

Writeup -Kinderriegel

Areeb Hussain
Mtr. 3133658

Questions:

1. Gain access to the "Kinderriegel" backend. There are two flags: The first is hidden in the database and the second is displayed on the main page.
2. Compromise the system. The flag can be found in the root directory of the server. Write down your actions.
3. Can you discover a way to bypass the authentication and exploit the vulnerability even if the password has been changed?
4. Show your l33t sk1llz: Can you extract the flag file on the root system without accessing the admin backend? Write down your actions.

Start and compile the container like the previous tasks

\$ sudo docker compose up -d in the container's directory and get the ip address

Ip addr: **172.17.0.3**

Put it into the web, we literally get a picture of animated kinderriegel chocolate nothing more

Lets do a nikto with the command **nikto -h 173.17.0.3** and see what directories are inside

```
+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ DEBUG HTTP verb may show server debugging information. See http://msdn.microsoft.com/en-us/library/e8z01xdh%28VS.80%29.aspx
+ Cookie PHPSESSID created without the httponly flag
+ OSVDB-3092: /admin/: This might be interesting...
+ Server leaks inodes via ETags, header found with file /tools/, fields: 0x7c 0x574284e7ecd00
+ OSVDB-3092: /tools/: This might be interesting...
+ OSVDB-3093: /admin/index.php: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3268: /images/: Directory indexing found.
+ OSVDB-3268: /images/?pattern=/etc/*&sort=name: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 6544 items checked: 0 error(s) and 10 item(s) reported on remote host
+ End Time:      2022-12-01 06:23:32 (GMT1) (22 seconds)
+ 1 host(s) tested
```

1. Gain access to the "Kinderriegel" backend. There are two flags: The first is hidden in the database and the second is displayed on the main page.

Lets look at **tools**, it opens a webpage with dbfrontend which leads to a adminer.php. This page looks exactly like a the adminer.php from the sql injection slides. Interesting lets try to manually brute force with commonly used/default credentials

ure | 172.17.0.3/tools/adminer.php

]

Login

System	MySQL
Server	localhost
Username	
Password	
Database	

☐ Permanent login

As I remember from sql injection, the adminer.php is used to look into the actual databse. After a couple of tries then going back to the slides I tried

username: root

password: root

database: kinderriegel

And I was in

got the flag from the table Read_flag

Language: English ▼

MySQL » [Server](#) » [kinderriegel](#) » Select: Read_Flag

Adminer 4.2.4 4.8.1

Select: Read_Flag

DB: kinderriegel ▼

[SQL command](#) [Import](#)
[Export](#) [Create table](#)

select Read_Flag
select user

Select data

[Show structure](#)

[Alter table](#)

[New item](#)

[Select](#)

[Search](#)

[Sort](#)

Limit

Text length

Action

[Select](#)

SELECT * FROM `Read_Flag` LIMIT 50 (0.001 s) [Edit](#)

<input type="checkbox"/> Modify	flag
<input type="checkbox"/> edit	n0_0n3_th0ught_ab0ut_4dm1n3r

(1 row) ☐ whole result

Modify

[Save](#)

Selected (0)

[Edit](#)

[Clone](#)

[Delete](#)

[Export \(1\)](#)

[Import](#)

Frontend Flag

n0_One_th0ught_ab0ut_4dm1n3r

Looking around there was another table user with two users.

Language: English

Adminer 4.2.4 4.8.1

DB: kinderriegel

SQL command [Import](#)
[Export](#) [Create table](#)

select Read_Flag
select user

MySQL » [Server](#) » [kinderriegel](#) » Select: user

Select: user

[Select data](#) [Show structure](#) [Alter table](#) [New item](#)

[Select](#)

[Search](#)

[Sort](#)

Limit

Text length

Action
[Select](#)

SELECT * FROM `user` LIMIT 50 (0.001 s) [Edit](#)

<input type="checkbox"/> Modify	id	username	password
<input type="checkbox"/> edit	1	manager	make_money_money
<input type="checkbox"/> edit	2	support	turn_it_on_and_off

(2 rows) ☐ whole result

Modify
[Save](#)

Selected (0)
[Edit](#) [Clone](#) [Delete](#)

[Export \(2\)](#)

[Import](#)

Now lets look into the other directory **/admin**. It's a simple admin login page which maybe could be exploited using sql injection

172.17.0.3/admin/

Please log in

Login

As suspected the users table from the database list the admin login credentials.

Enter the credentials

username: manager

password: make_money_money

Kinderriegel Backend

Flag: flag_n1c3_y0u_mad3_1t_t1ll_h3r3

Today is: 2023/01/13.

Sales figures

Year	Sold bars	Used milk (gallons)	Used sugar (pounds)
2012	20 0000	20	400
2013	21 0000	23	623
2014	18 0000	20	423
2015	18 9000	21	433
2016	19 0200	23	213
2017	19 0300	20	220

print sale figures as pdf

Backend Flag

flag_n1c3_y0u_mad3_1t_t1ll_h3r3

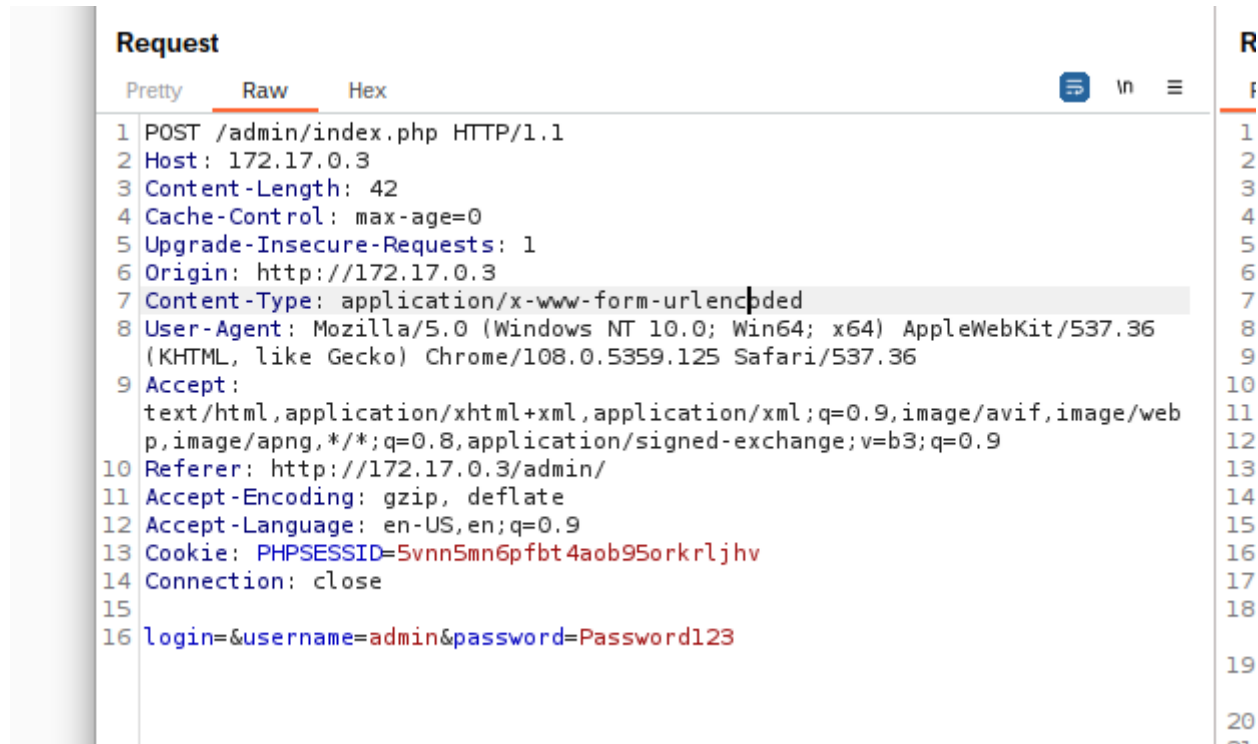
3. Can you discover a way to bypass the authentication and exploit the vulnerability even if the password has been changed?

As I researched online a way to bypass authentication if the password has been changed is by using the session id. But provided that sessid is used in url??

```
Request
Pretty Raw Hex
1 POST /admin/index.php HTTP/1.1
2 Host: 172.17.0.3
3 Content-Length: 51
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://172.17.0.3
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/107.0.5304.107 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
  plication/signed-exchange;v=b3;q=0.9
10 Referer: http://172.17.0.3/admin/index.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=ig2hnnvnei039ldkop4eripudl
14 Connection: close
15
16 login=&username=support&password=turn_it_on_and_off
```

```
9:/var/lib/php/sessions# ls
22vlb3poj0a4k1q sess_b1p4u13qb7ta629k05oq5ng9lt sess_i5jdd29mm566fiq26uavfh9u0h sess_q7q4bfscb6o36rkm3sd9uvqu3
nsog5m9mpdkoiti sess_cjpi6lg3h5k0ogc155helts8gu sess_ig2hnnvnei039ldkop4eripudl sess_rplqvvtus283icbr0qbi57ns7
sil3f7ic4jn3991 sess_e89sndiqnlupun1iq6sbnk4kt sess_juej4ehf4dr0rpa0m2pqrre0 sess_sellp5m34576spbmi3d3hp3d6j
ldb83s8q6jun4br sess_f4egf3u80td0hmvfpia7p4qrmn sess_om3v2jsa70gd91rbsuso0c2h4a sess_t9ajsalptruq5f4l722admams2
```

But just to see if sql injection exploitation can be used on the /admin login page. I will use sqlmap. Simply open Burpsuite and open browser. Enter the /admin page and enter any credentials. Check the burpsuite http history and copy the post request and save it as a txt file



We can provide the data being passed in the POST request body to scan by the SQLMap tool with the following command

\$ sqlmap -r <path to the request> -p username

```
[12:33:40] [INFO] testing 'brute AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to
reduce the number of requests? [Y/n] n
[12:33:58] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[12:34:00] [WARNING] POST parameter 'username' does not seem to be injectable
[12:34:00] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options i
f you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you coul
d try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[12:34:00] [WARNING] your sqlmap version is outdated
```

It seems to be not injectable.

lets try to enumerate the database with the command

\$sqlmap -r /home/areeb/kinderriegel.txt -dbs

```
13:46:43 [WARNING] Most likely web server instance hasn't recovered yet from previous timed based payload. If the problem persists, please wait for a few minutes and rerun without flag 'T' in option '--technique' (e.g. '--flush-session --technique=BEUS') or try to lower the value of option '--time-sec' (e.g. '--time-sec=2')
13:46:44 [WARNING] POST parameter 'login' does not seem to be injectable
13:46:44 [WARNING] POST parameter 'username' does not appear to be dynamic
13:46:44 [WARNING] heuristic (basic) test shows that POST parameter 'username' might not be injectable
13:46:44 [INFO] testing for SQL injection on POST parameter 'username'
13:46:44 [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
13:46:44 [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
13:46:44 [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
13:46:44 [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
13:46:44 [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
13:46:45 [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
13:46:45 [INFO] testing 'Generic inline queries'
13:46:45 [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
13:46:45 [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
13:46:45 [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
13:46:45 [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
13:46:45 [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
13:46:45 [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
13:46:45 [INFO] testing 'Oracle AND time-based blind'
13:46:45 [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
13:46:46 [WARNING] POST parameter 'username' does not seem to be injectable
13:46:46 [WARNING] POST parameter 'password' does not appear to be dynamic
13:46:46 [WARNING] heuristic (basic) test shows that POST parameter 'password' might not be injectable
13:46:46 [INFO] testing for SQL injection on POST parameter 'password'
13:46:46 [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
13:46:46 [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
13:46:46 [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
13:46:46 [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
13:46:46 [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
13:46:46 [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
13:46:46 [INFO] testing 'Generic inline queries'
13:46:46 [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
13:46:46 [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
13:46:46 [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
13:46:46 [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
13:46:46 [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
13:46:47 [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
13:46:47 [INFO] testing 'Oracle AND time-based blind'
13:46:47 [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
13:46:47 [WARNING] POST parameter 'password' does not seem to be injectable
13:46:47 [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
13:46:47 [WARNING] your sqlmap version is outdated
```

Same result

Unfortunately I was not able to get root access