



WHITE PAPER

Common web application security problems and how to identify them

Part 8 Insecure storage

By

Lee J Lawson
Lead Penetration Tester, **dns**.

Introduction

This is **part 8** of a series of papers designed to raise the level of security knowledge regarding common security flaws in web applications. All papers revolve around the top 10 list of web application security issues as defined by the Open Web Application Security Project (OWASP). The last issue described the vulnerability known as **Improper error handling**. This paper concentrates on **Insecure storage**, the identification and resolution of this flaw.

Here is a list of the most common security flaws at the time of writing taken directly from the OWASP site.

1. **Unvalidated input**
2. **Broken access control**
3. **Broken authentication and session management**
4. **Cross site scripting**
5. **Buffer overflows**
6. **Injection flaws**
7. **Improper error handling**
8. **Insecure storage**
9. **Application denial of service**
10. **Insecure configuration management**

It should be noted that this paper is not intended to be used as a replacement to professional penetration testing. Rather it is aimed toward raising the level of security knowledge in the community. Professional penetration testing requires years of experience to be able to apply worthwhile interpretation to results from testing tools and gain the skills required to perform manual assurance testing.

Insecure storage

Web applications frequently use cryptographic functions to protect information and credentials. These functions and the code to integrate them have proven difficult to code properly, frequently resulting in weak protection.

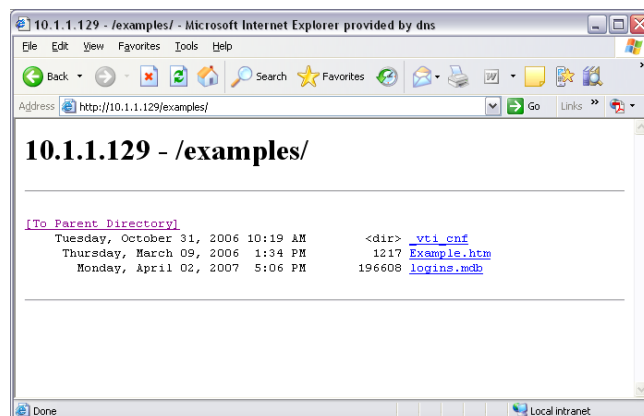
this information should be protected at all times

The Problem

All kinds of data are stored by web application servers such as usernames & passwords, credit card numbers and other personal data. This information should be protected at all times either for the security of the site, financial repercussions (PCI) or the law respectively.

Unfortunately this does not always happen. Credentials have been found in unencrypted text files which are viewable through a web browser, the highly sensitive information immediately useable to gain access to the system.

Other scenarios are also all too common; such as databases that are used for authentication have been found in accessible directories (see following image). This database was password protected however the information is still readable with text editors such as Notepad etc. Also, password protected Microsoft databases present a delay of about 10 seconds before an attacker has the password and opened the database!

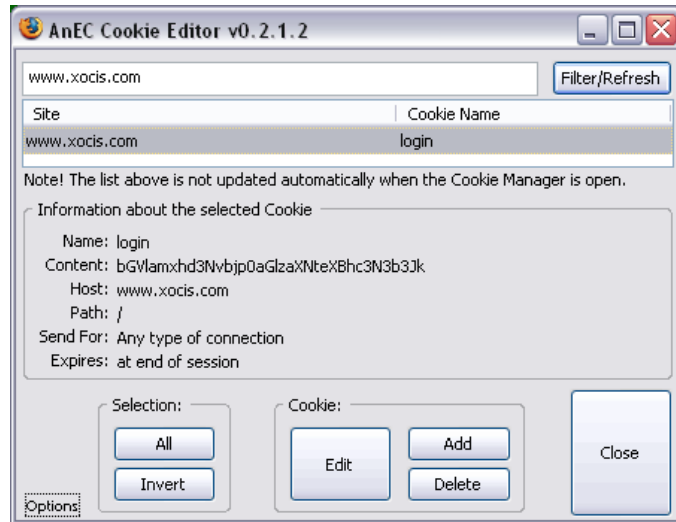


storing sensitive information in insecure locations is all too common

Storing sensitive information in insecure locations is all too common yet easily avoided with simple processes. But what about protecting the information with some kind of algorithm?

encoding is not encryption

Encoding is not encryption. Just because you cannot read something, does not mean that it cannot be read. These statements become apparent when considering that sensitive information has been stored with encoding systems like Base64. The following screenshot shows a cookie that has been created by a test web application. The content of the cookie is unreadable, or is it? It makes no sense to the human, but a simple decoding tool, freely available on the Internet will return that supposedly secure information into the plain text version.



Here, we can see the decoded text clearly revealing highly sensitive information.

Decode Safe Decode As Text

Decoded Output

Here is the decoded output of your Base 64 input:

```
leejlawson:thisismypassword
```

Encryption is not encoding, but even encryption algorithms are not always trustworthy. The Data Encryption Standard (DES) algorithm has long been exposed as a weak encryption system. RSA, one of the most influential cryptography organisations post a challenge once a year to crack a secret message. The first winner of this competition cracked the code in over 90 days, but this time has been plummeting since. DES encrypted messages are now being cracked in about 20 hours!!

This clearly shows that just by using encryption is not enough; research should be conducted to ensure that the algorithm in use reaches industry standards. AES, the Advanced Encryption Standard has now superseded DES as the algorithm of choice however even that only reaches 256 bit key lengths. Other algorithms now use keys up to 2k.

The general rule of thumb for encryption is that the bigger the key space, and the higher the number of encryption rounds, the better the algorithm is. That is true, however the human element should also be considered as they tend to be the weakest link in any computer system. An organisation may use the latest and

the human element should also be considered as they tend to be the weakest link in any computer system

greatest encryption systems such as Quantum algorithms, but if they store the key in an insecure fashion, the whole stack of cards come tumbling down.

a full penetration test is required to identify any lapses in security due to insecure storage

Identification

Ideally, a full penetration test is required to identify any lapses in security due to insecure storage; however there are a number of actions an organisation can take to attempt to find these problems themselves.

Identify what directories are accessible through web applications. This will give you an idea as to the 'attack surface', or the possible entry points to the system.

Attempt to get a list of the contents of those directories through the web application also. This should then be compared to the actual contents of the file system directory. If a file is in a web directory, it is likely to be downloadable. Attempting to hide files in web accessible directories by giving it an obscure name is not a viable security measure (security through obscurity) as you are reliant on the hope that attackers will not stumble across the filename and download the information.

look for files of all types, .txt, .doc, .xls, .mdb etc for Windows based environments

Countermeasures

Document grinding refers to the systematic searching of all items in a file system for a specific reason. The reason in this case would be to search for files that contain potentially sensitive information. Look for files of all types, .txt, .doc, .xls, .mdb etc for Windows based environments. For *nix environments, searches should not be limited by any file extension as they play a much lesser role than in Windows environments. Analysis of all accessible files should be conducted.

Encryption analysis should be performed to ensure that the algorithms in use sufficiently protect information in accordance with the classification of that data. For example, the recipe for Coca-Cola will need to be secure for many years to come, therefore a suitable encryption algorithm should be used. Ensure that the use of outdated algorithms is preferably eradicated, or limited to securing data that has a short life span (DES = < 24 hours protection etc).

Restrict access to files that are not absolutely necessary to run the web application. Place sensitive files in file system directories that are not within the web root and ensure that the relevant Access Control Lists (ACLs) are in place.

The next part of this series concentrates on **Application denial of service** and how they are used in exploitation.