



WAPT

Web Application Penetration Testing



5.1 Активный фаззинг веб-приложений

Оглавление

Burp Suite	4
Dirb	7
Gobuster	9
Wfuzz	12
Фаззинг директорий, файлов и параметров в url	13
Фаззинг POST-запросов	15
Фаззинг Куков	15
Фаззинг Хедеров	15
Ffuf	18
Фаззинг директорий и файлов	19
Брут формы авторизации	20
Фаззинг GET-параметров	20
Dirsearch	21
Словари	23
Вывод	25
Fuzzing tools	25

Впервые термин «фаззинг» появился в работе Барта Миллера под названием «The Fuzz Generator» в 1988.

Выделяется три подхода к выявлению недостатков системы: тестирование методом черного, серого и белого ящиков. Различие между ними определяется теми ресурсами, которые доступны во время тестирования.

Метод черного ящика – чаще всего используется при работе с удаленными веб-сервисами или веб-приложениями. При этом данные на входе могут подаваться в виде запросов, а на выходе получают какие-то веб-страницы или значения, с которыми и продолжается работа.

Проводить такое тестирование вручную, без использования автоматизации, обычно не очень хорошее решение. Но его можно использовать, например, при свипинге (sweeping) – процессе поиска похожих уязвимостей в различных приложениях.

Одним из самых популярных на данный момент фаззеров для тестирования черного ящика является OWASP JBroFuzz. Он бесплатен, но при этом позволяет проводить самые различные проверки: на межсайтовый скриптинг (XSS), на SQL-инъекции, на переполнение буфера, целочисленное переполнение и пр. Он работает с самыми популярными сетевыми протоколами: HTTP, SOAP, XML, LDAP и др.

Метод серого ящика – представляет собой комбинацию из метода черного ящика и восстановления кода (RCE – reverse code engineering). Сложно переоценить наличие исходного кода для тестирования безопасности, но даже если исходного кода нет – не все потеряно.

Метод белого ящика – может применяться только в том случае, если доступен сам исходный код. Как и для метода черного ящика, проверка может выполняться вручную или с помощью инструментов. Но, как и в черно ящичном случае – проверка вручную трудоемкая и долгая.

Для проверки исходного кода используются: средства проверки на этапе компиляции, браузеры исходного кода и автоматические инструменты проверки исходного кода.

Средства проверки на этапе компиляции обычно уже встроены в компиляторы и ищут недостатки после создания кода.

Браузеры исходного кода созданы для того, чтобы облегчать мануальный анализ кода.

По методу управления данными фаззинг можно разделить на генерацию и мутацию.

- Генерация – использование случайного набора данных выдавая их за конкретные типы;
- Мутация – внесение изменений в валидные данные.

Также фаззеры можно разделить по «интеллекту»:

- dump фаззер – ничего не знает о структуре данных;
- smart фаззер – имеет некоторое представление о структуре данным и может производить манипуляции.

Активным фаззингом чаще всего называют автоматическое или полуавтоматическое тестирование программ (в нашем случае веб-приложений). Суть данного метода в том, чтобы отправить приложению какие-либо данные (например, заведомо некорректные) и посмотреть, как оно отреагирует. На практике это может быть тестирование параметров в запросах к web-серверу, дирбастинг (перебор директорий или файлов) и автоматическое тестирование на такие уязвимости как ххе, cmd injection, sql injection (да-да, sqlmap – тоже фаззер).

Вариаций фаззинга веб приложения может быть различное множество, главное – найти точки соприкосновения. Само же тестирование можно вести как самописным, так и специально заточенными под это программами. Ниже будет рассмотрено несколько популярных утилит для фаззинга, и затронута тема о том, где именно искать точки соприкосновения.

Burp Suite

Без этой утилиты не обходится, пожалуй, ни одно глубокое тестирование веб приложений. С помощью неё мы можем манипулировать http-запросами так, как захотим.

Рассмотрим работу с burp suite repeater на примере двух самых часто используемых методов http-протокола: GET и POST.

После запуска burp запросы перехватываются и сохраняются во вкладке proxy > history. Там мы можем посмотреть все сделанные запросы и ответы на них. Но сейчас нас интересует утилита repeater. В proxy > history выбираем нужный нам запрос, жмём правую кнопку мыши и выбираем «send to repeater». Во вкладке Repeater появится точно такой же запрос (Рис. 1):

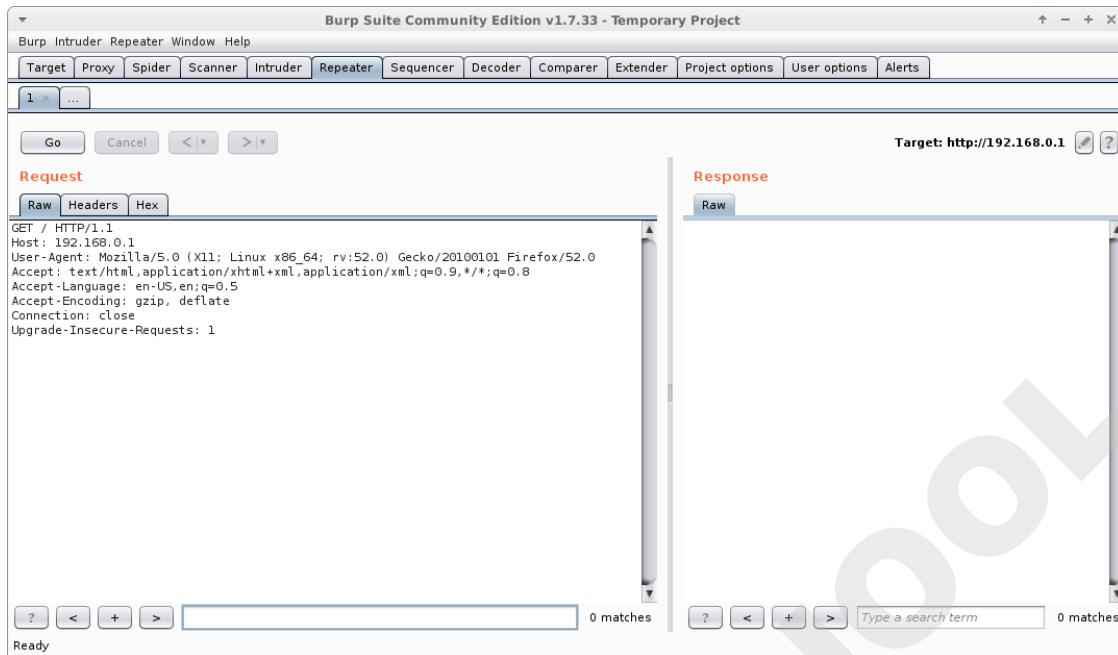


Рис. 1. Отправка перехваченного запроса в Repeater

Это крайне удобно при ручном фаззинге или тестировании какой-либо уязвимости, ведь теперь мы можем изменять запрос так, как захотим, отправлять его на сервер и сразу же анализировать ответ.

При настоящем тестировании веб приложения стоит «погулять» по сайту через браузер, перехватывая все запросы в burp suite. После чего мы можем посмотреть историю в proxy > history и выбрать интересующий нас запрос (чаще всего – запрос с какими либо параметрами)

Рассмотрим запрос с параметрами (Рис. 2):

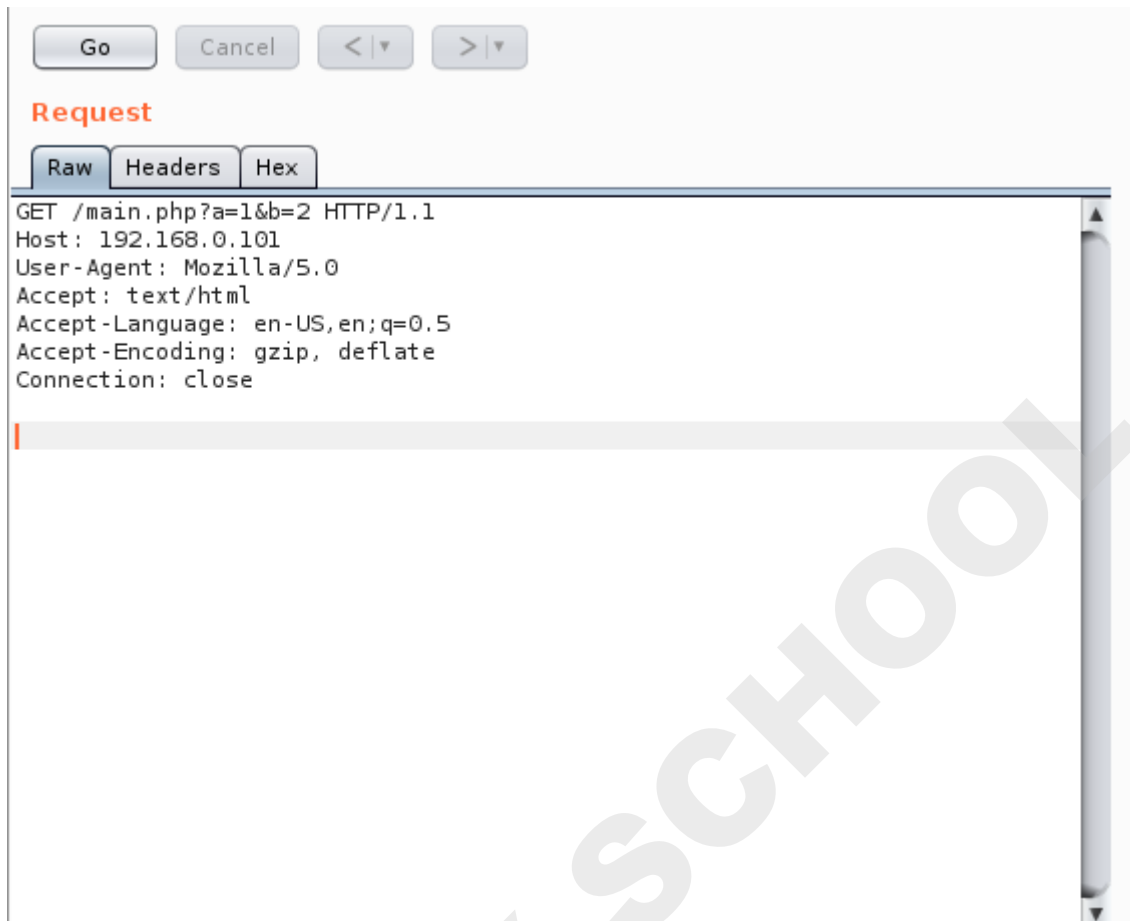


Рис. 2. GET-запрос с параметрами

В этом запросе, как и в прошлом, мы можем полностью его контролировать и изменять. Но нужно уделить особое внимание параметрам *a* и *b*, так как в них будут передаваться данные, отвечающие за логику работы веб приложения.

POST-запросы (Рис. 3), в отличие от GET, передают данные в теле, а не в URL. Но суть остаётся той же: если мы нашли какие-либо интересные параметры, то стоит остановиться на них и протестировать.

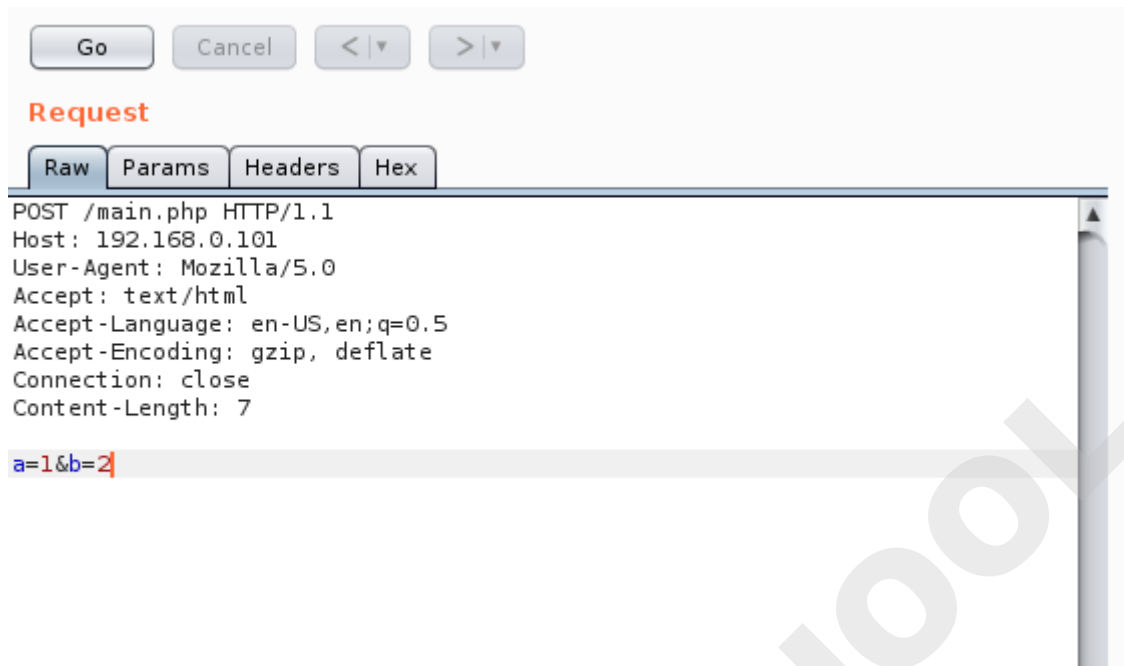


Рис. 3. POST-запрос с параметрами

В Burp suite есть утилита для автоматического фаззинга – Intruder. При помощи ее можно фаззить параметры, куки, искать директории и файлы, брутить формы авторизации, перебирать пейлоады для различных уязвимостей, раскручивать слепые sql-инъекции и многое другое. Но в community версии сильно урезаны возможности (специально занижена скорость перебора), поэтому мы не будем останавливаться на этом инструменте, а рассмотрим аналоги далее по тексту.

Dirb

Перебор директорий – это тоже фаззинг. DIRB – это сканер веб-контента. Он ищет существующие (возможно, скрытые) веб-объекты. В основе его работы лежит поиск по словарю, он формирует запросы к веб-серверу и анализирует ответ.

DIRB поставляется с набором настроенных на атаку словарей для простого использования, но вы можете использовать и ваш собственный список слов. Также иногда DIRB можно использовать как классический CGI сканер, но помните, что в первую очередь это сканер содержимого, а не сканер уязвимостей.

Главная цель DIRB – это помочь профессионалам в аудите веб-приложений. Особенно в тестах, ориентированных на безопасность. Она покрывает некоторые дыры, не охваченные классическими

сканерами веб-уязвимостей. DIRB ищет специфические веб-объекты, которые другие сканеры CGI не ищут. Она не ищет уязвимости и не ищет веб-содержимое, которое может быть уязвимым (Рис. 4).

Справка по DIRB

Использование:

`dirb <базовый_адрес> [<список(и)_словарей>] [опции]`

Примечание

<базовый_адрес> : Базовый URL для сканирования. (Используйте -resume для возобновления сессии)

<список(и)_словарей> : Списки слов.

(список_слов1,список_слов2,список_слов...)

Горячие клавиши

'n' -> Перейти к следующей директории.

'q' -> Остановить сканирование. (Сохранить состояние для возобновления)

'r' -> Сохранить статистику сканирования.

Опции

-a <строка_агента> : Задайте ваш пользовательский USER_AGENT.

-c <строка_кукиз> : Установите куки для HTTP запроса.

-f : Забавный тюнинг при выявлении NOT_FOUND (404).

-H <строка_заголовка> : Задайте пользовательский заголовок HTTP запроса.

-i : Использовать поиск без учёта регистра.

-l : Печатать заголовок "Location" когда найден.

-N <nf_code>: Игнорировать ответы с этим HTTP кодом.

-o <файл_для_вывода> : Сохранить вывод на диск.

-p <прокси[:порт]> : Использовать прокси. (Порт по умолчанию 1080)

-P <proxy_username:proxy_password> : Аутентификация на прокси.

-r : Не искать рекурсивно.

-R : Интерактивная рекурсия. (Спрашивать для каждой директории)

-S : Молчаливый режим. Не показывать тестируемые слова. (Для простых терминалов)

-t : Не принуждать к конечному слешу '/' в URL.

-u <пользователь:пароль> : HTTP аутентификация.

-v : Показывать также страницы NOT_FOUND.

-w : Не показывать сообщений WARNING.

-X <расширение> / -x <расширения_файла> : Применить эти расширения к каждому слову.

-z <миллисекунды> : Добавить миллисекунды, для задержки, чтобы не стать причиной экстенсивного флуда.


```
(codeby@Codeby)-[~]
$ dirb http://testphp.vulnweb.com

_____
DIRB v2.22
By The Dark Raver
_____

START_TIME: Tue Sep  6 14:52:27 2022
URL_BASE: http://testphp.vulnweb.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____

GENERATED WORDS: 4612

— Scanning URL: http://testphp.vulnweb.com/ —
⇒ DIRECTORY: http://testphp.vulnweb.com/admin/
+ http://testphp.vulnweb.com/cgi-bin (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/cgi-bin/ (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/crossdomain.xml (CODE:200|SIZE:224)
⇒ DIRECTORY: http://testphp.vulnweb.com/CVS/
+ http://testphp.vulnweb.com/CVS/Entries (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/CVS/Repository (CODE:200|SIZE:8)
+ http://testphp.vulnweb.com/CVS/Root (CODE:200|SIZE:1)
⇒ Testing: http://testphp.vulnweb.com/default_pages
```

Рис. 4. Работа программы Dirb

Gobuster

Еще одной хорошей программой для фаззинга директорий является Gobuster. В отличие от «классического» Dirb у gobuster есть такие преимущества, как многопоточность, простой интерфейс unix-like интерфейс и возможность точной настройки. Это инструмент командной строки, написанный на Go, он не выполняет рекурсивный брут-форс, позволяет одновременно брутфорсить папки и несколько расширений, компилируется на множестве платформ, работает быстрее интерпретируемых скриптов (таких как Python), не требует среды выполнения.

В Kali Linux Gobuster устанавливается из официальных репозиториях:

```
apt-get install gobuster
```

Gobuster не предоставляет словарей для брутфорса директорий из коробки. Поэтому придётся скачать их отдельно. В Kali Linux есть словари директорий в `/usr/share/dirb/wordlists` и `/usr/share/dirbuster/wordlists`. Если же вы используете другую систему, то эти словари можно скачать отдельно. **directory-list-2.3-**

medium.txt достаточно хороший словарь, чтобы использовать его в реальных тестах.

Справка по Gobuster

Справка к программе вызывается опцией «-h» (Рис. 5).

```
(codeby@Codeby)-[~]
$ gobuster -h
Usage:
  gobuster [command]

Available Commands:
  dir      Uses directory/file enumeration mode
  dns      Uses DNS subdomain enumeration mode
  fuzz     Uses fuzzing mode
  help     Help about any command
  s3       Uses aws bucket enumeration mode
  version  shows the current version
  vhost    Uses VHOST enumeration mode

Flags:
  --delay duration    Time each thread waits between requests (e.g. 1500ms)
  -h, --help          help for gobuster
  --no-error          Don't display errors
  -z, --no-progress   Don't display progress
  -o, --output string  Output file to write results to (defaults to stdout)
  -p, --pattern string File containing replacement patterns
  -q, --quiet         Don't print the banner and other noise
  -t, --threads int   Number of concurrent threads (default 10)
  -v, --verbose       Verbose output (errors)
  -w, --wordlist string Path to the wordlist

Use "gobuster [command] --help" for more information about a command.
```

Рис. 5. Вызов справки к программе Gobuster

Для вызова справки к каждому режиму необходимо указать режим и опцию «-h» (Рис. 6).

```
(codeby@Codeby)-[~]
$ gobuster dir -h
Uses directory/file enumeration mode

Usage:
  gobuster dir [flags]

Flags:
  -f, --add-slash          Append / to each request
  -c, --cookies string     Cookies to use for the requests
  -d, --discover-backup    Upon finding a file search for backup files
  --exclude-length ints   exclude the following content length (completely ignores the status). Supply multiple times to exclude multiple sizes.
  -e, --expanded          Expanded mode, print full URLs
  -x, --extensions string File extension(s) to search for
  -r, --follow-redirect    Follow redirects
  -H, --headers stringArray Specify HTTP headers, -H 'Header1: val1' -H 'Header2: val2'
  -h, --help              help for dir
  --hide-length           Hide the length of the body in the output
  -m, --method string     Use the following HTTP method (default "GET")
  -n, --no-status         Don't print status codes
  -k, --no-tlsvalidation  Skip TLS certificate verification
  -P, --password string   Password for Basic Auth
  --proxy string          Proxy to use for requests [http(s)://host:port]
  --random-agent          Use a random User-Agent string
  -s, --status-codes string Positive status codes (will be overwritten with status-codes-blacklist if set)
  -b, --status-codes-blacklist string Negative status codes (will override status-codes if set) (default "404")
  --timeout duration     HTTP Timeout (default 10s)
  -u, --url string        The target URL
  -a, --useragent string  Set the User-Agent string (default "gobuster/3.1.0")
  -U, --username string   Username for Basic Auth
  --wildcard             Force continued operation when wildcard found

Global Flags:
  --delay duration    Time each thread waits between requests (e.g. 1500ms)
  --no-error          Don't display errors
  -z, --no-progress   Don't display progress
  -o, --output string  Output file to write results to (defaults to stdout)
  -p, --pattern string File containing replacement patterns
  -q, --quiet         Don't print the banner and other noise
  -t, --threads int   Number of concurrent threads (default 10)
  -v, --verbose       Verbose output (errors)
  -w, --wordlist string Path to the wordlist
```

Рис. 6. Вызов справки к режиму dir программы Gobuster

Общие опции командной строки

- fw: Принудительная обработка доменов при обнаружении поддержки групповых символов.
- q: отключить вывод банеров/линий
- t <потоки>: количество одновременных потоков (по умолчанию: 10).
- u <url/домен>: полный URL (включая схему) или базовое имя домена.
- v: вербальный вывод (показывать все результаты).
- w <словарь>: путь до используемого для брут-форса словаря.

Опции командной строки для режима dns

- cn: показывать CNAME записи (не может использоваться с опцией '-i').
- i: показывать в результате все IP адреса.

Опции командной строки для режима dir

- a <строка пользовательского агента>: указать строку пользовательского агента для отправки в заголовках запросов.
- c <http cookies> — отправлять с каждым запросом эти кукиз (симулирование аутентификации).
- e: расширенный режим, печатает полные URL.
- f: добавлять / (слэш) для брут-форса директорий.
- k: Пропустить верификацию SSL сертификатов.
- l: показать длину ответа.
- n: режим "без статуса", отключить вывод в результатах кода статуса.
- o <файл>: имя файла для записи вывода.
- p <проху url> — прокси для использования со всем запросами (схема должна соответствовать URL схеме).
- r: следовать редиректам.
- s <коды статуса>: разделённый запятой набор из списка кодов статуса, которые считаются "положительными" (по умолчанию: 200, 204, 301, 302, 307).
- x <расширения>: список расширений для проверки, можно не указывать.
- P <пароль>: Пароль HTTP авторизации (только Basic Auth, если пропущено, появится запрос).
- U <имя пользователя>: Имя пользователя HTTP авторизации (только Basic Auth).

В качестве примера рассмотрим поиск директориев при помощи программы Gobuster, для чего запустим ее с минимальным набором опций (Рис. 7).

```
gobuster dir -u http://testphp.vulnweb.com -w
/usr/share/dirbuster/wordlists/directory-list-2.3-
medium.txt -t 20 -s "200,204,301,302,404" -x php,html
```

```

--(codeby@Codeby)-[~]
$ gobuster dir -u http://testphp.vulnweb.com -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -t 20 -s "200,204,301,302,404" -x php,html

Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://testphp.vulnweb.com
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Extensions: php,html
[+] Timeout: 10s

2022/09/06 15:03:43 Starting gobuster in directory enumeration mode

/images (Status: 301) [Size: 169] [→ http://testphp.vulnweb.com/images/]
/index.php (Status: 200) [Size: 4958]
/search.php (Status: 200) [Size: 4732]
/cgi-bin.html (Status: 403) [Size: 276]
/cgi-bin (Status: 403) [Size: 276]
/login.php (Status: 200) [Size: 5523]
/product.php (Status: 200) [Size: 5056]
/disclaimer.php (Status: 200) [Size: 5524]
/signup.php (Status: 200) [Size: 6033]
/admin (Status: 301) [Size: 169] [→ http://testphp.vulnweb.com/admin/]
  
```

Рис. 7. Работа программы Gobuster

Здесь **-u** – URL или IP-адрес сайта, **-w** – словарь, **-t** – количество потоков, **-x** – расширения файлов и **-s** – ответы сервера, которые будут отображаться.

По умолчанию Gobuster отображает ответы 200, 204, 301, 302, 307 и 403. Ответ (access denied) тоже стоит учитывать, так как это может помочь в фингерпринтинге веб приложения. Допустим, у серверов apache есть файл server-status, который недоступен обычным пользователям.

Другой пример – директория .git. На сайте может быть Git-репозиторий, и при обращении к директории .git вы скорее всего получите ответ 403 (если там есть git-репозиторий). Его можно попробовать выкачать с помощью утилиты rip-git.

Ещё одна очень полезная функция **-x** позволяет добавлять расширения файлов к словарным данным. Так мы сможем нацелено находить php-файлы или бекапы с расширением .bak.

Wfuzz

Возвращаясь к фаззингу как к средству поиска и эксплуатации уязвимостей, невозможно не упомянуть данный инструмент. Wfuzz —

это огромный фреймворк для фаззинга веб-приложений, включающий в себя api на питоне (сам wfuzz тоже написан на питоне), прикладные утилиты (например, wffrayload для генерации полезных нагрузок), и сам фаззер, к которому, кстати, можно подключать плагины.

В Kali Linux он уже предустановлен, установка на других системах сводится к установке модуля python:

```
pip install wfuzz
```

Подробное руководство по программе можно посмотреть по адресу: <https://russianblogs.com/article/6886816499/>

С помощью этой утилиты можно фаззить буквально всё веб приложение: параметры GET, POST и прочих запросов, тело запроса, куки, директории сайта и т.д.

Рассмотрим типичные примеры использования программы:

Фаззинг директорий, файлов и параметров в url

```
wfuzz -z file,wordlists/directory-list-2.3-medium.txt --hc 404 http://testphp.vulnweb.com/FUZZ
```

Ключевое слово FUZZ будет заменяться на полезную нагрузку в процессе фаззинга. Это довольно гибкое решение, так как мы можем подставлять FUZZ в любое место запроса (Рис. 8).

```
(codeby@Codeby) [~]
$ wfuzz -z file,/usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt --hc 404 --hw 388 http://testphp.vulnweb.com/FUZZ
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****
Target: http://testphp.vulnweb.com/FUZZ
Total requests: 220560
```

ID	Response	Lines	Word	Chars	Payload
000000016:	301	7 L	11 W	169 Ch	"images"
000000035:	403	9 L	28 W	276 Ch	"cgi-bin"
000000259:	301	7 L	11 W	169 Ch	"admin"
000000465:	301	7 L	11 W	169 Ch	"pictures"
000001481:	301	7 L	11 W	169 Ch	"vendor"
000002279:	301	7 L	11 W	169 Ch	"Templates"

Рис. 8. Поиск директорий в программе Wfuzz

Для поиска файлов необходимо запустить поиск по двум словарям, один с именами файлов, другой с расширениями файлов (Рис. 9).

```

[~]codeby@Codeby:~$ wfuzz -z file,/usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt --hc 404 --hw 388 http://testphp.vulnweb.com/FUZZ.FUZZ22
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****
Target: http://testphp.vulnweb.com/FUZZ.FUZZ22
Total requests: 441120

```

ID	Response	Lines	Word	Chars	Payload
000000054:	200	103 L	364 W	4732 Ch	"search - php"
000000106:	200	119 L	432 W	5523 Ch	"login - php"
000000069:	403	9 L	28 W	276 Ch	"cgi-bin - html"
000000314:	200	110 L	377 W	5056 Ch	"product - php"
000000434:	200	121 L	446 W	6033 Ch	"signup - php"
000000412:	200	114 L	463 W	5524 Ch	"disclaimer - php"
000000580:	200	116 L	503 W	6115 Ch	"categories - php"
000000702:	302	38 L	96 W	1246 Ch	"comment - php"
000000822:	200	108 L	384 W	4903 Ch	"cart - php"

Рис. 9. Поиск файлов в программе Wfuzz

Параметр `--hc 404` указывает, что нужно пропускать все ответы 404 (страница не найдена). Всего есть 8 таких параметров:

- `--hc` — не показывать ответы с указанным кодом ответа
- `--hl` — не показывать ответы с указанным количеством строк
- `--hw` — не показывать ответы с указанным количеством слов
- `--hh` — не показывать ответы с указанным количеством символов
- `--sc` — показывать ответы только с указанным кодом ответа
- `--sl` — показывать ответы только с указанным количеством строк
- `--sw` — показывать ответы только с указанным количеством слова
- `--sh` — показывать ответы только с указанным количеством символов

Полезная нагрузка указывается с помощью `-z file`, путь_до_файла (или `-w` путь_до_файла, если используются словари). Кроме нагрузки типа `file` можно использовать и другие. Например, с помощью `range` можно сделать список из чисел, что удобно при поиске рабочих параметров в url:

```
wfuzz -c -z range,0-10 --hc 404
```

Полный список полезных нагрузок можно посмотреть с помощью команды:

```
wfuzz -e payloads
```

Фаззинг POST-запросов

Фаззить параметры post-запроса тоже очень просто. В параметре `-d` указывается тело запроса с ключевым словом FUZZ в месте фаззинга:

```
wfuzz -z file,common_pass.txt -d  
"uname=FUZZ&pass=FUZZ" --hc 302  
http://example.com/login.php
```

Фаззинг Куков

Куки указываются через параметр `-b`, каждое значение отдельно:

```
wfuzz -z file,common.txt -b cookie1=FUZZ -b  
cookie2=value2 http://example.com/index.php
```

Wfuzz будет генерировать такие http запросы:

```
GET /index.php HTTP/1.1 Host: example.com Accept: */*  
Content-Type: application/x-www-form-urlencoded  
Cookie: cookie1=FUZZ; cookie2=value2  
User-Agent: Wfuzz/2.2.11  
Connection: close
```

Где вместо FUZZ будут подставляться значения из словаря.

Фаззинг Хедеров

Можно добавлять свои хедеры в запрос, а также изменять уже существующие. В параметр `-H` указываются данные в формате «хедер: значение»

```
wfuzz -z file,headers.txt -H "myheader: FUZZ" -H  
"User-Agent: Googlebot-News" http://example.com/
```

Запрос к серверу будет таким:

```
GET / HTTP/1.1
Host: example.com Accept: */*
Content-Type: application/x-www-form-urlencoded
myheader: FUZZ
User-Agent: Googlebot-News
Connection: close
```

Кодировка

В некоторых случаях веб приложение принимает закодированные данные (например, base64 или sha1). Wfuzz позволяет кодировать значения из словаря перед отправкой на сервер. Для этого в параметре -z после указания типа итератора (файл, последовательность, стандартный ввод stdin и т. д.) и самого итератора, через запятую указывается тип кодировки:

```
wfuzz -u example.net/main.php?page=FUZZ -z
file,wordlists/common.txt,base64 --sw 40
```

Полный список возможных способов кодирования с описанием (Рис. 10) можно посмотреть, введя команду:

```
wfuzz -e encoders
```


Available encoders:

Category	Name	Summary
url_safe, url	urlencode	Replace the char
url_safe, url	double urlencode	Applies Letters,
url	uri_double_hex	Encodes
html	html_escape	Convert
html	html_hexadecimal	Replaces
hashes	base64	Encodes
url	double_nibble_hex	Replaces
db	mssql_char	Converts
url	utf8	Replaces
hashes	md5	Applies
default	random_upper	Replaces
url	first_nibble_hex	Replaces
default	hexlify	Every by n.
url	second_nibble_hex	Replaces
url	uri_hex	Encodes
default	none	Returns
hashes	sha1	Applies
url	utf8_binary	Replaces
url	uri_triple_hex	Encodes
url	uri_unicode	Replaces
html	html_decimal	Replaces
db	oracle_char	Converts
db	mysql_char	Converts

Рис. 10. Способы кодирования Wfuzz

Кодировки можно применять по цепочке, разделяя их знаком «-», тогда wfuzz будет по очереди кодировать и отсылать на сервер полезную нагрузку с разными кодировками (Рис. 11).

```
wfuzz -u "http://example/main.php?page=FUZZ" -z list,just_one_payload,none- base64-md5
```

```

Total requests: 3
=====
ID      Response  Lines   Word    Chars    Payload
=====
000001:  C=200      26 L    42 W    420 Ch   "just_one_payload"
000002:  C=200      26 L    42 W    420 Ch   "anVzdF9vbmVfcGF5bG9hZA=="
000003:  C=200      26 L    42 W    420 Ch   "b40ffc433aee7c30cc018f5d6ed059d5"

Total time: 0.095090
Processed Requests: 3
Filtered Requests: 0
Requests/sec.: 31.54900

```

Рис. 11. Пример использования кодировок

Ffuf

Также в контексте рассматриваемой темы нельзя не упомянуть фаззер ffuf, который по назначению, функционалу и синтаксису довольно похож на вышерассмотренный wfuzz. В некоторых ситуациях использование ffuf может быть даже более предпочтительным, благодаря более продуманной оптимизации работы с памятью, а также более активной поддержке со стороны разработчиков.

Установить ffuf можно как из официального репозитория на github, так и через менеджер пакетов Kali Linux командой:

```
sudo apt install ffuf
```

Функционал данного инструмента действительно впечатляет и позволяет проводить тестирование, учитывая огромное количество параметров. Несмотря на схожесть с утилитой wfuzz, перед использованием ffuf необходимо сначала ознакомиться с его ключами командой ffuf -h, --help.

Наиболее полное руководство можно найти по адресу: <https://zen.yandex.ru/media/cisoclub/rukovodstvo-po-fuzz-faster-u-fool-ffuf-612504efb37e226a924a89f9>

Важно: Перед началом работы следует знать отличительные особенности ffuf от wfuzz. Ffuf по умолчанию не подставляет в ваш запрос никаких заголовков. Например, если форма на сайте требует отправлять данные с заголовком Content-Type, то обязательно надо выставить соответствующую настройку для ffuf (-H «Content-Type: <нужное вписать>»), в противном случае запросы будут отправляться без данного заголовка и сервер может их просто не обработать.

Вы же не увидите никаких ошибок, и будете думать, что все запросы отправляются как надо.

При запуске фаззинга ffuf покажет вам все параметры, с которыми он отправляет серверу запросы. Они будут удобочитаемо выведены в консоль. Убедитесь, что все параметры выставлены верно.

Рассмотрим на предыдущем примере работу программы ffuf (Рис. 12).

Фаззинг директорий и файлов

```
ffuf -u "http://testphp.vulnweb.com/FUZZ" -fs
16,153,276,4958 -c -t 100 -e .php,.html -w
/usr/share/dirbuster/wordlists/directory-list-2.3-
medium.txt -mc all
```

где:

-u – URL, -w – словари, -fs исключаем ответы с размером 16,153,276,4958 (выбираем размер ответов от неудачных попыток, чтобы отфильтровать ненужные результаты), -c – выделение цветом, -t – количество потоков (по умолчанию 10), -e .php,.html – указываем расширения файлов, которые ищем, -mc all – все коды ответа сервера.

```
codeby@Codeby:~$ ffuf -u "http://testphp.vulnweb.com/FUZZ" -fs 16,153,276,4958 -c -t 100 -e .php,.html -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -mc all

v1.5.0 Kali Exclusive <3>

:: Method      : GET
:: URL         : http://testphp.vulnweb.com/FUZZ
:: Wordlist    : FUZZ: /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
:: Extensions : .php .html
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 100
:: Response status: all
:: Matcher     : Response size: 16,153,276,4958

images [Status: 301, Size: 169, Words: 5, Lines: 8, Duration: 254ms]
search.php [Status: 200, Size: 4732, Words: 482, Lines: 104, Duration: 326ms]
login.php [Status: 200, Size: 5523, Words: 557, Lines: 120, Duration: 237ms]
product.php [Status: 200, Size: 5056, Words: 490, Lines: 111, Duration: 238ms]
discclaimer.php [Status: 200, Size: 5524, Words: 574, Lines: 135, Duration: 232ms]
signup.php [Status: 200, Size: 6035, Words: 547, Lines: 122, Duration: 250ms]
admin [Status: 301, Size: 169, Words: 5, Lines: 8, Duration: 225ms]
categories.php [Status: 200, Size: 6115, Words: 656, Lines: 117, Duration: 245ms]
comment.php [Status: 302, Size: 1246, Words: 125, Lines: 39, Duration: 230ms]
cart.php [Status: 200, Size: 4903, Words: 502, Lines: 109, Duration: 242ms]
pictures [Status: 301, Size: 169, Words: 5, Lines: 8, Duration: 251ms]
logout.php [Status: 200, Size: 4830, Words: 492, Lines: 107, Duration: 230ms]
vendor [Status: 301, Size: 169, Words: 5, Lines: 8, Duration: 250ms]
guestbook.php [Status: 200, Size: 5390, Words: 515, Lines: 113, Duration: 234ms]
404.php [Status: 200, Size: 5269, Words: 529, Lines: 112, Duration: 230ms]
artists.php [Status: 200, Size: 5328, Words: 503, Lines: 105, Duration: 238ms]
Templates [Status: 301, Size: 169, Words: 5, Lines: 8, Duration: 247ms]
userinfo.php [Status: 302, Size: 14, Words: 3, Lines: 1, Duration: 279ms]
Flash [Status: 200, Size: 169, Words: 5, Lines: 8, Duration: 220ms]
CWS [Status: 301, Size: 169, Words: 5, Lines: 8, Duration: 223ms]
AJAX [Status: 301, Size: 169, Words: 5, Lines: 8, Duration: 241ms]
secured [Status: 301, Size: 169, Words: 5, Lines: 8, Duration: 231ms]
showimage.php [Status: 200, Size: 3, Words: 2, Lines: 2, Duration: 240ms]
```

Рис. 12. Фаззинг директорий и файлов в программе Ffuf

Брут формы авторизации

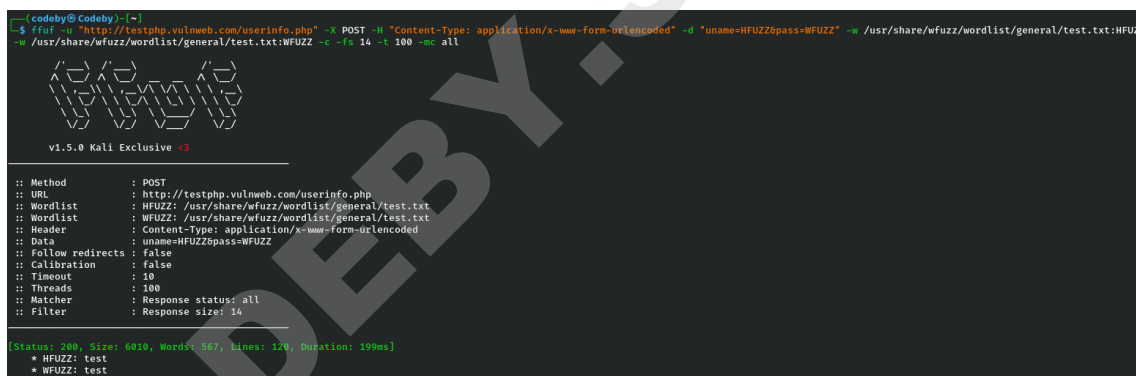
Брут логинов и паролей это тоже фаззинг. Брутить можно как по одному параметру, так и по нескольким. Чтобы не повторяться проведем перебор сразу по двум полям: логину и паролю.

Подготовим следующую команду (Рис. 13):

```
ffuf -u "http://testphp.vulnweb.com/userinfo.php" -X POST -H "Content-Type: application/x-www-form-urlencoded" -d "uname=HFUZZ&pass=WFUZZ" -w /usr/share/wfuzz/wordlist/general/test.txt:HFUZZ -w /usr/share/wfuzz/wordlist/general/test.txt:WFUZZ -c -fs 14 -t 100 -mc all
```

где:

-u – URL, -X – тип запроса (POST), -H – заголовок, -d – POST-параметры, -w – словари, -c – выделение цветом, -fs – фильтрация, -t – количество потоков, -mc – все коды ответов.



```
(codeby@Codeby)~$ ffuf -u "http://testphp.vulnweb.com/userinfo.php" -X POST -H "Content-Type: application/x-www-form-urlencoded" -d "uname=HFUZZ&pass=WFUZZ" -w /usr/share/wfuzz/wordlist/general/test.txt:HFUZZ -w /usr/share/wfuzz/wordlist/general/test.txt:WFUZZ -c -fs 14 -t 100 -mc all
```

```
v1.5.0 Kali Exclusive <3>
```

```

:: Method      : POST
:: URL         : http://testphp.vulnweb.com/userinfo.php
:: Wordlist     : HFUZZ: /usr/share/wfuzz/wordlist/general/test.txt
:: Wordlist     : WFUZZ: /usr/share/wfuzz/wordlist/general/test.txt
:: Header      : Content-Type: application/x-www-form-urlencoded
:: Data        : uname=HFUZZ&pass=WFUZZ
:: Follow redirects : false
:: Calibration   : false
:: Timeout      : 10
:: Threads      : 100
:: Matcher      : Response status: all
:: Filter       : Response size: 14

```

```
[Status: 200, Size: 6010, Words: 567, Lines: 120, Duration: 199ms]
+ HFUZZ: test
+ WFUZZ: test
```

Рис. 13. Брут формы авторизации в программе Ffuf

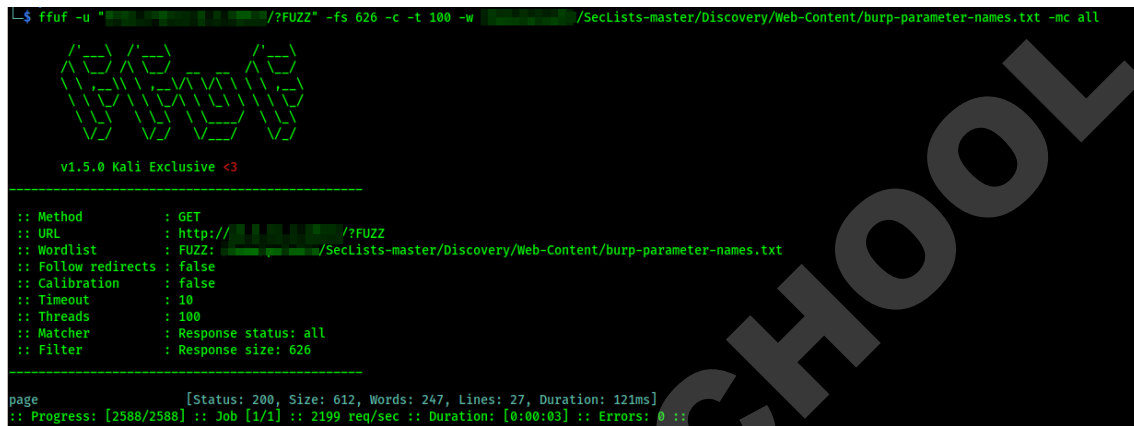
Фаззинг GET-параметров

Предположим необходимо найти имя GET-параметра в веб-приложении (Рис. 14). Для этого задаем такую команду для фаззера:

```
ffuf -u "http://web-site/?FUZZ" -fs 626 -c -t 100 -w /SecLists-master/Discovery/Web-Content/burp-parameter-names.txt -mc all
```

где:

-u –URL, -w – словари, -c – выделение цветом, -fs – фильтрация (после первого запуска команды появится просто «лавина» ответов от сервера, нам необходимо выбрать для фильтра размер большинства ответов), -t – количество потоков, -mc – все коды ответов.



```

L$ ffuf -u "http://10.10.10.10/?FUZZ" -fs 626 -c -t 100 -w /SecLists-master/Discovery/Web-Content/burp-parameter-names.txt -mc all

v1.5.0 Kali Exclusive <3

:: Method      : GET
:: URL         : http://10.10.10.10/?FUZZ
:: Wordlist    : FUZZ: /SecLists-master/Discovery/Web-Content/burp-parameter-names.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 100
:: Matcher     : Response status: all
:: Filter      : Response size: 626

page [Status: 200, Size: 612, Words: 247, Lines: 27, Duration: 121ms]
:: Progress: [2588/2588] :: Job [1/1] :: 2199 req/sec :: Duration: [0:00:03] :: Errors: 0 ::

```

Рис. 14. Фаззинг GET-параметров в программе Ffuf

Dirsearch

И еще один инструмент, который заслуживает нашего внимания, это Dirsearch. Dirsearch – это простой инструмент командной строки, предназначенный для брут-форса (поиска путём полного перебора) директорий и файлов в веб-сайтах.

Возможности:

- Многопоточность;
- Keep alive соединений;
- Поддержка множества расширений (-e|--extensions asp,php);
- Составление отчётов (простой текст, JSON);
- Эвристическое выявление невалидных веб-страниц;
- Рекурсивный брут-форсинг;
- Поддержка HTTP прокси;
- Случайные User agent;
- Пакетная обработка.

Справка по dirsearch

Использование:

dirsearch.py [-u|--url] цель [-e|--extensions] расширения [опции]

Опции:

-h, --help – вызов справки

Обязательные:

-u URL, --url=URL – URL цели

-L URLLIST, --url-list=URLLIST – список URL целей

-e РАСШИРЕНИЯ, --extensions=РАСШИРЕНИЯ – список расширений, разделённых запятой (Пример: php,asp)

Настройки словаря:

-w СЛОВАРЬ, --wordlist=СЛОВАРЬ

-l, --lowercase – все записи в словаре переводить в нижний регистр

-f, --force-extensions –принудительные расширения для каждой записи в словаре (как в DirBuster)

Общие настройки:

-s ЗАДЕРЖКА, --delay=ЗАДЕРЖКА – задержка между запросами

-r, --recursive – рекурсивный брутфорс

--suppress-empty, --suppress-empty

--scan-subdir=SCANSUBDIRS, --scan-subdirs=SCANSUBDIRS – сканировать поддиректории данного -u|--url (разделённые запятой)

--exclude-subdir=EXCLUDESUBDIRS, --exclude-subdirs=EXCLUDESUBDIRS – исключить следующие поддиректории во время рекурсивного сканирования (разделены запятой)

-t THREADSCOUNT, --threads=THREADSCOUNT – количество потоков

-x EXCLUDESTATUSCODES, --exclude-status=EXCLUDESTATUSCODES – исключить коды статусов, разделены запятыми (пример: 301, 500)

-c COOKIE, --cookie=COOKIE

-ua=USERAGENT, --user-agent=USERAGENT

-F, --follow-redirects следовать перенаправлениям

-H HEADERS, --header=HEADERS – заголовки для добавления (например: --header "Referer: example.com" --header "User-Agent: IE"

--random-agents, --random-user-agents случайные пользовательские агенты

Настройки соединения:

--timeout=TIMEOUT – тайм аут соединения

--ip=IP – преобразовывать имя до IP адреса

--proxy=HTTPPROXY, --http-proxy=HTTPPROXY – http-прокси (пример: localhost:8080)

--max-retries=MAXRETRIES максимальное количество попыток

-b, --request-by-hostname – по умолчанию для скорости dirsearch будет делать запросы по IP. Эта настройка принуждает делать запросы по имени хоста.

Отчёты:

--simple-report=SIMPLEOUTPUTFILE – только найденные пути

--plain-text-report=PLAINTEXTOUTPUTFILE – найденные пути с кодами статуса

--json-report=JSONOUTPUTFILE

Попробуем работу программы на предыдущем нашем примере (Рис. 15):

```
dirsearch -u http://testphp.vulnweb.com/ -w
/usr/share/dirbuster/wordlists/directory-list-2.3-
medium.txt -e php,html -f
```

```
(codeby@Codeby)~$ dirsearch -u http://testphp.vulnweb.com/ -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -e php,html -f
dirsearch v0.4.2
Extensions: php, html | HTTP method: GET | Threads: 30 | Wordlist size: 882180
Output File: /home/codeby/.dirsearch/reports/testphp.vulnweb.com/_22-09-06_12-17-29.txt
Error Log: /home/codeby/.dirsearch/logs/errors-22-09-06_12-17-29.log
Target: http://testphp.vulnweb.com/
[12:17:30] Starting:
[12:17:31] 200 - 5KB - /index.php
[12:17:31] 200 - 377B - /images/
[12:17:31] 301 - 169B - /images → http://testphp.vulnweb.com/images/
[12:17:32] 200 - 5KB - /search.php
[12:17:32] 403 - 276B - /cgi-bin/
[12:17:32] 403 - 276B - /cgi-bin/
[12:17:32] 403 - 276B - /cgi-bin.html
[12:17:33] 200 - 5KB - /login.php
[12:17:37] 200 - 5KB - /product.php
[12:17:39] 200 - 5KB - /disclaimer.php
[12:17:39] 200 - 6KB - /signup.php
[12:17:41] 301 - 169B - /admin → http://testphp.vulnweb.com/admin/
[12:17:41] 200 - 262B - /admin/
[12:17:42] 200 - 6KB - /categories.php
[12:17:44] 302 - 1KB - /comment.php → ./index.php
[12:17:46] 200 - 5KB - /cart.php
```

Словари

FuzzDB – это проект, объединяющий в себе большое количество фаззинг-баз, упорядоченных по своему назначению. В FuzzDB входят:

- распространенные пути файлов и директорий, представляющих ценность для атакующего, например, пути логов и конфигурационных файлов;

- шаблоны атак – собственно те строки, которые отправляются приложению, вследствие чего возникают ошибки и исключения;
- шаблоны ответов – строки, с помощью которых можно идентифицировать наличие уязвимости;
- другие полезности, например, коллекция web-шеллов под большинство платформ и словари для брутфорса;
- документация.

Основой для FuzzDB являются базы таких известных фаззеров как jBroFuzz (проект OWASP), wapiti, SPIKE, а также собственные исследования автора, анализ логов, различные другие источники. Проект поддерживается лишь одним человеком, но, тем не менее, не стоит на месте.

По сути, FuzzDB – лишь набор текстовых файлов с шаблонами, отсортированных по платформам, типам атак, языкам. Использовать FuzzDB можно, где угодно, например, в самописном web-сканнере директорий или в профессиональном инструменте для проведения пентестов, например, в том же Burp.

Скачивание через Github:

```
git clone https://github.com/fuzzdb-project/fuzzdb.git
```

SecLists – ещё одна база с хорошими фаззинг-листами, которую полезно держать при себе. В ней можно найти словари с популярными логинами, паролями, директориями сайтов, а также фаззинг-листы для различных уязвимостей: sqli, xss, lfi, etc...

```
git clone https://github.com/danielmiessler/SecLists.git
```

Payloads All The Things — один из наиболее популярных репозиториях полезных нагрузок, содержит следующие категории (которых хватит практически на «все случаи жизни» тестирования веб-приложений): CRLF injection; Methodology and Resources; Remote commands execution; Web cache deception; CSV injection; NoSQL injection; Server Side Template injections; XPATH injection; CVE Exploits; OAuth; SQL injection; XSS injection; File Inclusion - Path Traversal; Open redirect; SSRF injection; XXE injections; Insecured source code management; PHP juggling type; Tar

commands execution; Java Deserialization; PHP serialization; Traversal directory.

Вывод

Активный фаззинг веб-приложений — широко применяющаяся и очень важная техника при тестировании веб приложений. С его помощью находится большинство уязвимостей.

У активного фаззинга имеются как отрицательные, так и положительные стороны. К числу первых можно отнести низкую скрытность процесса тестирования, так как после проверки всех возможных параметров остаются следы, которые невозможно не заметить, если, конечно, проводится мониторинг журналов и нагрузки на систему. Среди положительных моментов выделяется полная автоматизация процесса, позволяющая значительно сэкономить время. Эффективность активного фаззинга в значительной степени зависит от программы-фаззера и базы, с которой она работает

Как может показаться на первый взгляд активный фаззинг достаточно простое занятие, но это совершенно не так, с практикой появляется опыт и приходит понимание о том, как именно нужно фаззить тот или иной параметр, что лучше протестировать на ту или иную уязвимость и какие полезные нагрузки для этого использовать.

Fuzzing tools

Наименование	Описание	Где скачать
0d1n	Инструмент для фаззинга HTTP входов, написан на C с поддержкой libCurl	https://github.com/CoolerVoid/0d1n
Ajpfuzzer	Инструмент для фаззинга протокола ajp13 Apache JServ	https://github.com/doyensec/ajpfuzzer
Astra	Автоматическое тестирование REST API	https://github.com/flipkart-incubator/astra
Atscan	Сканер уязвимостей серверов и сайтов	https://github.com/AlisamTechnology/ATSCAN
Bbscan	Небольшой сканер веб уязвимостей	https://github.com/lijieie/bbscan

Наименование	Описание	Где скачать
Bing-lfi-rfi	Скрипт для поиска сайтов через Bing уязвимых к RFI или LFI	https://packetstormsecurity.com/files/121590/Bing-LFI-RFI-Scanner.html
Brutexss	Брутфорсер XSS уязвимостей	https://github.com/shawarkhanethicalhacker/BBruteXSS-1
Cmsfuzz	Фаззер для таких CMS как: wordpress, cold fusion, drupal, Joomla, and phpnuke.	https://github.com/nahamsec/CMSFuzz
Conscan	Сканер уязвимостей для Concre5 CMS	http://nullsecurity.net/tools/scanner.html
Crlf-injector	Скрипт для тестирования CRLF инъекций	https://github.com/rudSarkar/crlf-injector
Darkbing	Скрипт для обнаружения с помощью Bing сайтов, восприимчивых к SQL инъекциям	https://packetstormsecurity.com/files/111510/darkBing-SQL-Scanner.1.html
Dpscan	Сканер уязвимостей в Drupal CMS	https://github.com/intfr/Blue-Sky-Information-Security
Easyfuzzer	Очередной фаззер веб уязвимостей, удобный вывод результата в csv файл	http://www.mh-sec.de/downloads.html.en
Fhttp	Фреймворк для атак, связанных с HTTP. Имеет прокси для отладки и манипуляции.	https://packetstormsecurity.com/files/104315/FHTTP-Attack-Tool.3.html
Filebuster	Очень быстрый и гибкий фаззер файлов	https://github.com/henshin/filebuster
Hexorbase	Приложение для тестирования нескольких серверов баз данных одновременно из одного места. Способен производить атаки на такие БД как: MySQL, SQLite, Microsoft SQL Server, Oracle, PostgreSQL	https://github.com/savio-code/hexorbase

Наименование	Описание	Где скачать
Httpforge	Набор инструментов позволяющих манипулировать, отправлять, получать и анализировать HTTP сообщения.	https://packetstormsecurity.com/files/98109/HTTPForge.02.01.html
Joomlavs	Сканер уязвимостей Joomla CMS	https://github.com/rastating/joomlavs
Jsql-injection	Приложение, написанное на Java для эксплуатации SQL инъекций	https://github.com/ron190/jsql-injection
Leviathan	Инструмент для массового обнаружения сайтов подверженных различным уязвимостям	https://github.com/tearsecurity/leviathan
Lfi-fuzzploit	Приложение позволяет обнаруживать и эксплуатировать LFI и Linux based PHP уязвимости	https://packetstormsecurity.com/files/106912/LFI-Fuzzploit-Tool.1.html
Lfi-scanner	Сканер проверяющий LFI уязвимость на конкретном сайте	https://packetstormsecurity.com/files/102848/LFI-Scanner.0.html
Lfi-spoiter	Очередной инструмент позволяющий обнаруживать и эксплуатировать LFI уязвимость на конкретном сайте	https://packetstormsecurity.com/files/96056/Simple-Local-File-Inclusion-Exploiter.0.html
Lfimap	Скрипт позволяет получить максимальную выгоду от использования LFI уязвимости	https://code.google.com/archive/p/lfimap/
Liffy	Еще один инструмент для эксплуатации LFI уязвимости	https://github.com/hvqzao/liffy
Nikto	Сканер веб серверов для проведения различных тестов	https://github.com/sullo/nikto

Наименование	Описание	Где скачать
Owtf	Большой фреймворк для тестирования на проникновение	https://www.owasp.org/index.php/OWASP_OWTF
Pappy-proxy	Приложение для тестирования на проникновение через прокси	https://github.com/rogilew/pappy-proxy
Shortfuzzy	Скрипт использует для фаззинга свыше 800 векторов атак и до 8 уровней вложенности рекурсии	https://packetstormsecurity.com/files/104872/Short-Fuzzy-Rat-Scanner.html
Skipfish	Полностью автоматический инструмент активной разведки веб-приложений	https://code.google.com/archive/p/skipfish/
Spaf	Скрипт для определения всех точек входа в приложение	https://github.com/Ganapati/spaf
Spartan	Программа успешно находит контейнеры TrueCrypt, VeraCrypt, CipherShed, Encrypt-файлы Encrypt, зашифрованные сообщения PGP / GPG, закрытые ключи OpenSSH и PEM, базы паролей и файлы, состоящие из случайных данных.	https://github.com/sensepost/SPartan
Sqlbrute	Брутфорс данных БД с помощью слепой SQL инъекции	https://github.com/GDSecurity/SQLBrute
SqlNinja	Инструмент для эксплуатации SQL уязвимостей.	http://sqlNinja.sourceforge.net/
Uniscan	Простой сканер для обнаружения LFI, RFI и CMD уязвимостей	https://sourceforge.net/projects/uniscan/
Uppwn	Анализатор уязвимостей системы загрузки файлов	https://github.com/ferry1/UpPwn

Наименование	Описание	Где скачать
Vane	Сканер уязвимостей для Wordpress CMS	https://github.com/delvelabs/vane
Vbscan	Сканер уязвимостей VBulletin CMS	https://github.com/reza-sp/vbscan
W3af	Огромный фреймворк для атак на веб приложения	http://w3af.org/
Wafninja	Инструмент для атаки на брандмауэры веб приложений	https://github.com/khalilbijou/WAFNinja
Wafpass	Анализ методов обхода WAF	https://github.com/wafpassproject/wafpass
Wapiti	Сканер основных уязвимостей	http://wapiti.sourceforge.net/
Webscarab	Фреймворк для анализа web приложений, которые обмениваются данными при помощи HTTP или HTTPS	https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project
Webshag	Многопоточный, мульти платформенный инструмент для анализа веб сервера	https://www.scrt.ch/en/attack/downloads/webshag
Webxploiter	Сканер уязвимостей от OWASP Top 10	https://github.com/a0xnirudh/WebXploiter
Witchxtool	Очередной сканер портов, LFI, RFI, SQLi и т. д.	https://packetstormsecurity.com/files/97465/Witchxtool-Port-LFI-SQL-Scanner-And-MD5-Bruteforcing-Tool.1.html
Wpscan	Сканер уязвимостей Wordpress CMS	https://wpscan.org/

Наименование	Описание	Где скачать
Wsfuzzer	Инструмент для автоматического тестирования SOAP сервисов	https://www.owasp.org/index.php/Category:OWASP_WSFuzzer_Project