not a MANual Tool for Risk Analysis

Risk Analysis Report [Design Constraints]

This document describes the features needed to be implemented in the design step to avoid the most risky threats to the system

*** Report Generated by: guifre**

*** Date: Fri Oct 26 10:25:54 CEST 2012**

*** Issued by: Computer Architecture and Operating Systems (CAOS)**
Universitat Autonoma de Barcelona.

Department of Architecture and Operating Systems

Universitat Autònoma de Barcelona

This is an alpha release, if for some arbitrary reason this got to you and have any comment suggestion question idea, you can ping me at guifre.ruiz at the gmail dot com server ;-).

caos
Department of Architecture and Operating Systems
UAB
Universitat Autònoma de Barcelona

1. Cross Site Request Forgery (CSRF)

1.1. Affected Components

* Path[0] Threat Agent[Anonymous User] Asset Element[VO Admin].
* Path[1] Threat Agent[Anonymous User] Asset Element[Identified User].
* Path[2] Threat Agent[Identified User] Asset Element[VO Admin].
* Path[3] Threat Agent[Identified User] Asset Element[Identified User].

1.2. Description

Summary

An attacker crafts malicious web links and distributes them (via web pages, email, etc.), typically in a targeted manner, hoping to induce users to click on the link and execute the malicious actionagainst some third-party application. If successful, the action embedded in the malicious link will be processed and accepted by the targeted application with the users' privilege level. This type of attack leverages the persistence and implicit trust placed in user session cookies by many web applications today. In such an architecture, once the user authenticates to an application and a session cookie is created on the user's system, all following transactions for that session are authenticated using that cookie including potential actions initiated by an attacker and simply "riding" the existing session cookie.

Example

```
The following code executes arbitrary HTTP requests in the users' broser:
<form name="badform" method="post"
  action="http://bank/Transfer">
  <input type="hidden" name="destinationAccountId" value="2" />
  <input type="hidden" name="amount" value="1000" />
 </form>
 <script type="text/javascript">
 document.badform.submit();
 </script>
•••

On the other hand, the code of the bank site is:
String id = response.getCookie("user");
userAcct = GetAcct(id);
If (userAcct != null)  {
   deposits.xfer(userAcct, toAcct, amount);
}

Since the credential is stored in the cookie, the victim will automatically be logged
in and the transferm carried out.
```

References

http://capec.mitre.org/data/definitions/62.html
http://cwe.mitre.org/data/definitions/352.html
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_ Sheet
http://www.codeguru.com/forum/showthread.php?t=371569
http://en.wikipedia.org/wiki/Cross-site_request_forgery

1.3. Software Specifications (AKA countermeasures)

Summary: Synchronizing a Secret token pattern in all HTML form requests

In order to facilitate a ”transparent but visible” CSRF solution, developers are encouraged to adopt the Synchronizer Token Pattern (http://www.corej2eepatterns.com/Design/PresoDesign.htm). The synchronizer token pattern requires the generating of random ”challenge” tokens that are associ- ated with the user’s current session. These challenge tokens are the inserted within the HTML forms and links associated with sensitive server-side operations. When the user wishes to invoke these sensitive operations, the HTTP request should include this challenge token. It is then the responsibility of the server application to verify the existence and correctness of this token. By including a challenge token with each request, the developer has a strong control to verify that the user actually intended to submit the desired requests. Inclusion of a required security token in HTTP requests associated with sensitive business functions helps mitigate CSRF attacks as successful exploitation assumes the attacker knows the randomly generated token for the target victim’s session. This is analogous to the attacker being able to guess the target victim’s session identifier. The validity of the token can also be limited to a small window of time, such as five minutes.For instance:

```
<form action="/transfer.do" method="post">
 <input type="hidden" name="CSRFToken"
value="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEwYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJj
ZDE1ZDZjMTViMGYwMGEwOA==">
…
</form>
```

2. Insecure Cryptographic Storage

2.1. Affected Components

* Path[0] Threat Agent[Anonymous User] Asset Element[config].
* Path[1] Threat Agent[Anonymous User] Asset Element[Relational Database].

2.2. Description

Summary

Protecting sensitive data with cryptography has become a key part of most applications. Simply failing to encrypt sensitive data is very widespread. Applications that do encrypt frequently contain poorly designed cryptography, either using inappropriate ciphers or making serious mistakes using strong ciphers. These flaws can lead to disclosure of sensitive data and compliance violations.

Example

```
> select * from users;
id username password
1 Brett 5f4dcc3b5aa765d61d8327deb882cf99 |
2 Dan 3c3662bcb661d6de679c636744c66b62 |
The passwords in these table are 32 characters long. Could these passwords be MD5
hashes?

As with all hashing algorithms, MD5 hashes can't be reversed. However, they can be pre-
computed. Using a hash table lookup we can identify what the password is before it was
ran through the MD5 hashing algorithm.

After inserting 5f4dcc3b5aa765d61d8327deb882cf99 into the hash table lookup the
resulting password is returned. In this example, the password is "password."
```

References

http://cwe.mitre.org/data/definitions/326.html
https://www.owasp.org/index.php/Top_10_2007-Insecure_Cryptographic_Storage
http://cwe.mitre.org/data/definitions/327.html
http://bretthard.in/2009/09/insecure-cryptographic-storage/

2.3. Software Specifications (AKA countermeasures)

Summary: Using strong cyptographic algorithms to encrypt sensitive data

Select a well-vetted algorithm that is currently considered to be strong by experts in the field Carefully manage and protect cryptographic keys