COMP SCI 7412 Secure Software Engineering Assignment 1

Tinson Lai
a1812422

1 Part 1

1.1 pmicore/pmicore

Repository: pmicore/pmicore
Commit: 162ced923c78d8f012aa0c3c6005dfa6c0804e6e
Stars: 1861
Contributors: 298
File: bundles/AdminBundle/Resources/public/js/pimcore/document/tree.js
Language: JavaScript
CWE ID: 79 (XSS)
Lines: 1022-1029
Fix: f20a21d1a6e72e502defb5453e6e94734a7ce783
Comment: The fix presented in the git history shows that the contributors have already patched the file. The original version of the file didn't perform any neutralisation on input and output it directly to the browser. Now it includes a specific renderer for the HTML rendering process to sanitise the input so it can mitigate the effect if malware injects a specially crafted JavaScript snippet into the text box. However, by following the code it adds to fix the vulnerability, I think the fix is enough for this circumstance as it escapes everything in a text area.

1.2 forkcms/forkcms

Repository: forkcms/forkcms
Link: 8c003bb6122ea884de0eb018ab4a74b0288d4a30
Stars: 1095
Contributors: 85
File: src/Backend/Core/Engine/Base/AjaxAction.php
Language: PHP
CWE ID: 352 (CSRF)
Lines: 20-22
Fix: c58b9bccc0bff4e01c68c69d1fcf8322f31abc22
Comment: The commit to fix the issue has already been merged into the repository. The defined PHP class, `class AjaxAction`, is inherited multiple times. By the implication of this base class name, those inherited classes perform AJAX actions to handle form submissions. Prior to the fix, there is no CSRF token checking process exists in the code, hence malicious forged request sent from external

website will be accepted without any further verification. The fix has patched this file (and relative files) and add extra checking for CSRF token during execution, so by the nature of CSRF tokens, it can prevent most of the CSRF attacks.

1.3 osTicket/osTicket

Repository: osTicket/osTicket
Commit: 00518b4f768939d53841619c8b7924a91a1f6741
Stars: 1922
Contributors: 86
File: include/staff/category.inc.php
Language: PHP
CWE ID: 79 (XSS)
Lines: 153
Fix: 1aef8906ea4a0839b7217f8af46ca7d7ed8af613
Comment: The original version of file actually contains a bit of effort to avoid XSS, as shown in line 36, by using `Format::htmlchars` which wraps the PHP's builtin routine `htmlspecialchars`. However, the system was not further sanitising the text so that a succeeding call to parse the rich text area can trigger an XSS execution by injecting a specially crafted XSS payloads into some of the form fields. Thus the patch has added an extra level of sanitisation process to enhance the protection.

2 Part 2

The first row of the data is my own answer. Actually, I personally think the survey is too long for most of the people as it contains more than 30 questions (30 original questions plus 2 more questions I made). As most of my friends filled in this questionnaire are in Melbourne and they are still experiencing the stage 4 lockdown, I can feel the anxiety and boringness and they seem to give me a few repetitive answers to a portion of the questions. I think they are not quite interested in these kinds of mobile applications, probably because this is now useless during a complete lockdown, and some are slightly concerning about the privacy. In my memory, however, the Australian federal government has imposed a new law to legislate against all kinds of improper and unauthorised use of the data available in the COVIDSafe application.
My first question is related to vulnerabilities. I don't think I have ever heard of any news about vulnerability has been discovered in the application, but every application will potentially contain bugs or errors. I think none of them not using the application based on chatting with them, so this question seems to be pointless to them.
The last question is about trust. If a people recommends an application to others, usually this implies that the people has used this application, and it conveys the message that this people think this application is useful and trustworthy. It also reflects the overall impression the people have about an application. From the result, I think they are quite neutral about the application, so I would say this application is not useful in their mind, or at least, not attractive to them to install the application.
Honestly, the last question is a bit too general compared to other questions, but I think most of the provided questions have already covered most of the aspects of a mobile application, so probably a satisfaction type of question will be more appropriate to end a questionnaire.

3 Part 3

3.1 CVE-2018-1000616

The corresponding repository of the vulnerability identified by the given CVE ID, CVE-2018-1000616, is actually hosted on the self-hosting instance of Gerrit maintained by the developer group. The one on GitHub is just a mirror as described in README. It is easy to identify the repository link from NVD website by looking at the referencing hyperlink section. Unfortunately, neither the README file in the repository nor the official wiki provides a redirect link to the GitHub repository, so the only way is to search for this repository on GitHub.
There is no way for me to identified the bug report as GitHub Issues is closed for this repository. Meanwhile, their own Jira instance does not open to the public and Gerrit is only used as a code review platform. However, the Gerrit instance includes the merge request and its relevant code review status so we can have a glance on the fix[1]. As this is a quite old merge request (around 2 years ago), everything including the CI pipeline and code reviewing has been finalised, and the fix has already been merged into the repository. The given hash of the fix commit match the actual one, so there is no more thing for this part.
For the scoring part, I've found utilities (CVSS 2 Calculator[2] & CVSS3 Calculator[3]) online provided by NVD. I think my CVSS 3 vector is essentially the same as the official one except that I've added temporal metrics to the score so the base scores are the same. The score computed is based on the following reasoning:

- There is no doubt that RL:OF (RL:O in CVSS 3) and RC:C should present in both versions, as the vulnerability has been reported and confirmed, and obviously the actual fix is now patched to the repoisotry by the contributors. I can't find an example of attack on the website, so I will leave E:X unchanged.
- AV:N in both versions as indicated in the bug report. The injection can be executed by sending a malicious XML to the controller.
- Either Access Complexity in CVSS 2 or Attack Complexity in CVSS 3 has a value of AC:L as there is no specialised circumstance for the attacker to access to the controller.
- Based on the report and the nature of XXE, the attack doesn't require privileges to be granted prior to the attack, So this is PR:N in CVSS3. Also it doesn't require authentication to send a malicious XXE to the controller, so it's Au:N in CVSS 2.
- Unlike injections like XSS, XXE is usually injected by the attacker to have unauthorised access to the system files, so there is no need to interact with other users, thus it's UI:N in CVSS 3.
- The scope will not be changed as usually the attacker can only have the same privileges as the running user of the controller, thus it's a S:C in CVSS 3. However, the attacker might launch a subsequent attack to gain root access to the system, but this is not related to this vulnerability.
- The reason all C, I and A have a value of High in CVSS 3 is that these metrics are focusing on the component but not the system. As the controller's file are sharing the same user privileges as,

[1]`https://gerrit.onosproject.org/c/onos/+/18894`
[2]`https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator`
[3]`https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator`

it will definitely affect the whole component. But in CVSS 2 it's focusing on the impacts on the whole system, so all C, I and A have a value of Partial, as the controller isn't run as the root user, and there should be more access control stuff exists in the system.

- Environmental score metrics are not related to this question.

3.2 CVE-2017-15719

The repository is given in NVD as well. NVD doesn't provide a specific issue number or pull request number, and I can't find the bug report in the repository as well. The only information available is the Wiki page in the repository. As pointed out in the wiki, the vulnerability was identified and reported in Apache OpenMeeting. The commit itself doesn't actually contains useful information in the description. However, we can clearly see that the contributor has added an extra level of sanitisation to a component in the commit, so it should be relative to the vulnerability reported. The parent commit and child commit of this commit doesn't have any related information about the vulnerability, so I am pretty sure the given fix commit hash is correct.
The score computed still contain a base score which is the same as the oficial one, but as said previously, I've taken temporal metric into account. Here's the explanation about scoring:

- The temporal metrics are the same as the previous questions.
- The WSYIWYG editor is an online only editor, so it's AV:N in both CVSS versions.
- AC in CVSS 3 is Attack Complexity, so it's AC:L as it's quite easy to complete an XSS attack if the victim trigger the attack. However, AC in CVSS 2 is Access Complexity. As there is no point to trigger the attack in attackers' sides, it must have interactions with victim users, thus it's AC:M in CVSS 2 and it also implies a UI:R in CVSS 3.
- The editor doesn't necessarily have authorisation, so it's Au:N in CVSS 2.
- The scope is changed as the attacker should not have any access to the user's scope, but the attack has granted the attacker the same access level as the user, meaning that it can access user's confidential information without prior authorisation from the user. So it's S:C in CVSS 3.
- Similarly, the attacker might share the same access level as the user, but the user won't be able to access the system behind the editor unless it contains other privilege elevation vulnerabilities, but again, this is not related to this vulnerability. So it's C:N in CVSS 2 but C:L in CVSS 3 since it can access user's information but not all information available in the system. As it can take over control of the user's account, it's possible to modify the information of the user's account (probably password etc.) but, again, not the whole system or component. So it's I:P in CVSS 2 and I:L in CVSS 3. XSS usually won't trigger any availability impact, so it's A:N in both CVSS 2 and CVSS 3.

3.3 CVSS 2 vs CVSS 3

Generally, I think CVSS 3 is more specific than CVSS 2 as it is more focusing on a component, but CVSS 2 is just saying the system. While a system can be the component, the component may not be able to affect the system, so CVSS 3 can emphasise more on the impacts of the individual component. This is especially useful for attacks like XSS as it doesn't have too many things about the system, it's all about the user’s data. Also CVSS 3 has listed a little bit more specific metrics Attack Complexity, UI, PR and S to address different types of attack instead of using metrics Access Complexity and Au in CVSS 2 which are too general.