Security Book

Patrick Günthard

June 13, 2016

Contents

1 Vulnrabilities defined by OWASP **1**
1.1 A4: Insecure Direct Object Reference 1
1.1.1 Introduction 1
1.1.2 Example 2
1.1.3 How to prevent 2

2 Symetrical encryption **3**
2.1 How does it work? 3
2.2 Examples 3

3 Authentication / Authorization **3**

4 Crypt Workshop **3**
4.1 Reflexion 3

5 Signatures **3**
5.1 Thoughts about collisions 3
5.2 THoughts about signatures of passwords and files 3

6 Key exchange **3**

7 Encryption in the Java programming language **3**

1 Vulnrabilities defined by OWASP

1.1 A4: Insecure Direct Object Reference

1.1.1 Introduction

Insecure Direct Object Reference is a common vulnrability which exists in web applications. It occurs if a parameter (e.g. a GET parameter) references a object in the system.

The atacker normally has to be authorized to this system but does not have access to all data.

1.1.2 Example

A URL which looks like this: `http://example.net/page.php?user=`*`myuser`* provides a page which shows the user data of the logged in user. One can easily change the parameter to show the data of another user: `http://example.net/page.php?user=`*`someotheruser`*

1.1.3 How to prevent

Session Based

- No *Direct Object Reference* has to be sent to the client, the references can be saved on the session
- In the case references are needed, they can differ from the server side data (i.e. database) an can be remapped on the server

Authorization

- Every access is checked if the user is authorized to do that. Example: A random token can be created for each user which then is checked every time the user accesses the page

Advantages

	Advantage	**Disadvantage**
Session Based	Only one authorization has to be done, access data for Database etc. is saved on the server and is not accessible by the attacker	A session uses a lot of memory for each user. For applications with a high number of users, a session for each client is not possible i.e. a non-session solution has to be implemented
Authorization	No Session is needed i.e. less memory is used and more users can access the application	Authorization is needed every time the user accesses data which is more complex to implement

2 Symetrical encryption

2.1 How does it work?

In symetrical encryption you en- & decrypt with the same key.

2.2 Examples

- AES

3 Authentication / Authorization

4 Crypt Workshop

4.1 Reflexion

5 Signatures

5.1 Thoughts about collisions

5.2 THoughts about signatures of passwords and files

6 Key exchange

7 Encryption in the Java programming language