Vulnerability minimization model in web distributed applications

ION IVAN, DRAGOS PALAGHITA, SORIN VINTURIS, MIHAI DOINEA
Informatics Economic Department
Academy of Economic Studies
Bucharest, ROMANIA
ionivan@ase.ro, dpalaghita@gmail.com, sorin.vinturis@yahoo.com, mihai.doinea@ie.ase.ro

Abstract: The paper whishes to emphasize the importance of analyzing the vulnerabilities of a web distributed application in order to block malicious attacks and prevent possible damage inflicted to distributed systems. Types of vulnerabilities are analyzed with insights on the distributed aspects. An analysis of the main vulnerabilities of authentication process is made and a model for minimizing them is described. A risk analysis is conducted to reveal the importance of such approach.

Key-words: security, vulnerabilities, risks, optimization, distributed applications.

1. Vulnerability Types

Vulnerabilities are constituted in classes which are susceptible to certain types of attacks.

Authentication is the vulnerability class that is opened to attacks that aim to corrupt validation procedures meant to establish the identity of application users. Here are some ways of attack:

- brute force the method of trying all combinations of symbols to form a password; this is a long process that is based on a dictionary of words. This type of attack is hampered by:
  - using complex passwords that contain a combination of alphanumeric characters with symbols;
  - blocking access if you have a given number of unsuccessful login attempts.
- exploitation of non-efficient authentication caused by poor programming error which allows access to protected resources without the need for identity verification. This type of attack is prevented by:
  - implementation of efficient validation rules;
  - form based authentication in order to not to permit accessing a protected page by inputting its address.

Arbitrary code execution represents the class of vulnerabilities that are susceptible to remote code that is able to run on the distributed application system. Types of attacks related to it are:

- buffer overflow is a common method to overwrite memory through which system instability is obtained by writing code on the stack according to [1] this type of attack is prevented by using strict length restrictions for fields that accept input from outside the application;
- SQL injection is represented by inserting SQL statements that run on the the SQL database; to avoid this type of situation it is required to implement meticulous validation of input supplied by the user and not use it directly in SQL syntax; according to [2] SQL injection damage is prevented by ensuring that the database system used

runs with the minimum privileges and is different from the system or SYSDBA;

- SSI injection is achieved by inserting code in the application that is executed on the server when the page is delivered; the command “<- # exec cmd = "ls" -->” lists current directory contents in a Unix system; this type of attack is prevented by spell checking user input and ban them if they do not satisfy the set character restrictions;

Vulnerabilities that disclose confidential information make up a separate class and are mainly caused by mismanagement of application resources:

- existence of pages to serve administrative purposes unprotected from unauthorized access;
- leaks are obtained by error messages that reveal snippets of source code lines or phrases about database information to the attacker and facilitates his attempts to destabilize the application, they are prevented by careful examination of the data if the information appears through an error, it is best to treat custom errors in the application and sanitize the output in order to prevent data disclosure;
- editing hidden fields in the pages that access the website properties according to [1] altering them by changing the page content values stored in fields intended for internal use by the application; such type of attack is prevented by not using hidden controls if necessary by blocking external access to the values stored in them.

The security level is in a directly proportional relationship with the degree of validation of user inputs data. The best method of validation is to treat all non-application entries as bad as it diminishes the chances to produce an attack in this way.
The human factor represents a large array of security problems:

- using the same password in more than one application is hazardous as it may cause a chain reaction if one of the applications is malicious and registers the user account information;
- storing the password phrase in a place that is accessible to foreign individuals that can gain access to it; it is also unsafe to use applications that store the user credentials in clear text without using any type of encryption;
- using passphrases that are directly related to the individual like birthdays, family name or others that are easy to guess.

The human factor influence is reduced by specific training on user account management, security procedures and computer security. It is important to point out the risks that the individual and organizational damage that can be inflicted by the misinterpretation of security policies.

2. Methods for Analyzing Vulnerabilities of Authentication Process

Use case analysis represents the activities undertaken to determine according to regular use cases attack possibilities and representing them as misuse cases for the AVIO product.

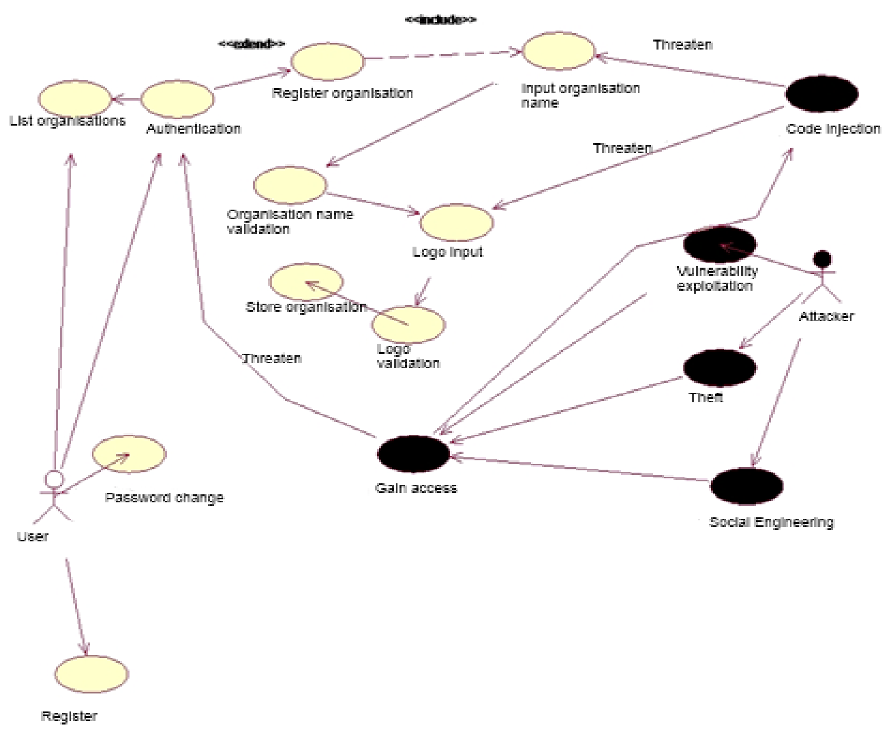

Figure 1 – Misuse case AVIO

The developed analysis identifies paths that are open to cybernetic attacks initiated by inside or outside entities.
Figure 1 presents the result of the use case analysis integrating attack methods that affect AVIO standard operation. These results of the use case analysis activities are defined by the identification of possible attacks on operations allowed within AVIO by determining the situations that are favorable to an attacker and through which gains access to confidential information or provokes damage to the software system.

In the use case analysis protection method identification is done through finding means and measures that handle the unwanted effects of a cybernetic attack or directly prevent it from happening. To this extent the use case diagram is altered by adding methods that aim to improve the security level of AVIO. These methods are presented in Figure 2 which presents new elements that are meant to develop an efficient security system by tackling attack types identified in the use case analysis.

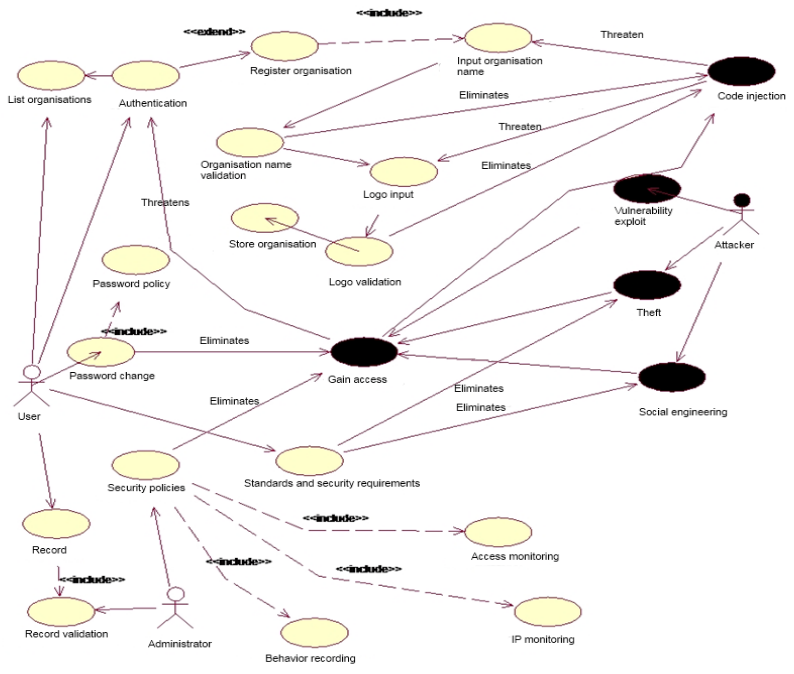

Figure 2 – Misuse case for AVIO and countermeasures

By determining the countermeasures aimed and stopping or minimizing the effects of informatics attack necessary information is obtained for risk analysis and management. *The structural analysis of AVIO* is represented by planned activities in order to determine the methods that produce attacks by taking advantage of structural deficiencies in AVIO.

3. Vulnerability Minimization Model

The vulnerability minimization is viewed as a process of improvement by transforming *A* into *B* with resource costs, but with some performance improvements which helps the entire system to be more efficient because:

- it uses the existing set of resources;
- achieves better quality, measured by specialized built-in metrics.

The minimization of vulnerabilities must have all the components required for an optimization problem:

- input – all the information given about the problem which need to be solved;
- output – the searched result which is obtained by resolving the problem in question.

These two components are represented by:

- the function *VLF*$(x) = y$ defined as $VLF: \mathcal{M} \to \mathcal{N}$;
- *constraints* over the $x\epsilon\mathcal{M}$ and $f(x)\epsilon\mathcal{N}$.
- $x_0\epsilon\mathcal{M}$ for which $f(x_0) \leq f(x)$ referring to vulnerability minimization;
- $x^*\epsilon\mathcal{M}$ defined as a local minimum for which there exists some $\delta > 0$, so that for $\forall x$ such that $\|x - x^*\| \leq \delta$, the expression $f(x^*) \leq f(x)$ holds.

Minimizing the level of vulnerability for a web based distributed application means improving security. The minimization of system's vulnerabilities can be traced and realized at the following levels:

- the physical level of vulnerability which improved can increase the safety of hardware equipments and access areas;
- the communication vulnerability level which can be achieved by decreasing the degree in which sensitive information is partially or totally revealed to end-users;
- the authentication vulnerability level, achieved by an adaptive algorithm which tries to lower the number of authentication flaws;
- the integrity vulnerability level given by data inconsistencies which must be checked from unauthorized access;
- the availability vulnerabilities that are threatening the user's access to resources.

Minimizing the vulnerability level of a security component, part of the security system of a distributed application means that a higher trustfulness in that particular area is achieved.

In [3] is described how vulnerabilities resulted from unsafe communication between distributed clients can be minimized by lowering in a simple fashion the degree of sensitive data, *DSD* which is available for access to end users.

$$DSD = 1 - \frac{TSDP(Kb)}{TSD(Kb)},$$

where:

TSDP – amount of sensitive information which is protected by user's access;

TSD – amount of total sensitive data processed by the distributed application.

If there is none sensitive data available in the system than there is nothing to protect. But if the system is processing information which shouldn't be available to public view than as presented in [3] we will have the situation depicted in Table 1, resulting that vulnerabilities can be treated through different approaches like optimization problem.

Table 1 – Security improvement showed by DSD

Without Security	**Relation**	**With Security**
$TSDP_0\ (Kb)$	<	$TSDP_1(Kb)$
$\frac{TSDP_0(Kb)}{TSD(Kb)}$	<	$\frac{TSDP_1(Kb)}{TSD(Kb)}$
$1 - \frac{TSDP_0(Kb)}{TSD(Kb)}$	>	$1 - \frac{TSDP_1(Kb)}{TSD(Kb)}$

The DSD indicator is getting values between [0; 1] with:

- DSD = 0; when the total amount of sensitive information is protected entirely in which case *TSD (Kb) = TSDP (Kb)* resulting *DSD = 1 – 1 = 0*;
- DSD = 1; when none of the sensitive information is protected, meaning *TSDP (Kb) = 0* resulting:

$$DSD = 1 - \frac{TSDP(Kb)}{TSD(Kb)} = 1 - 0 = 1.$$

An improvement of this metric is achieved when its value is lowering, describing a minimization problem.

In [4] are presented some directions for minimizing the vulnerabilities of the authentication process in web based distributed system. The approach is targeting some unique characteristics that only a particular person, in this case the authorized user can have them and no one else is capable of repeating them with high level of accuracy. Like biometric authentication is using facial characteristics, fingerprints matches or other individual unique aspects, the use of user's characteristics that reveals patterns of how the application is accessed in terms of location, time, the percentage of similarity between the password stored in the system and the one given at the authentication time and others unique features is meant to improve the capability of the authentication process to reject impersonations even when the credentials are right.

Based on these aspects, a further model can be developed which will take into account the users behavior and compute the characteristics into a single result for the authentication process.

From the normal process of authentication, a wide range of information can be recorded about the user's behavior such as:

- the IP addresses from which a user is used to access the web based distributed resources; using this information, the dispersion can be computed, based on which further access addresses can be tested for partial validity;
- the time interval in which users are normally access the system; exceptions can be checked more rigorously if they happen;
- the resources that are attempted to be accessed by users, denoting a different behavior than usual;
- the degree of similarity between the wrong passwords and the true one which can denote whether is a identity theft or a user which forgotten his password.

The process of developing a vulnerability minimization model for the authentication process in a web distributed application has the following steps:

- the process of data acquisition from which the main characteristics used in the later analysis are extracted;
- the process of data aggregation which provides an important view on the relevancy of each characteristic in the overall frame;
- the process of model validation through which data sets used for training will be also used for testing the system in a repetitive adjustment cycle until the values obtained will be statistically validated.

When a user tries to login the system is found himself in the following situations:

- he is the legitimate person knowing the password based on which he accesses the resources available according to his privileges;
- he forgot the password and he is trying to access the system with different passwords similar or not to the correct one;
- tries to hack someone's account by knowing the username and having a guess about the correct password of the real user in this way triggering a complete different behavior from the other two aforementioned situations.

The final output of the minimization model is represented by the number of password guessing attempts.

This number is considered to be uniquely depending on the user's behavior as mentioned above. The behavior is analyzed and a possible attack rate is given to each one of the accounts that exists on the system, meaning that for each user is determined a safety measure given by a lower or a higher number of missed passwords attempts. As the number is lower the higher probability of a targeted attack is assimilated with that specific user.

The flow of data gathering and processing is made based on the sequence diagram depicted below in Figure 3, which treats the main possibilities on how a particular user is interpreted by the system in order to calculate the attack probability of a certain login process.

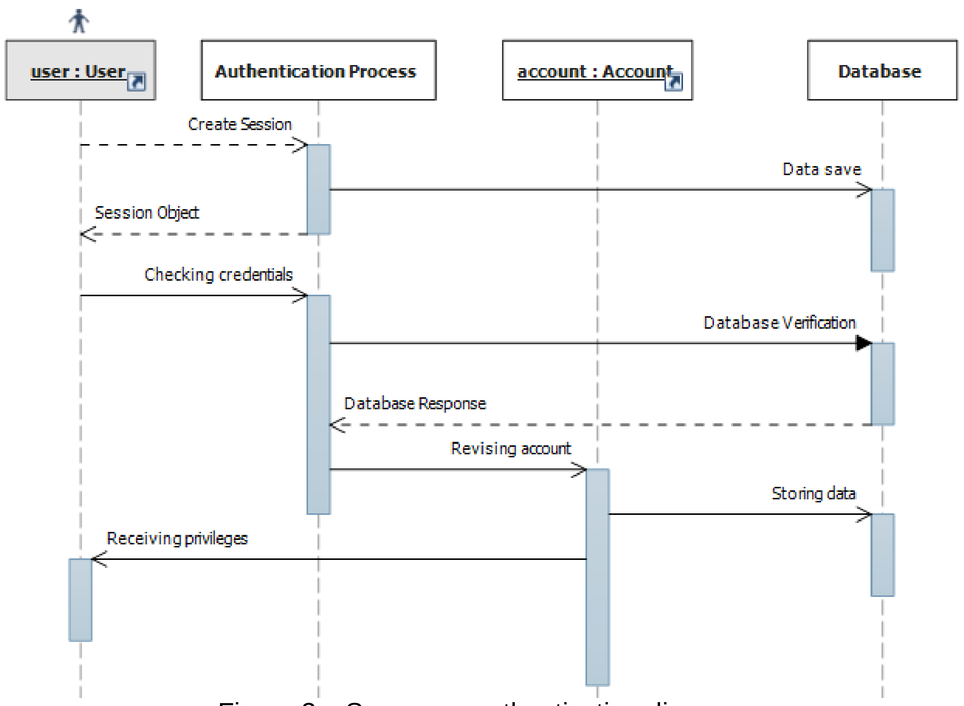

Figure 3 – Sequence authentication diagram

Every process, part of the sequence diagram, must be carefully analyzed because the results returned by them will represent the input into the following, excepting the process of returning privileges to user which will give access or not to the resources designated for each account.

4. AVIO Risk Analysis

An analysis can be made to tackle a little bit the quality – cost efficiency of the final model. This sort of approach takes into consideration the advantages and disadvantages of the implemented model such as:

- processing resources involved in the analysis;
- in which measure, the results returned by the model, are improving the overall system's security, implicitly the authentication process.

For the process of data acquisition the available resources are used in order to capture information that is already running through the network. The process of data aggregation can be attached to the main process of authentication, and for each user that is authenticated into the system or not the process is revalidating the user's account in terms of trustfulness. The process of model validation should only be run at the beginning of system's implementation when the model is trained and validated using different sets, interchanged.
Risk analysis is a set of actions meant to identify the main risks, Table 2, to which a web based distributed system is exposed analyzed from different perspectives such as:

- online operation – the risks concerning the services offered online to each users;

- functionality aspects – risks that are threatening the results quality of the application;
- technologies – aspects regarding risks that are focused on the technological functions for providing resources to users;
- data manipulation – the category of risks that threat the quality information characteristics.

.

Table 2 – AVIO risk analysis

AVIO Application		**Vulnerabilities**	**Threats**	**Risks**	**Measures**
Source code	C#	Validation	Code injection	AVIO database manipulation	Input validation [5]
		Unsigned library	Library alteration	Library modification and AVIO components control	Library signing [6]
		Privileges not verified when calling methods	Unauthorized access to methods	Accessing undisclosed information	Protecting methods from unauthorized access [7]
Operati on environ ment	IIS 7.0	Extended protection for authentication	Code injection	AVIO malfunction	Input validation [5]
	Windows Server 2008 R2 x64	Authorization	Instructions execution	Availability losses for AVIO	Input validation [8]
	ASP.NET	Viewstate management	XSS execution through _VIEWSTAT E	Unauthorized content modification	Input validation [9]
		CLR interface management	Code injection	AVIO highjacking	Input validation [5]
		HTML request management	Code injection	AVIO availability losses	Input validation [5]

		Types equality	Code injection	AVIO highjacking	Input validation [5]
	SQL Server 2008	-	-	-	-
Exploitation	Interface	Internal error	Code injection	AVIO highjacking	Input validation [5]
		Stack exposure	Information disclosure	Gathering information on how to run the source code allowing the attacker to identify AVIO logical structure of source code and focusing attacks	Errors management
		File uploading	Instructions execution	AVIO highjacking	Input validation [8]
		ASP .NET exposure	Information disclosure	Opportunities growth for attack by exploiting specific ASP.NET version vulnerabilities used by AVIO application	Errors management, deleting ASP version from retrieved pages
		Internal path disclosure	Reveal system directory structure information	Allow attack focusing based on the gathered information by examination of system directories.	Errors management

By making a connection between vulnerabilities and the main risks' categories, an entire picture of the framework in which the web based distributed application can be constructed, thus being possible to detect risks and vulnerabilities that are interdependent.

5. Conclusions

Vulnerabilities in web based distributed applications represents the main gate through which damage can be inflicted to such systems. There are vulnerabilities for which risk avoidance can be achieved, but also exists some that can't be managed.
Authentication vulnerabilities are an important piece from the main puzzle of web based distributed applications security. These must be seriously addressed due to high severity level to which they are assigned. If the authentication gate is breached, access to all resources available in the system is compromised so the entire infrastructure can be jeopardized.
For this reason this paper intends to give a feasible solution to how can the process of authentication can be improved by minimizing its weaknesses. The concept presented is meant to be further discussed as important feedback will be obtained from the scientific community.

Bibliography

[1] David WATSON – Web application attacks, *Network Security*, Issue 11, November, Elsevier, 2006
[2] David MORGAN – Web application security – SQL injection attacks, *Network security*, Issue 4, Elsevier, April 2006
[3] Mihai DOINEA – Security optimization of a distributed application for calculating daily calories consumption, *JISOM*, Vol. 4, No. 1, pp. 12-22, ISSN 1843 -4711
[4] Mihai DOINEA, Sorin PAVEL – Security Optimization for Distributed Applications Oriented On Very Large Data Sets, *Informatica Economica Journal*, Vol. 14, No. 2, 2010, pp. 72 – 85, ISSN 1453-1305
[5] Failure to Control Generation of Code – [Online], Available at: http://cwe.mitre.org/data/definitions/94.html
[6] Assemblies should have valid strong names – [Online], Available at: http://msdn.microsoft.com/en-us/library/ms182127(VS.80).aspx
[7] Do not indirectly expose methods with link demands – [Online], Available at: http://msdn.microsoft.com/en-us/library/ms182303.aspx
[8] Improper Input Validation – [Online], Available at: http://cwe.mitre.org/data/definitions/20.html
[9] Improper Neutralization of Input During Web Page Generation – [Online], Available at: http://cwe.mitre.org/data/definitions/79.html

Acknowledgements

This article is a result of the project POSDRU/6/1.5/S/11 „Doctoral Program and PhD Students in the education research and innovation triangle”. This project is co funded by European Social Fund through The Sectorial Operational Programme for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies, project no. 7832, Doctoral Program and PhD Students in the education research and innovation triangle, DOC-ECI.