Bug Report

Date: March 19, 2011
Reporter: Chris Jarabek (cjjarabe@ucalgary.ca)

Software: Kimai
Version: 0.9.1.1205
Website: http://www.kimai.org
Description: Kimai is a web based time-tracking application. Users can create tasks and track the amount of time worked on said tasks. These tasks are linked to projects and customers which can be created by system administrators. The application features a reporting mechanism such that administrators can build exportable reports of the amount of time worked on a project or task broken down by users and other factors.

Kimai is built with PHP utilizing a MySQL database. The application frequently uses AJAX for communicating between the client and the server. The project is open-source available under GPL. It was registered with Source Forge on January 1 2007, and remains active, with the last release on September 29, 2010, and the last update on March 15, 2011. Since it was registered, the project has received 39 community recommendations and has been downloaded 33,465 times.

Bug 1

Summary: Unprivileged users can delete timesheet records of other users, including administrators.

Vulnerability type: Unverified user input.

Description: By utilizing an intermediate proxy between the client browser and the server it is possible to intercept requests made to the server, change the values of variables within the request, and the server will process the request with these modified variables.

Reproduction: For the purposes of this bug report, the exploit will be demonstrated on a clean installation of Kimai. Steps 1 through 7 are setting up the environment for the exploit, step 9 and onwards are the steps of the exploit. This exploit will be demonstrated in Mozilla Firefox 3.6.15, with the OWASP WebScarab build from January 18 2011 (http://dawes.za.net/gitweb.cgi?p=webscarab.git no version number is available). However, given the nature of the bug, it is likely to work on almost any browser type and version.

1. Log into Kimai as the administrator.
2. Create a new user:
    a. Click on “Users”
    b. Type in a new user name, e.g. “mrFoo”
    c. Click “Add User”
    d. Provide a valid password, rate, e-mail, nickname
        i. Make sure “Status” is set to “Regular User”

e. Click “OK”
f. Click the padlock icon under “Status” to instate the user

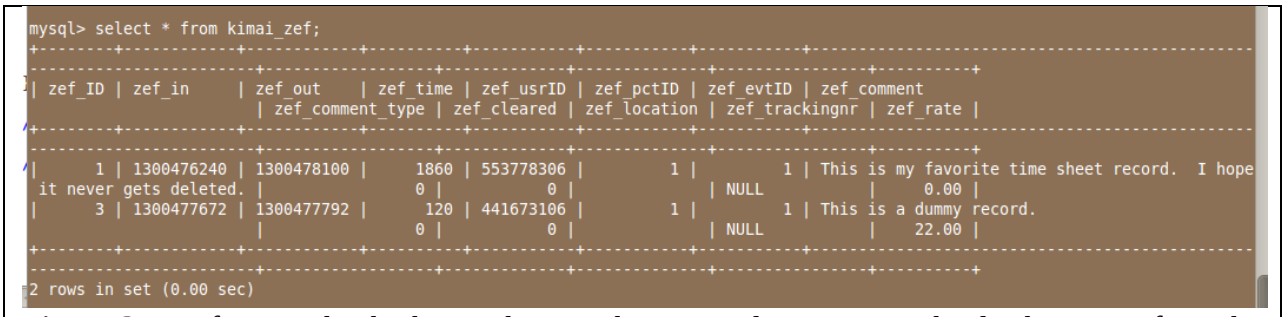
3. Create a new timesheet tracking entry:
   a. Click on “Timesheet tab”
   b. Click the plus button to add a new tracking entry
   c. The only field that needs to be changed is the “Time” field, so that the start / end times are different, also add a comment so it is easy to distinguish this record.
   d. Click “OK”

Figure 1. Administrator timesheet tracking entry.

```
mysql> select * from kimai_zef;
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------------------
-----------------------+-----------------+-------------+-------------+---------------+----------+
| zef_ID | zef_in     | zef_out    | zef_time | zef_usrID | zef_pctID | zef_evtID | zef_comment
                       | zef_comment_type | zef_cleared | zef_location | zef_trackingnr | zef_rate |
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------------------
-----------------------+-----------------+-------------+-------------+---------------+----------+
|      1 | 1300476240 | 1300478100 |     1860 | 553778306 |         1 |         1 | This is my favorite time sheet record.  I hope
 it never gets deleted. |                0 |           0 |             | NULL          |     0.00 |
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------------------
-----------------------+-----------------+-------------+-------------+---------------+----------+
1 row in set (0.00 sec)
```

Figure 2. Verification that this record exists in the database table **`kimai_zef`**.

4. Logout as the administrator
5. Login as newly created unprivileged user
6. Create a new timesheet tracking entry:
    a. Click the plus button to add a new tracking entry
    b. The only field that needs to be changed is the "Time" field, so that the start / end times are different, also add a comment so it is easy to distinguish this record.
    c. Click "OK"



```
mysql> select * from kimai_zef;
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------------
-------------------------+-----------------+------------+-------------+---------------+----------+
| zef_ID | zef_in     | zef_out    | zef_time | zef_usrID | zef_pctID | zef_evtID | zef_comment
                         | zef_comment_type | zef_cleared | zef_location | zef_trackingnr | zef_rate |
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------------
-------------------------+-----------------+------------+-------------+---------------+----------+
|      1 | 1300476240 | 1300478100 |     1860 | 553778306 |         1 |         1 | This is my favorite time sheet record.  I hope
 it never gets deleted. |                 0 |          0 |             | NULL          |     0.00 |
|      3 | 1300477672 | 1300477792 |      120 | 441673106 |         1 |         1 | This is a dummy record.
                        |                 0 |          0 |             | NULL          |    22.00 |
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------------
-------------------------+-----------------+------------+-------------+---------------+----------+
2 rows in set (0.00 sec)
```

Figure 3. Verification that both timesheet tracking records are now in the database, one from the administrator, one from the unprivileged user.

7. Launch WebScarab:
    a. Open WebScarab
    b. Click on the "Intercept" tab
    c. Ensure that "Intercept requests" and "Intercept responses" are both unchecked.
8. Configure Firefox to use WebScarab as a proxy:
    a. Click "Tools"
    b. Click "Options"
    c. Click "Advanced" tab
    d. Click "Network" tab
    e. Click "Settings"
    f. Select "Manual proxy configuration"
    g. Set HTTP proxy to "localhost"
    h. Set port to "8008"
    i. Click "Use this proxy for all protocols"
    j. Click "Ok"
    k. Click "Ok"

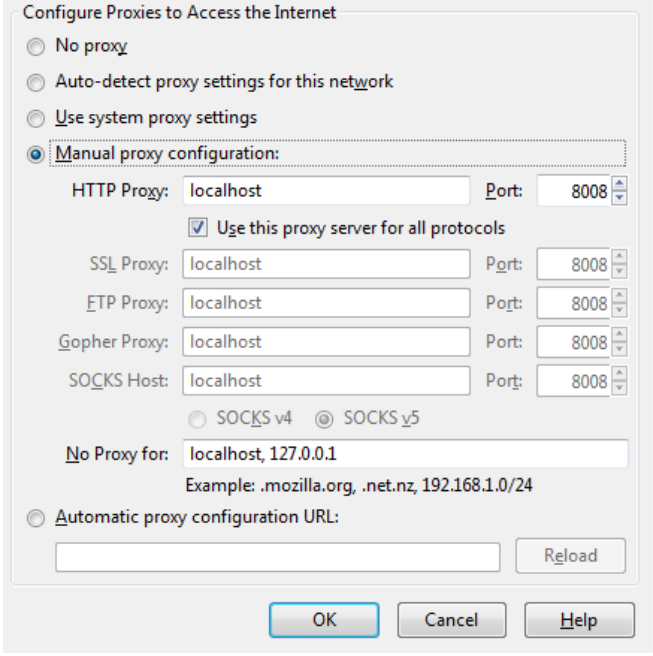


Figure 4. Firefox network configuration to use WebScarab.

9. Click the "edit" icon in Kimai for the newly created timesheet record
10. In WebScarab, check the "Intercept requests" box
11. Delete the timesheet entry in Kimai:
    a. Click the three dots in the top right corner of the edit dialog
    b. Check the "erase entry" box
    c. Click "OK"
12. Change the ID of the deleted entry in WebScarab:
    a. WebScarab will automatically intercept the request from step 11c
    b. Scroll through the URL encoded request data and locate "ID"
    c. Change "ID" to 1, the ID of the administrator's timesheet record (Fig. 3)
    d. Click "Accept changes"
    e. You may get addition interceptions, click "Accept Changes" for these as well

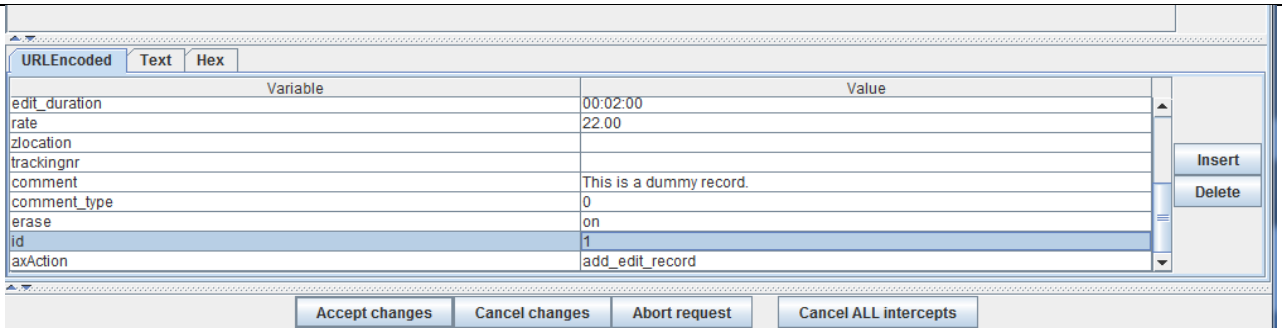


Figure 5. WebScarab interface for modify request values.

The administrator's timesheet record has now been deleted. You will notice that the unprivileged user's timesheet record remains. This can be verified by looking directly at the database or by logging out as the unprivileged user and re-logging in as the administrator.

```
mysql> select * from kimai_zef;
+--------+------------+------------+----------+-----------+-----------+-----------+-------------------------+-----------------+--
----------+-------------+---------------+----------+
| zef_ID | zef_in     | zef_out    | zef_time | zef_usrID | zef_pctID | zef_evtID | zef_comment             | zef_comment_type | z
ef_cleared | zef_location | zef_trackingnr | zef_rate |
+--------+------------+------------+----------+-----------+-----------+-----------+-------------------------+-----------------+--
----------+-------------+---------------+----------+
|      3 | 1300477672 | 1300477792 |      120 | 441673106 |         1 |         1 | This is a dummy record. |               0 |
        0 |             | NULL       |          |    22.00 |
+--------+------------+------------+----------+-----------+-----------+-----------+-------------------------+-----------------+--
----------+-------------+---------------+----------+
1 row in set (0.00 sec)
```

Figure 6. Database table **`kimai_zef`** showing only the unprivileged user's timesheet record, the administrator's timesheet record has been deleted.

Attack Variation: An even more direct attack is possible simply by using the Firebug Extension (http://getfirebug.com/) for Firefox. This attack proceeds from step 6 above:

7. Prepare to delete the timesheet entry:
    a. Click the three dots in the top right corner of the edit dialog
    b. Check the "erase entry" box
8. Modify the delete ID:
    a. Right click on the "OK" button
    b. Click "Inspect" element to launch Firebug
    c. Locate the hidden "ID" field 4 lines above the input tag
    d. Change the "ID" value to "1"
    e. Close firebug
9. Click "OK" to delete the administrator's record

Figure 7. Firebug console showing the hidden ID field that needs to be modified.

Implications: Unfortunately, this attack is already fully formed; it is not simply a potential vulnerability. The above steps demonstrate that an unprivileged system user can delete any timesheet tracking record from the system. By examining the HTML source of the user's various timesheet records, they can enumerate their own timesheet IDs. In this way a user could ensure that they do not accidently delete one of their own timesheet records. Fortunately, without a way to examine the database, or another user's session, the malicious user is only shooting in the dark when they delete records from the database. However, with the

vulnerability described below will demonstrate that it is in fact very possible that a user could gain access to another user's session.

Cause: It is challenging to remove all record IDs from a database powered web application. If these IDs displayed in the HTML source and then used as inputs to the application, they must be treated as being malicious. In this case, the application does not do any input validation on the ID passed to the delete action. As a result the application simply deletes the record with the supplied ID, regardless of whether the requesting user is the owner of that record.

Fix: Kimai uses a session identifier in a browser cookie for user identification. In order to ensure that the user deleting the record is the correct user, the session ID should be passed along with the delete request or any other request where actions are based on IDs coming from the client's browser. Then the server can deny the delete request if the session ID belongs to a different user than the ID of the record being deleted.

Code: An examination of /extensions/timesheets/processor.php shows the delete function as being in the called from '**`add_edit_record`**' (lines 227-232), and '**`quickdelete`**' (lines 119-122). This function, **`zef_delete_record($id)`**, appears to be an atomic function. This means that the security check must take place prior to any call to this function. The session ID of the requesting user, and the timesheet ID need to be examined to ensure that the requested delete operation is valid.

Bug 2

Summary: System users can insert JavaScript code into Kimai, and have this code executed on the browsers of other users.

Vulnerability type: Stored Cross-Site Scripting (XSS) Vulnerability.

Description: Users are able to insert JavaScript into the comments of timesheet tracking records which, when viewed by a user, will cause the execution of the script in question.

Reproduction: This bug report will be demonstrated on a clean installation of Kimai. Many of the early steps are identical to those in Bug 1, above. They are included here for completeness. This exploit will be demonstrated in Mozilla Firefox 3.6.15.

1. Log into Kimai as the administrator.
2. Create a new user:
    a. Click on "Admin panel"
    b. Click on "Users"
    c. Type in a new user name, e.g. "mrFoo"
    d. Click "Add User"
    e. Provide a valid password, rate, e-mail, nickname
        i. Make sure "Status" is set to "Regular User"
    f. Click "OK"
    g. Click the padlock icon under "Status" to instate the user

3. Logout as the administrator
4. Login as newly created unprivileged user
5. Create a new malicious timesheet tracking entry:
    a. Click the plus button to add a new tracking entry
    b. Change the "Time" field, so that the start / end times are different
    c. Add XSS code into the comment:
        i. For the purposes of demonstration, set comment to: "**`<script>alert(document.cookie);</script>`**"
    d. Click "OK"
6. Take note that the XSS is immediately noticeable as the cookie is displayed in a JS alert box.

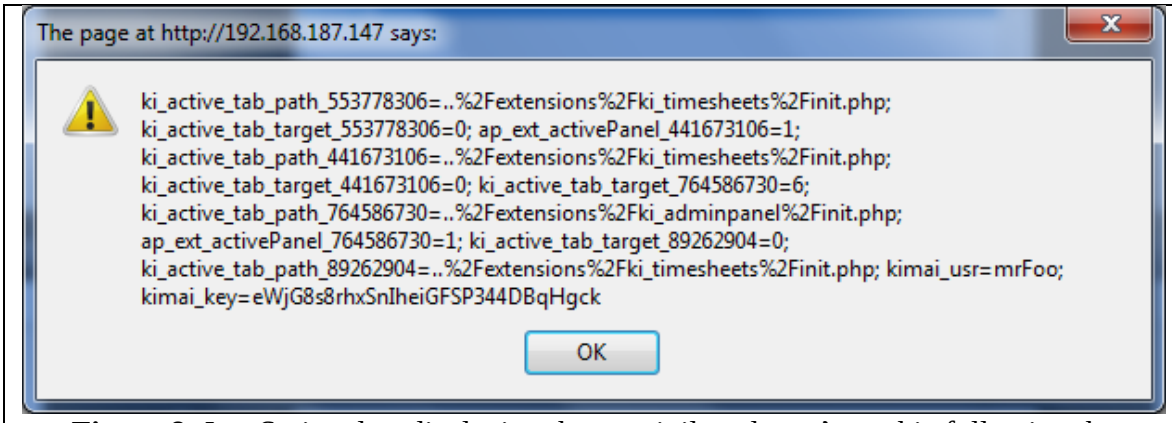


Figure 8. JavaScript alert displaying the unprivileged user's cookie following the addition of a new timesheet tracking entry.

7. Log out as the unprivileged user
8. Activate the XSS on the administrator's account:
    a. Log back in as the administrator
    b. Click on the "Export" tab
    c. Under the "Users" box, click the filter icon next to the unprivileged user (mrFoo)
9. Notice that the alert box script added by the unprivileged user is executed on the administrator's account. Additionally, notice that all the cookie values (including the session ID) correspond to the administrator.

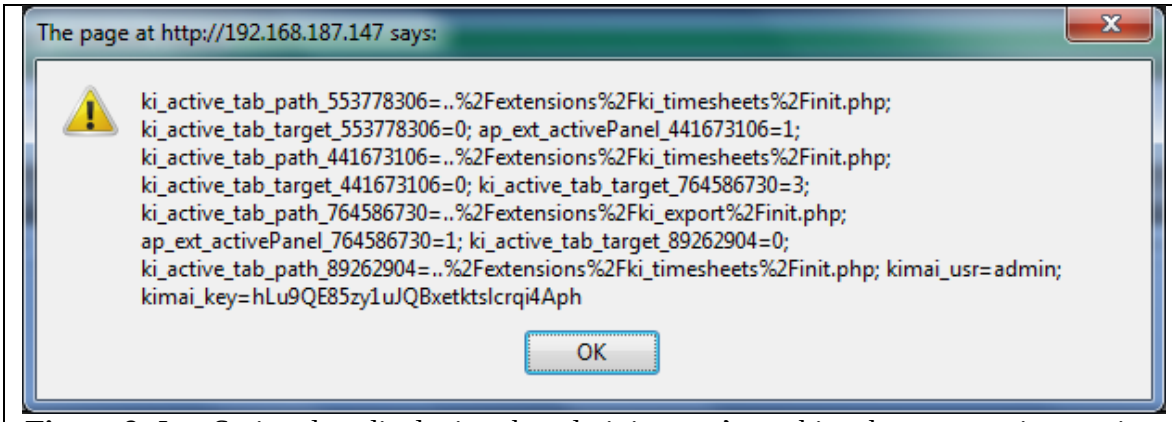


Figure 9. JavaScript alert displaying the administrator's cookie when attempting to view the timesheet tracking entry added by the unprivileged user.

Attack Variation: An examination of the database table **`kimai_zef`** shows that the JavaScript is not escaped and is stored in the format supplied by the malicious user. This means that the XSS could be exploited on any page where the comment field of the timesheet entry is rendered in HTML. Other fields were tested for potential XSS injection. The unprivileged user does not have access to many fields which supply input to the database, and most of those fields are too short to hold a malicious piece of JavaScript. However, it should be noted that field lengths specified in the browser can easily be changed, and therefore, all field inputs should be checked for correct length on the server prior to insertion into the database. The administrator has access to numerous other fields which were shown to support XSS. For the purposes of this bug, the focus of this report will be limited to the attack path described above. However, it should be noted that **all** user supplied input (regardless of which user) should be sanitized using the techniques described below.

```
mysql> select * from kimai_zef;
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------+----
-------------+-------------+--------------+----------------+----------+
| zef_ID | zef_in     | zef_out    | zef_time | zef_usrID | zef_pctID | zef_evtID | zef_comment                             | zef
_comment_type | zef_cleared | zef_location | zef_trackingnr | zef_rate |
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------+----
-------------+-------------+--------------+----------------+----------+
|      1 | 1300559468 | 1300560908 |     1440 |  89262904 |         1 |         1 | <script>alert(document.cookie);</script> |
           0 |           0 |              | NULL           |    22.00 |
+--------+------------+------------+----------+-----------+-----------+-----------+-----------------------------------------+----
-------------+-------------+--------------+----------------+----------+
```

Figure 10. The database table **`kimai_zef`** showing the fully formed XSS in the comment field.

Implications: In this bug, it was shown that a user could display the cookie of another user in an alert box when the target user viewed a comment supplied by the attacker. This technique could be used to easily hijack the session of another user, even an administrator. Rather than having the cookie appear in an alert box, the cookie could be sent to a server controlled by the attacker via the JavaScript XMLHttpRequest (AJAX) object. The attacker could then take this cookie, insert it into their browser and in doing so, hijack the target's session. XSS could also be used to rewrite content on the webpage being viewed by the victim, potentially contributing to a phishing attack.

Cause: This exploit stems from a failure to correctly sanitize user supplied inputs prior to insertion into the database. As a result, the application assumes that the user's comment is

benign, when in reality it is being recognized as valid JavaScript by the browser's JavaScript engine and executed when the page displaying the comment is rendered.

Fix: Every single input that comes from the client's browser needs to be treated as being potentially malicious. The XSS vulnerability described above was using the simplest form of XSS injection. This type of attack can be eliminated by transforming all '<' and '>' characters into their respective HTML entity characters: '**`<`**' and '**`>`**'. It should be noted that this will only combat the most trivial form of XSS. To fully mitigate XSS vulnerabilities in an application, it is recommended that all user supplied inputs be processed by OWASP's Enterprise Security API XSS sanitization library (http://www.owasp.org/index.php/ESAPI).

Code: An examination of /extensions/timesheets/processor.php shows the adding of a new timesheet tracking record occurs in '**`add_edit_record`**' (lines 214 onward). Here, the comment from the client request is placed in the array **`$data`**. This array is later passed to **`zef_create_record`**. This function can be found on line 2048 of /includes/db_layer_mysql.php. In this function, the elements of the **`$data`** array are correctly sanitized to prevent MySQL injection attacks. This would be the ideal location to also sanitize text for XSS injection. All of the database insertion functions should have the user supplied input escaped by the ESAPI described above prior to inserting any of the data.