FMEA – Web Application Security

Luboš Bretschneider <bretik@gmail.com>

Abstract

This document should describe and suggest solution of most common security problems during creating and running Web Application – I will consider UNIX web server and PHP application, but most of risks are applicable to every web application on every type of web server. For such analysis, I will try to find most common security issues and also find the most common and possible solutions according to the causes of failure. Failure in this context means security breakthrough, or unauthorized access to any protected part of application.

Possible risks

There are many types of risks, when someone is trying to break the Web Application security. I will concentrate on those, which can programmer or server administrator prevent:

- EXTERNAL CONTENT EXECUTION: If is possible to somehow execute external content (script) on our server, intruder can take control over whole virtual server and get everything he wants – including all data and password to database.
- SQL INJECTION: Thanks to SQL Injection, intruder can execute any SQL query on your database.
- UPLOADED CONTENT EXECUTION: If some file is uploaded on the server and intruder know, where the file is, he can take control over your server through his uploaded script.
- UNSECURE MANIPULATION WITH USER INPUT DATA: there are two different ways, how our server can be attacked. First of them can be very serious, the second one can “only” make our server busy.
  - EXECUTING DATA: If user of application is able to insert some executable content into input, it can make serious damage to application and/or server.
  - LARGE AMOUNT OF TEXT: While user is inserting some large amount of text, it can make server busy for short time, or take the server down.
- LOW-SECURITY PASSWORDS: Unsecure and predictable user password can be problem, intruder can gain access to application without permission.
- DIRECTORY LISTING: If some attacker is allowed to list directories on server, it is easier to imagine structure of data on server – exploring is first step of attacking some Web Application.
- UNAUTHORIZED ACCESS TO NON-PUBLIC PAGE: If pages are only hidden (not regularly protected) for unauthorized person, it could be easy just to find out address of hidden page.
- PLAIN-TEXT PASSWORDS IN DATABASE: If some attacker gains access to read your database, it can be disaster, if password of all users including administrator are not encrypted – attacker can take control over whole application.
- UNSECURE COOKIES: Cookies with some private content or plain-text passwords can help attacker to gain access to application without right to access it.

Failure Modes

Failure modes definition and is important for next analysis. Every possible risk has its own failure mode – from failure modes, we can define how serious problem represents every single risk and how common the problem is.

Failure modes for our possible risks are:

- *External content execution*: Any external content is executed, our server can be silently attacked and can be watched for long time without our attention
- *SQL Injection*: Database is corrupted or updated to attackers advantage
- *Uploaded content execution*: Some uploaded content is executed on our server – single powerful attack, attacker took some our personal data and made some damage on our server/application

- *Unsecure manipulation with user data*: some external content is executed, or server is busy (unavailable)
- *Low-security passwords*: attacker gained access to application without permission just thanks to guessing password
- *Directory listing*: attacker discovered structure of our data on server
- *Unauthorized access to non-public page*: intruder gain access to unprotected page, which is not intended to be public
- *Plain-text passwords in database*: attacker got all user-data including password in plain-text form and can use it to attack single person registered in our application
- *Unsecure cookies*: intruder attacked users computer and got cookie with plain-text information about user account in our application -> unauthorized access

Mechanisms and Causes of Failure - Potential Effects of Failure

For each potential failure mode, we will now define and analyze possible mechanisms and causes of failure and its effects to application and server functionality. There are few situations, where more failure modes has the same mechanism or cause of failure, that's because the right setup of server, or right working with data can prevent a lot of security issues.

External content execution

- *Direct inclusion of files as attributes in address bar*: This can be very serious problem – it's very easy to make some script, copy it to own server and just change one attribute in address bar. The result of this can be complete access to all data on server.
- *Using executable external content as images*: Including external executable content in output of our application can lead to session and cookie stealing – it's not recommended to use external executable content at all. Stealing of user cookies can lead to unauthorized access to our application.

SQL Injection

- *Direct using values from address bar in queries*: Very often mistake of programmers beginners, but not only beginners. Direct using user-readable and user-editable parameters in SQL queries is real security hole. Attacker is able to execute addition commands on our database, including destructive queries.
- *Direct using input values from form*: Practically the same problem as above – not so often, but still destructive. If we are directly using values filled and submitted by user, attacker can for example very easily access our application administration area.
- *Not using magic_quotes_gpc server property:* This is problem, when we are changing server, or our web-hosting provider is changing server setup. Without using this property, the above two causes of failure are very real, in other way, we can fast and easily eliminate them.

Uploaded Content Execution

- *Allowing uploaded content execution*: If we have allowed some user data upload and at the same time we have allowed execution of uploaded content, it can be easy for attacker to guess our upload directory and run his own uploaded script, what can be disaster for us – he can gain total control over our data.
- *Allowing executable content uploading*: Sometimes we need to allow user to upload executable content. In this case, we should consider the risks above.

Unsecure Manipulation with User Data

- *Printing user input directly to page*: If we are printing any user input to the page, we should consider checking it before printing. It can content some inappropriate content such as JavaScript. That can lead to using our application for illegal advertisement, showing unauthorized pop-up windows, or stealing cookies of logged users.
- *Not trimming input text to limited size*: If we have allowed any user input, we should consider maximal length of content, they can post and maximal relevant length for current action. A lot of content posted from some form can make server busy for short time, or in rare cases it can shut down the server.
- *Not optimizing allowable post size by post_max_size server property:* This is very important property of server setup – with this property set, we can prevent above cause of failure, if it's not set at all, it can lead to server busy or shut down from user side.

Low-Security Passwords

- *Not informing users about password security*: Without informing users about importance of their password security, we cannot expect, that most of them will choose secure password.
- *Not offering password generator*: Even if users know about importance of "strong" password, it's often difficult to choose some, because everything what come into humans mind is just association with something else and it leads to making easy-guessing passwords.
- *Accepting unsecure password in registration process:* If we accept "weak" passwords in our registration process, it's not dangerous only for user, but for our application too – attacker can gain access to registered area without permission.

Directory Listing

- *Allowing listing directories by default*: If we allow listing directories on server, it's generally big mistake which is first step to break our security.
- *Using .htaccess to allow listing*: Sometimes, we need to allow listing some directories. Using .htaccess is the most dangerous way, how to do that. If we allow listing of one directory, all subdirectories will inherit this setting – it can be problem if we make some new subdirectories and forget about this setting.

Unauthorized Access to Non-public Page

- *Granting access without session check*: If we have some area with user restrictions, it means some user can access it, others cannot, it's not enough to hide the page from others eyes. Registered and authorized user can send link to hidden page to unauthorized one and if we do not check the login session, it is security hole, which allow unauthorized users accessing protected areas.

Plain-text Passwords in Database

- *Storing password without encrypting*: If some attacker access our database, it's really bad, but not that bad as if we store all user information including passwords in plain-text, in this case, attacker can in few seconds read administrator username and password and gain access to all parts and settings of our application.

Unsecure Cookies

- *Using cookies for storing any information*: If we store some information in cookies, we should not really trust to it, when we are reading it back. Cookies are stored on the user's computer, and may be altered by web pages or malware or by the user. Or the server may be invoked by a script, not a browser. So cookies are good for storing some non-critical informal such as selected language or preferred text color.
- *Storing plain-text private information in cookies*: Cookies can be used for permanent login, or persistent storing any information, but storing private data on user's computer especially in plain-text could be really dangerous – any software, or user can read and alter the cookies. In combination with cause of failure above, it can lead to unauthorized access of any other user, who read the cookies.

Recommended Corrective Actions

In this chapter, I will suggest corrective actions in purpose to reduce Risk Priority Number (RPN). RPN indicates the reliability (security) of our application, lower RPN means more secure application. RPN is result of multiplication of three variables. Its values are from one (good) to ten (bad):

Occurrence x Severity x Detection = Risk Priority Number

Occurrence (O) in the formula means probability how often the problem can occur, Severity (S) means severity of the problem and Detection (D) represents the possibility of detection of the problem by our application itself, or system administrator.

This chapter will only describe possible corrective actions, all numbers and results can be seen in the Appendix A of this work

External content execution

If we want to avoid execution of external content, we should:

- Not at all use script inclusion from address bar variable
- Not to use unauthorized, or non-trusted external content in output of our application such as PHP images, frames with external content etc.

Executing external content on our server is critical security issue and has to be considered again and again. Every single change in the application should immediately rise the question - did we eliminated the chance to execute any external content on our server? If yes, our application is much more secure.

SQL Injection

In case we want to minimize possibility of SQL injection we must take following actions:

- Make input content secure – add slashes (make it not-executable) to strings used in SQL queries
- Enable magic_quotes_gpc property on server to automatically add slashes to COOKIES, GET and POST data

Through SQL injection is possible to alter data in database, or even delete it. Direct using of any user input is potential risk and programmer should think about using it before he really did it. Thanks to magic_quotes_gpc, we can add slashes automatically to any user editable input, but we cannot rely on it, we should add slashes manually too – this can make our application really immune from SQL injection.

Uploaded Content Execution

If we want to eliminate possibility of execution of uploaded content we must:

- Disable possibility to directly execute uploaded content
- Prevent uploading executable content if not necessary

Possibility of execution of uploaded content is as serious as execution of external content, it the same, but we can easily discover, that someone tried to do that and it's not so often problem, because the attacker has to know the application structure before executing the content.

Unsecure Manipulation with User Data

To lower the risk resulting from manipulation with user input data, we should:

- Use proper functions to make content secure – add slashes and remove HTML tags
- Limit maximal amount of data posted by form
- Set post_max_size server property to relevant value

Using corrective actions above, we can prevent any misusing of our application to illegal advertisement and prevent our server of being busy because of submitting large amount of user data. Adding slashes and removing HTML tags will help us to prevent executing any client-side JavaScript and setting limits can make our server less busy with data processing.

LOW-SECURITY PASSWORDS

To make user passwords more secure and raise whole application security, we can:

- Display password security warning
- Provide password strength meter
- Offer random generated secure password
- Forbid using dictionary passwords (common words) and passwords exactly same like username

Low security password is very often problem of all web applications. User should be informed that his password is weak and easily guessable. Providing simple password strength meter is very effective way to decently make user to think about stronger password – weak password means less secure application. In other way, id we really need to be sure, that user is using strong (secure) password, we can provide some password generator to him and forbid him to use dictionary (easily guessable) words as password.

DIRECTORY LISTING

To stop attacker to discover structure of our application we must:

- Disable directory listing by default
- Use HTML directory lists instead of .htaccess permission modification

If some attacker is trying to get into some application, he starts with discovering it – the less information we provide to attacker, the better. Attacker without information about internal structure of application has much more harder work – good for us. If we want to allow someone to list directory on our server, it's better to make HTML page with all accessible files, instead of allowing to list whole directory, including subdirectories.

UNAUTHORIZED ACCESS TO NON-PUBLIC PAGE

To prevent unauthorized accesses, we should:

- Check session on every non-public page

This mistake is not so common – if we are making some secure zone, it's necessary to make it really secure. Hiding files from menu is not secure at all, checking, if we have really authorized user on page is the most effective way.

Plain-text Passwords in Database

To prevent giving all passwords to intruder, we must:

- Encrypt password in database with some encryption or hashing algorithm such as MD5 or SHA1

Every complex web application has its users with different access privileges. If someone gains access to our database, it's disaster. If we store all user passwords as plain-text, it's total disaster. It's fast and effective way to encrypt passwords using some built-in algorithms such as SHA1, or MD5.

Unsecure Cookies

To avoid misuse of cookies, we can:

- Do not rely on persistent data in cookies
- Do not store private information
- If storing private information, encrypt it

Cookies are useful for users, who regularly visit our application. We can store some settings such as language, filter etc. and "remember" it thanks to cookies. Cookies are often used to "permanent login", but this is potential risk – anyone can steal cookie from users' computer and read it, or use it for login. It's better not to store any sensitive information in it and if we need to store some, it's better to encrypt it and store user-unreadable content.

Conclusion

In today world, Web application became important part of our lives. We are providing our personal information every day and web administrators are storing it in databases. As users, we never know about potential risks and we are surprised every time, when our personal data just appear somewhere else – this is result of computer criminality, but that's not only computer hackers fault, it is web application programmers and web server administrators fault too.

Before providing any our personal data, we should think about reliability and security of web application. Sometimes just short view can help us with considering this issue. This analysis can help programmers to prevent some really serious security holes and also to common users to check out possible risks and make their decision about providing personal information to web application.

Appendix A

No.	Item	Potential Failure Mode	Mechanisms and Causes of Failure	O	S	D	RPN	Recommended Corrective Actions	O	S	D	RPN
1	External Content Execution	Serious server security breakthrough	Direct inclusion of files	8	10	9	720	Not to use script inclusion from address bar variable	1	3	9	27
			Using External Executable Content	6	8	8	384	Not to inlcude unathorized external content in input	2	2	8	32
2	SQL Injection	Data loss or corruption	Direct using editable values in queries	8	9	7	504	Add slashes to input values	1	2	7	14
			Direct using form input in queries	7	9	6	378	Add slashes to input values	1	2	6	12
			No using magic_quotes_gpc	5	6	2	60	Enable magic_quotes_gpc property on server	1	1	2	2
3	Uploaded Content Execution	Unauthorizes control over server	Allowing uploaded content execution	4	7	4	112	Set automatically uploaded content to "not executable"	1	1	4	4
			Allowing executable content upload	6	5	3	90	Restrict executable content upload	2	3	3	18
4	Unsecure Manipulation with User Data	External data execution, server busy	Printing user input directly to page	9	6	2	108	Use proper functions to make input secure	2	3	2	12
			Not trimming input to max size	9	4	3	108	Decide max lenght of input content and limit it by HTML	2	1	3	6
			Not using post_max_size property	4	5	5	100	Consider re-setting post_max_size property on server	1	1	5	5
5	Low-Security Passwords	Unauthorized access	Not informing user about password security	8	6	1	48	Display warning and "password strength meter"	5	5	1	25
			Not offering password generator	10	2	1	20	Offer password generator in registration process	4	1	1	4
			Accepption unsere password through registration	9	3	2	54	Forbig dictionaty passwords and passwords exactly same like username	1	1	2	2
6	Directory Listing	Providing information about application structure	Allowing listing directories by default	7	8	6	336	Disable directory listing by default	1	1	6	6
			Using .htaccess to allow listing	6	6	7	252	Use HTML page for listing instead of .htaccess	1	2	7	14
7	Unauthorized Access to Non-Public Page	Providing access to unathorized area	Granting access withous session check	3	9	4	108	Check session on every page in secured area	1	3	4	12
8	Plain-text Passwords in Database	All user-passwords given to attacker	Storing passwords without encryption	4	8	1	32	Encrypt passwords in database (MD5, SHA1)	2	3	1	6
9	Unsercure Cookies	Unauthorized access, application error	Storing information in cookies	5	5	4	100	Do not rely on stored data, store only common information	2	2	4	16
			Storing plain-text private information on cookies	4	7	5	140	Do not store private information at all	1	1	5	5
				4	7	5	140	Encrypt stored information	2	2	5	20