Security Book

Patrick Günthard

July 11, 2016

Contents

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

3.1 Authentication

Authentication is the process of checking if someone *(e.g. a user of a multiuser application)* is the persone he or she pretends to be.

3.2 Authorization

Authentication is the process of checkin if an authenticated User

4 Crypt Workshop

4.1 Reflexion

5 Signatures

5.1 Thoughts about collisions

In some signature algorythms like MD5 collisions have been found. This means that two different values result in the same signature. Eventhought they happem rarely, this can be a security issue because in some cases a bruteforce attack can be easier. There is no completely secure method to solve this problem but always using the newest and most complex algorythms makes it harder for the attacker. E.g. there are no known collisions with algorythms like SHA-256.

5.2 Thoughts about signatures of passwords and files

- Passwords should be hashed and not encrypted so there is no other possibility to crack it other than brutforce. If a password is encryptet the attacker can access the data if he or she gets the key.
- If a signature of a file is available which can be downloaded, this hash can be compared with the hash of the downloaded file. If the two hashes are equal, the user can be sure that the downloaded file is actually the one which was intended to be distributet and not something else like a virus. *(There still are some security issues, see section about signature collisions.)*

6 Key exchange

7 Encryption in the Java programming language

In the java programming language there are some built-in (i.e. in the JDK) libraries for the en- and decryption of data.

Example of a class for encrypting data with the AES algorythm:

```
package ch.patrickguenthard.crypt.aes;

import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

public class Encrypt {
private byte[] key;
private byte[] data;
private Cipher c;
private SecretKeySpec k;


public Encrypt(String key, String data){

try {
c = Cipher.getInstance("AES");
} catch (NoSuchAlgorithmException | NoSuchPaddingException e) {
e.printStackTrace();
}

setData(data);
setKey(key);
}

public void setKey(String key){
setKey(key.getBytes(Charset.defaultCharset()));
}

public void setKey(byte[] key){
this.key = key;
k = new SecretKeySpec(key, "AES");
try {
c.init(Cipher.ENCRYPT_MODE, k);
} catch (InvalidKeyException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}


public void setData(String data){
```

```
setData(data.getBytes(Charset.defaultCharset()));
}

public void setData(byte[] data){
this.data = data;
}

public byte[] encrypt(){
try {
return c.doFinal(data);
} catch (IllegalBlockSizeException | BadPaddingException e) {
// TODO Auto-generated catch block
e.printStackTrace();
return null;
}
}
}
```

References

[marke] Literaturangabe