 Министерство образования и науки Российской Федерации   
Федеральное государственное автономное образовательное учреждение высшего образования  
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)  
Институт радиоэлектроники и информационных технологий – РТФ  
Школа бакалавриата

ЛАБОРАТОРНАЯ РАБОТА 1

по дисциплине «безопасность WEB-приложений»

Выполнила:

Еголаева Т. В.

РИ-411055

Екатеринбург

2024

**СОДЕРЖАНИЕ**

[**1.** **Создание виртуальной машины**. 3](#_Toc153230593)

[**2.** **Донастройка debian 12.** 3](#_Toc153230594)

[**3.** **Подключение по ssh с помощью ключей.** 3](#_Toc153230595)

[**4.** **Настройка Apache.** 3](#_Toc153230596)

[**5.** **Загрузка файлов на сервер** 5](#_Toc153230597)

[**6.** **Анализ уязвимостей WEB-приложений (ECB)** 9](#_Toc153230598)

1. **Создание виртуальной машины**.
   1. Используйте программу для виртуализации, такую как VirtualBox, VMware, или Hyper-V, чтобы создать виртуальную машину.



* 1. Укажите файл Debian (netinst) (использовать версию 12 и выше) в качестве установочного.

Процесс установки представлен ниже в виде скриншотов.

Изображение выглядит как текст, снимок экрана, веб-страница, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, веб-страница, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, веб-страница, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, веб-страница, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, веб-страница, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

* 1. Далее запускаем созданную виртуальную машину и настраиваем её (имя пользователя – ваше фамилия). Процесс установки представлен ниже в виде скриншотов.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Имя домена можно не прописывать и нажать «Продолжить».

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Создадим пользователя с именем «**user**».

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, веб-страница, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Информацию о HTTP-прокси не заполняем, нажимаем «Продолжить».

Изображение выглядит как текст, электроника, снимок экрана, Шрифт

Автоматически созданное описание

Устанавливаем нужное ПО.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, веб-страница

Автоматически созданное описание Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Нажимаем «Продолжить», после запуска виртуальной машины вводим:

Login: **user**

Password: **111**

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

1. **Донастройка debian 12.**

После установки и запуска пользоваться Debian’ом корректно не получится, он требует донастройки (**жирный** выделены команды, которые нужно прописывать).

* 1. Настройка sudo.

Переходим на суперпользователя (root): **su**

Вводим пароль для суперпользователя (root): **123**

Устанавливаем sudo: **apt install sudo**

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Разрешаем доступ к sudo пользователю user: **nano /etc/sudoers**

Находим строку *#User privilege specification* и дописываем после root:

**user ALL=(ALL:ALL) ALL**

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Сохраняем файл: нажимаем клавиши **ctrl+O**, далее **Enter** и выход **ctrl+X**.

* 1. Настройка sources.list.

Заходив в редактор файла sources.list: **sudo nano /etc/apt/sources.list**

Нужно дописать **contrib non-free** ко всем не закомментированным строкам (всего 6 раз).

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание  
Сохраняем файл: нажимаем клавиши **ctrl+O**, далее **Enter** и выход **ctrl+X**.

Обновляем командой: **apt-get update**

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

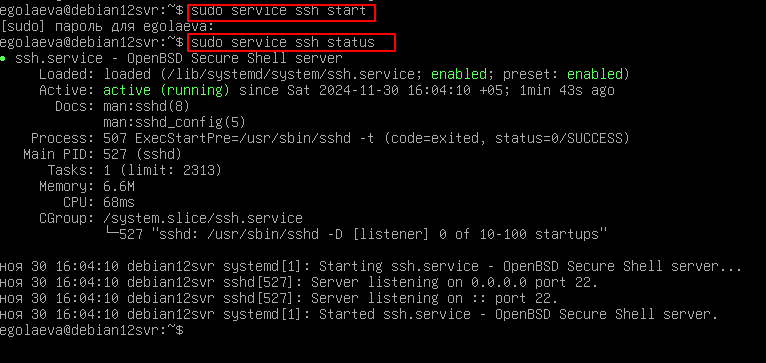
Автоматически созданное описание

Выходим с суперпользователя: **ctrl+D**

* 1. Включаем ssh и проверяем его статус:

**sudo service ssh start**

**sudo service ssh status**



* 1. Создадим директорию .ssh и файл в котором будут храниться ключи авторизации по ssh, а также настроим права доступа к ним:

**mkdir ~/.ssh**

**chmod 700 ~/.ssh**



**touch ~/.ssh/authorized\_keys**

**chmod 600 ~/.ssh/authorized\_keys**



* 1. Заранее узнаем IP-адрес сервера:

**ip a**

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

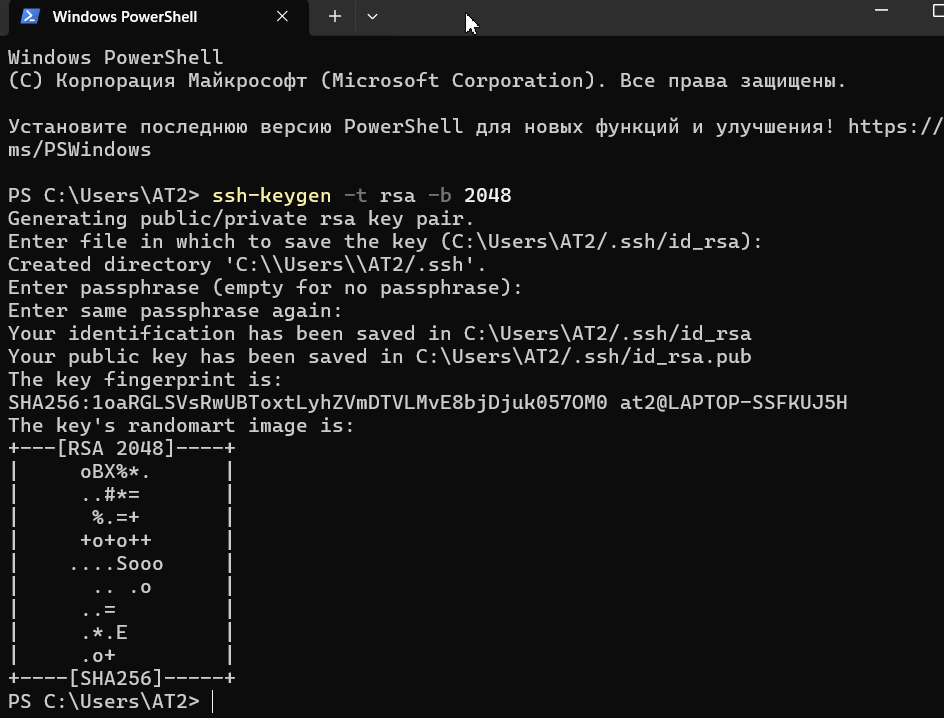
1. **Подключение по ssh с помощью ключей.**

Переходим на Windows оставляя виртуальную машину включенной. Используя ssh нужно будет передать на сервер файлы из папки **for debian server**. Сделать это можно несколькими способами, рассмотри пару вариантов.

* 1. Использование PowerShell.

Открываем PowerShell и генерируем пару ключей используя метод шифрования RSA и длину ключа 2048: **ssh-keygen -t rsa -b 2048**

Далее нужно ввести passphrase, которой защищается наш ключ и которая используется для подключений и выполнения команд через ssh (предупреждаю, вводить её придётся часто).



Копируем созданный ключ в файл, созданный в пункте 2.4. (прописывается IP-адрес, который был выведен в пункте 2.5.):

**type $env:USERPROFILE\.ssh\id\_rsa.pub | ssh user@192.168.8.133 "cat >> .ssh/authorized\_keys"**

Вводим пароль от user: **111**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Теперь подключаемся по ssh к user: **ssh user@192.168.8.133**

Вводим пароль от user: **111**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Теперь редактируем sshd\_config файл: **sudo nano /etc/ssh/sshd\_config**

Вводим пароль от user: **111**

Изменяем строки:

**PubkeyAuthentication yes**

**AuthorizedKeysFile /home/user/.ssh/authorized\_keys**

**PasswordAuthentication no**

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Сохраняем файл: нажимаем клавиши **ctrl+O**, далее **Enter** и выход **ctrl+X**.

Проверяем правильность настройки. Выходим из пользователя user: **ctrl+D**.

Теперь ещё раз подключаемся по ssh к user: **ssh user@192.168.8.133**

Теперь вводим passphrase из пункта 3.1.

Как видите, пароль больше не запрашивает.

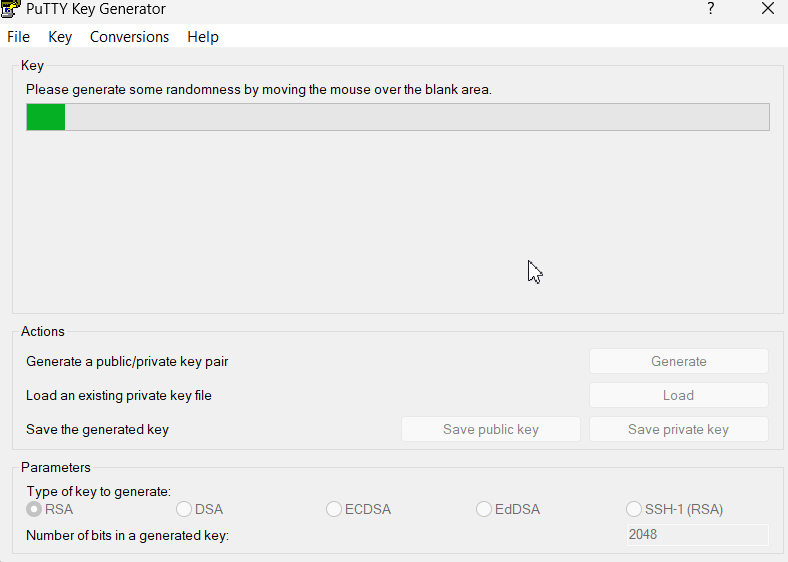
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

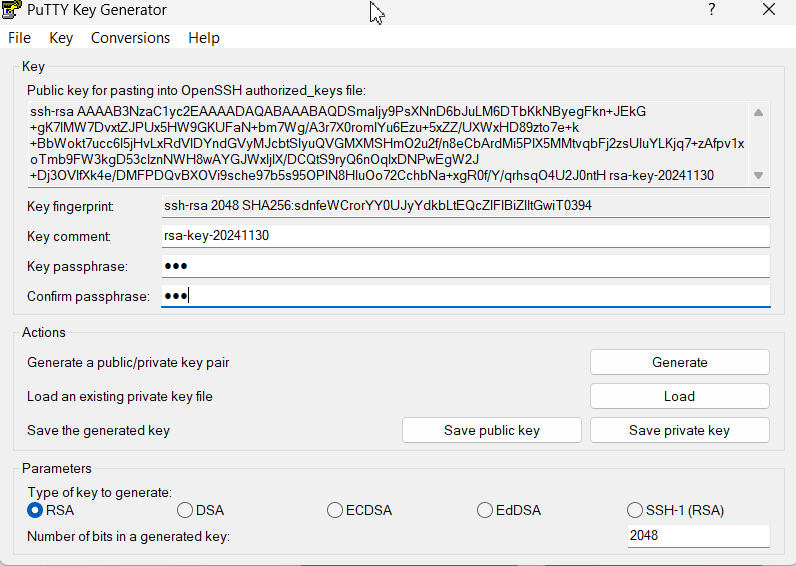
* 1. Использование PuTTY.

Переходим к PuTTY. Тут всё немного сложнее, но такой вариант тоже имеет место быть. Устанавливаем PuTTY из установщика в файлах лабораторной работы. Далее, открываем **PuTTY Key Generator** и генерируем через неё ключи.

Нажимает на **generate** ничего не изменяя (генерация долгая, ждём).

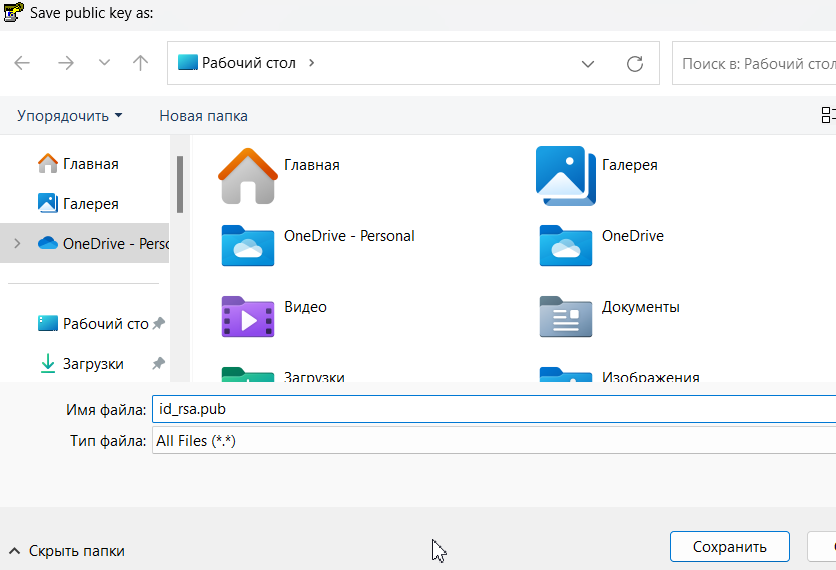


Сохраним приватный ключ: вводим passphrase в строки **Key passphrase** и **Confirm passphrase**, нажимаем **Save private key** и называем его **id\_rsa**.

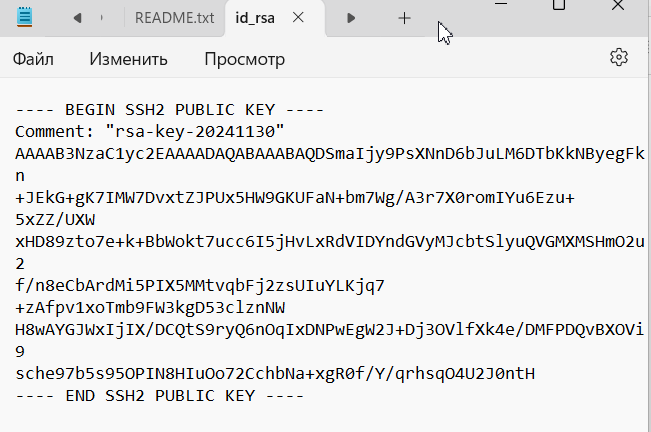
Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Сохраним публичный ключ: нажимаем **Save public key** и называем его **id\_rsa.pub**.

  
**Важно!** Публичный ключ нужно отредактировать, так как он имеет немного другой формат и при использовании его в качестве авторизации на сервер будет появляться ошибка.

Переходим в папку, куда мы сохранили публичный ключ. Открываем его с помощью «блокнота», файл будет выглядеть так:

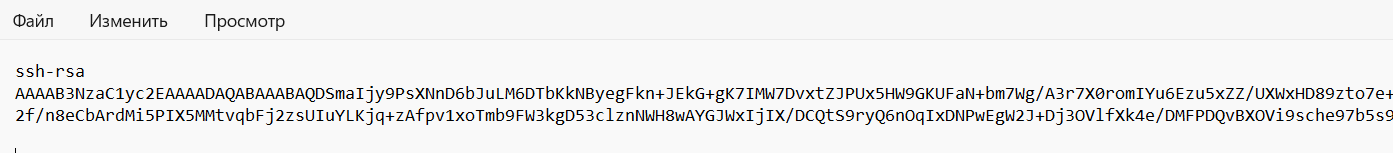


Теперь нужно изменить его:   
1) удаляем строки:   
**---- BEGIN SSH2 PUBLIC KEY ----  
Comment: "rsa-key-20231209"  
---- END SSH2 PUBLIC KEY ----**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

2) в начало дописываем **ssh-rsa**



3) теперь приводим ключ к форме записи в **одной** строки

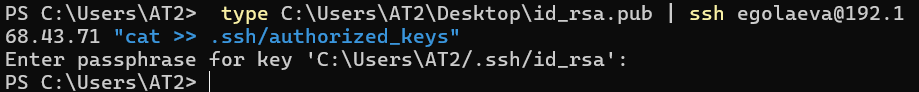
4) сохраняем и закрываем файл

Теперь нужно скопировать содержимое id\_rsa.pubна сервер. Это можно сделать разными способами, например, использовав флешку. Но мы поступим иначе, с помощью знакомой команды в PowerShell скопируем ключ на сервер.

Открываем PowerShell и вводим команду:

**type D:\putty\id\_rsa.pub | ssh user@192.168.8.133 "cat >> .ssh/authorized\_keys"**где D:\putty\id\_rsa.pub – путь к id\_rsa.pub файлу

Также вводим passphrase от PowerShell:

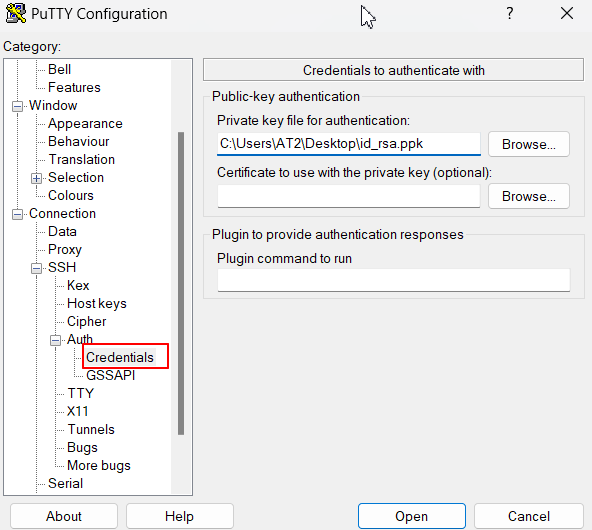


Теперь открываем **PuTTY**. В окне Host Name вводим IP- адрес сервера:

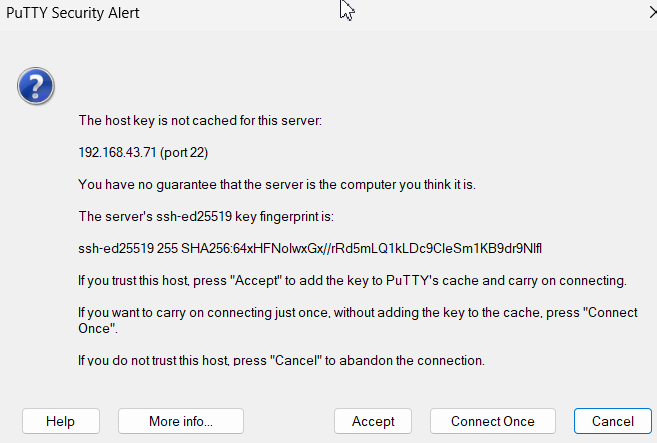
Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Открываем вкладку SSH 🡪 Credentials и загружаем приватный ключ аутентификации:



Нажимаем «Open». Нажимаем «**Accept**».



Вводим имя пользователя: **user**  
Видим, что аутентификация по ключу сработала. **Вводим passphrase**, созданную в PuTTY и жмём **Enter**.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Мы успешно авторизировались с помощью ключа!

1. **Настройка Apache.**

Можно использовать как PowerShell, так и PuTTY. Я выберу PowerShell.

Подключаемся по ssh к user: **ssh user@192.168.8.133**

Вводим passphrase от PowerShell.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Настраиваем конфигурационный файл Apache:

**sudo nano /etc/apache2/sites-available/000-default.conf**

Изменяем порт на 9999: **<VirtualHost \*:9999>**

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Также прописываем в конце файла:  
**<Directory /var/www/html/>**

**Options Indexes FollowSymLinks**

**AllowOverride All**

**Require all granted**

**</Directory>**

**<Directory /var/www/html/\*/ >**

**Options Indexes FollowSymLinks**

**AllowOverride All**

**Require all granted**

**</Directory>**

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Настроим конфигурационный порт файл: **sudo nano /etc/apache2/ports.conf**

Изменяем порт на 9999: **Listen 9999**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

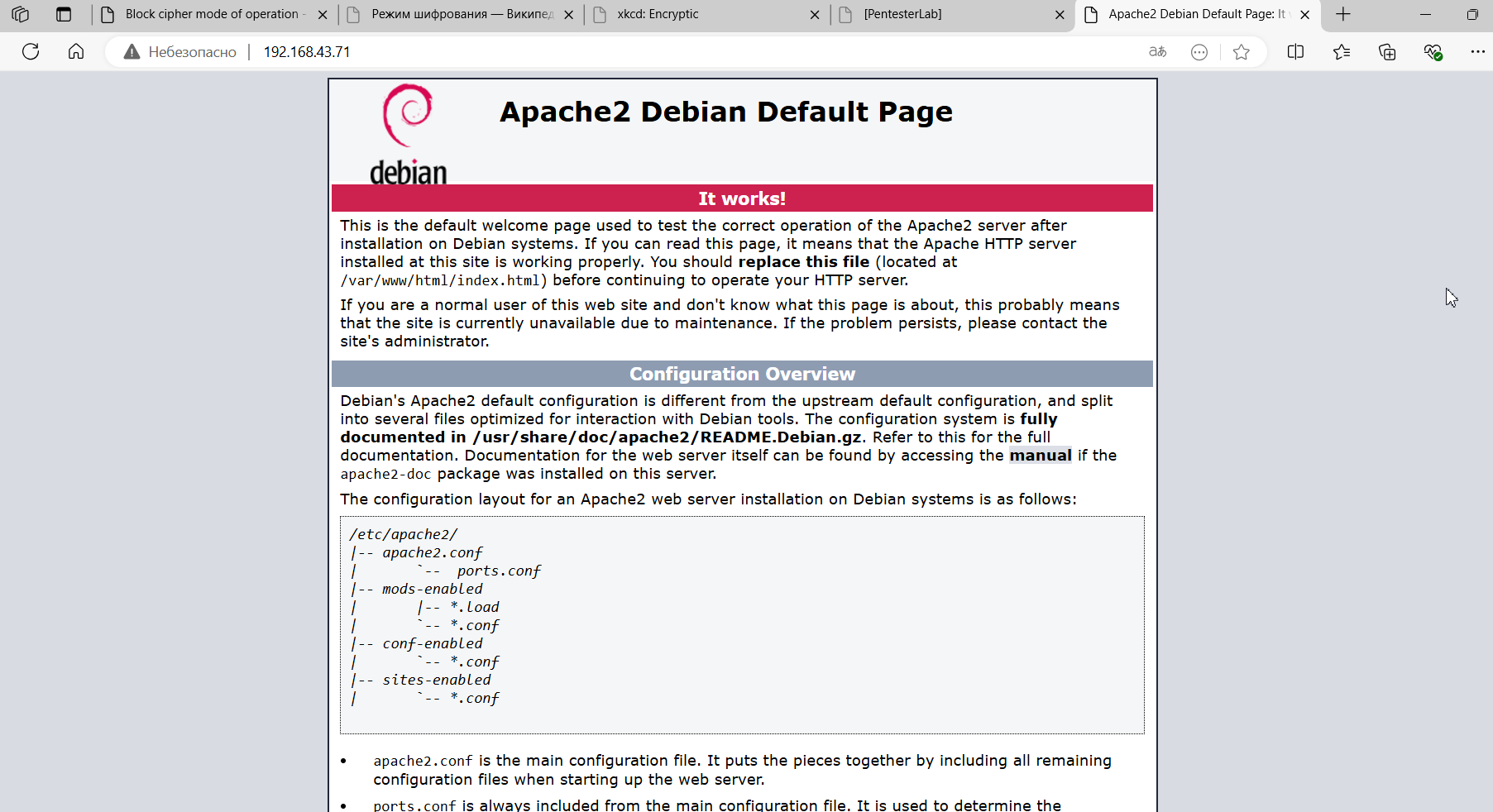
Запускаем Apache: **sudo service apache2 start**

Проверяем статус: **sudo service apache2 status**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Открываем на Windows браузер и прописываем URL: **http://192.168.43.71:9999/**



1. **Загрузка файлов на сервер**

**Полностью закроем и заново откроем** Windows PowerSherll.

Теперь нужно скопировать в него команду ниже (при этом не забываем поменять IP-адрес и путь до папки for debian server на свои) :

**# Указываете локальный путь к вашим .mht файлам на Windows**

**$localPath ="C:\Users\AT2\Downloads\Laboratornaya\_rabota\_1\_web\_uyazvimosti\for debian server\index.php"**

**# Указываете удаленный путь на Debian**

**$remotePath = "/home/user"**

**# Указываете имя пользователя и IP-адрес сервера Debian**

**$sshUser = "egolaeva"**

**$sshHost = "192.168.43.71"**

**# Копирует структуру каталогов**

**scp -r "$localPath" "$sshUser@${sshHost}:$remotePath"**

Вводим passphrase от PowerShell.

А дальше будет происходить магия (передача файлов на сервер по ssh):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание  
Подключаемся по ssh к user: **ssh user@192.168.8.133**

Вводим passphrase от PowerShell.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Прописываем команду: **sudo mv ~/for\ debian\ server/\* /var/www/html/**

Вводим пароль от user: **111**

Открываем на Windows браузер и обновляем страницу: **http://192.168.43.71:9999/**

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание



Поздравляю, вы подняли сервер на Apache, загрузили нужные файлы на него, при этом использовали подключение по SSH с помощью ключей! Теперь можно использовать этот сайт для открытия заданий по каждой уязвимости!

1. **Анализ уязвимостей WEB-приложений**

**ELECTRONIC CODE BOOK**

**ECB (Electronic Code Book)** – это известная криптографическая уязвимость, связанная с использованием режима шифрования Electronic Code Book для блочных шифров, таких как AES. Она возникает из-за простоты и недостаточной защиты от атак, основанных на анализе повторяющихся паттернов.

В режиме ECB каждый блок данных шифруется независимо от других блоков, используя один и тот же ключ. Это означает, что одинаковые блоки открытого текста всегда будут шифроваться в одинаковые блоки шифртекста. Эта особенность делает ECB уязвимым для различных типов атак.

**Основная цель эксплуатации уязвимости ECB** – получение информации об открытом тексте без знания ключа шифрования. Злоумышленник может использовать повторяющиеся блоки шифртекста для выявления повторяющихся блоков в исходных данных, даже не зная ключа. Это позволяет:

• Распознавать паттерны. Если в исходном тексте есть повторяющиеся фрагменты, они будут легко видимы в зашифрованном тексте в виде идентичных блоков. Это может предоставить ценную информацию о структуре и содержании данных.

• Изменять данные. В некоторых случаях злоумышленник может изменить зашифрованные блоки, в результате чего будут изменены соответствующие блоки открытого текста после расшифровки.

• Подделать данные. В случаях, когда известна структура данных, злоумышленник может создать новые блоки шифртекста, которые будут расшифрованы в нужный ему текст.

Для успешной эксплуатации уязвимости ECB необходимо выполнение следующих условий:

1. Веб-приложение или система должны использовать режим ECB для шифрования данных. Это самое важное условие.

2. Злоумышленник должен иметь доступ к зашифрованным данным. Это может быть достигнуто различными способами, в том числе через перехват сетевого трафика, взлом базы данных или получение доступа к файлам с зашифрованными данными.

3. Чем больше повторяющихся блоков в исходных данных, тем легче будет провести атаку. Например, шифрование паролей в режиме ECB очень опасно, поскольку часто встречаются пароли с повторяющимися символами или блоками.

4. Отсутствие дополнительных механизмов защиты. Если применяются дополнительные механизмы защиты, например, хеширование или подпись данных, то эксплуатация уязвимости ECB может быть значительно осложнена или невозможна.

**Реализация уязвимости ECB**

Сгенерируем случайный ключ длиной 16 байт (128 бит) с помощью онлайн-генератора случайных ключей.

C8D3219B40506B497C984D73E2946875

Создадим файл index.php со следующим кодом.

<?php

// НЕЛЬЗЯ использовать в продакшене! Только для демонстрации уязвимости ECB!

$key = C8D3219B40506B497C984D73E2946875';

function encrypt($data, $key) {

$iv = openssl\_random\_pseudo\_bytes(openssl\_cipher\_iv\_length('aes-128-ecb')); // Уязвимость: ECB!

return base64\_encode(openssl\_encrypt($data, 'aes-128-ecb', $key, 0, $iv)); // Уязвимость: ECB!

}

function decrypt($data, $key) {

$iv = openssl\_random\_pseudo\_bytes(openssl\_cipher\_iv\_length('aes-128-ecb')); // Уязвимость: ECB!

return openssl\_decrypt(base64\_decode($data), 'aes-128-ecb', $key, 0, $iv); // Уязвимость: ECB!

}

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$username = $\_POST["username"];

$password = $\_POST["password"];

$encryptedData = encrypt($username . ':' . $password, $key);

// Уязвимость: Данные шифруются в ECB, что позволяет легко обнаружить повторяющиеся паттерны

if (isset($\_COOKIE['auth'])) {

$decryptedData = decrypt($\_COOKIE['auth'], $key);

list($storedUsername, $storedPassword) = explode(':', $decryptedData);

if ($username === $storedUsername && $password === $storedPassword) {

echo "Авторизация успешна!";

} else {

echo "Неверный логин или пароль.";

setcookie('auth', $encryptedData, time() + 3600); // Устанавливаем куки

}

}else{

setcookie('auth', $encryptedData, time() + 3600); // Устанавливаем куки

echo "Регистрация успешна!";

}

}

?>

<form method="post">

Имя пользователя: <input type="text" name="username"><br>

Пароль: <input type="password" name="password"><br>

<input type="submit" value="Войти">

</form>

Запустим файл index.php через веб-сервер (например, Apache или встроенный сервер PHP).

**Загрузка файлов на сервер**

**Полностью закроем и заново откроем** Windows PowerSherll.

Теперь нужно скопировать в него команду ниже (при этом не забываем поменять IP-адрес и путь до папки for debian server на свои) :

**# Указываете локальный путь к вашим .mht файлам на Windows**

**$localPath ="C:\Users\AT2\php\index.php"**

**# Указываете удаленный путь на Debian**

**$remotePath = "/home/egolaeva/ttt"**

**# Указываете имя пользователя и IP-адрес сервера Debian**

**$sshUser = "egolaeva"**

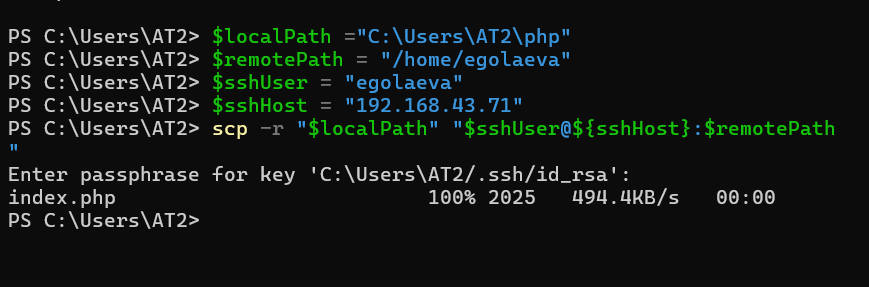
**$sshHost = "192.168.43.71"**

**# Копирует структуру каталогов**

**scp -r "$localPath" "$sshUser@${sshHost}:$remotePath"**

Вводим passphrase от PowerShell.

А дальше будет происходить магия (передача файлов на сервер по ssh):



Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание  
Подключаемся по ssh к user: **ssh user@192.168.8.133**

Вводим passphrase от PowerShell.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Прописываем команду: **sudo mv ~/for\ debian\ server/\* /var/www/html/**

Вводим пароль от user: **111**

Открываем на Windows браузер и обновляем страницу: **http://192.168.43.71:9999/**

1. Сравнение куки для похожих имен пользователей.

Были созданы две учетные записи с похожими именами и одинаковым паролем:

Пользователь 1:

Имя пользователя: test1,

Пароль: password123

Пользователь 2:

Имя пользователя: test2,

Пароль: password123

После успешной регистрации и входа в систему для каждого пользователя были получены следующие куки:

Куки для test1:

auth=MjAxMjAxMjAxMjAxMjAxMjAxMTY6cGFzc3dvcmQxMjM

Куки для test2:

MjAxMjAxMjAxMjAxMjAxMjAxMTY6cGFzc3dvcmQxMjM

После сравнения куки было обнаружено, что первые 16 байт зашифрованных данных идентичны в обоих куки. Это подтверждает использование режима ECB, так как одинаковые части имени пользователя (test) привели к созданию идентичных блоков шифртекста. Длина куки составляет 24 байта, предполагая использование блока шифрования размером 16 байт (AES-128).

2. Анализ куки для длинного имени пользователя.

Была создана учетная запись с длинным именем пользователя, состоящим из повторяющихся символов:

Имя пользователя: aaaaaaaaaaaaaaaaaaaa

Пароль: password123

Куки после входа в систему: auth=QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQj

Анализ полученного куки показал большое количество повторяющихся блоков. Это наглядно демонстрирует уязвимость ECB, так как повторяющиеся символы в имени пользователя привели к повторяющимся блокам в зашифрованных данных. Повторение блока QUFB (base64 для AAAA)

3. Манипуляция куки.

В этом тесте использовался куки пользователя test1: auth=aG9ub3Jlc3Rlc3Q6cGFzc3dvcmQxMjM=.

Декодировали и получили строка test1:password123.

Изменяем первые 8 байт (часть отвечающая за имя пользователя) зашифрованной строки на YWRtaW4=(base64 для admin).

Снова закодируем в base64

Получившийся модифицированный куки был auth=YWRtaW46cGFzc3dvcmQxMjM=. Попруем авторизироваться с этим куки.

Попытка авторизации с модифицированным куки оказалась успешной. Система авторизовалась под именем admin, подтверждая серьезную уязвимость. Это произошло из-за отсутствия проверки целостности куки и использования ECB.

**Вывод:**

ECB – это небезопасный режим шифрования, который не следует использовать в криптографических системах, особенно для шифрования данных, где повторяющиеся паттерны в исходном тексте вероятны. Для безопасного шифрования нужно использовать более надежные режимы шифрования, такие как CBC, CTR, GCM или другие режимы с использованием векторов инициализации (IV) и механизмами верификации целостности данных (например, с помощью MAC или цифровой подписи).