injection	

In addition, the WASP classification indicates where in the development life cycle (design, implementation, and deployment) of an application, a particular type of threat is most likely to be applied.

Classification of vulnerabilities according to OWASP

Unlike WASC, which describes all possible attacks, the Open Web Application Security Project (OWASP) only considers the top 10 security threats every 3 years. It publishes a ranking of the 10 most dangerous security vulnerabilities in the "OWASP Top 10" document and allows the project team to focus on protecting the web application from the most important threats [10].

The ranking (in descending order of number) of the top ten web application vulnerabilities according to the OWASP 2021 report is shown in Table 2 [11].

Table 2. *Top 10 web application vulnerabilities according to OWASP 2021 report*

Location	Type of vulnerability
A1	Broken access control
A2	Cryptographic failures
A3	Injections
A4	Insecure design
A5	Incorrect security setting
A6	Vulnerable and outdated components
A7	Failures iniIdentification and authentication
A8	Failures in software and data integrity
A9	Security logging and crash monitoring
A10	Server side request forgery

WEB APPLICATION VULNERABILITY SCANNERS

The following main tasks are assigned to web application vulnerability scanners [12, 13]:

- search for all types of vulnerabilities described in OWASP top 10 [11];
- reporting an attack that demonstrates a vulnerability;
- definition of an attack with indication of the location of the script, input data and context;
- definition of a vulnerability with a name semantically equivalent to the names from the OWASP top 10;
- providing the ability to authenticate in the application and maintain a connected state;
- providing a sufficiently low level of false positive results.

Architecture of web application vulnerability scanners

The basic architecture of web application scanners (WAVS) is shown in Figure 3.

Figure 4. *Basic architecture of web application vulnerability scanners*

A. Bypass module

The traversal is performed by a component called a crawler. The crawler examines a web application to recover and identify web pages related input vectors such as HTML form input fields, GET and POST request parameters, and cookies. In addition, the scanner creates an indexed list of all pages viewed. Detection of web vulnerabilities significantly depends on the quality of the crawler. If the crawler is mediocre, then the scanner will most likely miss the vulnerability [14, 15].

B. Attack module (fuzzing)

Fuzzing is performed by a component called a fuzzer. The fuzzer parses the page URLs and input vectors viewed by the crawler and then sends potential attack patterns to the entry points defined in the previous step. This component generates incorrect, random, or unexpected values to run a threat check for each entry and vulnerability type that the Web Application Scanner checks. For example, to check for the possibility of cross-site scripting (Cross-Site Scripting, hereinafter XSS vulnerability), the attacking module (fuzzer) will try to inject malicious Javascript code [14, 15].

C. Analysis module

This module analyzes the results of the fuzzing step to detect the presence of vulnerabilities and provide data to other modules. If the pages returned in response to the input tests for SQL injection contain a database error message, the analysis module will conclude that there is a SQL vulnerability [14, 15].

Preliminary analysis of security scanners

When examining scanners, two approaches are used. In one case, scanners are checked to identify one specific vulnerability, as, for example, in [14, 15]. In another case, researchers use a large number of types of vulnerabilities in their test applications [16-21].

In [16], an analysis of the effectiveness of 8 commercial vulnerability scanners on well-known applications was carried out. Most scanners found the SQL injection vulnerability and reflected the XSS vulnerability. Other vulnerabilities were not detected at all or were detected at a very low frequency.

In [17] several scanners were tested that failed to detect reflected XSS and SQL injection, but were able to detect stored XSS and cross site request.

In [18], an analysis of open source scanners was carried out in accordance with the OWASP Top 10

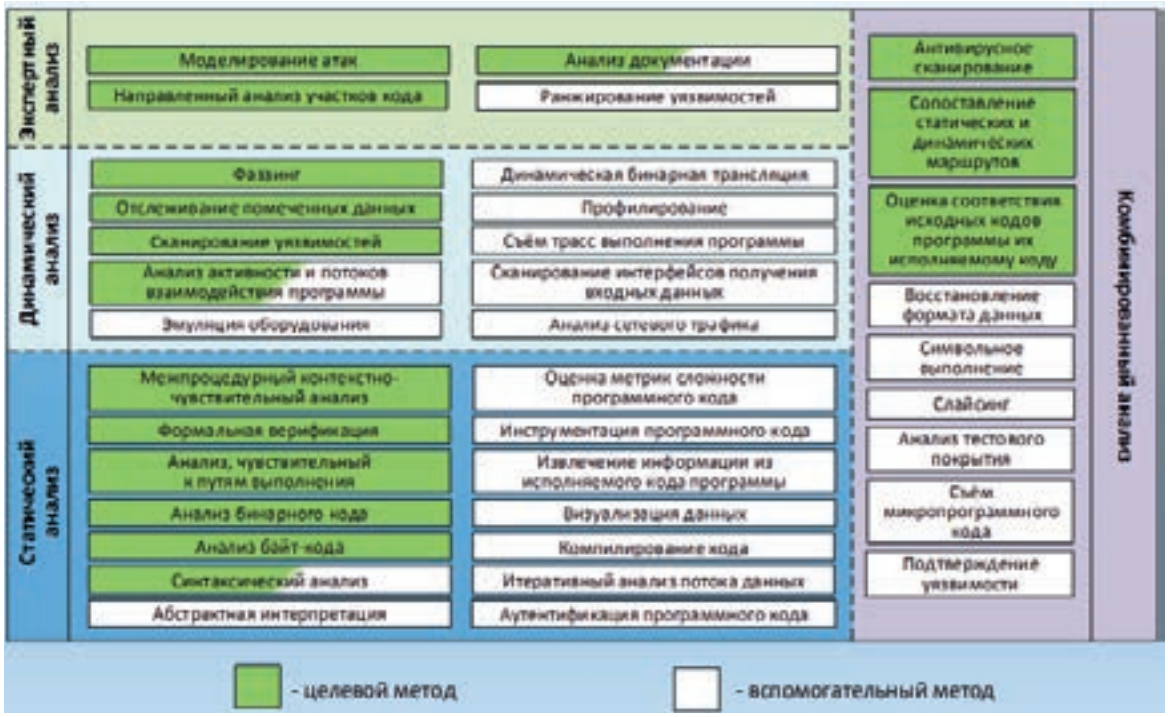

Figure 5. *General methodology for finding vulnerabilities and undeclared software features*

according to the criterion of detectability, using an average indicator.

In [19], 3 scanners were tested for 3 different web applications in terms of code vulnerabilities detection.

In [20], the main attention was paid to the detection of stored XSS, based on the results of the analysis [21], the previously obtained estimates of the weaknesses and limitations of scanners were confirmed.

Reference [21] presents some evaluation reports on the results of running QualysGuard WAS and Acunetix WVS against the selected test bed.

However, in these and other well-known works, a comprehensive study of the effectiveness of scanners, including their performance, was not carried out. This paper evaluates eleven web application scanners (both commercial and open source) against comprehensive performance metrics.

Methodology for Searching for Web Application Vulnerabilities

The general methodology for searching for vulnerabilities and undeclared software features is described in [22] and shown in Fig. 2. 4.

Figure 6. *Methodological process*

For web applications, the procedure for searching for vulnerabilities that should be performed by the scanner is shown in Fig. 5.

A. Choice of reference

Vulnerable test applications are required to evaluate and test scanners. These applications must be described in detail, contain a list of all the weaknesses introduced into the application, so that it is possible to fully determine all true positive, false positive and false negative scanner responses and their aggregate values based on the scan results (hereinafter referred to as IP, LP, LO results, respectively).

True positive results reflect the number of vulnerabilities found that actually exist. In a web application, this indicator should tend to zero.

False positive results indicate the number of vulnerabilities found that are not actually present in the application.

False negatives show the number of vulnerabilities introduced into the application but not detected by the scanner.

The study used a vulnerable web application from the WAVSEP (Web Application Vulnerability Scanner Evaluation Project) [23]. This evaluation platform contains a collection of unique vulnerable web pages that can be used to test various properties of web application scanners.

The used vulnerable web application WAVSEP BENCHMARK is written in java JSP and is designed to evaluate the capabilities, quality and accuracy of web application vulnerability scanners [24].

The WAVSEP benchmark includes the test cases shown in Table 3.

Table 3. *Total number of vulnerabilities for each type in WAVSEP*

Vulnerability type	Number of true positive cases	Number of false positive cases
SQL Injection	136	10
Reflected XSS	66	7
RFI - Remote File Inclusion	108	6
Path traversal /LFI - Local File Inclusion	816	8

B. Selection of scanners to study

The study was conducted on commercial and open source scanners, details of which are presented in Table 4 (commercial scanners in italics).

C. Selecting scanner performance metrics

The study was conducted using the following performance indicators of scanners [25, 26].

A) Accuracy, as the ratio of correctly detected vulnerabilities to the total number of test vulnerabilities:

$$Accuracy = \frac{IP}{IP + LP}$$

The higher the Accuracy, the lower the number of false positives, hence the scanner is more accurate in detecting vulnerabilities.

Table 4. *General characteristics of the studied scanners*

Scanner	Company / developer	Version	License	Technology
BurpSuite	PortSwiger	1.6.12	Commercial / Free (Limited Capability)	Java
Acunetix	Acunetix	10	Commercial / Free (Limited Capability)	Perl
Wapiti	InformaticaGesfor	2.3.0	Open Source	Python
SkipFish	Google	2.10	Open Source	C
Netsparker	Mavituna Security	2.3	Commercial	.Net
W3AF	W3af devel	1.2	Open Source	Python
AppSpider	Rapid 7	6.0	Commercial	Java
IronWASP	L. Kuppan	0.9.7.1	Open Source	.Net
Arachni	Tasos Laskos	2.2.1	Commercial	Ruby
ZAP	OWASP	2.3.1	Open Source	.Net
Vega	Subgraph	1.0 (beta)	Open Source	Java

B) Sensitivity, as the ratio of correctly detected vulnerabilities to the total number of detected vulnerabilities:

$$Sensitivity = \frac{IP}{IP + LO}$$

The more Sensitivity, the less false negative results. Hence, the tool is better at detecting vulnerabilities.

C) F-measure, which represents the harmonic mean of accuracy and sensitivity, given by:

$$F\text{–}measure = \frac{2 \times Accuracy \times Sensitivity}{Accuracy + Sensitivity}$$

These three metrics can be used to rank scanners according to the goals of the benchmark user.

Results of evaluation of web application scanners

Scanners were examined using the WAVSEP test suite.

Tables 5, 6 show the test results. Values highlighted in bold indicate the total number of vulnerabilities in WAVSEP mentioned in Table 5. Values highlighted in yellow indicate the level of detection, while the rest indicate the number of IP, LP, LO found.

Tables 7-10 present the results of the study of scanners according to the criteria of accuracy, sensitivity and F-measure.

D) Generalization of test results

Summary graphs of accuracy, sensitivity, and F-measure for the studied scanners and vulnerabilities are shown in Fig. 6-9.

Table 5. *Results of WAVSEP tests for SQLI and XSS vulnerabilities by IP, LP, LO values*

Scanners	Test results					
	SQLI			XSS		
	IP	LO	LP	IP	LO	LP
	136		**10**	**66**		**7**
BurpSuite	136	0	3	62	4	0
	100%		30%	93.93%		0%
Wapiti	136	0	2	44	22	3
	100%		20%	66.66%		42.85%
Acunetix	136	0	0	66	0	0
	100%		0%	100%		0%
SkipFish	102	34	0	65	1	0
	75%		0%	98.48%		0%
Netsparker	136	0	3	64	2	0
	100%		30%	96.96%		0%
W3AF	81	55	3	19	47	3
	59.55%		30%	28.78%		42.85%
AppSpider	132	4	0	66	0	0
	97.05%		0%	100%		0%
IronWASP	136	0	5	52	14	0
	100%		50%	78.78%		0%
Arachni	136	0	5	63	3	0
	100%		50%	95.45%		0%
ZAP	136	0	0	63	3	0
	100%		0%	95.45%		0%
Vega	136	0	2	66	0	0
	100%		20%	100%		0%

Table 6. *WAVSEP test results for RFI and LFI vulnerabilities by IP, LP, LO values*

Scanners	Test results					
	RFI			LFI		
	IP	LO	LP	IP	LO	LP
	108		**6**	**816**		**8**
BurpSuite	80	28	0	496	320	1
	74.07%		0%	60.78%		12.5%
Wapiti	64	44	0	414	402	1
	59.25%		0%	50.73%		12.5%
Acunetix	87	21	0	292	524	0
	80.55%		0%	35.78%		0%
SkipFish	39	69	1	312	504	2
	36.11%		16.66%	38.23%		25%
Netsparker	57	51	0	467	349	0
	52.77%		0%	57.23%		0%
W3AF	13	95	1	461	355	1
	12.03%		16.66%	56.49%		12.5%
AppSpider	80	28	0	660	156	1
	74.07%		0%	80.88%		12.5%
IronWASP	106	2	0	288	528	1
	98.14%		0%	35.29%		12.5%
Arachni	46	62	0	162	654	0
	42.59%		0%	19.85%		0%
ZAP	136	0	1	590	226	0
	100%		16.66%	72.30%		0%
Vega	108	0	0	519	297	5
	100%		0%	63.60%		62.5%

Table 7. *Results of WAVSEP tests by values of accuracy, sensitivity and F-measures for SQLI vulnerability*

Scanners	Test results		
	SQLI		
	Accuracy, %	Sensitivity, %	F-measure, %
BurpSuite	97.84	100	98.90
Wapiti	98.55	100	99.26
Acunetix	100	100	100
SkipFish	100	100	100
Netsparker	97.84	100	98.90
W3AF	96.42	59.55	73.62
AppSpider	100	97.05	98.50
IronWASP	96.45	100	98.19
Arachni	96.45	100	98.19
ZAP	100	100	100
Vega	98.55	100	99.26

Table 8. *WAVSEP test results for accuracy, sensitivity and F-measures for XSS vulnerability*

Scanners	Test results		
	XSS		
	Accuracy, %	Sensitivity, %	F-measure, %
BurpSuite	100	93.93	96.87
Wapiti	93.61	66.66	77.86
Acunetix	100	100	100
SkipFish	100	100	100
Netsparker	100	96.96	98.45
W3AF	86.36	28.78	43.17
AppSpider	100	100	100
IronWASP	100	78.78	88.13
Arachni	100	95.45	97.67
ZAP	100	95.45	97.67
Vega	100	100	100

Table 9. *WAVSEP test results for accuracy, sensitivity and F-measures for RFI vulnerability*

Scanners	Test results		
	RFI		
	Accuracy, %	Sensitivity, %	F-measure, %
BurpSuite	100	74.07	85.10
Wapiti	100	59.25	74.41
Acunetix	100	80.55	89.22
SkipFish	100	80.55	89.22
Netsparker	100	52.77	69.08
W3AF	92.85	12.03	21.30
AppSpider	100	74.07	85.10
IronWASP	100	98.14	99.06
Arachni	100	42.59	59.73
ZAP	99.08	100	99.53
Vega	100	100	100

Table 10. *WAVSEP test results for accuracy, sensitivity and F-measures for LFI vulnerability*

Scanners	Test results		
	LFI		
	Accuracy, %	Sensitivity, %	F-measure, %
BurpSuite	99.79	60.78	75.54
Wapiti	99.75	50.73	67.25
Acunetix	100	35.78	52.70
SkipFish	100	35.78	52.70
Netsparker	100	57.23	72.79
W3AF	99.78	56.49	72.13
AppSpider	99.84	80.88	89.36
IronWASP	99.65	35.29	52.12
Arachni	100	19.85	33.12
ZAP	100	72.30	83.92
Vega	99.04	63.60	77.45

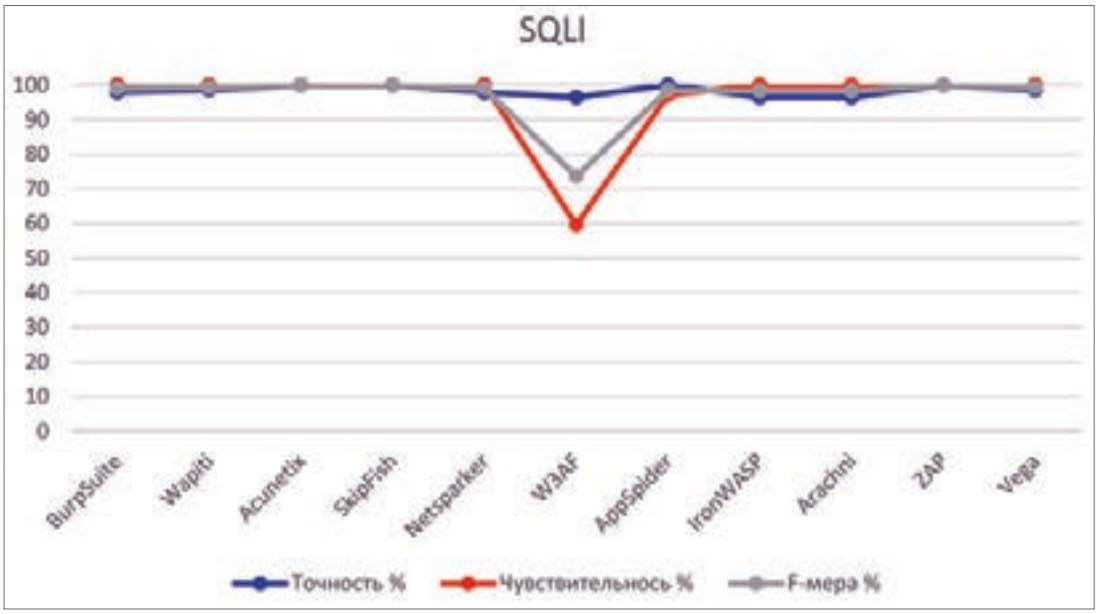

Figure 7. *Accuracy, sensitivity, and F-measures of SQLI vulnerability in WAVSEP*

As can be seen, regarding the vulnerability of SQLI scanners Acunetix, SkipFish, ZAP Acunetix, SkipFish, ZAP have the highest F-measure values (100%). Scanners BurpSuite, Netsparker, AppSpider, IronWASP, and Arachni showed F-measures of 98%, while the W3AF scanner had an F-measure of 73.62%.

So besides W3AF scanners, open source and commercial options work well for the SQLI vulnerability.

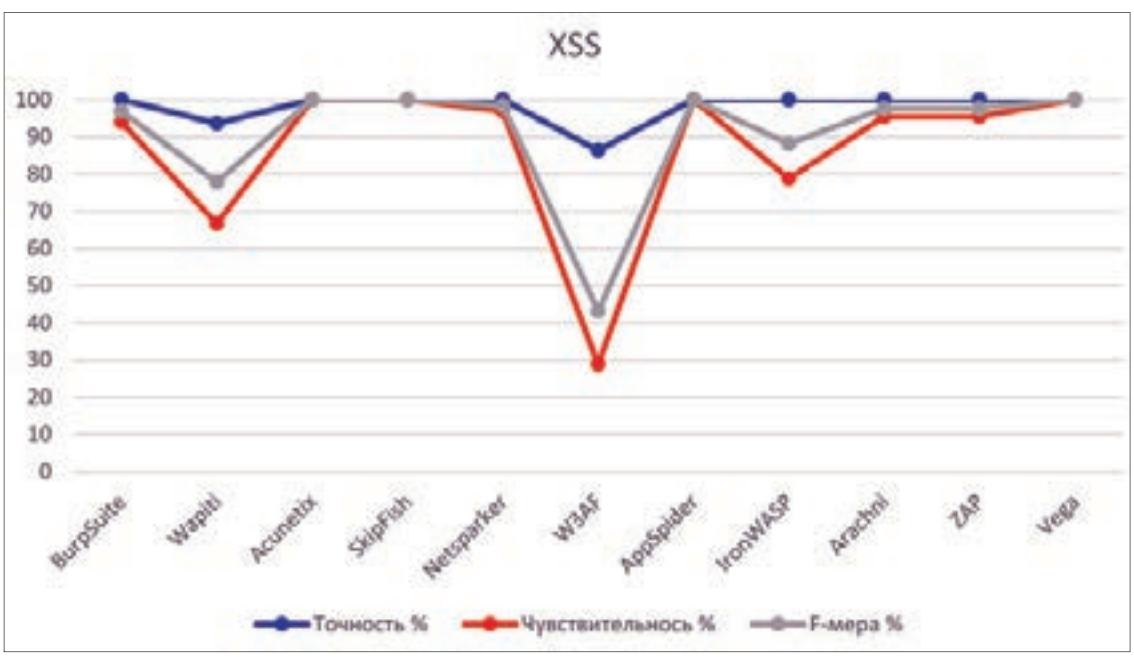

Figure 8. *XSS precision, sensitivity, and F-measures of vulnerability in WAVSEP*

With respect to the XSS vulnerability, the F-measure value of the Acunetix, Skipfish, Appspider, and Vega scanners was 100%. Similar values show scanners Netsparker, Arachni, Zap, Burp suite and IronWASP. But the efficiency of Wapiti and W3AF scanners is much lower in all respects.

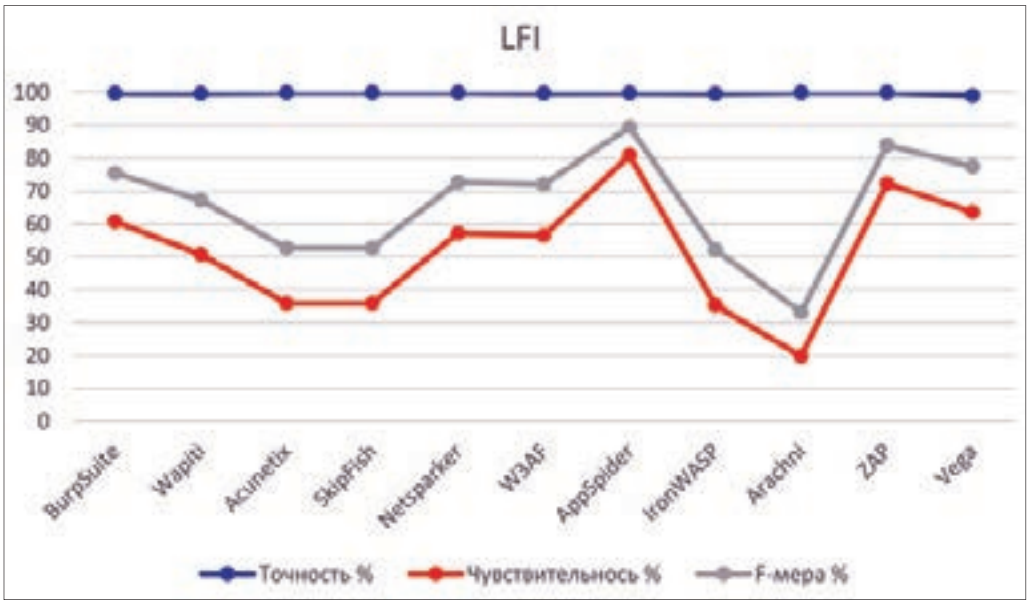

Figure 9. *Accuracy, sensitivity and F-measures of LFI vulnerability in WAVSEP*

Regarding the LFI vulnerability, the best results were obtained by the AppSpider and ZAP scanners. At the same time, no scanner showed 100% efficiency in terms of sensitivity and F-measure.

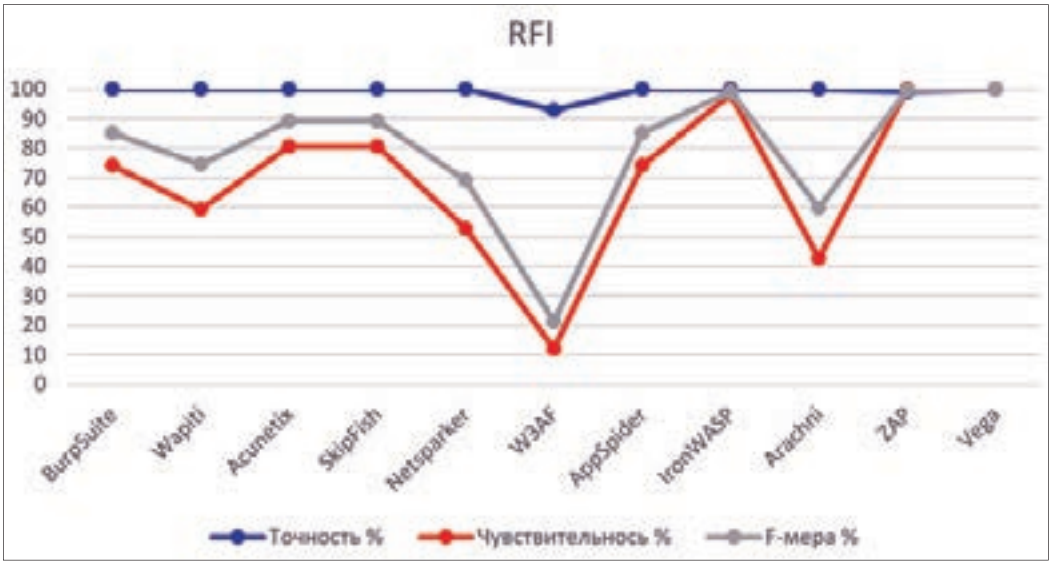

Figure 10. *Accuracy, sensitivity and F-measures to RFI vulnerability in WAVSEP*

For the RFI vulnerability, 100% results are from IronWASP, Zap, and Vega open source scanners. Not a single commercial scanner has shown 100% efficiency. The lowest results were obtained with the W3AF scanner.

Conclusion

The effectiveness of the studied web application vulnerability scanners depends on the types of vulnerabilities. At the same time, the effectiveness of all scanners in relation to SQLI and XSS vulnerabilities is higher than in relation to LFI and RFI vulnerabilities.

In terms of LFI and RFI vulnerabilities, Vega and ZAP scanners, both open source scanners, performed the best, outperforming all commercial scanners.

The availability of open source scanners that show high efficiency (100% for some types of vulnerabilities) allows us to conclude that it is possible to create a good web application vulnerability scanner that covers the types of vulnerabilities specific to ITS.

Information about authors

Mikhalevich Igor Feodosevich - Associate Professor, Department of Information Management and Security, Russian University of Transport (MIIT)
academic degree: candidate of technical sciences
academic title: senior researcher
e-mail: mif-orel@mail.ru

Fedorenko Bogdan Nikolaevich - student of the department "Management and Information Protection" at the Russian University of Transport (MIIT)

Shelamov Maxim Dmitrievich - student of the department "Management and Information Protection" at the Russian University of Transport (MIIT)

Исследование сканеров уязвимостей веб-приложений интеллектуальных транспортных систем

Михалевич Игорь Феодосьевич, Федоренко Богдан Николаевич, Шеламов Максим Дмитриевич

Российский университет транспорта (МИИТ), Москва, mif-orel@mail.ru

Оригинальная научная статья

Аннотация: Интеллектуализация транспортных систем сопровождается широким применением веб-приложений. В работе представлена система критериев оценки эффективности сканеров уязвимостей веб-приложений интеллектуальных транспортных систем, особенности функционирования которых накладывают дополнительные требования по безопасной разработке приложений, используемых в критической информационной инфраструктуре и системах, взаимодействующих с ней. Проведено исследование наиболее известных сканеров уязвимостей веб-приложений.

Ключевые слова: Атака, веб-приложение, информационная безопасность, интеллектуальная транспортная система, компьютерная атака, критическая информационная инфраструктура, сканер уязвимостей, угроза информационной безопасности, уязвимость.

Введение

Построение интеллектуальных транспортных систем (далее – ИТС) основано на концепции V2X [1, 2]. Данная концепция предусматривает, что каждое интеллектуальное транспортное средство взаимодействует с другими транспортными средствами и транспортной инфраструктурой, а также с любыми другими объектами, функционирование которых способно оказать влияние на транспортную безопасность. С другой стороны ИТС, как разновидности транспортных систем, являются объектами критической информационной инфраструктуры (далее – КИИ) [3, 4]. Безопасность ИТС (КИИ) существенно зависит от информационной безопасности программного обеспечения (далее – ПО).

Вследствие указанных факторов при разработке и эксплуатации приложений владельцы ИТС (КИИ) обязаны принимать меры по обеспечению информационной безопасности ПО на всех этапах его жизненного цикла, в том числе соблюдать правила безопасной разработки, распространения, эксплуатации, технической поддержки и вывода из эксплуатации ПО, использовать доверенные средства разработки ПО, а также доверенные средства контроля безопасности ПО, используемого на объектах ИТС (КИИ) [5, 6].

Интеллектуализация транспортных систем сопровождается широким применением веб-приложений. Это обусловлено, в частности, возможностью их разработки для кроссплатформенного применения и сравнительно слабыми требованиями к вычислительной мощности устройств и объемам их памяти. Веб-приложения характеризуются также сравнительной простотой и высокой скоростью разработки, что, наряду с достоинствами, влечет риски появления недостатков в архитектуре, коде, поддержке и другие.

Уточним, что под веб-приложением понимается программа, выполняемая веб-сервером, отвечающая на динамические запросы веб-страницы по протоколу HTTP [7].

Для веб-приложений ИТС актуален широкий спектр угроз, которые могут проявляться в физической, сетевой и беспроводной областях информационной безопасности и создавать многообразные векторы компьютерных атак на ИТС (рис. 1 [8]). В физической области угрозы веб-приложениям ИТС возникают вследствие возможности прямого доступа к устройствам, встроенным в транспортные средства и объекты транспортной инфраструктуры. Нарушения информационной безопасности веб-приложений могут быть вызваны возможностью осуществления атак на транспортные средства, объекты транспортной инфраструктуры и устройства иных объектов, находящихся в непосредственной близости к атакуемым ИТС, посредством беспроводного доступа. Уязвимости веб-приложений ИТС могут эксплуатироваться также удаленно через сети.

Рисунок 1. *Области информационной безопасности ИТС*

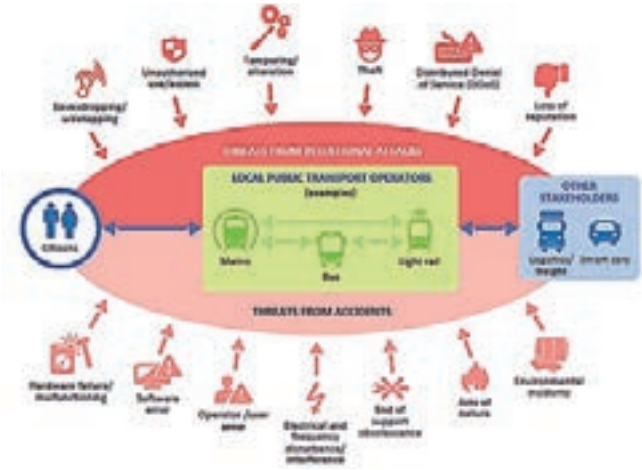

Рисунок 2. *Ландшафт угроз информационной безопасности ИТС*

Успешные атаки на веб-приложения могут нарушить безопасность информации владельцев ИТС, транспортных средств и других объектов ИТС, повлечь нарушения конфиденциальности, целостности, доступности, подлинности, неотслеживаемости, подотчетности и других важных свойств информации.

Важным инструментом обеспечения информационной безопасности являются сканеры уязвимостей ПО. Эффективность сканеров уязвимостей (их производительность, точность выявления уязвимостей и другие свойства) существенно зависит от полноты используемых баз, объемы которых могут быть весьма значительными. В этих условиях целесообразно адаптировать алгоритмы функционирования сканеров под конкретные виды ПО или использовать специализированные сканеры. Под сканером уязвимостей веб-приложений понимается автоматизированная программа безопасности, которая ищет программные уязвимости в веб-приложениях [7]. Данные сканеры широко представлены как в форме свободного ПО, так и в проприетарных (коммерческих, платных) версиях.

Учитывая особенности архитектуры веб-приложений, а также интенсивный рост угроз именно в данной области информационных технологий, в настоящей работе рассмотрены сканеры уязвимостей веб-приложений (далее – WAVS, web application vulnerability scanners).

Архитектура веб-приложений

Веб-приложение состоит из набора скриптов, которые находятся на веб-сервере и взаимодействуют с базами данных или другими источниками динамического контента. Используя инфраструктуру Интернета, веб-приложения позволяют поставщикам услуг и клиентам обмениваться информацией и манипулировать ею независимо от платформы.

Веб-приложение имеет распределенную *n*-уровневую архитектуру. Как правило, существует клиент (веб-браузер), веб-сервер, сервер приложений (или несколько серверов приложений) и сервер баз данных (система хранения данных). На рисунке 1 предста-

влен упрощенный вид веб-приложения. Между веб-клиентом и веб-сервером может функционировать брандмауэр.

Рисунок 3. *Многоуровневая архитектура веб-приложений*

Классификация уязвимостей веб-приложений

Растущее число уязвимостей на веб-сайтах побудило многие организации критически взглянуть на качество безопасности своих веб-приложений привело к образованию, в том числе, крупных международных сообществ, стремящихся к повышению уровня безопасности веб-приложений. Данными сообществами предложено несколько классификаций для угроз, уязвимостей и атак на веб-приложения. Среди этих сообществ можно выделить такие, как OWASP (Open Web Application Security Project) и WASC (Web Application Security Consortium).

Классификация уязвимостей по версии WASC

Консорциум по безопасности веб-приложений (WASC) выделяет 49 видов угроз, классифицированных по двум базовым признакам: слабости (причина возникновения уязвимости) и атаки (виды атак) [9]. Данная классификация представлена в таблице 1.

Таблица 1. *Классификация угроз по версии WASC*

Угрозы	
Атаки	**Слабости**
Злоупотребление функциональными возможностями	Неправильная конфигурация приложения
Брутфорс	Индексация каталогов
Переполнение буфера	Неправильные разрешения файловой системы
Подмена информации	
Прогнозирование сеансов/учетных данных	Неправильная обработка ввода
Межсайтовый скриптинг (XSS)	Неправильная обработка вывода
Подделка межсайтовых запросов	Утечка информации
Отказ в обслуживании	Небезопасная индексация
Снятие отпечатков пальцев	Недостаточная защита от автоматизации
Ошибка форматирования строки	
Контрабанда HTTP-ответов	Недостаточная аутентификация
Разделение HTTP-ответов	Недостаточная авторизация
Контрабанда HTTP-запросов	Недостаточное восстановление пароля
Разделение HTTP-запросов	Неполная проверка процесса
Целочисленные переполнения	Недостаточный срок действия сеанса
LDAP-инъекции	Недостаточная защита транспортного уровня
Внедрение почтовых команд	
Внедрение нулевого бита	Неправильная конфигурация сервера
Управление ОС	
Обход пути	
Предсказывание местоположения ресурсов	
Удаленное включение файлов (RFI)	
Маршрутный обход	
Фиксация сеанса	
Злоупотребление массивом SOAP	
SSI-инъекции	
SQL-инъекции	
Злоупотребление переадресацией URL	
XPath-инъекции	
Расширение XML-атрибута	
Внешние объекты XML	
Расширение объектов XML	
XML-инъекция	
XQuery-инъекция	

Помимо этого, в классификации WASP указывается, где в жизненном цикле разработки (при проектировании, реализации и развертывании) приложения вероятнее всего может быть применен конкретный тип угрозы.

Классификация уязвимостей по версии OWASP

В отличие от WASC, который описывает все возможные атаки, проект Open Web Application Security Project (OWASP) рассматривает только

10 основных угроз безопасности каждые 3 года. Он публикует рейтинг 10 наиболее опасных уязвимостей безопасности в документе «OWASP Top 10» и позволяет команде проекта сосредоточиться на защите веб-приложения от наиболее важных угроз [10].

Рейтинг (в порядке убывания номера) десяти основных уязвимостей веб-приложений по отчету OWASP 2021 года приведен в таблице 2 [11].

Таблица 2. *Топ 10 уязвимостей веб-приложений по отчету OWASP 2021 года*

Место	Тип уязвимости
A1	Нарушенный контроль доступа
A2	Криптографические сбои
A3	Инъекции
A4	Небезопасный дизайн
A5	Неправильная настройка безопасности
A6	Уязвимые и устаревшие компоненты
A7	Сбои в идентификации и аутентификации
A8	Сбои в целостности программного обеспечения и данных
A9	Ведение журнала безопасности и мониторинг сбоев
A10	Подделка запроса на стороне сервера

Сканеры уязвимостей веб-приложений

На сканеры уязвимостей веб-приложений возлагается выполнение следующих основных задач [12, 13]:

- поиск всех типов уязвимостей, описанных в OWASP top 10 [11];
- сообщение об атаке, которая демонстрирует уязвимость;
- определение атаки с указанием местоположения скрипта, входных данных и контекста;
- определение уязвимости с именем, семантически эквивалентным названиям из OWASP топ-10;
- предоставление возможности проходить аутентификацию в приложении и поддерживать подключенное состояние;
- обеспечивать достаточно низкий уровень ложноположительных результатов.

Архитектура сканеров уязвимостей веб-приложений

Базовая архитектура сканеров веб-приложений (WAVS) представлена на рис. 3.

Рисунок 4. *Базовая архитектура сканеров уязвимостей веб-приложений*

А. Модуль обхода

Обход выполняется компонентом, называемым краулер (Crawler). Краулер исследует веб-приложение для восстановления и идентификации веб-страниц, связанных векторов ввода, таких как поля ввода HTML-форм, параметры запроса GET и POST и файлы cookie. Кроме того, сканер создает индексированный список всех просмотренных страниц. Обнаружение веб-уязвимостей существенно зависит от качества краулера. Если краулер посредственный, то сканер наверняка пропустит уязвимость [14, 15].

Б. Атакующий модуль (фаззинг)

Фаззинг выполняется компонентом под названием фаззер (fuzzer). Фаззер анализирует URL-адреса страниц и входные векторы, просматриваемые краулером, а затем отправляет потенциальные шаблоны атак в точки входа, определенные на предыдущем шаге. Этот компонент генерирует неправильные, случайные или неожиданные значения для запуска проверки наличия угроз для каждой записи и типов уязвимостей, которые проверяет сканер веб-приложения. Например, для проверки наличия возможности межсайтового скриптинга (Cross-Site Scripting, далее уязвимость XSS) атакующий мо-

дуль (fuzzer) попытается внедрить вредоносный код Javascript [14, 15].

В. Модуль анализа

Этот модуль анализирует результаты, полученные на этапе фаззинга, чтобы обнаружить наличие уязвимостей и предоставить данные другим модулям. Если страницы, возвращенные в ответ на входные тесты для SQL-внедрения, содержат сообщение об ошибке базы данных, модуль анализа сделает вывод о наличии уязвимости SQL [14, 15].

Предварительный анализ сканеров безопасности

При исследовании сканеров применяется два подхода. В одном случае сканеры проверяются на предмет выявления одной конкретной уязвимости, как, например, в [14, 15]. В другом случае исследователи в своих тестовых приложениях используют большое количество видов уязвимостей [16-21].

В [16] проведен анализ эффективности 8 коммерческих сканеров уязвимостей на хорошо известных приложениях. Большинство сканеров обнаружило уязвимость SQL-инъекции и отразили уязвимость XSS. Другие уязвимости вообще не были обнаружены или обнаруживались с очень низкой частотой.

В [17] проверено несколько сканеров, которые не смогли обнаружить отраженный XSS и SQL-инъекцию, но смогли обнаружить сохраненный XSS и межсайтовый запрос.

В [18] проведен анализ сканеров с открытым исходным кодом в соответствии с OWASP Top 10 по критерию возможности обнаружения, используя средний показатель.

В [19] протестированы 3 сканера для 3 различных веб-приложений по критерию обнаружения уязвимостей кода.

В [20] основное внимание уделено обнаружению сохраненных XSS, основываясь на результатах анализа [21] были подтверждены полученные ранее оценки слабостей и ограниченных возможностей сканеров.

В [21] представлены некоторые оценочные отчеты по результатам запуска QualysGuard WAS и Acunetix WVS в противовес выбранному испытательному стенду.

Однако, в этих и других известных работах не проводилось комплексное исследование эффективности сканеров, в том числе их производительности. В данной работе представлена оценка одиннадцати сканеров веб-приложений (как коммерческих, так и с открытым исходным кодом) по комплексным показателям эффективности.

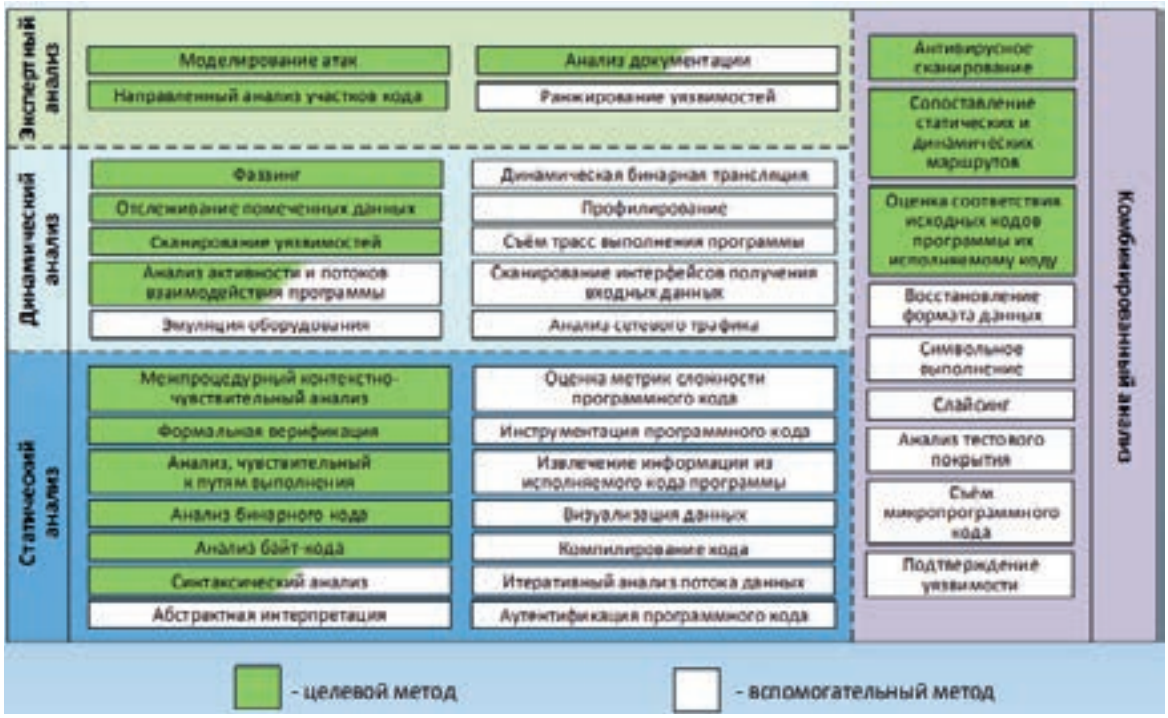

Рисунок 5. *Общая методология поиска уязвимостей и недекларированных возможностей ПО*

Рисунок 6. *Методологический процесс*

Методология поиска уязвимостей веб-приложений

Общая методология поиска уязвимостей и недекларированных возможностей ПО изложена в [22] и представлена на рис. 4.

Применительно к веб-приложениям порядок поиска уязвимостей, который должен выполняться сканером, представлен на рис. 5.

А. Выбор эталона

Для оценки и тестирования сканеров необходимы уязвимые тестовые приложения. Эти приложения должны быть подробно описаны, содержать перечень всех внесенных в приложение слабостей, чтобы была возможность полно определить все истинно положительные, ложноположительные и ложноотрицательные срабатывания сканера и их совокупные значения по результатам сканирования (далее – ИП, ЛП, ЛО результаты соответственно).

Истинно положительные результаты отражают количество обнаруженных уязвимостей, которые действительно существуют. В веб-приложении данный показатель должен стремиться к нулю

Ложно положительные результаты говорят о количестве обнаруженных уязвимостей, которых в приложении на самом деле нет.

Ложно отрицательные результаты показывают число уязвимостей, внесенных в приложение, но не обнаруженных сканером.

В исследовании было использовано уязвимое веб-приложение из проекта WAVSEP (Web Application Vulnerability Scanner Evaluation Project) [23]. Эта оценочная платформа содержит коллекцию уникальных уязвимых веб-страниц, которые могут быть использованы для тестирования различных свойств сканеров веб-приложений.

Использованное уязвимое веб-приложение WAVSEP BENCHMARK написано на java JSP и предназначено для оценки возможностей, качества и точности сканеров уязвимостей веб-приложений [24].

WAVSEP benchmark включает в себя тестовые примеры, представленные в таблице 3.

Таблица 3. *Общее количество уязвимостей для каждого типа в WAVSEP*

Тип уязвимости	Количество истинно положительных кейсов	Количество ложноположительных кейсов
SQL-инъекция (SQL Injection)	136	10
Отраженные XSS (Reflected XSS)	66	7
Удаленное включение файлов (RFI - Remote File Inclusion)	108	6
Обход пути / включение локального файла (Path traversal / LFI - Local File Inclusion)	816	8

Б. Выбор исследуемых сканеров

Исследование проведено в отношении коммерческих сканеров и сканеров с открытым исходным кодом, сведения о которых представлены в таблице 4 (коммерческие сканеры выделены курсивом).

Таблица 4. *Общие характеристике исследуемых сканеров*

Сканер	Компания / разработчик	Версия	Лицензия	Технология
BurpSuite	PortSwiger	1.6.12	Commercial / Free (Limited Capability)	Java
Acunetix	Acunetix	10	Commercial / Free (Limited Capability)	Perl
Wapiti	InformaticaGesfor	2.3.0	Open Source	Python
SkipFish	Google	2.10	Open Source	C
Netsparker	Mavituna Security	2.3	Commercial	.Net
W3AF	W3af devel	1.2	Open Source	Python
AppSpider	Rapid 7	6.0	Commercial	Java
IronWASP	L. Kuppan	0.9.7.1	Open Source	.Net
Arachni	Tasos Laskos	2.2.1	Commercial	Ruby
ZAP	OWASP	2.3.1	Open Source	.Net
Vega	Subgraph	1.0(beta)	Open Source	Java

В. Выбор показателей эффективности сканеров

Исследование проводилось с использованием следующих показателей эффективности сканеров [25, 26].

А) Точность, как отношение правильно обнаруженных уязвимостей к общему числу тестовых уязвимостей:

$$\text{Точность} = \frac{\text{ИП}}{\text{ИП} + \text{ЛП}}$$

Чем выше точность, тем меньше количество ложноположительных результатов, следовательно сканер более точен в обнаружении уязвимостей.

Б) Чувствительность, как отношение правильно обнаруженных уязвимостей к общему числу количеству обнаруженных уязвимостей:

$$\text{Чувствительнось} = \frac{\text{ИП}}{\text{ИП} + \text{ЛО}}$$

Чем больше чувствительность, тем меньше ложноотрицательных результатов. Следовательно, инструмент лучше обнаруживает уязвимости.

В) F-мера, которая представляет среднее гармоническое значение точности и чувствительности, определяемая формулой:

$$F\text{-мера} = \frac{2 \times \text{Точность} \times \text{Чувствительность}}{\text{Точность} + \text{Чувствительность}}$$

Эти три показателя могут быть использованы для установления рейтинга сканеров в соответствии с целями пользователя бенчмарка.

Результаты оценки сканеров веб-приложений

Сканеры исследовались с использованием набора тестов WAVSEP.

В таблицах 5, 6 приведены результаты тестирований. Выделенные жирным шрифтом значения указывают на общее количество уязвимостей в WAVSEP, упомянутых в таблице 5. Выделенные желтым цветом значения указывают на уровень обнаружения, в то время как остальные - количество найденных ИП, ЛП, ЛО.

Таблица 5. *Результаты тестов WAVSEP для SQLI и XSS уязвимостей по значениям ИП, ЛП, ЛО*

Сканеры	Результаты тестирования					
	SQLI			XSS		
	ИП	ЛО	ЛП	ИП	ЛО	ЛП
	136		**10**	**66**		**7**
BurpSuite	136	0	3	62	4	0
	100%		30%	93,93%		0%
Wapiti	136	0	2	44	22	3
	100%		20%	66,66%		42,85%
Acunetix	136	0	0	66	0	0
	100%		0%	100%		0%
SkipFish	102	34	0	65	1	0
	75%		0%	98,48%		0%
Netsparker	136	0	3	64	2	0
	100%		30%	96,96%		0%
W3AF	81	55	3	19	47	3
	59,55%		30%	28,78%		42,85%
AppSpider	132	4	0	66	0	0
	97,05%		0%	100%		0%
IronWASP	136	0	5	52	14	0
	100%		50%	78,78%		0%
Arachni	136	0	5	63	3	0
	100%		50%	95,45%		0%
ZAP	136	0	0	63	3	0
	100%		0%	95,45%		0%
Vega	136	0	2	66	0	0
	100%		20%	100%		0%

Таблица 6. *Результаты тестов WAVSEP для RFI и LFI уязвимостей по значениям ИП, ЛП, ЛО*

Сканеры	Результаты тестирования					
	RFI			LFI		
	ИП	ЛО	ЛП	ИП	ЛО	ЛП
	108		**6**	**816**		**8**
BurpSuite	*80*	*28*	0	496	320	1
	74,07%		0%	60,78%		*12,5%*
Wapiti	64	44	0	414	402	1
	59,25%		0%	50,73%		12,5%
Acunetix	*87*	*21*	0	292	524	0
	80,55%		0%	*35,78%*		*0%*
SkipFish	39	69	1	312	504	2
	36,11%		16,66%	38,23%		25%
Netsparker	*57*	*51*	0	467	349	0
	52,77%		0%	*57,23%*		*0%*
W3AF	13	95	1	461	355	1
	12,03%		16,66%	56,49%		12,5%
AppSpider	*80*	*28*	0	660	156	1
	74,07%		0%	*80,88%*		*12,5%*

IronWASP	106	2	0	288	528	1
	98,14%		0%	35,29%		12,5%
Arachni	46	62	0	162	654	0
	42,59%		0%	*19,85%*		*0%*
ZAP	136	0	1	590	226	0
	100%		16,66%	72,30%		0%
Vega	108	0	0	519	297	5
	100%		0%	63,60%		62,5%

В таблицах 7-10 представлены результаты исследования сканеров по критериям точности, чувствительности и F-меры.

Таблица 7. *Результаты тестов WAVSEP по значениям точности, чувствительности и F-меры для уязвимости SQLI*

Сканеры	**Результаты тестирования**		
	SQLI		
	Точность, %	**Чувствительность, %**	**F-мера, %**
BurpSuite	97,84	100	98,90
Wapiti	98,55	100	99,26
Acunetix	100	100	100
SkipFish	100	100	100
Netsparker	97,84	100	98,90
W3AF	96,42	59,55	73,62
AppSpider	100	97,05	98,50
IronWASP	96,45	100	98,19
Arachni	96,45	100	98,19
ZAP	100	100	100
Vega	98,55	100	99,26

Таблица 8. *Результаты тестов WAVSEP по значениям точности, чувствительности и F-меры для уязвимости XSS*

Сканеры	**Результаты тестирования**		
	XSS		
	Точность, %	**Чувствительность, %**	**F-мера, %**
BurpSuite	*100*	*93,93*	*96,87*
Wapiti	93,61	66,66	77,86
Acunetix	*100*	*100*	*100*
SkipFish	100	100	100
Netsparker	*100*	*96,96*	*98,45*
W3AF	86,36	28,78	43,17
AppSpider	*100*	*100*	*100*
IronWASP	100	78,78	88,13
Arachni	*100*	*95,45*	*97,67*
ZAP	100	95,45	97,67
Vega	100	100	100

Таблица 9. *Результаты тестов WAVSEP по значениям точности, чувствительности и F-меры для уязвимости RFI*

Сканеры	Результаты тестирования		
	RFI		
	Точность, %	Чувствительность, %	F-мера, %
BurpSuite	*100*	*74,07*	*85,10*
Wapiti	100	59,25	74,41
Acunetix	*100*	*80,55*	*89,22*
SkipFish	100	80,55	89,22
Netsparker	*100*	*52,77*	*69,08*
W3AF	92,85	12,03	21,30
AppSpider	*100*	*74,07*	*85,10*
IronWASP	100	98,14	99,06
Arachni	*100*	*42,59*	*59,73*
ZAP	99,08	100	99,53
Vega	100	100	100

Таблица 10. *Результаты тестов WAVSEP по значениям точности, чувствительности и F-меры для уязвимости LFI*

Сканеры	Результаты тестирования		
	LFI		
	Точность, %	Чувствительность, %	F-мера, %
BurpSuite	99,79	60,78	75,54
Wapiti	99,75	50,73	67,25
Acunetix	100	35,78	52,70
SkipFish	100	35,78	52,70
Netsparker	100	57,23	72,79
W3AF	99,78	56,49	72,13
AppSpider	99,84	80,88	89,36
IronWASP	99,65	35,29	52,12
Arachni	100	19,85	33,12
ZAP	100	72,30	83,92
Vega	99,04	63,60	77,45

Г. Обобщение результатов тестирования

Сводные графики значений точности, чувствительности и F-меры для исследованных сканеров и уязвимостей приведены на рис. 6-9.

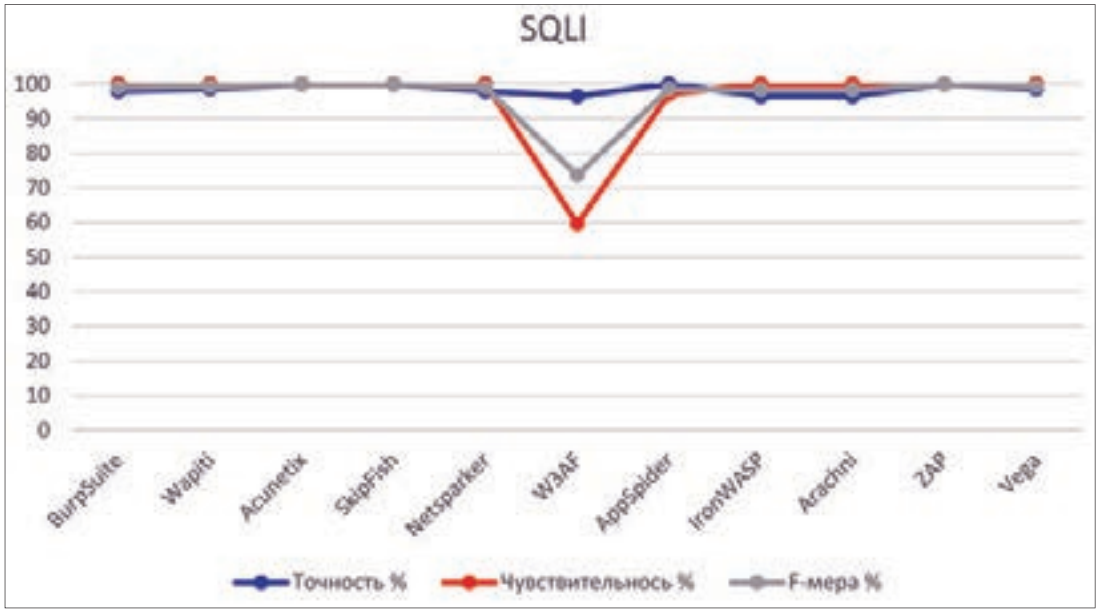

Рисунок 7. *Показатели точности, чувствительности и F-меры уязвимости SQLI в WAVSEP*

Как видно, в отношении уязвимости SQLI сканеры Acunetix, SkipFish, ZAP Acunetix, SkipFish, ZAP имеют самые высокие значения F-меры (100%). Сканеры BurpSuite, Netsparker, AppSpider, IronWASP и Arachni показали значения F-меры, равные 98%, в то время как у сканера W3AF значение F-меры составило73, 62%.

Таким образом, помимо сканера W3AF сканеры с открытым исходным кодом и коммерческие варианты хорошо работают в отношении уязвимости SQLI.

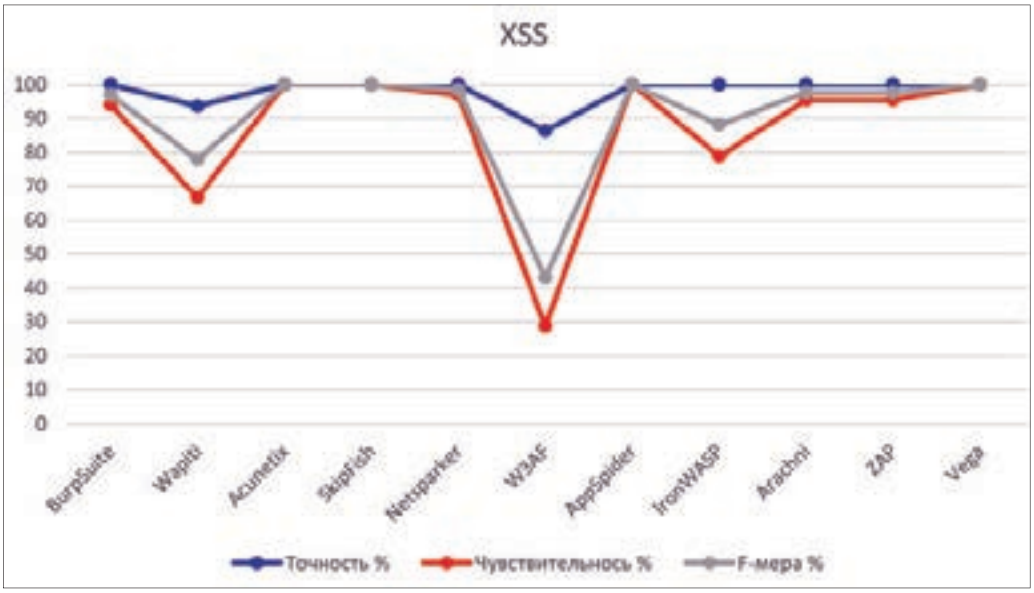

Рисунок 8. *Показатели точности, чувствительности и F-меры уязвимости XSS в WAVSEP*

В отношении уязвимости XSS значение F-меры у сканеров Acunetix, Skipfish, Appspider и Vega составило 100%. Близкие значения показывают сканеры Netsparker, Arachni, Zap, Burp suite и IronWASP. А вот эффективность сканеров Wapiti и W3AF по всем показателям значительно ниже.

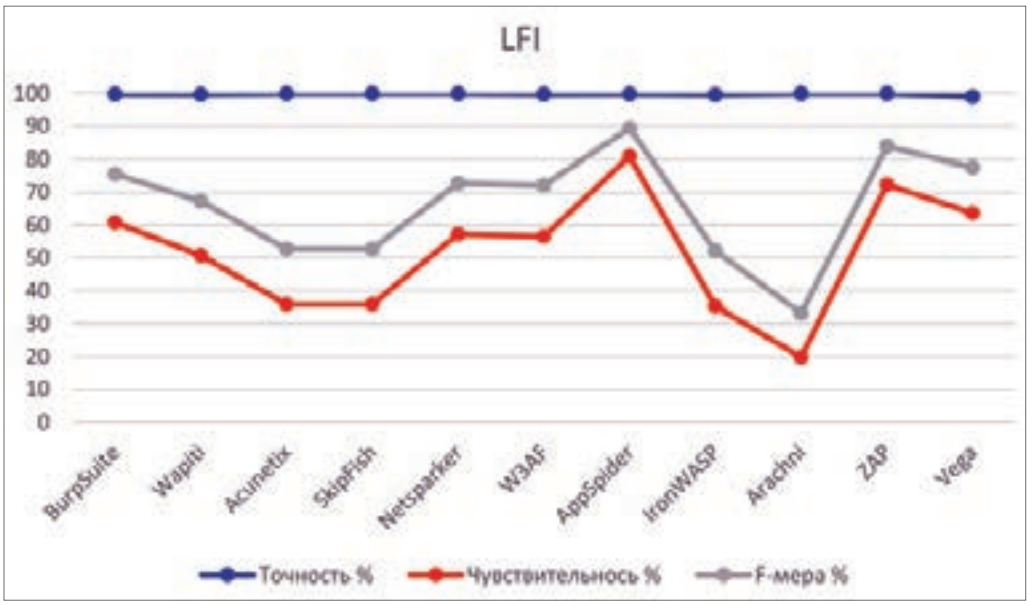

Рисунок 9. *Показатели точности, чувствительности и F-меры уязвимости LFI в WAVSEP*

В отношении уязвимости LFI наилучшие результаты получены сканерами AppSpider и ZAP. При этом ни один сканер не показал 100% эффективности по показателям чувствительности и F-меры.

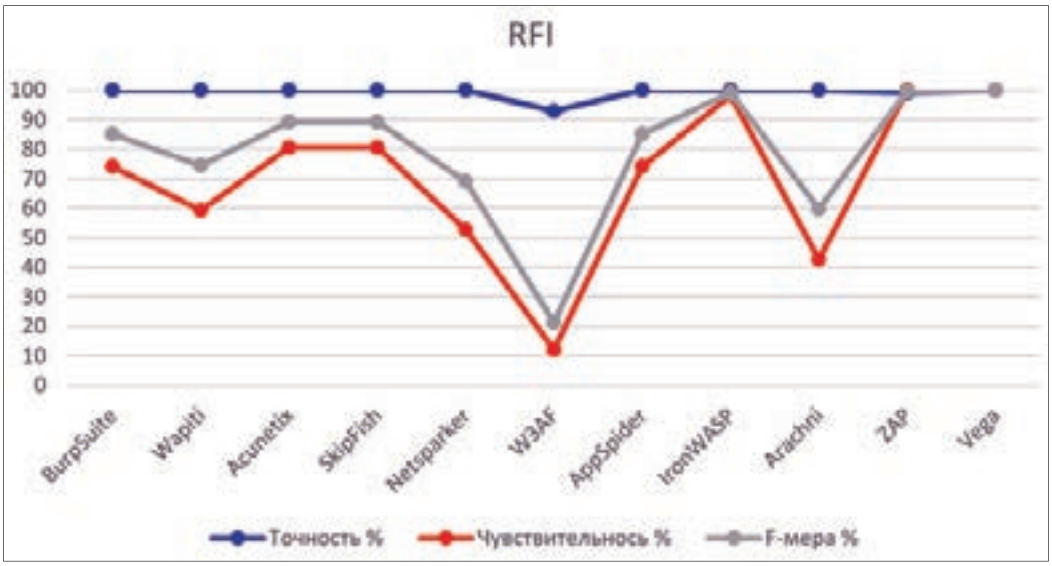

Рисунок 10. *Показатели точности, чувствительности и F-меры к RFI уязвимости в WAVSEP*

В отношении уязвимости RFI 100% результаты получены сканерами IronWASP, Zap и Vega с открытым исходным кодом. Ни один коммерческий сканер 100% эффективности не показал. Самые низкие результаты получены сканером W3AF.

Заключение

Эффективность исследованных сканеров уязвимостей веб-приложений зависит от видов уязвимостей. При этом эффективность всех сканеров в отношении уязвимостей SQLI и XSS выше, чем в отношении уязвимостей LFI и RFI.

В отношении уязвимостей LFI и RFI сканеры Vega и ZAP, являющиеся сканерами с открытым исходным кодом, показали наилучшие результаты, опередив все коммерческие сканеры.

Наличие открытого исходного кода сканеров, показывающих высокую эффективность (в отношении некоторых видов уязвимостей 100%) позволяет сделать вывод о имеющейся возможности создать хороший сканер уязвимостей веб-приложений, покрывающий виды уязвимостей, характерные именно для ИТС.

References / Список использованных источников

[1] A European strategy on Cooperative Intelligent Transport Systems, a milestone towards cooperative, connected and automated mobility, European Commission, Brussels, 30.11.2016, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52016DC0766&from=EN (доступ 21.10.22).

[2] Транспортная стратегия Российской Федерации до 2030 года с прогнозом на период до 2035 года (утв. распоряжением Правительства РФ от 27.11.2021 г. № 3363-р), http://www.consultant.ru/document/cons_doc_LAW_402052/d01744672f881dc4a07365eacfb4a9e5b4f6783f/ (доступ 21.10.22).

[3] Critical Infrastructure. Threat Information Sharing Framework. A Reference Guide for the Critical Infrastructure Community. USA Homeland Security, October 2016, https://www.cisa.gov/sites/default/files/publications/ci-threat-information-sharing-framework-508.pdf (доступ 21.10.22).

[4] Федеральный закон «О безопасности критической информационной инфраструктуры Российской Федерации» от 26.07.2017 № 187-ФЗ, http://www.consultant.ru/document/cons_doc_LAW_220885/ (доступ 21.10.22).

[5] I. F. Mikhalevich (2020), "Metods Ensuring the Secure of Software for Intelligent Transport Systems", 2020 Systems of Signals Generating and Processing in the Field of on Board Communications, https://ieeexplore.ieee.org/document/9078647 (доступ 21.10.22).

[6] Mikhalevich I.F. (2018), "Methodological foundations of creation of national protected hardware-software platforms for critical information infrastructures", T-Comm, vol. 12, no 3, pp. 75-81, https://cyberleninka.ru/article/n/metodologicheskie-osnovy-sozdaniya-natsionalnyh-zaschischennyh-apparatno-programmnyh-platform-dlya-kriticheskih-informatsionnyh/viewer (доступ 21.10.22).

[7] Web Security Glossary The WASC, http://www.webappsec.org/projects/glossary/ (доступ 21.10.22).

[8] ENISA, Cyber Security and Resilience of Intelligent Public Transport. Good practices and recommendations", January 12, 2016, https://www.enisa.europa.eu/publications/good-practices-recommendations (доступ 21.10.22).

[9] Threat Classification The WASC, http://projects.webappsec.org/w/page/13246978/Threat%20Classification (доступ 21.10.22).

[10] R. Jnena, 2013, "Modern Approach for WEB Applications Vulnerability Analysis", https://library.iugaza.edu.ps/thesis/109553.pdf (доступ 21.10.22).

[11] OWASP Ten Most Critical Web Application Security Vulnerabilities", https://owasp.org/Top10/ (доступ 21.10.22).

[12] A. Rajan and E. Erturk, 2016, "Web Vulnerability Scanners:

A Case Study", https://arxiv.org/abs/1706.08017 (доступ 21.10.22).

[13] Black, P. E., Fong, E., Okun, V., & Gaucher, R. National Institute of Standards and Technology (NIST). "Software Assurance Tools: Web Application Security Scanner Functional Specification", https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-269.pdf (доступ 21.10.22).

[14] F. Kagorora, J. Li, D. Hanyurwimfura, and L.Camara, 2015, "Effectiveness of Web Application Security Scanners at Detecting Vulnerabilities behind AJAX / JSON", https://www.researchgate.net/profile/Lancine-Camara/publication/309740460_Effectiveness_of_Web_Application_Security_Scanners_at_Detecting_Vulnerabilities_behind_AJAXJSON/links/5b1cf06fa6fdcca67b68b43d/Effectiveness-of-Web-Application-Security-Scanners-at-Detecting-Vulnerabilities-behind-AJAX-JSON.pdf (доступ 21.10.22).

[15] A. Doupe, M. Cova, and G. Vigna, 2010 "Why Johnny Can't Pentest: An Analysis of Black- Web Vulnerability Scanners," Proc. 7th Int. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 111-131, https://link.springer.com/chapter/10.1007/978-3-642-14215-4_7 (доступ 21.10.22).

[16] J. Bau,, E. Bursztein, D. Gupta, and J. Mitchell, 2010, "State of the Art: Automated Black-Box Web Vulnerability Testing" in Proc. 2010 IEEE Symposium on Security and Privacy, pp. 32-345, https://ieeexplore.ieee.org/abstract/document/5504795 (доступ 21.10.22).

[17] A. M. Ferreira, and H. Kleppe, "Effectiveness of Automated Application Penetration Testing Tools", Master dissertation., Master Education SNE/OS3, University of Amsterdam, Netherlands, https://www.os3.nl/_media/2010-2011/courses/rp1/p27_presentation.pdf (доступ 21.10.22).

[18] Fakhreldeen A. and Eltyeb E., 2014, "Assessment of Open-Source Web Application Security Scanners", College of Computer Science and Information Technology, KAU, Khulais, Saudi Arabia, https://www.researchgate.net/profile/Fakhreldeen-Saeed/publication/283353796_ASSESSMENT_OF_OPEN_SOURCE_WEB_APPLICATION_SECURITY_SCANNERS/links/58e0da5fa6fdcc41bf941a6f/ASSESSMENT-OF-OPEN-SOURCE-WEB-APPLICATION-SECURITY-SCANNERS.pdf (доступ 21.10.22).

[19] L. Suto, 2010, "Analyzing the Accuracy and Time Costs of Web Application Security Scanners", http://www.think-secure.nl/uk/Accuracy_and_Time_Costs_of_Web_App_Scanners.pdf (доступ 21.10.22).

[20] S. Alassmi, P. Zavarsky, D. Lindskog, R. Ruhl, A.Alasiri, and M. Alzaidi,,2012 "An analysis of the Effectiveness of Black-box Web Application Scanners in Detection of Stored XSSI Vulnerabilities", International Journal of Information Technology and Computer Science, vol. 4, No. 1, https://www.researchgate.net/profile/Pavol_Zavarsky/publication/259647343_An_Analysis_of_the_Effectiveness_of_Black-Box_Web_Application_Scanners_in_Detection_of_Stored_XSS_Vulnerabilities/links/58d2bd17aca2723c0a7773de/An-Analysis-of-the-Effectiveness-of-Black-Box-Web-Application-Scanners-in-Detection-of-Stored-XSS-Vulnerabilities.pdf (доступ 21.10.22).

[21] Yuliana M., 2012, "Security Evaluation of Web Application Vulnerability Scanners' Strengths and Limitations Using Custom Web Application", California State University, http://mcs.csueastbay.edu/~lertaul/YulianaThesis_V8.pdf (доступ 21.10.22).

[22] Сердечный А.Л. «Методика выявления уязвимостей и недекларированных возможностей в программном обеспечении», http://new.groteck.ru/images/catalog/70840/e75c72a254fcc880fa65657fdb144063.pdf (доступ 21.10.22).

[23] The Web Application Vulnerability Scanner Evaluation Project, https://code.google.com/archive/p/wavsep/ (доступ 21.10.22).

[24] Y. Makino and V. Klyuev, 2015, "Evaluation of web vulnerability scanners", Proc. 2015 IEEE 8th Int. Conf. Intell. Data Acquis. Adv. Comput. Syst. Technol. Appl. IDAACS 2015, vol. 1, no. September, pp. 399-402, https://ieeexplore.ieee.org/abstract/document/7340766 (доступ 21.10.22).

[25] D. M. W. POWERS, 2011, "Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation", J. Mach. Learn. Technol., vol. 2, no. 1, pp. 37-63, https://arxiv.org/abs/2010.16061 (доступ 21.10.22).

[26] N. Antunes and M. Vieira, 2010, "Benchmarking vulnerability detection tools for web services", ICWS 2010 - 2010 IEEE 8th Int. Conf. Web Serv., pp. 203210, https://ieeexplore.ieee.org/abstract/document/5552783 (доступ 21.10.22).

Received: April 19, 2022 / Получено: 19 апреля 2022 г.
Accepted: October 17, 2022 / Принято: 17 октября 2022 г.

Сведения об авторах

Михалевич Игорь Феодосьевич – доцент кафедры «Управление и защита информации» Российского университета транспорта (МИИТ)
ученая степень: кандидат технических наук
ученое звание: старший научный сотрудник
e-mail: mif-orel@mail.ru

Федоренко Богдан Николаевич – студент кафедры «Управление и защита информации» Российского университета транспорта (МИИТ)

Шеламов Максим Дмитриевич – студент кафедры «Управление и защита информации» Российского университета транспорта (МИИТ)

For citation

Mikhalevich Igor Feodosevich, Fedorenko Bogdan Nikolaevich, Shelamov Maxim Dmitrievich, Research of Vulnerability Scanners of Web Applications of Intelligent Transport Systems, *JITA – Journal of Information Technology and Applications, Banja Luka*, Pan-Europien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 12(2022) 2:127-151, (UDC: 621.86/.87:65.011.56), (DOI: 10.7251/JIT2202127F), Volume 12, Number 2, Banja Luka, December (65-172), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004