

Твой код глазами хакера

или 5 советов для разработчиков



Disclaimer

1. Вся информация предоставлена в образовательных целях.
2. Автор не несет ответственности за любые специальные, прямые, косвенные, побочные или случайные убытки или любые другие убытки, возникшие в результате или в связи с использованием информации из этого доклада.

Whoami

Kutlymurat Mambetniyazov (@manfromkz)

Cybersecurity expert at BTS Digital, OSCP, eWPTXv2, eCPTXv2

Research: CVE-2022-29938, CVE-2022-29939, CVE-2022-29940,
CVE-2021-34187, CVE-2020-29143, CVE-2020-29142, CVE-2020-29140,
CVE-2020-29139

Blog: <https://murat.one>

Channel: <https://t.me/onebrick>





Однажды зашел на
сайт, и увидел ...







О чем доклад?



#1_start. Секретные ключи

JSON Web Token (JWT) is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key.

Поиграть с JWT можно на <https://jwt.io>



Пример JWT

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjb25mljoiQmVlVGVjaCIsInNwZWFrZXliOiJAbWZnJvbWt6IiwiaXNfYWRtaW4iOiJF9iNQ5iNL4Gu5awCROqEnijqDaJU3eiBEvAusxkchJhhAUU

Первая часть => {"alg":"HS256","typ":"JWT"}

Вторая часть => {"conf":"BeeTech","speaker":"@manfromkz","is_admin":1}

Третья часть => сигнатура токена на основе секретного ключа



jwt java example spring boot



Барлығы

Бейне

Сурет

Басқа

Құралдар

Шамамен 753 000 нәтиже (0,65 секунд)



javainuse.com

https://www.javainuse.com > spring ▾ Осы бетті аудар

Spring Boot Security + JWT Hello World Example - JavainUse

In this **tutorial** we will be developing a **Spring Boot** Application to secure a REST API with **JSON Web Token (JWT)**. We will be generating a **JWT** and allowing ...

[Spring Boot + JWT + MYSQL...](#) · [RestTemplate](#) · [Online Bcrypt Generator](#)

Басқа адамдардың іздейтіндері:



spring security jwt token example

jwt authentication spring boot

spring-security-jwt spring boot 2

generate jwt token java example


how to validate jwt token in spring boot

jwt token spring boot microservices example

#1. JWT. Поиск примера реализации в Google

- Define the application.properties. As see in [previous JWT tutorial](#), we specify the secret key using which we will be using for hashing algorithm. The secret key is combined with the header and the payload to create a unique hash. We are only able to verify this hash if you have the secret key.

```
jwt.secret=javainuse
```



- JwtTokenUtil



"jwt.secret=javainuse"



 Барлығы

 Бейне

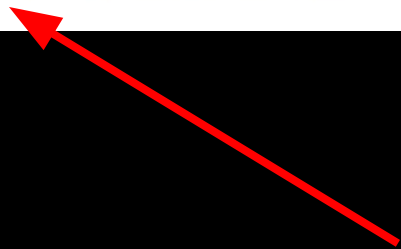
 Сурет

 Карта

 Басқа

Құралдар

Шамамен 163 нәтиже (0,30 секунд)



#1. JWT. Поиск секрета в Google

master IQproject / src / main / resources / application.properties

OlhaLozinska Added spring security (with JWT)

Latest co

1 contributor

13 lines (9 sloc) | 493 Bytes

```
1  ## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
2  spring.datasource.url=jdbc:postgresql://localhost:5432/IQdatabase
3  spring.datasource.username= postgres
4  spring.datasource.password=root
5
6  # The SQL dialect makes Hibernate generate better SQL for the chosen database
7  spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
8
9  # Hibernate ddl auto (create, create-drop, validate, update)
10 spring.jpa.hibernate.ddl-auto = update
11
12
13 jwt.secret=javainuse
```



#1. JWT. Секрет используется в репозиториях на Github

Словарь секретов

The goal for this project was to find as many public-available JWT secrets as possible to help developers and DevOpses identify it by traffic analysis at the Wallarm NGWAF level.

For now (10/02/2020) the list consists of 3502

We focused on Google search and GitHub dorks by using mainly two query patterns:

1. `jwt example +TECHNOLOGY` where the `TECHNOLOGY` is the language itself like PHP, Ruby, Rails, or framework like ExpressJS, Struts or Flask.
2. Google BigQuery search based on 3M GitHub projects

<https://github.com/wallarm/jwt-secrets>



```
3356 XY7kmzoNz1100
3357 xyz123
3358 xz?_fd-BE\#:K/+Fb[b7DQ'gPhWW^fU!7Hk$G7+7eX$EWK_N-r
3359 xz?_fd-BE\\#:K/+Fb[b7DQ'gPhWW^fU!7Hk$G7+7eX$EWK_N-r
3360 Y^$B,FVseZ~d,+mrP,#@DM4:U~BU6.=m9QDD+.U\'yL"bp-hTS#H+f;UR(+<xS%Hh
3361 y0urS3cr311<ey
3362 Y2InZKhVNuxi7HXsA2Bq1nCDqWQ9XLQH
3363 y4bqQ3ehGtb9swtNyObsi9wxrreff4VH
3364 y5oEr+Hcmnjff1soU4v0fw==
3365 yarimasune
3366 yassine
3367 yDzK7Cp1RhrL7AwiparOmDbcd9jtH3cLGJypRUUrwagibVc42kmBe6rFiBmZzOJ5
3368 yeiyeyeyeyeyeyeyeyeyeyeyeyeyeyeyei
3369 yetgTXUzFG3sLviN
3370 yeWAgVDfb$!MFn@MCJVN7uqkznHbDLR#
```

#1. JWT. Словарь секретов

Примеры с Hackerone

Mail.ru - <https://hackerone.com/reports/896649>

Trint - <https://hackerone.com/reports/638635>



#1. JWT. Ошибаются даже разработчики крупных компаний

#1_end. Вывод

- Секретный ключ не секретный, если его знает весь интернет.
- На каждую среду должен быть свой секретный ключ (dev, stage, prod).



#2_begin. Кодогенерация

CRUD (Create, Read, Update, Delete) - объединенное название нескольких операций работы над данными.

Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)

```
SELECT * FROM `users`
```

Profiling [\[Edit inline\]](#)

Show all | Number of rows: Filter rows: Sort by key:

+ Options

| | | id | fullname | department_id | role_id | is_active | email | phone | password |
|--------------------------|--|----|----------|---------------|---------|-------------|--------|---------|------------------|
| | | | ФИО | Департамент | Роль | Активирован | Почта | Телефон | Пароль |
| <input type="checkbox"/> | | 1 | admin | 1 | 1 | yes | admin | NULL | \$2y\$10\$BohLyc |
| <input type="checkbox"/> | | 5 | admin4 | 1 | 4 | yes | admin4 | admin4 | \$2y\$10\$phzxOg |
| <input type="checkbox"/> | | 6 | admin5 | 1 | 5 | yes | admin5 | admin5 | \$2y\$10\$WkIfQu |

Пример генерации кода в Yii2

```
$query->andWhere(['like', 'fullname', $this->fullname])
->andWhere(['like', 'is_active', $this->is_active])
->andWhere(['like', 'email', $this->email])
->andWhere(['like', 'phone', $this->phone])
->andWhere(['like', 'password', $this->password]);
```

[https://testsite.kz/users/index?UsersSearch\[PARAM\]=VALUE](https://testsite.kz/users/index?UsersSearch[PARAM]=VALUE)

testsite.kz/users/index?UsersSearch[name]=admin

Apps

NitroTest

Главная / Пользователи

Пользователи

Добавить пользователя

Showing 1-1 of 1 item.

| # | Почта | ФИО |
|---|-------|-------|
| 1 | admin | admin |

#2. Кодогенерация. Фильтрация по имени `UserSearch[name]=admin`

testsite.kz/users/index?UsersSearch[password]=\$2y\$10\$Bo

Apps

NitroTest

[Главная](#) / [Пользователи](#)

Пользователи

[Добавить пользователя](#)

Showing 1-1 of 1 item.

| # | Почта | ФИО |
|---|----------------------|----------------------|
| | <input type="text"/> | <input type="text"/> |
| 1 | admin | admin |

#2. Кодогенерация. Фильтрация по паролю UserSearch[password]=\$2y\$...

#2_end. Вывод

Модели поиска нужно редактировать самостоятельно, убирая из фильтров параметры чувствительных данных, особенно если их поиск не требуется



#3_begin. SQL-инъекции

SQL-инъекция - это техника, при которой злоумышленник использует недостатки в коде приложения, отвечающего за построение динамических SQL-запросов.

Пример уязвимого кода:

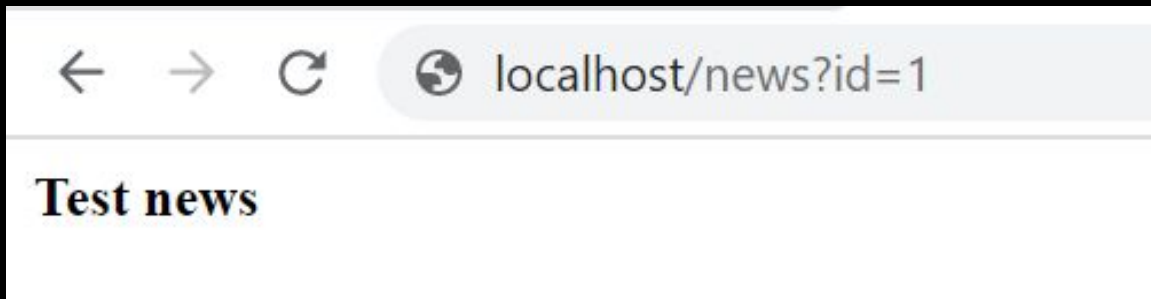
```
results, err := db.Query("SELECT title FROM news WHERE id = " + id)
```

https://localhost/news?id=1

```
results, err := db.Query("SELECT title FROM news WHERE id = " + id)
```

=>

```
SELECT title FROM news WHERE id = 1
```



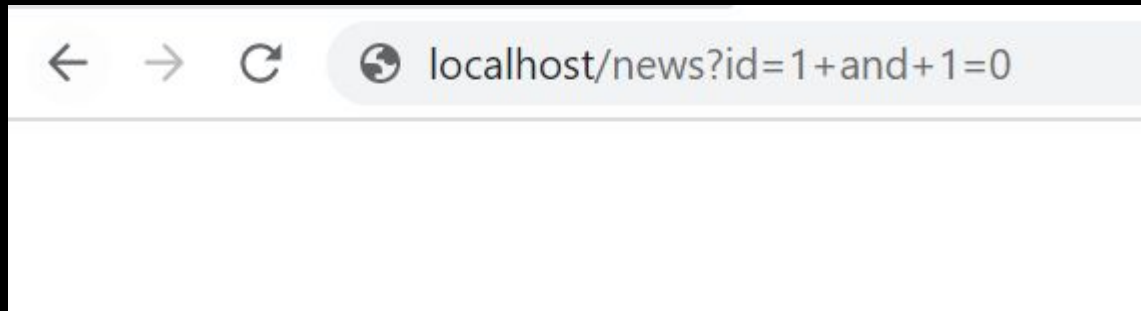
#3. SQL-инъекции. Что это такое?

<https://localhost/news?id=1+and+1=0>

```
results, err := db.Query("SELECT title FROM news WHERE id = " + id)
```

=>

```
SELECT title FROM news WHERE id = 1 and 1 = 0
```



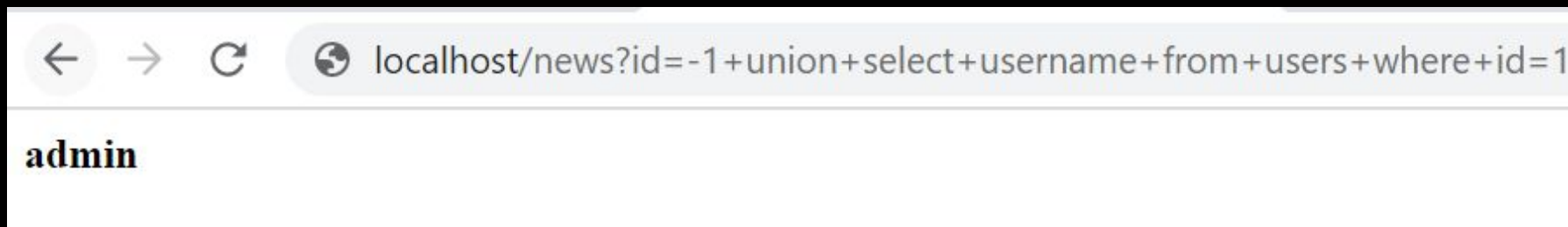
#3. SQL-инъекции. Что это такое?

<https://localhost/news?id=-1+union+select+username+from+users+where+id=1>

```
results, err := db.Query("SELECT title FROM news WHERE id = " + id)
```

=>

```
SELECT title FROM news WHERE id = -1 UNION SELECT username FROM users  
WHERE id = 1
```



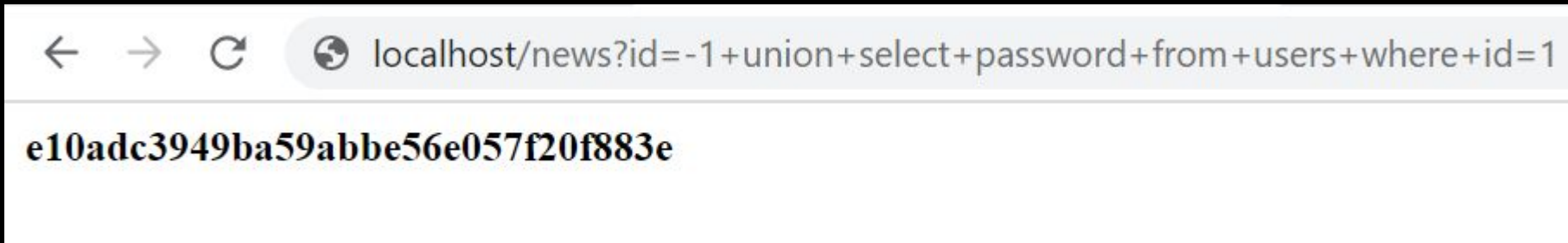
#3. SQL-инъекции. Что это такое?

<https://localhost/news?id=-1+union+select+password+from+users+where+id=1>

```
results, err := db.Query("SELECT title FROM news WHERE id = " + id)
```

=>

```
SELECT title FROM news WHERE id = -1 UNION SELECT password FROM users  
WHERE id = 1
```



#3. SQL-инъекции. Что это такое?

Почему это актуально?

Самые топовые SQL-инъекции (261 шт.) с Hackerone:

https://github.com/reddelexc/hackerone-reports/blob/master/tops_by_bug_type/TOPSQLI.md

Неполный список компаний, у которых нашли SQLi:

Starbucks, Razer, Mail.ru, QIWI, Acronis, Zomato, Uber,
OLX, GitLab, IBM, ...

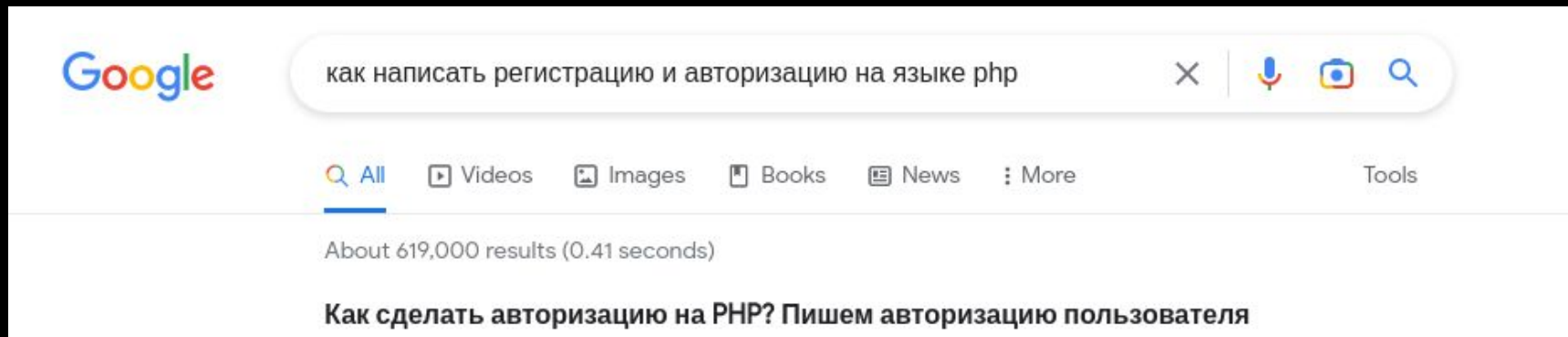


1. SQL Injection Extracts Starbucks Enterprise Accounting, Financial, Payroll Database to Starbucks - 739 upvotes, \$4000
2. SQL injection in https://labs.data.gov/dashboard/datagov/csv_to_json via User-agent to GSA Bounty - 666 upvotes, \$2000
3. Time-Based SQL injection at city-mobil.ru to Mail.ru - 622 upvotes, \$15000 ←
4. SQL injection at <https://sea-web.gold.razer.com/ajax-get-status.php> via txid parameter to Razer - 580 upvotes, \$2000
5. SQL Injection in <https://api-my.pay.razer.com/inviteFriend/getInviteHistoryLog> to Razer - 528 upvotes, \$2000
6. SQL injection on contactws.contact-sys.com in TScenObject action ScenObjects leads to remote code execution to QIWI - 4 \$5500
7. Blind SQL Injection to InnoGames - 432 upvotes, \$2000
8. SQL injection at fleet.city-mobil.ru to Mail.ru - 369 upvotes, \$10000 ←
9. SQL Injection in report_xml.php through countryFilter[] parameter to Valve - 347 upvotes, \$25000 ←
10. [windows10.hi-tech.mail.ru] Blind SQL Injection to Mail.ru - 327 upvotes, \$5000
11. SQL Injection on cookie parameter to MTN Group - 300 upvotes, \$0

#3. SQL-инъекции. Несколько примеров из практики гигантов

Гугл не в помощь

Первое, что делают веб-разработчики, изучая какой-либо язык, ищут в Google “как написать регистрацию и авторизацию на языке X”



#3. SQL-инъекции. Не верь туториалам

1

А теперь рассмотрим все функции, вызываемые в коде. Вот функция входа на сайт:

```
function enter ()
{
$error = array(); //массив для ошибок
if ($_POST['login'] != "" && $_POST['password'] != "") //если поля заполнены

{
    $login = $_POST['login'];
    $password = $_POST['password'];

    $rez = mysql_query("SELECT * FROM users WHERE login=$login"); //запрос
    if (mysql_num_rows($rez) == 1) //если нашлась одна строка, значит такой

    {
        $row = mysql_fetch_assoc($rez);
```

#3. SQL-инъекции. Не верь туториалам. Пример уязвимого кода

1

А теперь рассмотрим все функции, вызываемые в коде. Вот функция входа на сайт:

```
function enter ()
{
$error = array(); //массив для ошибок
if ($_POST['login'] != "" && $_POST['password'] != "") //если поля заполнены

{
    $login = $_POST['login'];
    $password = $_POST['password'];

    $rez = mysql_query("SELECT * FROM users WHERE login=$login"); //запрос
    if (mysql_num_rows($rez) == 1) //если нашлась одна строка, значит такой

    {
        $row = mysql_fetch_assoc($rez);
    }
}
```

#3. SQL-инъекции. Не верь туториалам. Пример уязвимого кода

2

```
<?php
if (isset($_POST['login'])) { $login = $_POST['login']; if ($login == '') { unset($login);} } //записываем введенный
if (isset($_POST['password'])) { $password=$_POST['password']; if ($password == '') { unset($password);} }
//записываем введенный пользователем пароль в переменную $password, если он пустой, то уничтожаем переменную
if (empty($login) or empty($password)) //если пользователь не ввел логин или пароль, то выдаем ошибку и останавливаем
{
    exit ("Вы ввели не всю информацию, вернитесь назад и заполните все поля!");
}
//если логин и пароль введены, то обрабатываем их, чтобы теги и скрипты не работали, мало ли что люди могут ввести
$login = stripslashes($login);
$login = htmlspecialchars($login);
$password = stripslashes($password);
$password = htmlspecialchars($password);
//удаляем лишние пробелы
$login = trim($login);
$password = trim($password);
// подключаемся к базе
include ("bd.php");// файл bd.php должен быть в той же папке, что и все остальные, если это не так, то просто изме
// проверка на существование пользователя с таким же логином
$result = mysql_query("SELECT id FROM users WHERE login='$login'", $db);
```

#3. SQL-инъекции. Не верь туториалам. Пример уязвимого кода

2

```
<?php
if (isset($_POST['login'])) { $login = $_POST['login']; if ($login == '') { unset($login);} } //записываем введенный
if (isset($_POST['password'])) { $password=$_POST['password']; if ($password == '') { unset($password);} }
//записываем введенный пользователем пароль в переменную $password, если он пустой, то уничтожаем переменную
if (empty($login) or empty($password)) //если пользователь не ввел логин или пароль, то выдаем ошибку и останавливаем
{
    exit ("Вы ввели не всю информацию, вернитесь назад и заполните все поля!");
}
//если логин и пароль введены, то обрабатываем их, чтобы теги и скрипты не работали, мало ли что люди могут ввести
$login = stripslashes($login);
$login = htmlspecialchars($login);
$password = stripslashes($password);
$password = htmlspecialchars($password);
//удаляем лишние пробелы
$login = trim($login);
$password = trim($password);
// подключаемся к базе
include ("bd.php");// файл bd.php должен быть в той же папке, что и все остальные, если это не так, то просто изме
// проверка на существование пользователя с таким же логином
$result = mysql_query("SELECT id FROM users WHERE login='$login'", $db);
```

#3. SQL-инъекции. Не верь туториалам. Пример уязвимого кода

3

```
// проверяем, не существует ли пользователя с таким именем
$query = mysqli_query($link, "SELECT user_id FROM users WHERE user_login='".mysqli_real_escape_string($link, $
if(mysqli_num_rows($query) > 0)
{
    $err[] = "Пользователь с таким логином уже существует в базе данных";
}

// Если нет ошибок, то добавляем в БД нового пользователя
if(count($err) == 0)
{

    $login = $_POST['login'];

    // Убираем лишние пробелы и делаем двойное хеширование
    $password = md5(md5(trim($_POST['password'])));

    mysqli_query($link, "INSERT INTO users SET user_login='".$login."', user_password='".$password.'");
    header("Location: login.php"); exit();
}
```

#3. SQL-инъекции. Не верь туториалам. Пример уязвимого кода


3

```
// проверяем, не существует ли пользователя с таким именем
$query = mysqli_query($link, "SELECT user_id FROM users WHERE user_login='".mysqli_real_escape_string($link, $_POST['login'])."'");
if(mysqli_num_rows($query) > 0)
{
    $serr[] = "Пользователь с таким логином уже существует в базе данных";
}

// Если нет ошибок, то добавляем в БД нового пользователя
if(count($serr) == 0)
{
    $login = $_POST['login'];

    // Убираем лишние пробелы и делаем двойное хеширование
    $password = md5(md5(trim($_POST['password'])));

    mysqli_query($link, "INSERT INTO users SET user_login='".$login."', user_password='".$password.'");
    header("Location: login.php"); exit();
}
```



#3. SQL-инъекции. Не верь туториалам. Пример уязвимого кода

3

```
// проверяем, не существует ли пользователя с таким именем
$query = mysqli_query($link, "SELECT user_id FROM users WHERE user_login='".mysqli_real_escape_string($link, $
if(mysqli_num_rows($query) > 0)
{
    $err[] = "Пользователь с таким логином уже существует в базе данных";
}

// Если нет ошибок, то добавляем в БД нового пользователя
if(count($err) == 0)
{
    $login = $_POST['login'];

    // Убираем лишние пробелы и делаем двойное хеширование
    $password = md5(md5(trim($_POST['password'])));

    mysqli_query($link, "INSERT INTO users SET user_login='".$login."', user_password='".$password.'");
    header("Location: login.php"); exit();
}
```

#3. SQL-инъекции. Не верь туториалам. Пример уязвимого кода

Промежуточный вывод

- Тutorials не всегда содержат безопасный код



#3. SQL-инъекции. Не верь туториалам

Сырые запросы

У многих была ситуация, когда SQL-запрос очень громоздкий, и его очень сложно (лень) писать через ORM фреймворка.

Запрос:

```
SELECT `id`, `email` FROM `user` WHERE `last_name` = 'Smith' LIMIT 10
```

ORM:

```
$rows = (new \yii\db\Query()) ->select(['id', 'email']) ->from('user')  
->where(['last_name' => 'Smith']) ->limit(10) ->all();
```

Сырые запросы

У многих была ситуация, когда SQL-запрос очень громоздкий, и его очень сложно (лень) писать через ORM фреймворка.

Запрос:

```
SELECT `id`, `email` FROM `user` WHERE `last_name` = 'Smith' LIMIT 10
```

ORM:

```
$rows = (new \yii\db\Query()) ->select(['id', 'email']) ->from('user')  
->where(['last_name' => 'Smith']) ->limit(10) ->all();
```

Сырые запросы

У многих была ситуация, когда SQL-запрос очень громоздкий, и его очень сложно (лень) писать через ORM фреймворка.

Запрос:

```
SELECT `id`, `email` FROM `user` WHERE `last_name` = 'Smith' LIMIT 10
```

ORM:

```
$rows = (new \yii\db\Query()) ->select(['id', 'email']) ->from('user')  
->where(['last_name' => 'Smith']) ->limit(10) ->all();
```


Пример с OpenEMR (CVE-2020-29142)

Функция `add_escape_custom()`.

```
function add_escape_custom($s)
{
    //prepare for safe mysql insertion
    $s = mysqli_real_escape_string($GLOBALS['dbh'], ($s ?? ''));
    return $s;
}
```



Разработчики OpenEMR используют эту функцию для обезопасывания своих SQL-запросов.

#3. SQL-инъекции. Сырые запросы

mysql*_real_escape_string()

Цитата:

Characters encoded are NUL (ASCII 0), \n, \r, \, ', ", and Control-Z.

Из документации:

<https://www.php.net/manual/en/mysqli.real-escape-string.php>



Неправильное использование

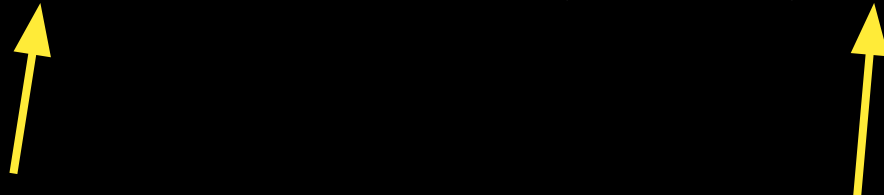
```
$q = ' SELECT a, b FROM rec WHERE id = ' . add_escape_custom($_GET['id']);
```

```
SELECT a, b FROM recs WHERE id = -1 union select password, ip from users limit 0,1
```

Вернет пароли пользователей.

Правильное использование

```
$q = ' SELECT a, b FROM rec WHERE id= "' . add_escape_custom($_GET['id']) . '" ';
```



```
SELECT a, b FROM records WHERE id = "-1 union select password, ip from users  
limit 0,1"
```

Вернет пустой результат.

Заметка по GORM

```
id := c.Query("id")
```

```
dbms.First(&user, id) // неправильно
```

```
dbms.First(&user, "id = ?", id) // правильно
```

Способ инъекции: `http://host/user?id=id=1)) or 1=1--`

Обсуждение: <https://github.com/go-gorm/gorm/issues/2517>



Заметка по GORM

PHP Yii2: `User::findOne($id);`

PHP Laravel: `Category::find($id);`

PHP Custom: `User->first($email);`

#3_end. Вывод

- В документации на официальном сайте PHP нет примеров по правильному и неправильному использованию функции, так как считается, что функция `mysql*_real_escape_string()` должна использоваться только для строк.
- Всегда нужен дополнительный ресерч и тест по используемым функциям.



#4_begin. Фильтры

Golang - <https://pkg.go.dev/net/mail#ParseAddress>

PHP - <https://www.php.net/manual/en/filter.filters.validate.php>

Python - <https://docs.python.org/3/library/email.utils.html#email.utils.parseaddr>

Wikipedia - https://en.wikipedia.org/wiki/Email_address

RFC - <https://rfc-editor.org/rfc/rfc5322.html>



`example-indeed@strange-example.com`

`test/test@test.com` (slashes are a printable character, and allowed)

`admin@mailserver1` (local domain name with no TLD, although ICANN highly discourages)

`example@s.example` (see the [List of Internet top-level domains](#))

`" @example.org` (space between the quotes)

`"john..doe"@example.org` (quoted double dot)

`mailhost!username@example.org` (bangified host route used for uucp mailers)

`"very.(),:;<>[]\".VERY.\"very@\\ \"very\".unusual"@strange.example.com`

(`very` and `unusual` are not valid domain names, the first one being double quoted)

`user%example.com@example.org` (% escaped mail route to `user@example.com` via example.com)

`user-@example.org` (local part ending with non-alphanumeric character from the list of

reserved characters)

`postmaster@[123.123.123.123]` (IP addresses are allowed instead of domains when

using the `mailto:` scheme)

`postmaster@[IPv6:2001:0db8:85a3:0000:0000:8a2e:0370:7334]` (IPv6 addresses are allowed



#4. Фильтры. Валидные e-mail

test@murat.one =



test+test1@murat.one

test+qwe@murat.one

test+sdfgdfghdfg@murat.one

test+anything@murat.one



тег/категория письма

#4. Фильтры. Валидные e-mail

```
1 package main
2
3 import (
4     "fmt"
5     "log"
6     "net/mail"
7 )
8
9 func main() {
10     e, err := mail.ParseAddress("test@murat.one")
11     if err != nil {
12         log.Fatal(err)
13     }
14
15     fmt.Println(e.Name, e.Address)
16
17 }
```

#4. Фильтры. Стандартный фильтр Golang

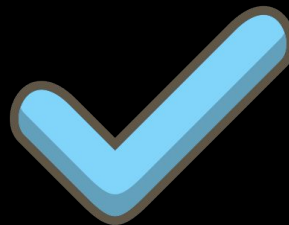
Golang

test@murat.one

test+test@murat.one

test{{7*7}}test@murat.one

te'+union+select+1--+g't@murat.one



Python

```
1 from email.utils import parseaddr
2 print(parseaddr('test@murat.one'))
```

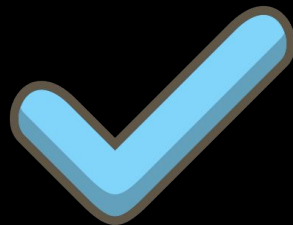


```
(' ', 'test@murat.one')
```

```
from email.utils import parseaddr
print(parseaddr('test@murat.one'))
print(parseaddr('tes"<svg/onload=alert(1)>"t@murat.one'))
print(parseaddr('tes"+union+select+1,2,3--+d"t@murat.one'))
print(parseaddr('test{{7*7}}test@murat.one'))
```



```
('', 'test@murat.one')
('', 'tes"<svg/onload=alert(1)>"t@murat.one')
('', 'tes"+union+select+1,2,3--+d"t@murat.one')
('', 'test{{7*7}}test@murat.one')
```



#4. Фильтры. Стандартный фильтр Python

Сюрприз от Gmail

Dots don't matter in Gmail addresses

If someone accidentally adds dots to your address when emailing you, you'll still get that email. For example, if your email is **johnsmith@gmail.com**, you own all dotted versions of your address:

- **john.smith@gmail.com**
- **jo.hn.sm.ith@gmail.com**
- **j.o.h.n.s.m.i.t.h@gmail.com**

Note: If you use Gmail through work, school, or other organization (like yourdomain.com or yourschool.edu), dots do change your address. To change the dots in your username, contact your [admin](#).

#4_end. Вывод

- Для e-mail лучше использовать фильтры с белым ограниченным списком символов
- Документации оставляют ссылку на RFC стандарт, который никто читать не будет, вместо того, чтобы напрямую писать о рисках



#5_begin. Отладка

PHP - <https://www.php.net/manual/en/function.phpinfo.php>

Django DEBUG=true -

<https://docs.djangoproject.com/en/4.1/ref/settings/#debug>



PHP

```
<?php phpinfo(); ?>
```

- Позволяет временно загружать файлы в папку %TMP%
- Позволяет смотреть защищенные cookies
- Выдает информацию по установленным модулям и ОС
- Выдает настоящий IP-адрес сервера

#5. Отладка. Коварный phpinfo()

```
POST /phpinfo.php HTTP/1.0
Content-Type: multipart/form-data; boundary=-----7db268605ae
Content-Length: 196

-----7db268605ae
Content-Disposition: form-data; name="dummyname"; filename="test.txt"
Content-Type: text/plain

Security Test
-----7db268605ae
```

PHP Variables

| Variable | Value |
|----------------------------------|--|
| <code>_FILES["dummyname"]</code> | Array ([name] => test.txt [type] => text/plain [tmp_name] => /tmp/phpjcmFa3 [error] => 0 [size] => 13 |

Django debug

Page not found (404)

Request Method: GET

Request URL: http://

Using the URLconf defined in `mdl_reports.urls`, Django tried these URL patterns, in this order:

1. `^api/1/`
2. `^api/1/`
3. `^api/1/`
4. `^admin/`
5. `^version/$`
6. `^api/1/`
7. `^docs/`

The empty path didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

Django debug + Sentry = hack

SENTRY_OPTIONS

```
{'cache.backend': 'sentry.cache.redis.RedisCache',  
'cache.options': {},  
'redis.options': {'hosts': {0: {'host': '127.0.0.1',  
                                'password': 'g7dRA06BjTXMtP3iXGJjrSkz2',  
                                'port': 6379}}}},  
'system.databases': {'default': {'ATOMIC_REQUESTS': False,  
                                  'AUTOCOMMIT': True,  
                                  'CONN_MAX_AGE': 0,  
                                  'ENGINE': 'sentry.db.postgres',  
                                  'HOST': 'localhost',  
                                  'NAME': 'sentry',  
                                  'OPTIONS': {},  
                                  'PASSWORD': u'*****',  
                                  'PORT': '',  
                                  'TEST_CHARSET': None,  
                                  'TEST_COLLATION': None,  
                                  'TEST_MIRROR': None,  
                                  'TEST_NAME': None,  
                                  'TIME_ZONE': 'UTC',  
                                  'USER': 'sentry'}}},  
'system.debug': True,  
'system.secret-key': 'c7f3a64aa184b7cbb1a7cbe9cd544913'}
```

Django debug + Sentry = hack

- Полученный секретный ключ позволяет подписать cookies, в том числе с вредоносным сериализованным объектом (PickleSerializer)
- В сериализованный объект можно записать любую команду, например `os.system("cat /etc/passwd")`
- Хакер заработал за такую находку на серверах Facebook 5000\$ - <https://blog.scrt.ch/2018/08/24/remote-code-execution-on-a-facebook-server/>



#5_end. Выводы

Отключаем режим отладки везде, если это не dev-среда



ИСТОЧНИКИ

<https://murat.one/?p=41>

<https://murat.one/?p=90>

https://en.wikipedia.org/wiki/JSON_Web_Token

<https://support.google.com/mail/answer/7436150>

<https://insomniasec.com/downloads/publications/LFI%20With%20PHPInfo%20Assistance.pdf>

<https://medium.com/@syedabuthahir/django-debug-mode-to-rce-in-microsoft-acquisition-189d27d08971>

<https://spy-soft.net/django-website-hack/>

<https://www.php.net/manual/ru/security.database.sql-injection.php>



Спасибо за внимание!

