ISSN 2277-3061

Security of Web Services: Methods and Contrivance

Malinka Ivanova
Technical University of Sofia, Sofia, Bulgaria

ABSTRACT

The increasing use of web services, the proved advantages of service-oriented architectures and continuously applied attacks to them require utilization of given secure mechanisms that ensure the security at different levels. The aim of the paper is to summarize the existing threats and attacks to web applications and web services. Contemporary security standards and good practices describing methods and contrivance for deciding security problems are explored too to reveal the present state in the field.

Indexing terms/Keywords

Web services; Security; Threats and attacks; SOAP; RESTful, WS-Security Framework

Academic Discipline And Sub-Disciplines

Informatics; Web Application security

SUBJECT CLASSIFICATION

Web services security

TYPE (METHOD/APPROACH)

Survey

Council for Innovative Research

Peer Review Research Publishing System

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Vol. 14, No. 11

www.ijctonline.com, editorijctonline@gmail.com

ISSN 2277-3061

INTRODUCTION

Development and usage of web applications and services is a sector in continuous growth with focus on social, financial, enterprise, mapping, ecommerce, government, science, messaging, payment and other fields. BBC reports on increase in demand for Amazon web services in the field of cloud computing, hosting, searching, analysis, storing, etc. [1]. Public available web services are listed in registers or directly on the providers' web sites like: XMethods [2], Membrane directory for public SOAP web services [3], Workday SOAP Web Services Directory [4], USGS site for SOAP and REST web services [5], etc. This wide use of web services via Internet and realization of service-oriented architectures in organizations requires adhering to secure issues ensuring their correct utilization and the achievement of the right effects to the end users. So, the design and development of secure web services is related to deciding a set of problems – several of them are "standard" and well described in the literature and security specifications/standards, others are unique for the concrete architecture and solution. Building secure web services requires observance of a number of steps: (1) identification and description of the security objectives; (2) understanding the use of web service in particular scenarios and context and related to that possible threats, attacks and vulnerabilities; (3) studying and applying the best practices, proven principles and patterns avoiding risks factors in security implementation; (4) choosing effective security engineering process based on specific activities ensuring the realization of the security objectives [6].

Web services are defined as software systems that perform interoperable machine-to-machine communication through network [7]. The main characteristics of web services, ensuring interoperability are described via Web Services Architecture (WSA) [7]. The WSA consists of four models representing through concepts different aspects of web services as it is shown in Table 1. WSA identifies three concepts revealing the security issues of web services: resources (anything that has an URI; web services are a kind of resource), mechanisms for securing the web services and policies (documents that describe resource constrains and that are prepared for machine processing). Also, the WSA summarizes the security requirements that should be taken into considerations at the web services implementation (Figure 1).

Table 1. Main characteristics of web services via models

Model	Focus	Concepts	Security concepts
Service oriented	Action	• Service is realized by agent 1 • Service is used by agent 2 • Messaging between agent 1 and agent 2 • Usage of meta-data	
Message oriented	Message	• Messages • Message structure • Message transport	• Message Layer Security
Resource oriented	Resource	• Representation • URI • Ownership by organization or person	• Resources security
Policy	Policy	• Policy about resource • Policy about action • Policy applied to agents • Established by organization or person	• Security mechanisms • Policies

ISSN 2277-3061

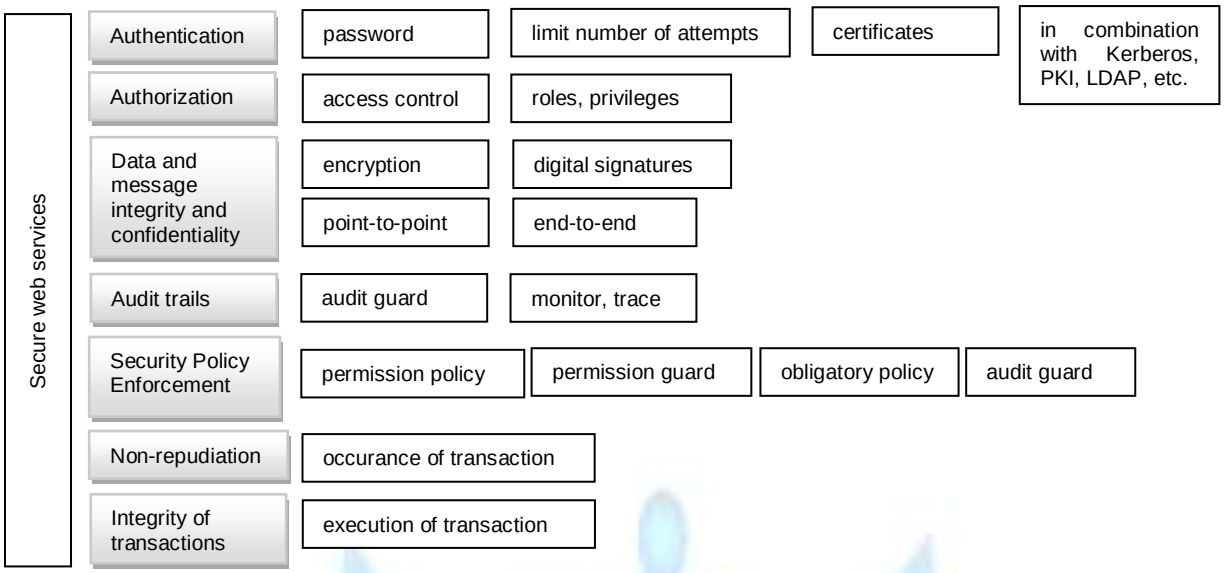

Fig 1: Security requirements to web services

Interface design is the main criterion that forms two groups of web services: (1) SOAP (Simple Object Access Protocol)- and WSDL (Web Service Definition Language)-based web services and (2) RESTful (REST - Representational State Transfer) web services. The differences in these two groups of web services lead to several specific characteristics concerning security issues.

The aim of the paper is to summarize the current state related to web services' security covering common threats and attacks at application level as well as to identify specific security points typical for SOAP-based and RESTful web services. It includes an examination of existing security specifications/standards and exploration of good practices in realization of security web services.

Web Services Vulnerabilities, Threats and Attacks

Vulnerabilities, threats and attacks to web services include all activities of the attacker against network infrastructure, web applications and host computers as well as all activities of designers/developers (during design/development process of web applications/web services) and administrators (during administration). Threats and attacks at application level are organized in several categories for their better understanding and analysis. Two classifications according to Microsoft Corporation [8] and Web Application Security Consortium [9] are presented in Table 2.

Table 2. Threats and attacks against web applications

Attack category	Microsoft Corporation [8]	Web Application Security Consortium [9]
Authentication	network eavesdropping brute force attacks dictionary attacks cookie replay credential theft	brute force insufficient authentication weak password recovery validation
Authorization	elevation of privilege disclosure of confidential data data tampering luring attacks	credential/session prediction insufficient authorization insufficient session expiration session fixation
Input validation	buffer overflow cross-site scripting SQL injection canonicalization	
Session management	session hijacking session replay man in the middle	

Parameter manipulation	cookie manipulation form field manipulation HTTP header manipulation	
Cryptography	poor key generation or key management weak encryption	
Sensitive data	network eavesdropping data tampering	
Configuration management	unauthorized access to administration interfaces unauthorized access to configuration stores retrieval of clear text configuration data	
Exception management	information disclosure denial of service	
Auditing and logging	user denies performing an operation attacker exploits an application without trace attacker covers his tracks	
Client-side Attacks		content spoofing cross-site scripting
Command Execution		buffer overflow format string attack LDAP Injection OS Commanding SQL Injection SSI Injection XPath Injection
Information Disclosure		directory indexing information leakage path traversal predictable resource location
Logical Attacks		abuse of functionality denial of service insufficient anti-automation insufficient process validation

Anyway, according to a statistical analysis, published at the web site of Web Application Security Consortium, the most applied vulnerabilities and attacks to web applications are: Cross-Site Scripting, Information Leakage, SQL Injection, Insufficient Transport Layer Protection, Fingerprinting and HTTP Response Splitting [10]. The experts reveal that the reasons for Cross-Site Scripting, SQL Injection and HTTP Response Splitting vulnerabilities are obligated to designers and developers of web services and the rest vulnerabilities are mistakes performed by administrators. Another statistic analysis of WhiteHat Security company reports that the most common vulnerabilities and attacks typical for web applications are Cross-Site Scripting, Information Leakage, Content Spoofing, SQL Injection, Cross-Site Request Forgery, Insufficient Transport Layer Protection, Abuse of Functionality, HTTP Response Splitting, Predictable Resource Location, Brute Force [11]. The analysis shows a dependence between the appeared vulnerabilities and used technologies for realization of web applications. The contemporary picture explaining the most utilized threats and attacks to web applications and web services are presented on Figure 2.

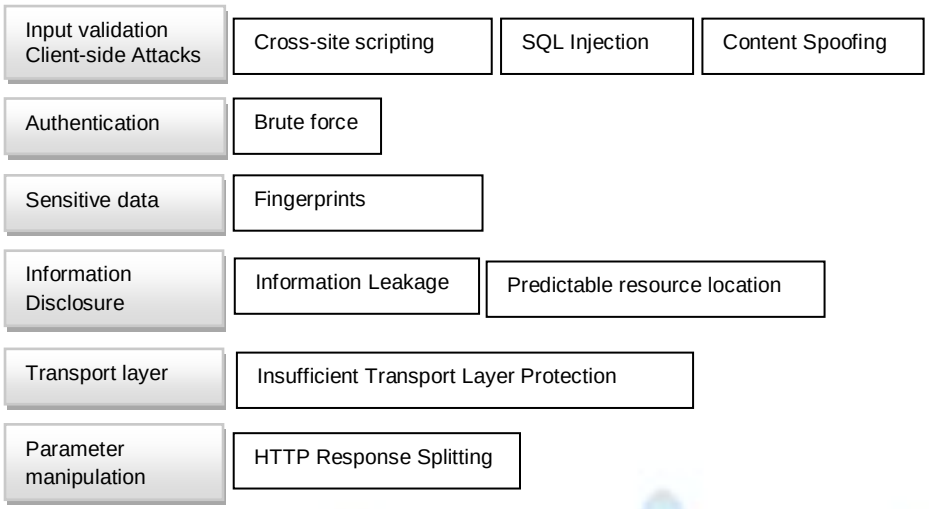

Fig.2: Contemporary threats and attacks

In this section many of the common threats and attacks to web applications and web services are described and solutions for different security problems proposed by the scientific society are discussed.

- *Cross-site scripting (XSS)*

An attacker could put executable code (for example JavaScript) in a data stream and the victim's web browser will execute it. Then the attacker will take private information and privileges. Two types of XSS attacks are defined: reflected (non-persistent) and stored (persistent) [12]. In the reflected XSS attack, the attacker prepares a special link that will direct the victim to the vulnerable web server. Clicking on the link, the executable code is ran in the victim's web browser. In the stored attack, the executable code is embedded in an HTML page or other data that are stored on the vulnerable web server. Then everybody who opens this contagious web page will run the executable code in the browser. Experts suggest several preventive measures to be taken: blocking the untrusted JavaScript code, using available possibilities in a browser for protection, disallowing external web sites to request internal resources, updating the used software and using a protective system against malware [12]. Kirda et al. have developed a client-side application Noxes to prevent XSS attacks [13]. It is built in a form of web firewall that is ran as background service and that alarms the victim for possible attacks. A survey describing existing vulnerable XSS web applications and current solutions divided in four categories (according to application location, type of analysis, method for detection and method for prevention) is proposed by Shanmugam and Ponnavaikko [14].

- *SQL Injection*

This is one of the most common method for attacking a web service. Malicious SQL code is placed in parameter field and is sent to the server that will execute it. SQL statement is created or modified in a false way by the attacker and in this way he will have access to the database and information revealing given privileges. Laranjeiro et al. propose a solution to secure web services against SQL and XPath injection attacks [15]. The solution consists of three phases: service assessment, statement learning and service protection. The performed experiment shows 100% effectiveness against all SQL and XPath injection attacks. Another approach for prevention against SQL injection attack is presented in [16]. It is called SQLDOM4J and it is created for Java online applications. SQLDOM4J builds and executes SQL queries protecting web applications and services. The authors recommend eight techniques for prevention.

- *Buffer Overflows*

The attacker writes more data in one or several fields causing buffer overflows and memory overfilling. This can lead to crash of the service, software or system. This attack allows gaining access to unauthorized information or unauthorized execution of processes Two types of buffer overflow are most common: Stack-Based Overflow and Heap-Based Overflow [17].

- *Brute Force Attack*

Every web application that requires user authentication through username and password could be a target of brute force attack. The attacker tries different combinations of values to guess the password and to gain access to the application/system. This attack is not hard for detection, but it is difficult for prevention and protection. Login attempts with different passwords for a short time should be limited and after a given number of attempts should be blocked. The password should be strong with different character types. Usage of encryption algorithms will decrease the attacker possibilities for reaching needed data. Adams et al. divide the brute force attacks in four types: "targeted attack" - the attacker tries to guess the user password and the success depends on the password strength; "trawling attack" - the attacker uses combination of characters for a password and tries it to find a user account with this password; "blind" attack – the attacker is looking for both in parallel: username and password applying a heuristic strategy or just applies a strategy of methodically attempts; "knowledge question" – it is a strategy that proposes a user possibility to answer a secure question when he has forgot his username or password [18]. The authors propose a security mechanism based of three

different ideas: decoupling protection and authentication activities; recognition of attacks directions and defense of the correct directions, not at all simultaneously; creation of a sliding window along each direction – when the number of attempts in this window reaches a given value then a defence measure will be taken.

- *Dictionary Attack*

Through a dictionary attack the attacker utilizes a set of methods for gaining access to a target system. He uses all strings in a list like all words in a dictionary. So, this attack is more focused than brute force attack and it could be done in shorter time. It includes encryption of a cryptograph algorithm and breaking an authentication mechanism, receiving the cipher key or passwords/usernames. Pinkas and Sander talk about RTT (Reverse Turing Test) protocol for securing passwords from a Dictionary attack [19]. They discuss its scalability, usability and security issues and show that the RTT authentication scheme is easy for implementation even for service providers plus the fact that it is not very costly. A modification of the RTT protocol called history-based protocol with RTTs is presented by Stubblebine and Oorschot [20]. The research team argues that this new protocol improves security against Dictionary attack, usability and parameter flexibility. Anyway, they suggest that this protocol is not enough for securing web applications – it needs a combination of RTT and additional techniques like: notifications of failed logins, solving the problem of consuming client resources by clients and returning verification of a login, RTT with an embedded warning instructing the user to not give an answer to RTT when the match is failed.

- *Cookie replay attacks*

The attacker uses a previously valid cookie deluding the server that this session is still in progress and active. He hijacks a session token and utilizes it to replay to application. Then attacker gains access to the application under a false identity. The Cookie replay attack could be prevented by the following countermeasures: ensuring of encrypted communication SSL channel at transferring of authentication cookies [21], usage of session timeout that will decrease the time interval for reply attacks performed by the attacker, through technique token regeneration [22].

- *Network Eavesdropping*

The attacker monitors the network packets, reads information searching for sensitive data (clear text passwords) or configuration information. This attack is known also under name sniffing [23]. The important information that is traveling via a network could be protected through: (1) encryption on transport level using SSL or IPSec in the case when the administrator controls sender and receiver points; (2) encryption of messages transferring among intermediary network points. Adida proposes a method SessionLock to ensure the security of web sessions against the network eavesdropping attack without the utilization of SSL [24]. A session secret is generated for usage by browsers and an authentication code for every HTTP request is created. Then the connection between HTTPS and HTTP is protected.

- *Credential Theft*

The attacker has the possibility to steal stored/cashed usernames and passwords – information that is required for login. Preventing countermeasures could be: usage of a strong password, block many attempts at login, verification of stored password, available option for user to choose or not the cashing of credentials [25].

- *Data tampering*

Data tampering is related to unauthorized modification of data on the network and in the storage. Prevention could be realized through: (1) ensuring strong control over access to data storage – just for authorized users; (2) ensuring of different privileges for users who can view and modify data. Pavlou and Snodgrass propose a method for detection of data tampering using cryptographic hash functions [26]. Forensic analysis is performed revealing when, how and by who the data tampering is done.

- *Elevation of privilege*

The attacker with low level privileges elevates his privileges at higher level taking control over accounts, processes and services. It is talking about vertical elevation – elevation from lower privileges to higher ones and horizontal elevation – gaining access to accounts with similar privileges. Countermeasures consists of returning the last privileged state, including accounts, processes and services [27].

- *Disclosure of confidential data*

Confidential data (identity, sensitive data, configuration data, system data) could be exposed in unwilling form and seen by unprivileged entities. Prevention includes ensuring secure storage and a secure transmission channel via the network. Software developers should remove any bugs leading to the disclosure of confidential data. One solution ensuring trusted middleware is called SaveWeb and is proposed by Hosek and al. [28]. Security labels are associated with transmitted and stored data and their statement is transparently tracked.

- *Man-in-the-middle*

The attacker intercepts a communication channel between client and server and interacts between them. He reads and probably modifies data with aim to perform other kind of attacks. Eriksson demonstrates in his research an easy way for creation of tool for performing a man-in-the-middle attack against server authenticated SSL sessions just through usage of existing programming libraries [29]. He suggests that the defence against such attacks includes: two-way trust relationship

between client and server during communication period and user's intentions to be signed. Another author Gangan in his review concludes that the man-in-the-middle-attack cannot be completely removed, but several security measures could be taken like: network software update, creation of secure network design, continues update of computer software [30].

- *Cookie manipulation*

Cookies store information about user preferences and about the realized session. Also, the cookies could be modified at the client side by the attacker and then they could be sent to a server. It helps the attacker to authenticate his false identity accessing the web service or web site. Several mitigation techniques are presented in [31]: user authentication through a session token, evaluation of cookies for false combination of values that identify tampering, cookies encryption to prevent tampering.

- *Denial of service (DoS)*

This is a process for network flooding, creating a high volume of traffic, then all network resources are consumed and web application is unavailable for use. The final goal is disruption of service. Distributed Denial of service attack (DDoS) is based on collaborative work of many computers for coordination of one DoS attack against one or more targets. DDoS attack addressing a web site is performed by flooding web servers with so many requests and as a result the web site is getting unavailable for use. For this attack there is no sufficient protection. Kargl et al. describe DoS and DDoS attacks classification, grouping them in three categories: "system attacked" – the target of the attacker is a system – this could be a web server clients, routers, a firewall system, a load-balancer, individual web servers, database servers; "part of the system attacked" – hardware of a given system, operating system, TCP/IP flow, router, web server, database server, scripts; "bug or overload" – DoS could be caused from an application bug or overload of components [32]. The authors propose a solution applying strategies related to the routing, based on classes and active monitoring of traffic.

- *Spoofing*

The attacker could use a false identity through stolen credentials of a user or false IP address to receive the system access. A misuse pattern of Spoofing web services is presented in [33]. This pattern explains how a false identity can be created, how the attack is started and how it can be stopped.

A testing tool WS-Attacker against specific attacks like WS-Addressing spoofing and SOAPAction spoofing is presented in [34]. The tool is evaluated utilizing four frameworks for deployment of web services: Apache Axis2, JBossWS native, JBossWS CXF and .NET Web Services and the results are promising.

Another attack is visual spoofing – in this case the secure connection of a browser is replaced with a fake connection created by attacker. The victim who thinks that he is using a secure web site and services sends private information directly to the attacker. The visual spoofing attack is a typical threat to secure web services related to single sign on services, online banking and shopping [35]. An additional attack to visual spoofing is the so called mounting attack – the victim's browser is directed to a web site that hosts the visual spoofing attack. The proposed effective countermeasures against the visual spoofing attack include: training the victim to increase his security vigilance and technical measure including several issues – personalization at authentication of browser secure connection indicators, using additional application that reports information from trustfully source (like IP address of web server), using application that renders and analyses the code of a web site [35].

- *Abuse of functionality*

The attacker takes advantages of the current functionality of a web application to attack this application or other. The result is information leaking, gaining access control or using resources.

Web Services Security Specifications

Specifications and standards concerning the security issues of web applications and web services are part of the working groups of World Wide Web Consortium (W3C) and Organization for the Advancement of Structured Information Standards (OASIS). In this section the key specifications and standards concerning web services security are listed:

- W3C XML Encryption Syntax and Processing – This specification describes a process for data encryption and examines the result presentation in XML format [36].
- W3C XML Signature Syntax and Processing – Specification provides description of syntax and rules for the creation of XML digital signature [37]. Digital signatures ensure integrity, message and signer authentication for any data inside a XML document.
- W3C XML Key Management Specification – This specification describes protocols related to the distribution and registration of public keys [38] and W3C recommends it to be used in conjunction with specifications XML Encryption Syntax and Processing and XML Signature Syntax and Processing.
- OASIS Security Assertion Markup Language (SAML) – This is a XML-based framework aiming to standardize security assertions related to authentication, entitlement, and attribute information between XML services [39].
- OASIS eXtensible Access Control Markup Language (XACML) – This is a standard XML-based language that possesses possibilities to describe security policies [40]. Access control rules are standardized in XML format.

- OASIS Web Services Security: SOAP Message Security – This specification reveals the models for building secure web services (including using secure models like PKI, Kerberos, and SSL) achieving integrity and confidentiality of the message content [41].
- OASIS Web Services Security X.509 Certificate Token Profile – The specification presents X.509 certificate relationship with a public key and a set of attributes [42]. It specifies the methods for X.509 authentication framework and it is used together with the Web Services Security: SOAP Message Security specification.
- OASIS Web Services Security Policy Language – It describes the set of policy assertions of web services in the form of constraints and requirements according to the security features proposed in specifications Web Services Security: SOAP Message Security, Web Services Trust Language and Web Services Secure Conversation Language [43].
- OASIS Web Services Trust Language – The specification is extension of the Web Services Security: SOAP Message Security and describes methods for designing of secure communication based on "trust" between the communicating parties [44].
- OASIS Web Services Secure Conversation Language – The specification is extension of the Web Services Security: SOAP Message Security specification and describes security contests and their application [45].

Security of Technologies for implementation of Web Services

Web services are developed upon well-known technologies like: XML, SOAP, WSDL, HTTP and utilization of security issues of these technologies could support the process of web services securing.

- Security of SOAP messages

The security of SOAP messages can be done through security mechanisms applied to different communication layers as well as ensured through a combination of methods. The right configuration of the SOAP layer and transport layer is a very important part of a strategy for realization of security.

Transport layer – This layer uses the SOAP messaging protocol to realize the transport of the messages. Security can be achieved through separate mechanisms applied to the transport layer or through a combination of mechanisms taking into considerations the security possibilities of a SOAP message. A security transport layer can ensure SOAP message integrity (integrity of the whole message, this mechanism cannot protect a part of the message), confidentiality (SSL/TLS defence depends on the used cryptograph algorithms and the key length), authentication (through certificate X.509, usage of digests).

Layer SOAP message - Mechanisms can ensure the integrity of the message part or combination of different message parts (through XML digital signature). The integrity is guaranteed not only during HTTPS session, but also after that. Confidentiality is realized for parts of a SOAP message through XML digital signature. The confidentiality defence continues after the end of a session.

Authentication of SOAP sender – The SOAP sender can propose authentication to one or more SOAP receivers through including one or more secure tokens in the head part of the SOAP messages. This security mechanism can be applied together with XML digital signature.

- Security at XML level

Security measures include defence of XML documents in their transmission from sender to the final destination passing them through several intermediary nodes. Different parts of one XML document can travel to different final destinations. So, every element and content of one XML document can be secured. The XML security is realized through technologies like: XML encryption and XML digital signature. XML security achieves the following:

Confidentiality – Just the right receiver reads a given part of the XML document. It is achieved through usage of cryptograph algorithms (XML encryption standard).

Integrity – It means that the XML is not modified during its transportation from sender to receiver. The integrity is guaranteed through usage of the standard XML digital signature.

Authenticity – This mechanism ensures that the XML is sent by this person for whose he is presented. Confirmation of the sender identity can be realized through XML digital signature. The receiver can validate the sender's digital signature through usage of the sender's public key. If the digital signature is valid, then the identity is confirmed.

Not-refuse – The sender cannot refuse the sent XML document. XML digital signature is generated through the private key of the sender and guarantee his identity.

- Security on HTTP

The security of HTTP is ensured through the cryptographic protocols SSL and TLS. SSL/TSL use X.509 certificate and asymmetric cryptograph algorithms for authentication on the opposite side in one communication. For encryption of the transmitted data between two sides a session key is used to guarantee integrity, confidentiality and authentication.

- Security of RESTful web services

RESTful web services possess different design in comparison with SOAP web services and this fact leads to several differences in their securing. Anyway, all threats and attacks are applicable to web services in despite of their design nature. Forsberg in his research work talks about the specific characteristics of RESTful web services and the problem of secure data caching [46]. He proposes a solution for the implementation of secure web services without the need to secure URLs. This increases the performance of the RESTful services and decreases the requirements about ensuring of secure data storages. Experts reveal that mechanisms and technologies like: HTTP authentication, SSL, XML Signature, XML Encryption, SAML, Cardspace, other could be applied for better securing of RESTful services [47]. A report of National Security Agency of USA describes the often applied threats and attacks to the RESTful web services: session hijacking, Cross-site scripting, SQL Injection, Format String, Inadequate authentication/authorization methods, Cross Site Request Forgery, Access control policies and also comments several protective and defensive techniques [48]. The report recommends that encryption is always used for data transfers. Two types of encryption should be applied: transport encryption (SLL/TSL) and data encryption (XML).

Conclusion

The paper presents the contemporary view about security issues concerning web applications and web services. Realization of secure web services is a complex topic and it depends on different stakeholders: software designers and developers, network/database administrators, the work and understanding of security department in an organization, security policy of organizations, organizations involved in creation of specifications and standards. The temp of emerging new threats and attacks depends on emerging new technologies and applications. Classifications of threats and attacks are created by experts for their better understanding and analysis. This review shows that scientific society proposes a wide variety of solutions for securing the web applications and web services. Based on different design, the SOAP and RESTful web services are characterized with several specific features related to their attacks and security. Table 3 summarizes the reported attacks to SOAP and RESTful web services. The security issues in specifications and standards for SOAP web services are better developed in comparison with these concerning RESTful services.

Table 3. Threats and attacks against SOAP and RESTful web services

SOAP [10], [11]	RESTful [48]
Cross-Site Scripting	Cross-site scripting
SQL Injection	SQL Injection
Content Spoofing	Cross Site Request Forgery
Information Leakage	Inadequate authentication/authorization methods
Abuse of Functionality	Format String
Predictable Resource Location	Session hijacking
HTTP Response Splitting	Access control policies
Fingerprinting	
Insufficient Transport Layer Protection	
Brute Force	

REFERENCES

[1] Amazon web services 'growing fast'. 2015. http://www.bbc.com/news/business-32442268.

[2] XMethods web site – directory for public SOAP services. http://www.xmethods.com/.

[3] Membrane - directory for public SOAP services. http://www.service-repository.com.

[4] Workday SOAP Web Services Directory. https://community.workday.com/custom/developer/API/index.html.

[5] USGS site for SOAP and REST web services. http://waterservices.usgs.gov.

[6] Microsoft Developer Network. Chapter 1: Security Fundamentals for Web Services patterns & practices. https://msdn.microsoft.com/en-us/library/ff648318.aspx.

[7] W3C Working Group. Web Services Architecture. 2004. http://www.w3.org/TR/ws-arch/.

[8] Microsoft Corporation. Improving Web Application Security: Threats and Countermeasures. Microsoft Press. 2 September 2003.

[9] Web Application Security Consortium. 2004. Threat Classification.Version 1.0. www.webappsec.org.

[10] Gordeychik, S. 2010. Web Application Security Statistics. The Web Application Security Consortium. http://projects.webappsec.org/w/page/13246989/Web%20Application%20Security%20Statistics.

[11] WhiteHat Security. 2014. Website Security Statistics Report. https://www.whitehatsec.com/resource/stats.html.

[12] The Mitigation Group “One Voice”. 2011. Protect Against Cross Site Scripting (XSS) Attacks. The Information Assurance Mission at NSA. https://www.nsa.gov/ia/_files/factsheets/xss_iad_factsheet_final_web.pdf.

[13] Kirda, E., Kruegel, C., Vigna, G. and Jovanovic, N. 2006. Noxes: A Client-Side Solution for Mitigating Cross-Site Scripting Attacks. Proceedings of the 2006 ACM symposium on Applied computing. 330-337.

[14] Shanmugam, J. and Ponnavaikko, M. 2008. Cross Site Scripting-Latest developments and solutions: A survey. International Journal Open Problems Compt. Math. vol. 1. No. 2. 8-28.

[15] Laranjeiro, N., Vieira, M. and Madeira, H. 2009. Protecting Database Centric Web Services against SQL/XPath Injection Attacks. Lecture Notes in Computer Science. vol. 5690. 271-278.

[16] Janot, E. and Zavarsky, P. 2008. Preventing SQL Injections in Online Applications: Study, Recommendations and Java Solution Prototype Based on the SQL DOM. Application Security Conference. https://www.owasp.org/images/5/57/OWASP-AppSecEU08-Janot.pdf.

[17] Auger, R. Buffer Overflow. The Web application Security Consortium. http://projects.webappsec.org/w/page/13246916/Buffer%20Overflow.

[18] Adams, C., Jourdan, G.-V., Levac, J.-P. and Prevost, F. 2010. Lightweight protection against brute force login attacks on Web applications. in 'PST'. IEEE. 181-188.

[19] Pinkas, B. and Sander, T. 2002. Securing Passwords Against Dictionary Attacks. Proceedings of the 9th ACM conference on Computer and communications security. 161-170.

[20] Stubblebine, S. and van Oorschot P. C. 2004. Addressing Online Dictionary Attacks with Login Histories and Humans-in-the-Loop. Lecture Notes in Computer Science. volume 3110. 39-53.

[21] Ghetau, V. 2012. Preventing cookie replay attacks. The Web Systems Engineering Blog. http://websystemsengineering.blogspot.com/2012/12/preventing-cookie-replay-attacks.html.

[22] Hill, C. 2009. Cookie replay attack protection. Blog post. http://www.chrisjhill.co.uk/article/cookie-replay-attack-protection.

[23] Udemy blog. Eavesdropping on the Network : Sniffing for Packets. 2014. https://blog.udemy.com/packet-sniffers/.

[24] Adida, B. 2008. SessionLock: Securing Web Sessions against Eavesdropping. In Proceedings of the WWW 2008. Beijing, China, 517-524.

[25] Epp, D. 2015. Credential Theft and How to Secure Credentials. https://technet.microsoft.com/en-us/security/dn920237.aspx.

[26] Pavlou, K. E. and Snodgrass, R. T. 2008. Forensic Analysis of Database Tampering. ACM Transactions on Database Systems. vol. 33, Issue 4, Article No. 30.

[27] OWASP Foundation. Testing for Privilege escalation. https://www.owasp.org/index.php/Testing_for_Privilege_escalation_(OTG-AUTHZ-003).

[28] Hosek, P., Migliavacca, M., Papagiannis, I., Eyers, D. M., Evans, D. Shand, B. Bacon, J. and Pietzuch, P. 2011. SafeWeb: A Middleware for Securing Ruby-Based Web Applications. Lecture Notes in Computer Science. vol. 7049. 491-511.

[29] Eriksson, M. An Example of a Man-in-the-middle Attack Against Server Authenticated SSL-sessions. http://www8.cs.umu.se/education/examina/Rapporter/MattiasEriksson.pdf.

[30] Gangan, S. 2015. A Review of Man-in-the-Middle Attacks. http://arxiv.org/ftp/arxiv/papers/1504/1504.02115.pdf.

[31] The Open Web Application Security Project. A Guide to Building Secure Web Applications and Web Services. http://webpages.uncc.edu/billchu/classes/fall03/itis5166/APPSECURITY.PDF.

[32] Kargl, F., Maier, J. and Weber, M. Protecting web servers from distributed denial of service attacks. Proceedings of the 10th international conference on World Wide Web. 514-524.

[33] .-Arteaga, J. M, Caudel-García, H., and Fernandez, E. B. 2011. Misuse Pattern: Spoofing Web Services. Proceedings of the 2nd Asian Conference on Pattern Languages of Programs. Article No. 11.

[34] Mainka, C., Somorovsky, J. and Schwenk, J. 2012. Penetration Testing Tool for Web Services Security. IEEE Eighth World Congress on Services. 163-170.

[35] Adelsbach, A., Gajek, S., Schwenk, J. 2005. Visual Spoofing of SSL Protected Web Sites and Effective Countermeasures. Lecture Notes in Computer Science. vol. 3439. 204-216.

[36] W3C XML Encryption Syntax and Processing version 1.1. 11 April 2013. http://www.w3.org/TR/xmlenc-core1/.

[37] W3C XML Signature Syntax and Processing, Second edition. 10 June 2008. http://www.w3.org/TR/xmldsig-core/.

[38] W3C XML Key Management Specification (XKMS 2.0) version 2.0. 28 June 2005. http://www.w3.org/TR/xkms2/.

[39] OASIS Security Assertion Markup Language v2.0. 25 March 2008. https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

[40] OASIS eXtensible Access Control Markup Language v2.0. 1 February 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[41] Web Services Security:SOAP Message Security 1.0. March 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

[42] Web Services Security X.509 Certificate Token Profile. March 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf.

[43] Web Services Security Policy Language. July 2005 Version 1.1. http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf.

[44] Web Services Trust Language. February 2005. http://specs.xmlsoap.org/ws/2005/02/trust/ws-trust.pdf.

[45] Web Services Secure Conversation Language. February 2005. http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf.

[46] Forsberg, D. 2009. RESTful Security. IEEE Symposium on Security and Privacy. Claremont Resort. Oakland. USA. http://w2spconf.com/2009/papers/s4p3.pdf.

[47] Scalable, Reliable, and Secure RESTful services. 2007. Presentstion. ApacheCon Europe 2007, Amsterdam. http://www.apachecon.com/eu2007/materials/Scalable,%20Reliable,%20and%20Secure%20RESTful%20services.pdf

[48] National Security Agency of USA. 2011. Guidelines for Implementation of REST. https://www.nsa.gov/ia/_files/support/guidelines_implementation_rest.pdf.