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 рада 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 туториал 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

все нужные скрипты и утилиты для работы в одном месте, имея разный спектр задач, и что бы программа служила своеобразным мультитулом.

Начнем с небольшого обзора что умеет данная программа:

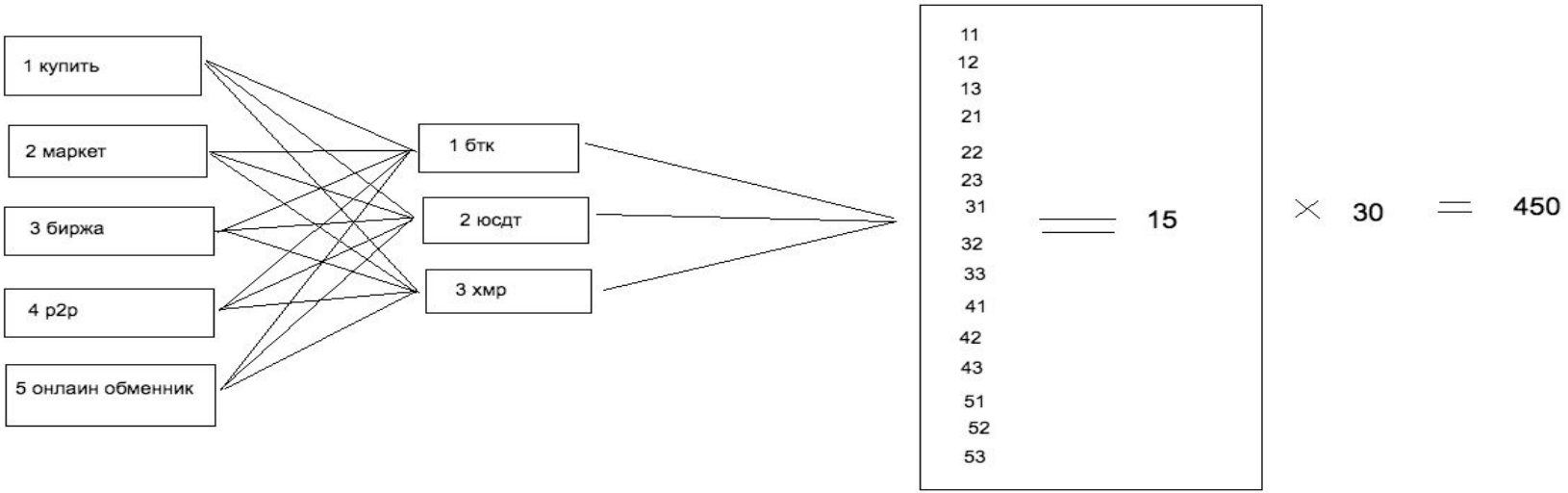
Есть 3 вида утилит: Database work, WEB attack, Phish attack

- 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.

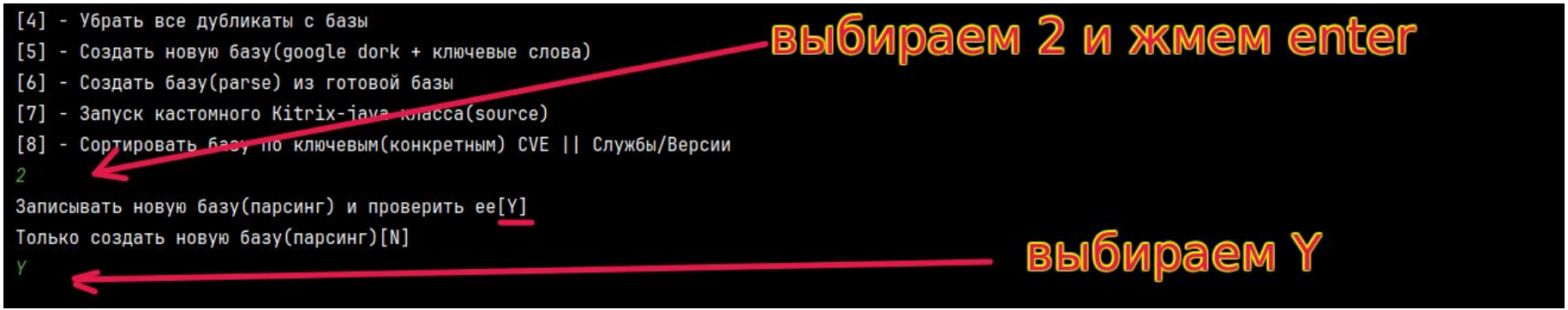
- Создать новую базу(парсинг)
- Создать новую базу(google dork + ключевые слова)
Тут мы будем парсить url сайтов, по нашим ключевым словам. На этом пункте остановимся по подробнее, я на скринах покажу что куда тыкать, и все остальные функции работают по аналогии. Отвечу сразу на вопрос, зачем если есть 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 получиться. 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 певром 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 ру), 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 ру и снг!

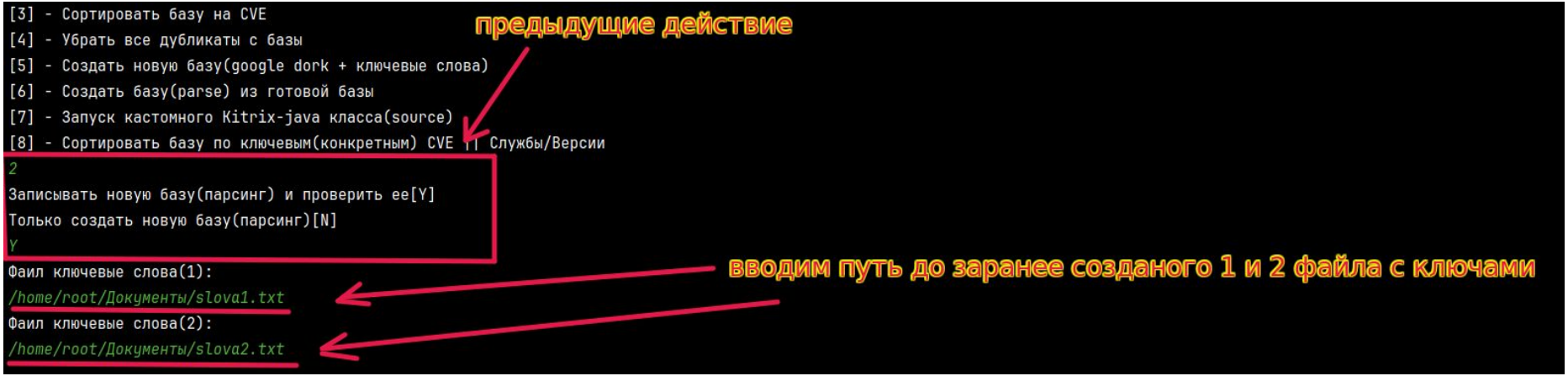
Сам алгоритм создания базы ключей из 2 баз слов выглядит так(для примера на тематике крипты привел пример):



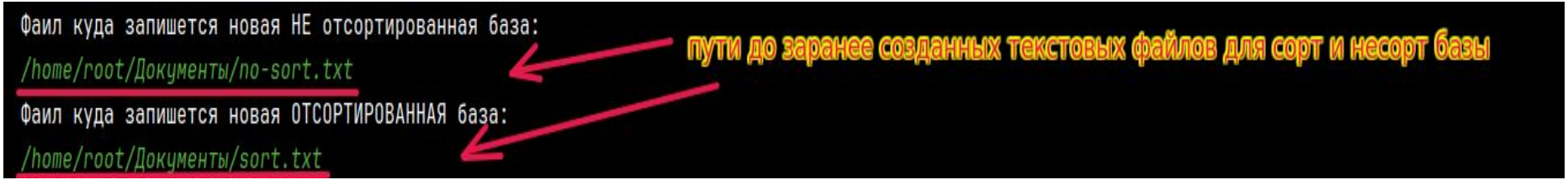
Так же мы говорим хотим ли мы создать только базу ключей, или же сразу пройтись по ней и создать базу сайтов:

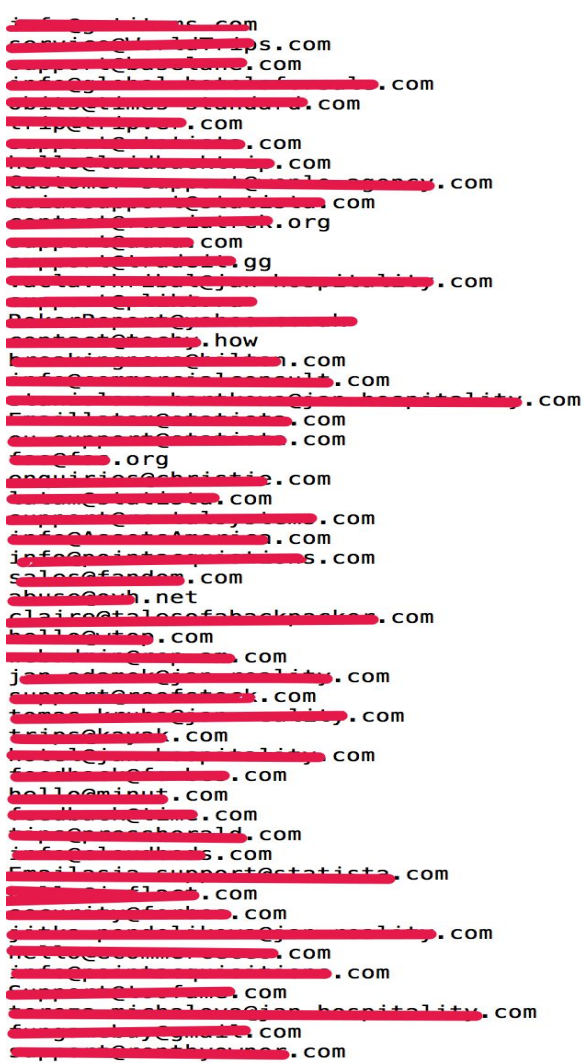


Потом вводим пути до наших файлов с ключевыми словами:



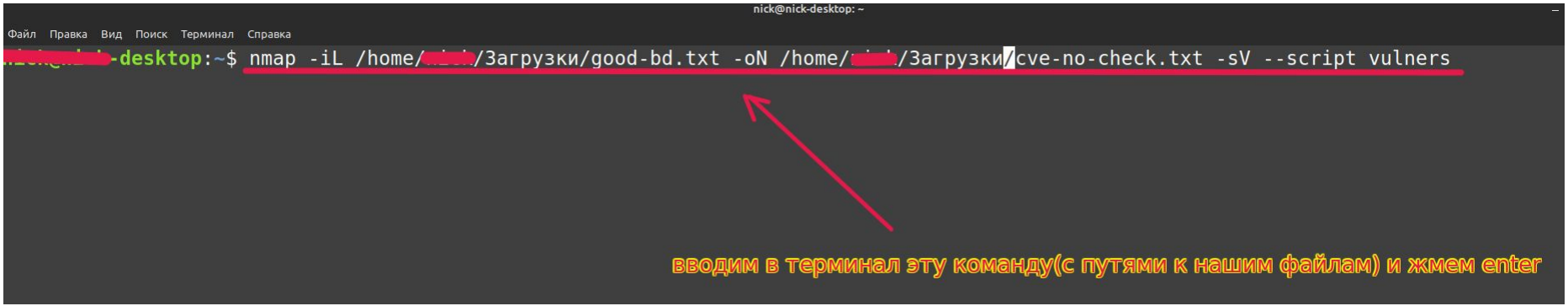
И путь куда запишется отсортированная и не отсортированная база:





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 получиться 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 эксплотируют ту или иную уязвимость.



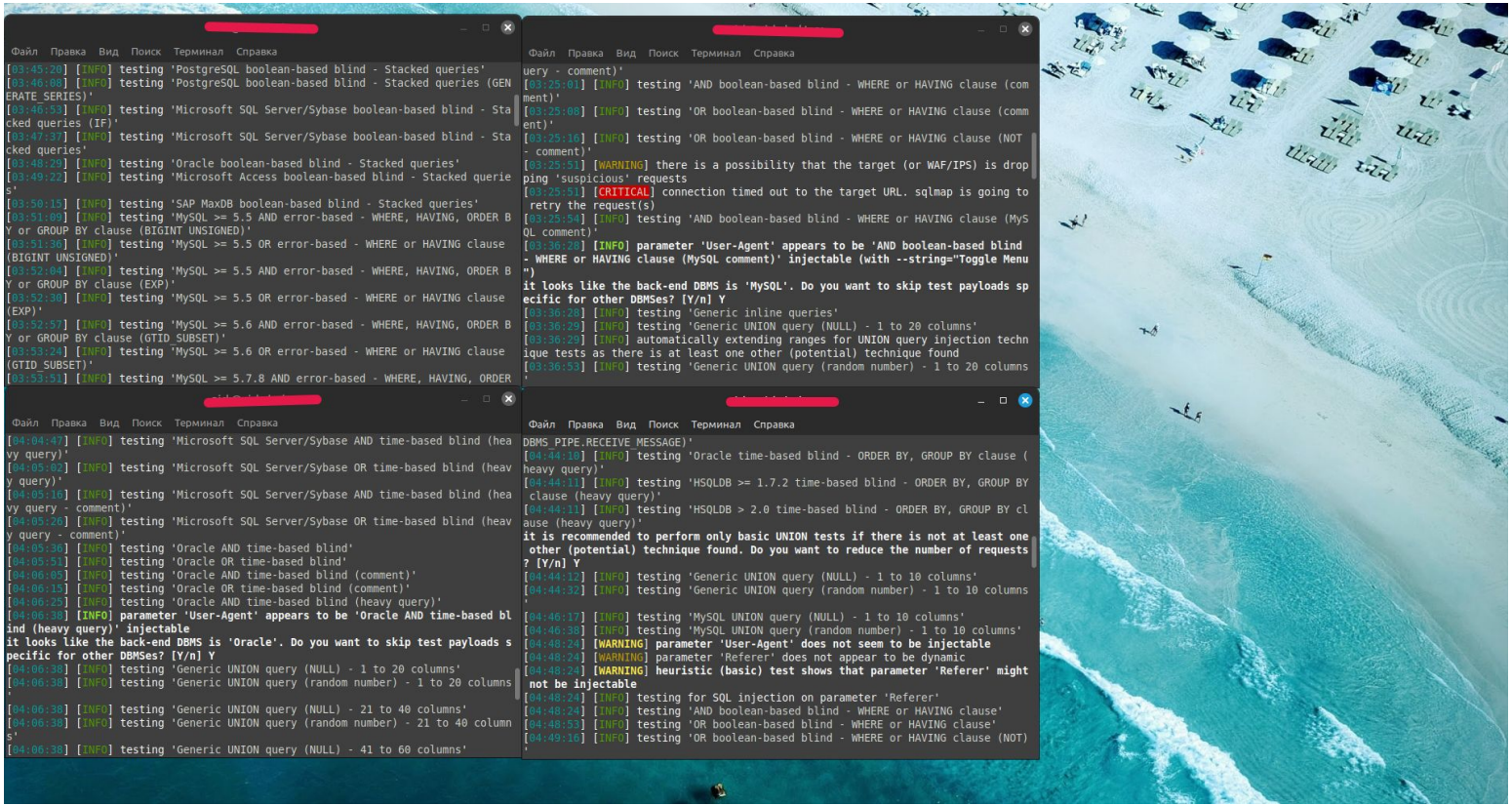
Далее можно по метасплойту найти подходящий экслоит, и пробовать раскрутить уязвимость:

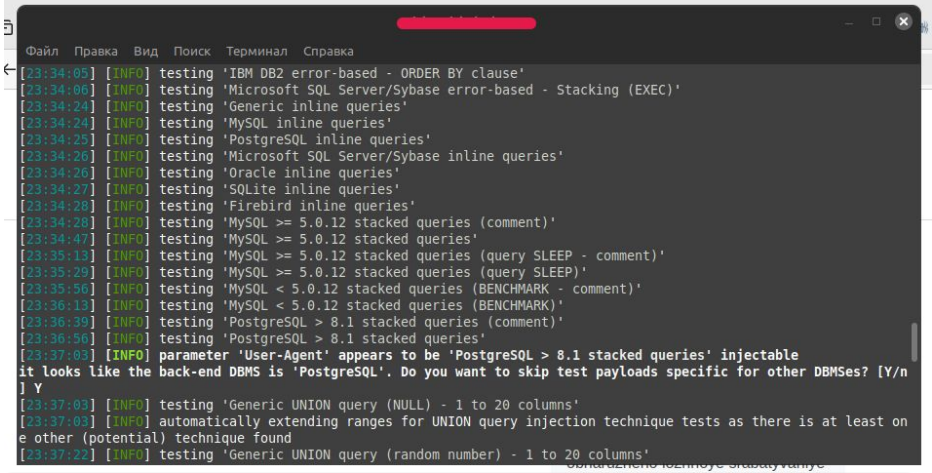
```
Not shown: 997 filtered ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:8.2p1:
|       PRION:CVE-2020-15778    6.8     https://vulners.com/prion/PRION:CVE-2020-15778
|       CVE-2020-15778  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*
```

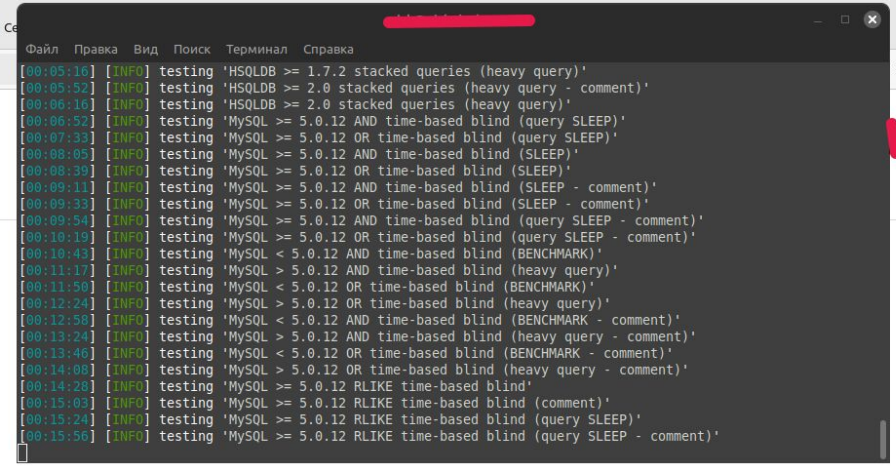
```
443/tcp open  ssl/http Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
| vulners:
|   cpe:/a:apache:http_server:2.4.41:
|       PACKETSTORM:176334      7.5     https://vulners.com/packetstorm/PACKETSTORM:176334      *EXPLOIT*
|       PACKETSTORM:171631      7.5     https://vulners.com/packetstorm/PACKETSTORM:171631      *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*
```

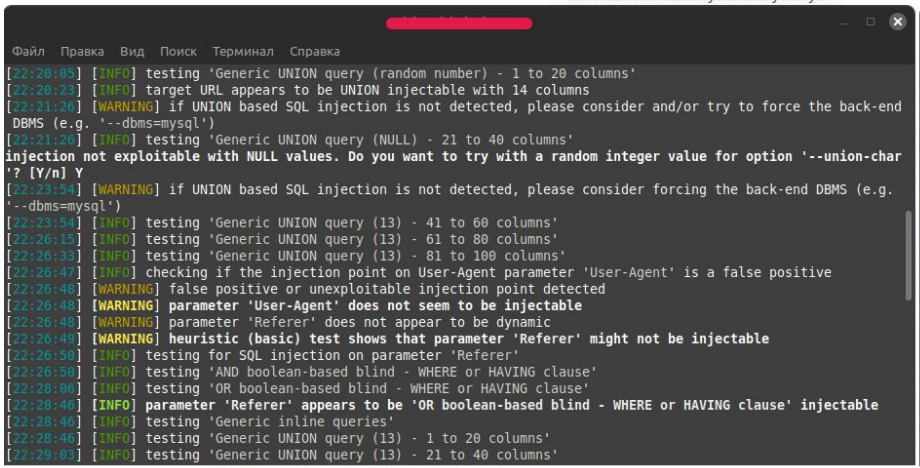
```
msf6 > search apache httpd

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
```

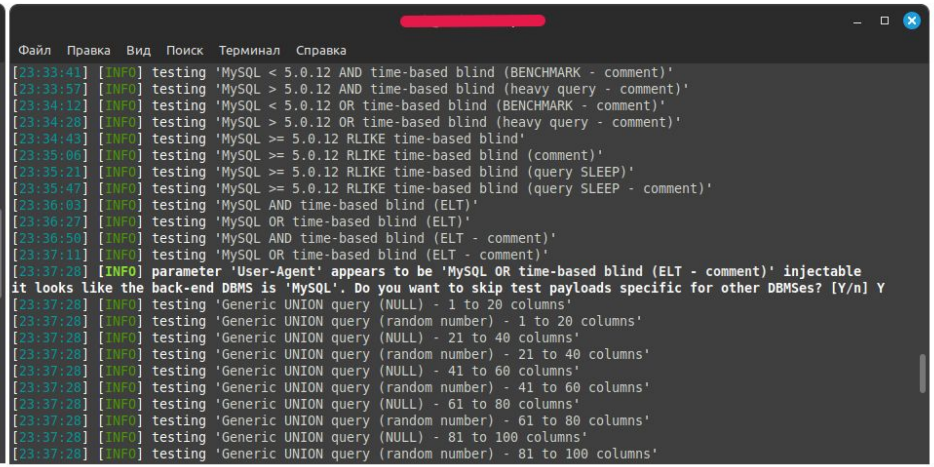
Или sqlmap для дальнейшей проверки на sql иньекцию:



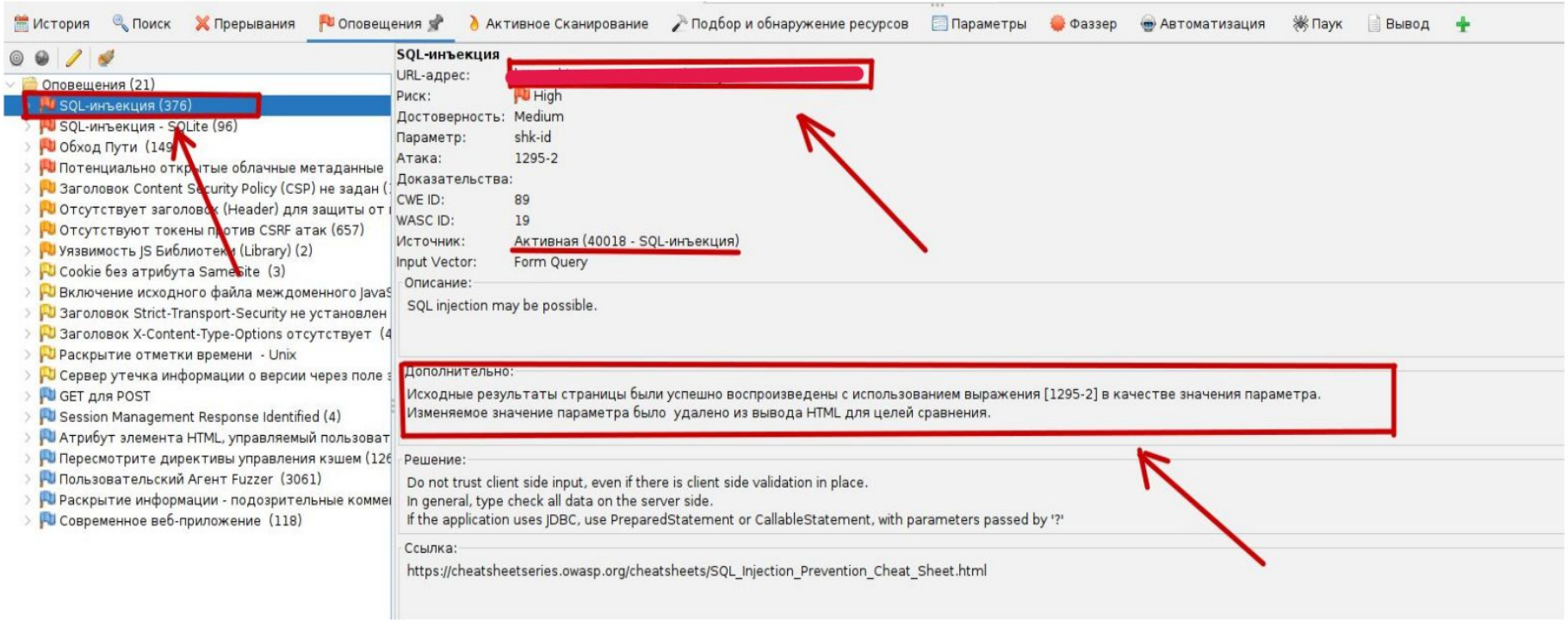

Файл Правка Вид Поиск Терминал Справка
[23:34:05] [INFO] testing 'IBM DB2 error-based - ORDER BY clause'
[23:34:06] [INFO] testing 'Microsoft SQL Server/Sybase error-based - Stacking (EXEC)'
[23:34:24] [INFO] testing 'Generic inline queries'
[23:34:24] [INFO] testing 'MySQL inline queries'
[23:34:25] [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 on
e other (potential) technique found
[23:37:22] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'

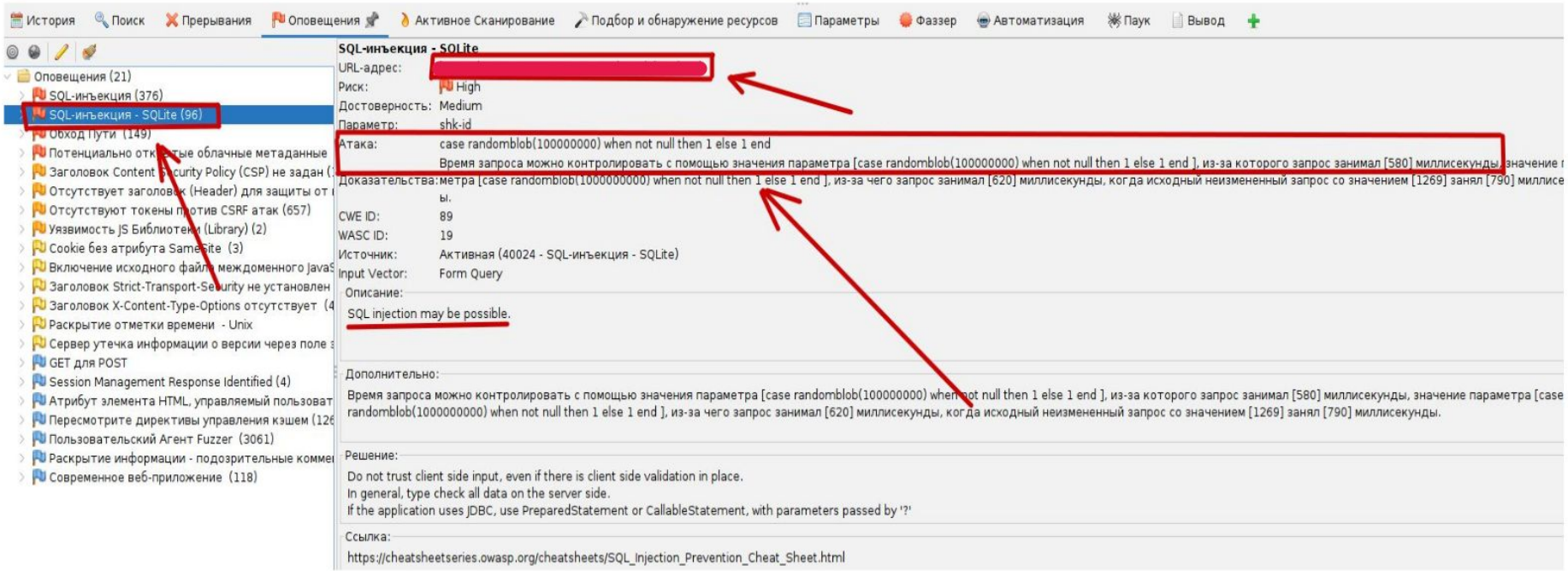

Файл Правка Вид Поиск Терминал Справка
[00:05:16] [INFO] testing 'HSQLDB >= 1.7.2 stacked queries (heavy query)'
[00:05:52] [INFO] testing 'HSQLDB >= 2.0 stacked queries (heavy query - comment)'
[00:06:16] [INFO] testing 'HSQLDB >= 2.0 stacked queries (heavy query)'
[00:06:52] [INFO] testing 'MySQL >= 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)'

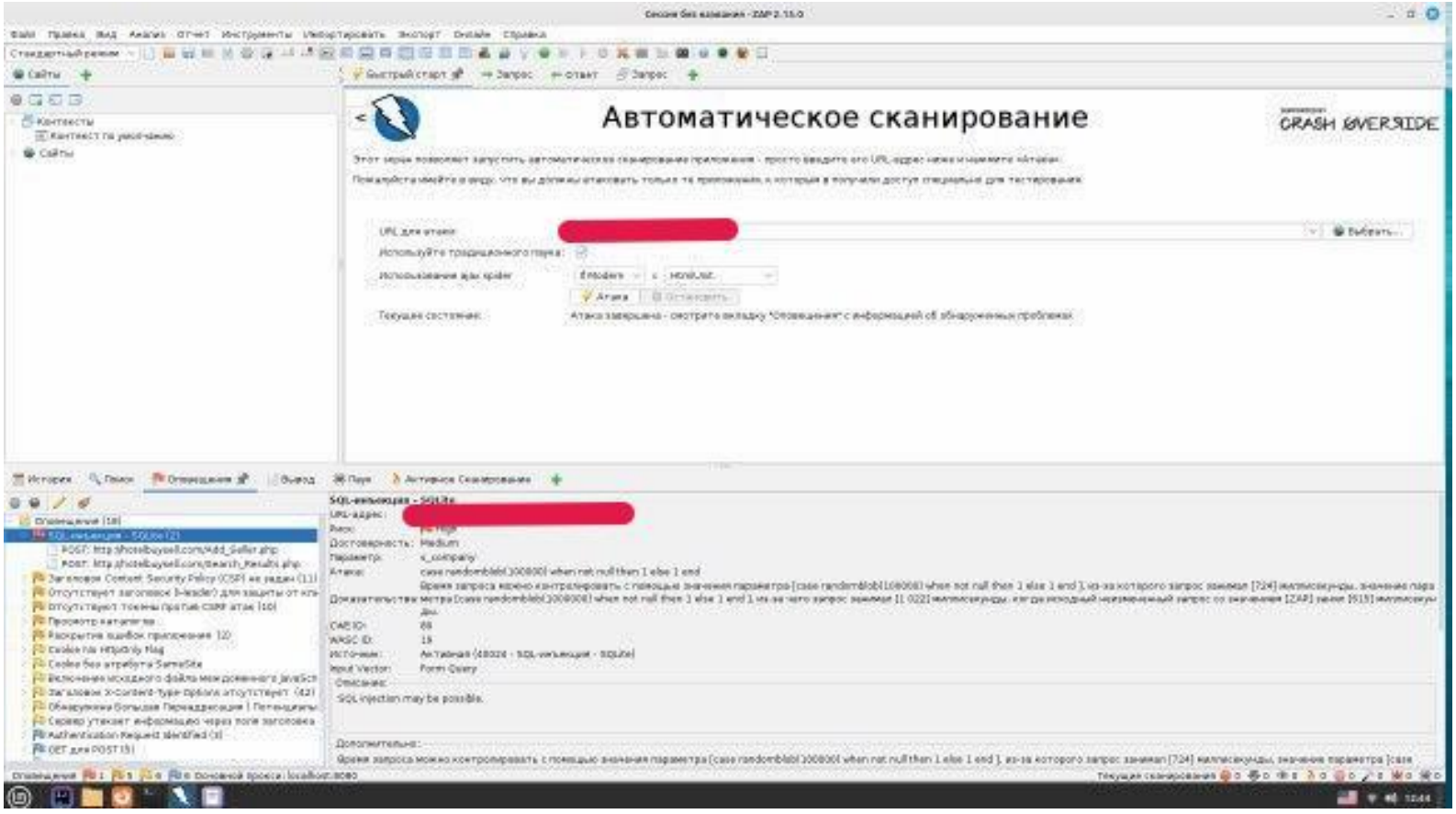

Файл Правка Вид Поиск Терминал Справка
[22:20:05] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
[22:20:23] [INFO] target URL appears to be UNION injectable with 14 columns
[22:21:20] [WARNING] if UNION based SQL injection is not detected, please consider and/or try to force the back-end
DBMS (e.g. '--dbms=mysql')
[22:21:20] [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
[22:26:49] [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'
[22:29:03] [INFO] testing 'Generic UNION query (13) - 21 to 40 columns'


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:




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
Отсутствует заголовок (Header) для защиты от
Отсутствуют токены против CSRF атак (657)
Уязвимость JS Библиотеки (Library) (2)
Cookie без атрибута SameSite (3)
Заголовок Strict-Transport-Security не установлен
Заголовок X-Content-Type-Options отсутствует
Раскрытие отметки времени - Unix
GET для POST
Session Management Response Identified (4)
Пересмотрите директивы управления кэшем
Пользовательский Агент Fuzzer (3061)
Современное веб-приложение (118)
SQL-инъекция - SQLite
URL-адрес:
Риск: High
Достоверность: Medium
Параметр: shk-id
Атака: case randomblob(100000000) when not null then 1 else 1 end
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
Доказательства: метра [case randomblob(1000000000) when not null then 1 else 1 end], из-за чего запрос занимал [620] миллисекунды, когда исходный неизмененный запрос со значением [1269] занял [790] миллисе
ы.
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 неизмененный 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


Автоматическое сканирование
CRASH OVERRIDE
Этот экран позволяет запустить автоматическое сканирование приложения - просто введите его URL-адрес ниже и нажмите «Атака».
Пожалуйста, имейте в виду, что вы должны атаковать только те приложения, к которым у вас есть разрешение на тестирование.
URL для атаки:
Выбрать...
Используйте традиционного паука:
Использование ajax spider:
Firefox
Атака
Остановить
Текущее состояние:
Атака завершена - смотрите вкладку "Оповещения" с информацией об обнаруженных проблемах.
SQL-инъекция - SQLite
URL-адрес:
Риск:
Достоверность: Medium
Параметр: x_company
Атака: case randomblob(1000000) when not null then 1 else 1 end
Время запроса можно контролировать с помощью значения параметра [case randomblob(1000000) when not null then 1 else 1 end], из-за которого запрос занимал [724] миллисекунды, значение пара
Доказательство: метра [case randomblob(1000000) when not null then 1 else 1 end], из-за чего запрос занимал [1 022] миллисекунды, когда исходный неизменный запрос со значением [ZAP] занял [613] миллисекун
ды.
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(1000000) when not null then 1 else 1 end], из-за которого запрос занимал [724] миллисекунды, значение параметра [case

Вот пример как выглядят доступ после поного цикла сканирований и работы с найдеными уязвимостями - админка сайта генератора прокси. Он использовал api royal для генерации резедентных проски, 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 документацие 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 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 фаил-ответ от nmap мы сортируем на только сайты где есть cve 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 фаил только с сайтами с интересующими уязвимостями.

- Перевод URL -> IP/IP+PORT
 Передаем на вход фаил 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.

- Обьеденить фаилы в один
общий
 Просто обьединяет 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 фаил 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 фаил. Далее этот фаил отрабатывается через метасплоит например. Вы можете добавить свои эксплоит 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 эксплоит.

- 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 использую 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 проходится 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 додйдем 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 перебируться 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 количивство в зависимости от конфигурации вашего пк. Усредненное значение написано к каждому методу в комментариях в исходниках. Так же рекомендуется поставить либо прокси, либ перекинуть весь трафик через тор, что бы ваш реальный ip адрес не попал в блек листы.

- Узнать IP сайта в обход WAF

Простой перевод 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 эксплотации нужно будет воспользоваться инструментом по мощнее, таким как zap/sqlmap/окунь. Данный функционал реализван для отбора самых слабых ресурсов. Но стоит отметить что та же sql иньекция проверяет через пост гет запросы, юзер агент, куки. Имеет несколько стандартных пайлаудов для иньекции. И проверка на каждую уязвимоть 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 выводиться информация о их версиях! Данный метод подходит что бы отсеить только сайты с определенным открытым портом(каким нибудь 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 эксплотации, 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 эксплотируем 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 подводиться 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 фаил. Который можно прикрепить к письму, для более красивого официального стиля.