

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по «производственной» практике

**Тема: Динамическая автоматизация сканирования периметра для
обнаружения уязвимостей с помощью OpenSource инструментов.**

Студент гр. 8306

Занин Д.С.

Руководитель

Юшкевич.И.А.

Санкт-Петербург

2023

ЗАДАНИЕ

НА «УЧЕБНУЮ» ПРАКТИКУ

Студент Занин Д.С.

Группа 8306

Тема практики: Динамическая автоматизация сканирования периметра для обнаружения уязвимостей с помощью Open Source инструментов.

Задание на практику:

Создание графического интерфейса для программного обеспечения предназначенного для сканирования opensource инструментов. Корректировка очередности вызова инструментов и самих инструментов, обработки результатов выполнения и дальнейшее перенаправления данных для оптимизации работы используемых утилит. Корректировка программного комплекса позволяющего проанализировать домены и/или IP адреса на наличие уязвимостей.

Все задачи были выполнены в соответствии с поставленным планом и календарным графиком.

Сроки прохождения практики: 01.12.2023 – 19.06.2023

Дата сдачи отчета: 20.12.2023

Дата защиты отчета: 27.12.2023

Студент		Занин Д.С.
Руководитель		Юшкевич И.А.

АННОТАЦИЯ

Сканирование сетевого периметра на наличие уязвимостей это то, что должно помочь организации найти прорехи в сетевом периметре и не дать злоумышленнику ими воспользоваться. В данной работе предоставляется результат автоматизации по данной этой теме. Демонстрируется разработанный графический интерфейс. Представлена работа тех инструменты, которые рассматривались в обзоре литературы [5] и использовались в данной работе для решения поставленной проблемы, а также прочих инструментов применяемых для создания программного обеспечения.

SUMMARY

Scanning the network perimeter for vulnerabilities is something that should help an organization find gaps in the network perimeter and prevent an attacker from taking advantage of them. This paper provides the result of automation on this topic. The developed graphical interface is demonstrated. The work of those tools that were considered in the literature review [5] and used in this work to solve the problem, as well as other tools used to create software, is presented.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	5
1. Постановка проблем и конкретизация задач.....	7
2. Создание программы.....	9
2.1. Замена инструментов сканирования.....	9
2.2. Написание программного обеспечения.....	9
2.2.1 Подготовка к запуску OpenSource инструментов.....	11
2.2.2 Поиск поддоменов с помощью subfinder	12
2.2.3 Сканирование с помощью nuclei.....	12
2.3. Добавим приложение на GitHub.....	13
2.4. Создадим графический интерфейс.....	13
3. Результаты работы приложения.....	21
3.1. После запуска docker.....	21
Выводы:.....	27
ПЛАН РАБОТЫ НА ВЕСЕННИЙ СЕМЕСТР.....	28
Описание предполагаемого метода решения.....	28
ЗАКЛЮЧЕНИЕ	29
Приложение А. Исходный код скрипта requirements.sh.....	30
Приложение Б. Исходный код скрипта requirements.sh.....	33
Приложение В. Исходный код Dockerfile.....	33
Приложение Г. Исходный код docker-compose.yml.....	33
Приложение Д. Исходный код index.html.....	34
Приложение Е. Исходный код styles.css	36
Приложение Ж. Исходный код server.py.....	37
Приложение З. Исходный код START_Windows.bat.....	38
Приложение И. Исходный код start_Windows.ps1	38
Приложение К. Исходный код start_scan_target.bat.....	38

Определения, обозначения и сокращения

программное обеспечение (ПО)

операционная система (ОС)

ВВЕДЕНИЕ

В данной работе будут представлено решение поставленной задачи и продемонстрирована работа написанного программного обеспечения.

Актуальность — Сканирование сетевого периметра на наличие уязвимостей - задача актуальная, и довольно часто требующая своего регулярного выполнения. Однако современные готовые продукты создаваемые для решения поставленной задачи, предлагаемые на рынке, имеют ряд недостатков. А именно, подобные решения, зачастую могут разрабатываться для конкретной компании или задачи, трудно встраиваются и адаптируются под необходимые требования, в недостатки также можно выделить высокую стоимость, и что часто бывает – медленную скорость работы. Зачастую требуется не доскональная проверка сетевого периметра на наличие всех возможных уязвимостей, а быстрая проверка на наличие критических уязвимостей в периметре сети. Совокупность всех перечисленных проблем, делает актуальным создание программы способной их решить. Opensource инструменты являются терминальными и не удобными для обычного пользователя, что делает актуальным создание графического интерфейса.

Проблема — Добавить графический интерфейс динамической в программу автоматизации сканирования периметра для обнаружения уязвимостей.

Объект исследования — исследование возможности добавления графического интерфейса в объединённую работу различных OpenSource инструментов для сканирования периметра.

Предмет исследования — Автоматизация, оценка и исследование возможности доступных инструментов для тестирования на проникновение.

Цель работы — Основная задача состоит в написании программного продукта, который позволяет автоматизировать сканирование сетевого периметра с использованием известных инструментов.

Задача данной работы: Реализовать графический интерфейс совместную работу OpenSource инструментов для сканирования цели на наличие уязвимостей в периметре её сети. Поскольку используемые инструменты имеют разный интерфейс и по-разному готовят выходные данные.

Практическая ценность работы: Результат работы позволит использовать бесплатные, очень мощные и гибкие Open Source инструменты для динамической автоматизации сканирования периметра с целью обнаружения уязвимостей используя простой в управлении графический интерфейс, которым смогут воспользоваться и обычные пользователи программного обеспечения, а не только профессионалы.

1. Постановка проблем и конкретизация задач

В соответствии с заданием на практику, аннотации и актуальности можно выделить ряд проблем которые требуют решения и которые я рассматривал в обзоре литературы по данной теме, в весеннем семестре [5]:

- Медленная работа программ по поиску уязвимостей.
- Высокая стоимость аналогов, т.е. готовых решений.
- Отсутствие гибкости используемых инструментов при сканировании сети.
- Автоматизация, процесса сканирования.
- Отсутствие графического интерфейса у большей части OpenSource инструментов.

Соответственно можно сформулировать ряд задач которые будут решаться в данной работе:

- Созданная программа должна использовать написанный графический интерфейс. Это позволит расширить круг людей желающих использовать гибкость OpenSource инструментов, но не имеющих для этого необходимого профессионального опыта пользования этими инструментами в терминале ОС.
- Созданная программа, для автоматизации сканирования сетевого периметра должна работать достаточно быстро, что бы её можно было запускать несколько раз в час или несколько раз в день, в зависимости от объёма сканируемой сети. Эта работа продельвается не для получения высоких результатов в показателях скорости работы разрабатываемой программы, а скорее в целях демонстрации возможностей использования инструментов, которые и были выбраны для реализации поставленной задачи. Решение проблемы скорости

работы программы предполагается при написании магистерской диссертации.

- Использование Open Source инструментов само по себе предполагает, что полученное решение будет бесплатным. Приведённые в данной работе инструменты я рассматривал в обзоре литературы по данной теме, в весеннем семестре [5], все кроме nslookup, которая является бесплатной утилитой Linux и доступна к установке через `apt install dnsutils`.
- Инструменты которые будут использоваться в данной работе, и которые уже рассматривались в обзоре литературы, очень гибкие [5]. Все инструменты представленные в данной работе являются терминальными утилитами, поэтому легко встраиваются в процесс сканирования. Желательно, что бы получившееся решение подразумевало возможность встраивания дополнительных утилит (при необходимости), которые не рассматриваются в данной работе. Также сюда можно отнести кроссплатформенность, т.е. желательно, чтобы полученное решение работало на любых операционных системах или по крайней мере на большинстве операционных систем (ОС).
- Полученная программа должна работать автоматически, т.е. без участия пользователя в поэтапном сборе, обработке и анализе полученной информации. Она должна лишь предоставлять результат своей работы, после её запуска и при необходимости отображать результаты своей работы в процессе нахождения уязвимостей. Если есть необходимость в передаче полученных данных с выхода одной утилиты на вход другой, это также должно происходить автоматически, без участия пользователя.

2. Создание программы

2.1. Замена инструментов сканирования

В прошлой версии программы использовался Amass в качестве сканера доменного имени цели на наличие поддоменов для составления карты сети. Из-за изменения в выводе работы программы, а также из-за приоритета в скорости работы было принято решение заменить Amass на Subfinder.

Замена инструмента привела к значительному приросту в скорости перебора поддоменов. И решила проблему изменённого вывода результатов в работе Amass, что приводило к проблеме передачи полученных данных следующим по очереди инструментам, которые уже были заточены на конкретный вывод результата работы сканера поддоменов целевого домена.

2.2. Написание программного обеспечения

Логика работы:

1. После клонирования программы из репозитория запускаем файл START_Windows.bat, который служен для запуска приложения.
 - а. Первое что делает этот .bat скрипт это запускает файл start_Windows.ps1, который распаковывает архив лежащий в протативном FireFox браузере. Этот браузер необходим для успешной работы скрипта на любой ОС windows не зависимо от того, какие браузеры установлены у пользователя на ПК. Открытие графического интерфейса и взаимодействие с ним происходят именно через протативный браузер FireFox. Причина по, которой было решено использовать именно Web страницу с графическим интерфейсом, а не оконное приложение состоит в том, что в дальнейшем планируется, не только задание начальных параметров для сканирования цели, но также планируется

воспользоваться этим при сохранении и отображении результатов сканирования в браузере.

- б. Далее приложение запускает локал хост `http://127.0.0.1:5000` на порту 5000. Это позволит работать пользователю с приложением именно как с удалённым сервером.

2. Перед пользователем открывается приложение в браузере:

Теги	Критичность
<input type="checkbox"/> CVE	<input type="checkbox"/> Info
<input type="checkbox"/> Panel	<input type="checkbox"/> High
<input type="checkbox"/> Wordpress	<input type="checkbox"/> Medium
<input type="checkbox"/> XSS	<input type="checkbox"/> Critical
<input type="checkbox"/> Exposure	<input type="checkbox"/> Low
<input type="checkbox"/> WP-Plugin	<input type="checkbox"/> Unknown
<input type="checkbox"/> OSINT	
<input type="checkbox"/> Tech	
<input type="checkbox"/> LFI	
<input type="checkbox"/> EDB	

Set Parameters START

Рис. 1 – вид графического интерфейса

3. Пользователь вводит домен цели и выбирает нужные теги и критичность уязвимостей, которые мы ищем.
4. После пользователь должен нажать кнопку “Set Parameters” для загрузки указанных параметров в текстовые файлы `domain_name.txt`, `severity_vulnerabilities.txt` и `tags_vulnerabilities.txt`, которые создаются в папке `automated_scanning`.

5. Затем пользователь нажимает кнопку “START” и запускается файл `start_scan_target.bat`, который отвечает за запуск `docker-compose`.
6. Т.к. файлы `domain_name.txt`, `severity_vulnerabilities.txt` и `tags_vulnerabilities.txt` были помещены в папку `automated_scanning`, то при запуске `docker` контейнера эти файлы также окажутся в контейнере и скрипт `script.sh`, который был передан, теперь берёт данные из файлов `domain_name.txt`, `severity_vulnerabilities.txt` и `tags_vulnerabilities.txt`.
 - a. Из `domain_name.txt` берётся домен цели и передаётся в `Subfinder`.
 - b. Из `severity_vulnerabilities.txt` и `tags_vulnerabilities.txt` берётся критичность уязвимости, которые мы собираемся искать, и теги уязвимостей, которые мы собираемся искать. Эти данные передаются `nuclei` и процесс сканирования запускается в соответствии с указанными пользователем параметрами.
7. Сохраняется результат, как и в прошлой версии программы.

2.2.1 Подготовка к запуску OpenSource инструментов.

```
3  # Считываем имя домена из файла
4  domen=`cat domain_name.txt`
```

Рис. 2 – Скрипт `script.sh` (часть 1)

Первое, что происходит это считывание из файла `domain_name.txt` доменного имени цели и сохранение полученной от пользователя информации в переменную.

```

16  ###-1 Добавляем необходимые флаги
17  severity_and_tags_flag="-tags "
18  severity_and_tags_flag+ "`cat tags_vulnerabilities.txt`"
19  severity_and_tags_flag+ " -severity "
20  severity_and_tags_flag+ "`cat severity_vulnerabilities.txt`"
21
22
23  while getopts "x d s" ARG; do
24      case "$ARG" in
25          x) xml="Yes";; # Флаг nmap - для сохранения вывода nmap в xml файле
26          d) dom="Yes";; # Флаг nmap - для сканирования поддоменов
27          # Флаг nuclei - указывающий на серьёзность цели сканирования -severity и используемые -tags
28          s) severity_and_tags=severity_and_tags_flag;;
29          {) echo "argument missing";;
30          \?) echo "Something is wrong";;
31      esac
32  done

```

Рис. 3 – Скрипт script.sh (часть 2)

Сохраняем полученную из файлов severity_vulnerabilities.txt и tags_vulnerabilities.txt информацию в правильном виде в переменную для дальнейшего использования.

2.2.2 Поиск поддоменов с помощью subfinder .

```

36  ###-1 Запускаем subfinder и сохраняем результат работы в файл:
37  subfinder -d $domain > ./ $name_folder/inf/subfinder_out.txt
38  echo "subfinder - закончил работу"

```

Рис. 4 – Скрипт script.sh (часть 3)

Запускаем subfinder для сбора поддоменов целевого ресурса.

2.2.3 Сканирование с помощью nuclei.

```

###-4 Запускаем nuclei на найденных открытых портах для найденных ip-адресов
nuclei -l ./ $name_folder/inf/sort_nmap_out_kopiy_ip_ports.txt -o ./ $name_folder/inf/nuclei_out.txt $severity_and_tags

```

Рис. 5 – Скрипт script.sh (часть 4)

Запускаем nuclei используя записанные в переменную (Рис. 3) параметры .

В остальном скрипт работает как и в прошлой версии.

2.3. Добавим приложение на GitHub

Это добавит возможность получить доступ к приложению через простое клонирование репозитория на локальную машину.

Ссылка для клонирования репозитория (ветка: test_4):
https://github.com/JackoPrograms/automated_scanning.git [3]

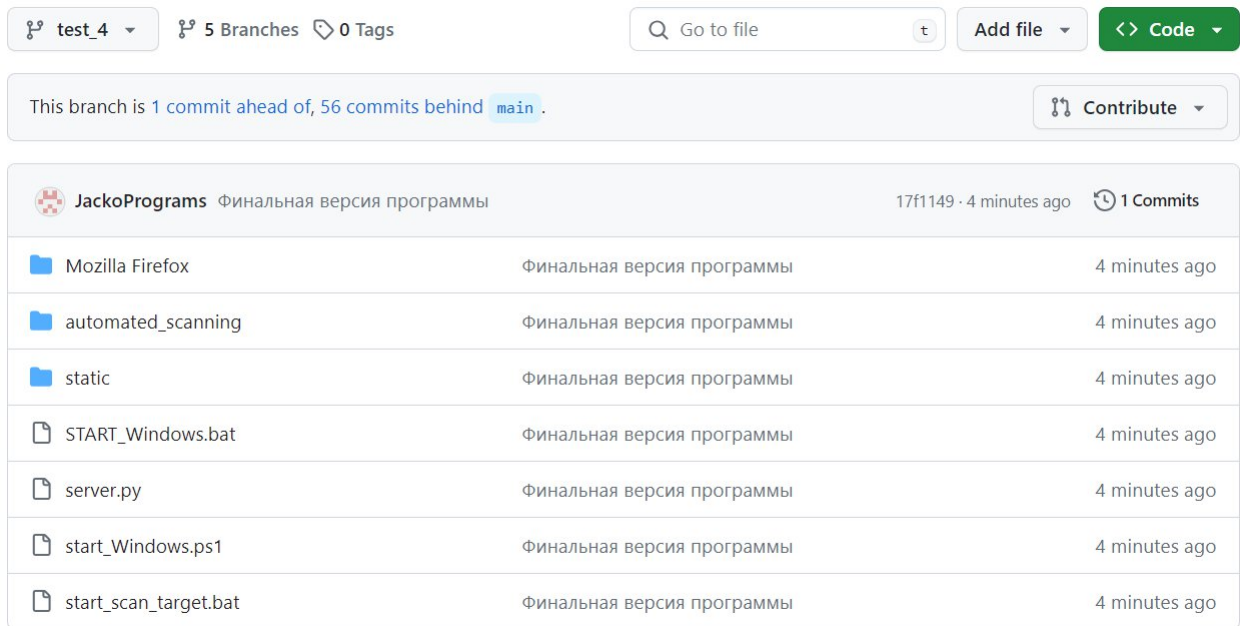


Рис. 6 – Проект загружен на GitHub

2.4. Создадим графический интерфейс

Для этого, напомним web приложение: html страницу, js скрипт, добавим css стили и напишем web сервер.

2.4.1 Создадим html страницу и добавим js скрипт.

Интерфейс будет состоять из поля input для ввода текста, чек поинтов дающим выбрать tag (теги) и severity (серьёзность) уязвимости. Ещё на странице присутствуют 2 кнопки. Кнопка “Set Parameters” необходима для загрузки указанных параметров в текстовые файлы domain_name.txt, severity_vulnerabilities.txt и tags_vulnerabilities.txt, которые создаются в папке

automated_scanning. Кнопка “START” запускает файл start_scan_target.bat, который отвечает за запуск docker-compose.

Такой подход поможет при написании магистерской диссертации добавить в приложение возможность настройки всех OpenSource инструментов.

Html файл имеет следующий вид:

```
<!DOCTYPE html>
<html>
<head>
  <title>Button Example</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div class="h1_title">
    <h1>Сканер уязвимостей</h1>
  </div>
  <input type="text" id="domain_name" placeholder="Введите доменное имя цели">
  <div class="container">
    <div class="column">
      <h5>Теги</h5>
      <label><input type="checkbox" id="cve"> CVE</label>
      <label><input type="checkbox" id="panel"> Panel</label>
      <label><input type="checkbox" id="wordpress"> Wordpress</label>
      <label><input type="checkbox" id="xss"> XSS</label>
      <label><input type="checkbox" id="exposure"> Exposure</label>
      <label><input type="checkbox" id="wp-plugin"> WP-Plugin</label>
      <label><input type="checkbox" id="osint"> OSINT</label>
      <label><input type="checkbox" id="tech"> Tech</label>
      <label><input type="checkbox" id="lfi"> LFI</label>
      <label><input type="checkbox" id="edb"> EDB</label>
    </div>
    <div class="column">
      <h5>Критичность</h5>
      <label><input type="checkbox" id="info"> Info</label>
      <label><input type="checkbox" id="high"> High</label>
      <label><input type="checkbox" id="medium"> Medium</label>
      <label><input type="checkbox" id="critical"> Critical</label>
      <label><input type="checkbox" id="low"> Low</label>
      <label><input type="checkbox" id="unknown"> Unknown</label>
    </div>
  </div>
</div>
```

Рис. 7 – файл index.html (часть 1)


```

<div class="button-container">
  <button onclick="sendAll()">Set Parameters</button>
  <button onclick="startScan()">START</button>
</div>
<script>
  function sendAll() {
    sendDomainName();
    sendVulnerabilityTags();
    sendSeverity();
  }

  function sendDomainName() {
    var userInput = document.getElementById('domain_name').value;
    var xhr = new XMLHttpRequest();
    xhr.open('POST', '/create_domain_file', true);
    xhr.setRequestHeader('Content-Type', 'text/plain');
    xhr.send(userInput);
  }

  function sendVulnerabilityTags() {
    var checkboxes = ['cve', 'panel', 'wordpress', 'xss', 'exposure', 'wp-plugin', 'osint', 'tech', 'lfi', 'edb'];
    var tags_vulnerabilities = checkboxes.map(function(id) {
      return document.getElementById(id).checked ? id.charAt(0).toUpperCase() + id.slice(1) + ',' : '';
    }).join('');

    var xhr = new XMLHttpRequest();
    xhr.open('POST', '/create_tags_file', true);
    xhr.setRequestHeader('Content-Type', 'text/plain');
    xhr.send(tags_vulnerabilities.slice(0, -1)); // Удаляем последнюю запятую
  }
}

```

Рис. 8 – файл index.html (часть 2)

```

  function sendSeverity() {
    var severities = ['info', 'high', 'medium', 'critical', 'low', 'unknown'];
    var severity = severities.map(function(id) {
      return document.getElementById(id).checked ? id.charAt(0).toUpperCase() + id.slice(1) + ',' : '';
    }).join('');

    var xhr = new XMLHttpRequest();
    xhr.open('POST', '/create_severity_file', true);
    xhr.setRequestHeader('Content-Type', 'text/plain');
    xhr.send(severity.slice(0, -1));
  }

  function startScan() {
    var xhr = new XMLHttpRequest();
    xhr.open('POST', '/start_scan_command', true);
    xhr.send(null);
  }
</script>
</body>
</html>

```

Рис. 9 – файл index.html (часть 3)

Описание js скрипта:

1. sendAll(): Эта функция вызывает три другие функции: sendDomainName(), sendVulnerabilityTags(), и sendSeverity(). Она

используется для отправки всех параметров (доменного имени, тегов уязвимостей и критичности) на сервер.

2. `sendDomainName()`: Эта функция извлекает значение, введенное пользователем в поле ввода с `id="domain_name"`, и отправляет его на сервер с помощью объекта `XMLHttpRequest`. Запрос отправляется методом `POST` на URL `/create_domain_file`, с использованием типа контента `text/plain`.
3. `sendVulnerabilityTags()`: Эта функция извлекает состояние всех чекбоксов тегов уязвимостей, создает строку тегов на основе состояния чекбоксов и отправляет ее на сервер методом `POST` на URL `/create_tags_file`, используя тип контента `text/plain`.
4. `sendSeverity()`: Эта функция работает аналогично функции `sendVulnerabilityTags()`, за исключением того, что она обрабатывает чекбоксы критичности уязвимостей.
5. `startScan()`: Эта функция отправляет команду на сервер для запуска сканирования методом `POST` на URL `/start_scan_command`.

2.4.2 Создадим css стили.

```
161  /* Устанавливает цвет фона и цвет текста для всего документа */
162  body {
163      background-color: ☐ #1E1E1E;
164      color: ☒ #E0E0E0;
165      margin: 0; /* Убирает отступы */
166      padding-top: 20px; /* Добавляет отступ сверху */
167  }
168
169  /* Стиль для поля ввода доменного имени */
170  #domain_name {
171      display: block; /* Устанавливает блочное отображение элемента */
172      width: 350px; /* Устанавливает ширину поля ввода */
173      box-sizing: border-box; /* Учитывает границы внутри ширины элемента */
174      margin-left: 5px; /* Устанавливает отступ слева */
175      margin-right: 5px; /* Устанавливает отступ справа */
176  }
177
178  /* Общий контейнер, в котором отображаются все элементы в виде горизонтальной линии */
179  .container {
180      display: flex;
181  }
182
183  /* Стиль для колонок в контейнере */
184  .column {
185      padding: 10px; /* Добавляет внутренний отступ */
186  }
187
188  /* Стиль для меток внутри колонок */
189  .column label {
190      display: block; /* Устанавливает блочное отображение для меток */
191      margin-bottom: 5px; /* Добавляет отступ снизу */
192  }
193
```

Рис. 10 – файл styles.css (часть 1)

```
194  /* Стиль для чекбоксов */
195  input[type="checkbox"] {
196      margin-right: 5px; /* Устанавливает отступ справа */
197  }
198
199  /* Стиль для текстовых полей, чекбоксов и кнопок */
200  input[type="text"],
201  input[type="checkbox"],
202  button {
203      background-color: ☐ #2E2E2E; /* Устанавливает цвет фона */
204      color: ☒ #E0E0E0; /* Устанавливает цвет текста */
205      border: none; /* Убирает границы */
206      padding: 10px; /* Добавляет внутренний отступ */
207      margin-bottom: 5px; /* Добавляет отступ снизу */
208      margin-left: 5px; /* Устанавливает отступ слева */
209  }
```

Рис. 11 – файл styles.css (часть 2)

```

211  /* Стилъ для контейнера с кнопками */
212  .button-container {
213      display: flex; /* Устанавливает отображение в виде горизонтальной линии */
214      justify-content: flex-start; /* Выравнивание кнопок по левому краю */
215  }
216
217  /* Стилъ для кнопок */
218  button {
219      cursor: pointer; /* Изменяет курсор при наведении на кнопку */
220      margin-right: 20px; /* Устанавливает отступ между кнопками */
221  }
222
223  /* Стилъ для заголовка h1 */
224  .h1_title {
225      margin-left: 10px; /* Устанавливает отступ слева */
226  }
227
228  /* Стилъ для поля ввода доменного имени в контейнере */
229  .input_domen {
230      margin-left: 5px; /* Устанавливает отступ слева */
231      margin-right: 5px; /* Устанавливает отступ справа */
232      width: 100%; /* Устанавливает ширину 100% от родительского элемента */
233  }

```

Рис. 12 – файл styles.css (часть 3)

2.4.3 Создадим сервер.

```

from flask import Flask, request, send_from_directory
import os

app = Flask(__name__, static_folder='static')

@app.route('/', methods=['GET'])
def index():
    return send_from_directory(app.static_folder, 'index.html')

@app.route('/create_domain_file', methods=['POST'])
def create_domain_file():
    text = request.data.decode('utf-8')
    with open('./automated_scanning/domain_name.txt', 'w') as file:
        file.write(text)
    return 'Domain name saved'

```

Рис. 13 – файл server.py (часть 1)

```

@app.route('/create_tags_file', methods=['POST'])
def create_tags_file():
    tags = request.data.decode('utf-8')
    with open('./automated_scanning/tags_vulnerabilities.txt', 'w') as file:
        file.write(tags)
    return 'Tags saved'

@app.route('/create_severity_file', methods=['POST'])
def create_severity_file():
    severity = request.data.decode('utf-8')
    with open('./automated_scanning/severity_vulnerabilities.txt', 'w') as file:
        file.write(severity)
    return 'Severity saved'

@app.route('/start_scan_command', methods=['POST'])
def start_scan_command():
    try:
        # Предполагается, что start_scan_target.bat находится в той же папке, что и server.py
        os.system('start_scan_target.bat')
        return 'Scan started'
    except Exception as e:
        return str(e), 500

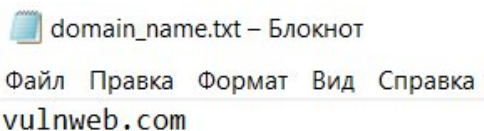
if __name__ == '__main__':
    app.run()

```

Рис. 14 – файл server.py (часть 2)

Часть приложения отвечающая за сканирование целевого домена в docker контейнере осталась неизменной. Единственное изменение это добавленные в docker контейнер 3 txt файлов () отвечающих за передачу в скрипт script.sh заданных пользователем параметров domain_name.txt, severity_vulnerabilities.txt и tags_vulnerabilities.txt.

2.4.4 Примерный вид txt файлов.

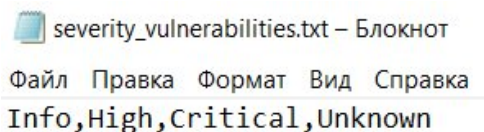


domain_name.txt – Блокнот

Файл Правка Формат Вид Справка

vulnweb.com

Рис. 15 – файл domain_name.txt



severity_vulnerabilities.txt – Блокнот

Файл Правка Формат Вид Справка

Info,High,Critical,Unknown

Рис. 16 – файл severity_vulnerabilities.txt

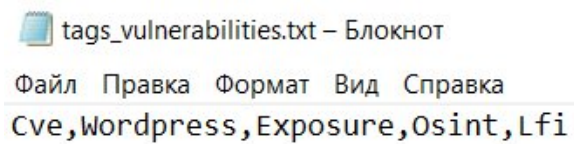


Рис. 17 – файл tags_vulnerabilities.txt

Так будут выглядеть файлы после выбора соответствующих чек-поинтов, ввода домена в поле input и нажатия кнопки “Set Parameters”.

A screenshot of a web application titled 'Сканер уязвимостей' (Vulnerability Scanner). The interface has a dark theme. At the top, there is a header with the title. Below it is a text input field containing 'vulnweb.com'. The main area is divided into two columns: 'Теги' (Tags) and 'Критичность' (Criticality). Each column contains a list of items with checkboxes. At the bottom, there are two buttons: 'Set Parameters' and 'START'.

Рис. 18 – Задание параметров сканирования.

По итогу, до запуска сканирования – проект имеет следующий вид:

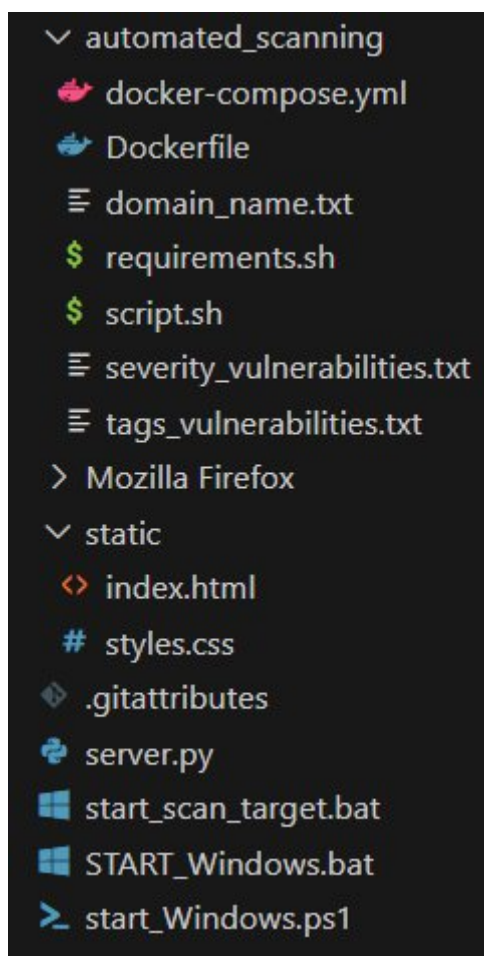


Рис. 19 – Структура проекта.

3. Результаты работы приложения.

3.1. После запуска docker.

Запуск приложения и процесс его работы. Происходит запуск docker-compose.

Происходит сборка и запуск докера, после чего запускается скрипт и нам предлагается ввести домен цели. В качестве цели были выбраны уязвимые тестовые веб-сайты для сканера веб-уязвимостей Acunetix [4].

```

$ docker-compose run docker_scan
[+] Building 0.0s (0/0)
[+] Creating 1/0
✓ Network automated_scanning_default Created 0.1s
[+] Building 58.3s (9/9) FINISHED
=> [docker_scan internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [docker_scan internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 181B 0.0s
=> [docker_scan internal] load metadata for docker.io/kalilinux/kali-rolling:latest 0.4s
=> [docker_scan 1/4] FROM docker.io/kalilinux/kali-rolling@sha256:5e4a532e4f4e2d5838c4867f8773a57182b75fca323d589029c19d6d6f97 6.1s
=> => resolve docker.io/kalilinux/kali-rolling@sha256:5e4a532e4f4e2d5838c4867f8773a57182b75fca323d589029c19d6d6f97d2cd 0.0s
=> => sha256:5e4a532e4f4e2d5838c4867f8773a57182b75fca323d589029c19d6d6f97d2cd 1.19kB / 1.19kB 0.0s
=> => sha256:9c5f1bf6e500619013035ad96d9fd8ef28738726b5b9d2c1cc10b6263ce3a952 429B / 429B 0.0s
=> => sha256:588662d6709a206b2e4c46a20f03eacebaedb23ebd951777c263ef1a122fbdc8 2.85kB / 2.85kB 0.0s
=> => sha256:29441601816ce6668dbecf58f20b29e879cc88865bad5a82429e7837d1c68482 51.21MB / 51.21MB 0.5s
=> => extracting sha256:29441601816ce6668dbecf58f20b29e879cc88865bad5a82429e7837d1c68482 5.3s
=> [docker_scan internal] load build context 0.0s
=> => transferring context: 49.59kB 0.0s
=> [docker_scan 2/4] WORKDIR /script 0.0s
=> [docker_scan 3/4] COPY . /script 0.1s
=> [docker_scan 4/4] RUN bash requirements.sh 45.8s
=> [docker_scan] exporting to image 5.7s
=> => exporting layers 5.7s
=> => writing image sha256:27f7abc46fdb1c4e6602f86c1156a2e971c2f734dfbd9c66423ef4d57b22c29 0.0s
=> => naming to docker.io/library/automated_scanning-docker_scan 0.0s

```

Рис. 20 – Запуск docker (часть 1)

После запуска, спустя какое-то время, высвечивается сообщение о том, что subfinder закончил свою работу. Затем высветилась информация о найденных ip-адресах, после этого вывелась информация о работе nmap.

```

subfinder - закончил работу
35.81.188.86
44.228.249.3
44.238.29.244
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-20 10:48 UTC
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.23 seconds
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-20 10:48 UTC
Nmap scan report for ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3)
Host is up (0.085s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
Nmap done: 1 IP address (1 host up) scanned in 8.23 seconds
80
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-20 10:48 UTC
Nmap scan report for ec2-44-238-29-244.us-west-2.compute.amazonaws.com (44.238.29.244)
Host is up (0.084s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
Nmap done: 1 IP address (1 host up) scanned in 10.01 seconds
80

```

Рис. 21– Запуск docker (часть 2)

И далее в терминал выводится результат работы nuclei, процесс работы которого, т.е. процесс нахождения каждой уязвимости мы можем наблюдать в процессе сканирования:

Рис. 22— Запуск docker (часть 3)

Рис. 23— Запуск docker (часть 4)

Как видно было создано две папки, в каждой из которых находится результат сканирования цели. Содержимое этих папок:

```
START_Windows.bat
automated_scanning
├── 20-12-2023_10-48-02_vulnweb.com
│   ├── inf
│   │   ├── nmap_out_full.txt
│   │   ├── nuclei_out.txt
│   │   ├── sort_nmap_out_kopiya_ip_ports.txt
│   │   ├── sort_nslookup_out_ip.txt
│   │   └── subfinder_out.txt
│   └── worker
│       ├── nmap_out_ports.txt
│       ├── nslookup_out_ip.txt
│       └── sort_nmap_out_ports.txt
├── Dockerfile
├── docker-compose.yml
├── domain_name.txt
├── requirements.sh
├── script.sh
├── severity_vulnerabilities.txt
└── tags_vulnerabilities.txt
server.py
start_Windows.ps1
start_scan_target.bat
static
├── index.html
└── styles.css
```

Рис. 26– Структура вывода после завершения работы ПО.

Как и ожидалось, была создана папка 20-12-2023_10-48-02_vulnweb.com с результатами сканирования.

Вывод содержимого файла nuclei_out.txt с результатами сканирования на наличие уязвимостей с заданными пользователем параметрами:

```

$ cat nuclei_out.txt
[ptr-fingerprint] [dns] [info] 3.249.228.44.in-addr.arpa [ec2-44-228-249-3.us-west-2.compute.amazo
naws.com.]
[ptr-fingerprint] [dns] [info] 244.29.238.44.in-addr.arpa [ec2-44-238-29-244.us-west-2.compute.ama
zonaws.com.]
[options-method] [http] [info] http://44.238.29.244:80 [OPTIONS, TRACE, GET, HEAD, POST]
[default-windows-server-page] [http] [info] http://44.238.29.244:80
[microsoft-iis-version] [http] [info] http://44.238.29.244:80 [Microsoft-IIS/8.5]
[tech-detect:ms-iis] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:x-frame-options] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:x-permitted-cross-domain-policies] [http] [info] http://44.238.29.2
44:80
[http-missing-security-headers:cross-origin-embedder-policy] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:cross-origin-opener-policy] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:cross-origin-resource-policy] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:strict-transport-security] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:content-security-policy] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:permissions-policy] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:x-content-type-options] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:referrer-policy] [http] [info] http://44.238.29.244:80
[http-missing-security-headers:clear-site-data] [http] [info] http://44.238.29.244:80
[waf-detect:aspgeneric] [http] [info] http://44.238.29.244:80/

```

Рис. 27– Содержимое файла nuclei_out.txt.

Выводы:

Программа работает корректно и справилась со всеми поставленными перед ней задачами. Она пригодна для сканирования сетевого периметра на регулярной основе, в том числе по несколько раз (при необходимости) в день или в час, в зависимости от объёма сканируемой сети. По стоимости данная программа является бесплатной. Гибкость настройки работы скрипта ограничивается возможностями используемых OpenSource инструментов, которые уже получили своё признание у пентестиров и разработчиков, что так же освещалась в обзоре литературы прошлого семестра [5]. ПО работает автоматически, без участия пользователя, и благодаря успешной реализации docker образа оно является кроссплатформенным. Однако в данной работе графический интерфейс был написан пока только для windows. В дальнейшем при написании магистерской диссертации планируется доработка ПО до кроссплатформенного. Составление очередности вызова инструментов, обработка результатов выполнения и дальнейшее перенаправления данных для оптимизации работы используемых утилит было успешно доработано заменой Amass на Subfinder и замена способа передачи данных в subfinder и nuclei, вместо терминального ввода пользователем теперь используется автоматическая загрузка данных из txt файлов с записанными в них пользователем параметрами (при взаимодействии с графическим интерфейсом), что и обеспечило автоматическую работу по сканированию периметра сети целевого домена, без участия пользователя. В процессе выполнения работы был создан графический интерфейс и доработан скрипт script.sh.

ПЛАН РАБОТЫ НА ВЕСЕННИЙ СЕМЕСТР

Доработать графический интерфейс программы добавив в него возможность использования большей части гибкости OpenSource инструментов subfinder, nmap, nuclei и прочих. Это будет иметь ценность так как перечисленные OpenSource инструменты являются гибкими но воспользоваться их гибкостью можно только в терминале. Перенос взаимодействия с инструментами данного ПО из терминального в графический без потери гибкости их настроек - расширит круг пользователей данного ПО, так как взаимодействие с данным ПО изначально подразумевала знание терминальных утилит и навыки владения терминалом Windows / Linux / Mac.

Описание предполагаемого метода решения

Работу данной практики можно назвать демонстрацией возможности создания графического интерфейса. Предполагается доработка графического интерфейса программы до уровня при котором была бы возможность использовать гибкость OpenSource инструментов на полную мощность без потери в скорости их работы. Это позволит расширить круг людей, которые смогут использовать данное ПО основанное на терминальных утилитах, не обладая навыками работы с терминалом, а используя вместо этого графический интерфейс позволяющей также гибко настроить процесс сканирования периметра сети на наличие уязвимостей. Также возможно будет расширен список используемых OpenSource инструментов, а также вероятно будет реализован способ сохранения полученных результатов в виде оформленного файла со статистикой.

ЗАКЛЮЧЕНИЕ

При прохождении практики, было написано ПО по теме практики, с использованием Open Source инструменты для решения поставленной проблемы. Цель практики достигнута и результаты этой учебной практики впоследствии будут использованы при написании магистерской диссертации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Быстрый и настраиваемый сканер уязвимостей на основе простого DSL на основе YAML, Nuclei Templates // DSL URL: <https://github.com/projectdiscovery/nuclei-templates> (дата обращения: 30.06.23).
2. Docker образ kali-rolling // DSL URL: <https://hub.docker.com/r/kalilinux/kali-rolling> (дата обращения: 30.06.23).
3. Ссылка на репозиторий GitHub с созданным приложением, для сканирования сетевого периметра на наличие уязвимостей // DSL URL: <https://github.com/projectdiscovery/nuclei-templates> (дата обращения: 30.06.23).
4. Уязвимые тестовые веб-сайты для сканера веб-уязвимостей Acunetix. // DSL URL: `docker-compose run docker_scan`
5. Отчёт обзора литературы по теме данной работы, Занина Д.С. // DSL URL: https://disk.yandex.ru/d/QZ_aqxSoY6q_qg

Приложение А. Исходный код скрипта requirements.sh

```
#!/bin/bash

# Считываем имя домена из файла
domen=`cat domain_name.txt`

# Создаём папку для текущего запуска
now=$(date +"%d-%m-%Y_%H-%M-%S")
r="_"
name_folder="$now$r$domen"
mkdir $name_folder
mkdir $name_folder/inf
mkdir $name_folder/worker

###-1 Добавляем необходимые флаги
severity_and_tags_flag="-tags "
severity_and_tags_flag+=`cat tags_vulnerabilities.txt`
severity_and_tags_flag+=" -severity "
severity_and_tags_flag+=`cat severity_vulnerabilities.txt`

while getopts "x d s" ARG; do
    case "$ARG" in
        x) xml="Yes";; # Флаг nmap - для сохранения вывода nmap в xml файле
        d) dom="Yes";; # Флаг nmap - для сканирования поддоменов
        # Флаг nuclei - указывающий на серьёзность цели сканирования -severity и
        используемые -tags
        s) severity_and_tags=severity_and_tags_flag;;
        :) echo "argument missing";;
        \?) echo "Something is wrong";;
    esac
done

###-1 Запускаем subfinder и сохраняем результат работы в файл:
subfinder -d $domen > ./$name_folder/inf/subfinder_out.txt
echo "subfinder - закончил работу"

###-2 Запускаем поиск ip-адресов по найденным поддоменам
for line in `cat ./$name_folder/inf/subfinder_out.txt`
do
    nslookup $line >> ./$name_folder/worker/nslookup_out_ip.txt
done

# Удаляем лишнюю информацию и оставляем только ip-адреса
sed -i '/^Server:/d' ./$name_folder/worker/nslookup_out_ip.txt
sed -i '/answer/d' ./$name_folder/worker/nslookup_out_ip.txt
sed -i '/^Name:/d' ./$name_folder/worker/nslookup_out_ip.txt
sed -i '/#/d' ./$name_folder/worker/nslookup_out_ip.txt
```

```

sed -i '/find/d' ./${name_folder}/worker/nslookup_out_ip.txt
sed -i '/^$/d' ./${name_folder}/worker/nslookup_out_ip.txt
sed -i 's/Address: //' ./${name_folder}/worker/nslookup_out_ip.txt
### Удаляем лишние ip-адреса
sed -i '/^127/d' ./${name_folder}/worker/nslookup_out_ip.txt
#sed -i '/^35/d' ./${name_folder}/worker/nslookup_out_ip.txt
#sed -i '/^44.238/d' ./${name_folder}/worker/nslookup_out_ip.txt

# Удаляем все повторяющиеся ip-адреса
sort -
u ./${name_folder}/worker/nslookup_out
_ip.txt > ./${name_folder}/inf/sort_nslookup_out_ip.txt

# Выводим результат работы nslookup записанный в файл
cat ./${name_folder}/inf/sort_nslookup_out_ip.txt

###-3 ----- Здесь цикл поиска открытых портов для каждого из найденных ip и
запись в файл
#                новый формат ip:порт
if [[ ${#dom} -ne 0 ]]
then
    if [[ ${#xml} -ne 0 ]]
    then
        xml_domen="-oX ./${name_folder}/worker/nmap_out-domen.xml"
    fi

    for line_ip in `cat ./${name_folder}/inf/subfinder_out.txt`
    do
        nmap -oN ./${name_folder}/worker/nmap_out-domen.txt $xml_domen $line_ip
        cat ./${name_folder}/worker/nmap_out-
domen.txt >> ./${name_folder}/inf/nmap_out-domen_full.txt
        if [[ ${#xml} -ne 0 ]]
        then
            cat ./${name_folder}/worker/nmap_out-
domen.xml >> ./${name_folder}/inf/nmap_out-domen_full.xml
        fi
    done
fi

for line_ip in `cat ./${name_folder}/inf/sort_nslookup_out_ip.txt`
do
    ###-3.1 Запускаем сканирование открытых портов по найденным поддоменам
    echo "----- Сканируется
$line_ip : " >> ./${name_folder}/worker/nmap_out.txt

    # Проверяем что нужно включить в вывод .xml
    if [[ ${#xml} -ne 0 ]]
    then
        xml_domen="-oX ./${name_folder}/worker/nmap_out.xml"
    fi

```

```

# Запускаем nmap
nmap -oN ./${name_folder}/worker/nmap_out.txt ${xml_domen} $line_ip

# Если в вывод xml включить нужно, то
if [[ ${#xml} -ne 0 ]]
then
    # Записываем результаты работы nmap в общий файл
    cat ./${name_folder}/worker/nmap_out.xml >> ./${name_folder}/inf/nmap_out_full.xml
fi

# Записываем результаты работы nmap в общий файл
cat ./${name_folder}/worker/nmap_out.txt >> ./${name_folder}/inf/nmap_out_full.txt

echo "" >> ./${name_folder}/worker/nmap_out.txt
echo "" >> ./${name_folder}/worker/nmap_out.txt

###-3.2 Удаляем все строки кроме тех в которых встречается слово open
sed -i '/open/!d' ./${name_folder}/worker/nmap_out.txt
###-3.3 Удаляем всю информацию кроме номера порта
sed -r 's/[/].+//' ./${name_folder}/worker/nmap_out.txt > ./${name_folder}/worker/nmap_out_ports.txt
###-3.4 Удаляем все повторяющиеся порты
sort -u ./${name_folder}/worker/nmap_out_ports.txt > ./${name_folder}/worker/sort_nmap_out_ports.txt

cat ./${name_folder}/worker/sort_nmap_out_ports.txt

# Записываем в файл отсортированные результаты в виде "ip:порт"
for line_port in `cat ./${name_folder}/worker/sort_nmap_out_ports.txt`
do
    echo
    $line_ip":"$line_port >> ./${name_folder}/inf/sort_nmap_out_kopiya_ip_ports.txt
done
rm ./${name_folder}/worker/nmap_out.txt
done

# Удаляем лишние символы из полученных строк
sed -i 's/ //' ./${name_folder}/inf/sort_nmap_out_kopiya_ip_ports.txt
sed -i 's/ //' ./${name_folder}/inf/sort_nmap_out_kopiya_ip_ports.txt

```



```
###-4 Запускаем nuclei на найденных открытых портах для найденных ip-адресов
nuclei -l ./${name_folder}/inf/sort_nmap_out_kopiya_ip_ports.txt -
o ./${name_folder}/inf/nuclei_out.txt $severity_and_tags
```

Приложение Б. Исходный код скрипта requirements.sh

```
apt-get -y update
apt install subfinder
apt install dnsutils -y
apt-get install nmap -y
apt install nuclei -y
apt-get install git -y
git clone https://github.com/projectdiscovery/nuclei-templates.git
```

Приложение В. Исходный код Dockerfile

```
FROM kalilinux/kali-rolling

WORKDIR /script

COPY . /script

RUN bash requirements.sh

ENTRYPOINT [ "bash", "script.sh" ]

CMD [ "", "", "" ]
```

Приложение Г. Исходный код docker-compose.yml

```
version: '3'
services:
  docker_scan:
    build: .
    volumes:
      - ../script
```

Приложение Д. Исходный код index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Button Example</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div class="h1_title">
    <h1>Сканер уязвимостей</h1>
  </div>
  <input type="text" id="domain_name" placeholder="Введите доменное имя цели">
  <div class="container">
    <div class="column">
      <h5>Теги</h5>
      <label><input type="checkbox" id="cve"> CVE</label>
      <label><input type="checkbox" id="panel"> Panel</label>
      <label><input type="checkbox" id="wordpress"> Wordpress</label>
      <label><input type="checkbox" id="xss"> XSS</label>
      <label><input type="checkbox" id="exposure"> Exposure</label>
      <label><input type="checkbox" id="wp-plugin"> WP-Plugin</label>
      <label><input type="checkbox" id="osint"> OSINT</label>
      <label><input type="checkbox" id="tech"> Tech</label>
      <label><input type="checkbox" id="lfi"> LFI</label>
      <label><input type="checkbox" id="edb"> EDB</label>
    </div>
    <div class="column">
      <h5>Критичность</h5>
      <label><input type="checkbox" id="info"> Info</label>
      <label><input type="checkbox" id="high"> High</label>
      <label><input type="checkbox" id="medium"> Medium</label>
      <label><input type="checkbox" id="critical"> Critical</label>
      <label><input type="checkbox" id="low"> Low</label>
      <label><input type="checkbox" id="unknown"> Unknown</label>
    </div>
  </div>
  <div class="button-container">
    <button onclick="sendAll()">Set Parameters</button>
    <button onclick="startScan()">START</button>
  </div>
  <script>
    function sendAll() {
      sendDomainName();
      sendVulnerabilityTags();
      sendSeverity();
    }

    function sendDomainName() {
      var userInput = document.getElementById('domain_name').value;
      var xhr = new XMLHttpRequest();
```

```

        xhr.open('POST', '/create_domain_file', true);
        xhr.setRequestHeader('Content-Type', 'text/plain');
        xhr.send(userInput);
    }

    function sendVulnerabilityTags() {
        var checkboxes = ['cve', 'panel', 'wordpress', 'xss', 'exposure', 'wp-
plugin', 'osint', 'tech', 'lfi', 'edb'];
        var tags_vulnerabilities = checkboxes.map(function(id) {
            return document.getElementById(id).checked ?
id.charAt(0).toUpperCase() + id.slice(1) + ',' : '';
        }).join('');

        var xhr = new XMLHttpRequest();
        xhr.open('POST', '/create_tags_file', true);
        xhr.setRequestHeader('Content-Type', 'text/plain');
        xhr.send(tags_vulnerabilities.slice(0, -1)); // Удаляем последнюю запятую
    }

    function sendSeverity() {
        var severities = ['info', 'high', 'medium', 'critical', 'low',
'unknown'];
        var severity = severities.map(function(id) {
            return document.getElementById(id).checked ?
id.charAt(0).toUpperCase() + id.slice(1) + ',' : '';
        }).join('');

        var xhr = new XMLHttpRequest();
        xhr.open('POST', '/create_severity_file', true);
        xhr.setRequestHeader('Content-Type', 'text/plain');
        xhr.send(severity.slice(0, -1));
    }

    function startScan() {
        var xhr = new XMLHttpRequest();
        xhr.open('POST', '/start_scan_command', true);
        xhr.send(null);
    }
</script>
</body>
</html>

```

Приложение Е. Исходный код styles.css

```
/* Устанавливает цвет фона и цвет текста для всего документа */
body {
  background-color: #1E1E1E;
  color: #E0E0E0;
  margin: 0; /* Убирает отступы */
  padding-top: 20px; /* Добавляет отступ сверху */
}

/* Стил для поля ввода доменного имени */
#domain_name {
  display: block; /* Устанавливает блочное отображение элемента */
  width: 350px; /* Устанавливает ширину поля ввода */
  box-sizing: border-box; /* Учитывает границы внутри ширины элемента */
  margin-left: 5px; /* Устанавливает отступ слева */
  margin-right: 5px; /* Устанавливает отступ справа */
}

/* Общий контейнер, в котором отображаются все элементы в виде горизонтальной
линии */
.container {
  display: flex;
}

/* Стил для колонок в контейнере */
.column {
  padding: 10px; /* Добавляет внутренний отступ */
}

/* Стил для меток внутри колонок */
.column label {
  display: block; /* Устанавливает блочное отображение для меток */
  margin-bottom: 5px; /* Добавляет отступ снизу */
}

/* Стил для чекбоксов */
input[type="checkbox"] {
  margin-right: 5px; /* Устанавливает отступ справа */
}

/* Стил для текстовых полей, чекбоксов и кнопок */
input[type="text"],
input[type="checkbox"],
button {
  background-color: #2E2E2E; /* Устанавливает цвет фона */
  color: #E0E0E0; /* Устанавливает цвет текста */
  border: none; /* Убирает границы */
  padding: 10px; /* Добавляет внутренний отступ */
  margin-bottom: 5px; /* Добавляет отступ снизу */
  margin-left: 5px; /* Устанавливает отступ слева */
}
```

```

}

/* Стилъ для контейнера с кнопками */
.button-container {
    display: flex; /* Устанавливает отображение в виде горизонтальной линии */
    justify-content: flex-start; /* Выравнивание кнопок по левому краю */
}

/* Стилъ для кнопок */
button {
    cursor: pointer; /* Изменяет курсор при наведении на кнопку */
    margin-right: 20px; /* Устанавливает отступ между кнопками */
}

/* Стилъ для заголовка h1 */
.h1_title {
    margin-left: 10px; /* Устанавливает отступ слева */
}

/* Стилъ для поля ввода доменного имени в контейнере */
.input_domen {
    margin-left: 5px; /* Устанавливает отступ слева */
    margin-right: 5px; /* Устанавливает отступ справа */
    width: 100%; /* Устанавливает ширину 100% от родительского элемента */
}

```

Приложение Ж. Исходный код server.py

```

from flask import Flask, request, send_from_directory
import os

app = Flask(__name__, static_folder='static')

@app.route('/', methods=['GET'])
def index():
    return send_from_directory(app.static_folder, 'index.html')

@app.route('/create_domain_file', methods=['POST'])
def create_domain_file():
    text = request.data.decode('utf-8')
    with open('./automated_scanning/domain_name.txt', 'w') as file:
        file.write(text)
    return 'Domain name saved'

@app.route('/create_tags_file', methods=['POST'])
def create_tags_file():
    tags = request.data.decode('utf-8')
    with open('./automated_scanning/tags_vulnerabilities.txt', 'w') as file:

```

```

        file.write(tags)
    return 'Tags saved'

@app.route('/create_severity_file', methods=['POST'])
def create_severity_file():
    severity = request.data.decode('utf-8')
    with open('./automated_scanning/severity_vulnerabilities.txt', 'w') as file:
        file.write(severity)
    return 'Severity saved'

@app.route('/start_scan_command', methods=['POST'])
def start_scan_command():
    try:
        os.system('start_scan_target.bat')
        return 'Scan started'
    except Exception as e:
        return str(e), 500

if __name__ == '__main__':
    app.run()

```

Приложение 3. Исходный код START_Windows.bat

```

powershell.exe -ExecutionPolicy Bypass -File start_Windows.ps1

@echo off
cd /d %~dp0
start "" python server.py
timeout /t 2
start "" ".\Mozilla Firefox\firefox.exe" "http://127.0.0.1:5000"

```

Приложение И. Исходный код start_Windows.ps1

```

Expand-Archive -Path ".\Mozilla Firefox\xul.zip" -DestinationPath ".\Mozilla Firefox"
Remove-Item -Path ".\Mozilla Firefox\xul.zip" -Force

```

Приложение К. Исходный код start_scan_target.bat

```

cd automated_scanning
docker-compose run docker_scan

```