Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Институт электронной техники и приборостроения

Кафедра Информационная безопасность автоматизированных систем

Специальность 10.05.03 Информационная безопасность автоматизированных систем

**КУРСОВАЯ РАБОТА**

по дисциплине «Безопасность сетей ЭВМ»

**Тема работы**

«Анализ уязвимостей в пакете программ OpenSSL»

|  |  |
| --- | --- |
|  | Выполнила: студент 4 курса  учебной группы с-ИБС42  очной формы обучения  Солодилов В.В. Руководитель работы:  доцент каф. ИБС Губенков А.А.  Комиссия по защите:  доцент каф. ИБС Губенков А.А.  доцент каф. ИБС Хороводова Н.Ю. |

Курсовая работа защищена на оценку \_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_2022 г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись члена комиссии)

«\_\_\_»\_\_\_\_\_\_2022 г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись члена комиссии)

Саратов 2022

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Кафедра Информационная безопасность автоматизированных систем

**Задание**

**на курсовую работу**

студенту 4 курса учебной группы с-ИБС42

Института электронной техники и приборостроения

Солодилову Владимиру Владимировичу

**Анализ уязвимостей в пакете программ OpenSSL**

Провести анализ уязвимостей в OpenSSL, воспользовавшись информацией из открытых источников (bdu.fstec.ru, web.nvd.nist.gov, cwe.mitre.org), сделать обзор уязвимостей. В обзор включать только уязвимости новее 2005 года.

Реализовать угрозу безопасности на примере конкретной уязвимости с помощью эксплойтов. Поиск эксплойтов осуществлять на сайтах metasploit.com, vulners.com, exploit-db.com. Оформить отчет по выполненной работе, процесс эксплуатации уязвимости продемонстрировать с помощью снимков экрана. В заключении дать рекомендации по защите систем от угроз безопасности информации.

Сроки выполнения 14.09.2022 г. - 23.12.2022 г.

Дата защиты 26.12.2022 г.

**Руководитель работы Губенков А.А.**

**Задание принял к исполнению Солодилов В.В.**

Содержание

**Введение**

В современном Интернете большинство соединений с сайтами устанавливается с использованием различных средств шифрования, включающих в себя протоколы SSL и TLS. Данные протоколы входят в OpenSSL – пакет программ, используемый для работы с различными криптографическими функциями библиотеки OpenSSL в консоли.

Протокол SSL был выпущен в 1995 году и был обновлен до TLS в 1999 году. С тех пор происходит активное развитие данного протокола. С появлением новых версий данного протокола большинство предыдущих его версий становятся устаревшими ввиду устаревания методов шифрования, в них используемых, а также появления различных уязвимостей, в том числе критических, которые ставят под угрозу возможное использование данного протокола. В связи с этим важно следить за выходом обновлений для протоколов SSL/TLS и использовать его актуальную версию.

В рамках курсовой работы проводится обзор протокола SSL/TLS, его особенностей, преимуществ и недостатков, а также пакета программ OpenSSL, использующий данный протокол. В дальнейшем рассматриваются выявленные уязвимости для OpenSSL, а также реализация угрозы с использованием уязвимости Heartbleed для OpenSSL 1.0.1 на Ubuntu Linux 12.04 LTS.

1. Основные термины и определения
   1. Понятие и состав OpenSSL

OpenSSL – это библиотека программного обеспечения для приложений, которые защищают данные при передаче их по открытым каналам связи от несанкционированного изменения, подделки или утечки. Он широко используется интернет-серверами, включая большинство HTTPS-сайтов.

OpenSSL содержит реализацию протоколов SSL/TLS с открытым исходным кодом. Основная библиотека, написанная на языке программирования C, реализует основные криптографические функции и предоставляет различные служебные функции. Среди криптографических алгоритмов шифрования, используемых в данной библиотеке, можно выделить DES, AES, Camellia, Blowfish, RC2, RC4.

Для версии OpenSSL 3.0 после компиляции из исходного кода возможно использование российских криптографических алгоритмов шифрования «Магма» и «Кузнечик», определенных ГОСТом Р 34.10-2015.

* 1. Особенности протокола SSL

Протокол SSL является протоколом прикладного уровня, выпущенным компанией Netscape в 1995 году для защиты информации передаваемых по Интернету данных кредитных карт. В настоящий момент данный протокол располагается поверх протокола HTTP, что обеспечивает соединение клиента с сервером посредством уже более безопасного протокола HTTPS.

Основополагающими средствами обеспечения безопасности передачи информации в HTTPS с использованием протокола SSL являются:

* Взаимная аутентификация (известна как mutual challenge) – заключается в аутентификации идентификационных данных другой стороны. При передачи данных между двумя собеседниками первоначально каждый из них должен аутентифицировать идентификационные данные друг друга, после чего становится возможным непосредственная передача самих сообщений.
* Конфиденциальность – перед отправкой сообщения происходит его шифрование для дальнейшей передачи по каналу связи. После чего получатель расшифровывает исходное сообщение. При перехвате зашифрованного сообщения злоумышленником понадобится большая вычислительная мощность для его дешифрования, в то время как исходная информация может устареть.
* Целостность – гарантия совпадения полученного сообщения с отправленным. Реализуется на основе использования хэш-функций, также входящих в состав OpenSSL.
  1. Основные принципы работы SSL

Принцип работы протокола SSL заключается в реализации двух фаз: **фазы рукопожатия** и **фазы передачи данных**. При данных действиях активно используются асимметричные и симметричные алгоритмы шифрования, первых из которых предназначен для аутентификации сторон, второй – для сохранения конфиденциальности передаваемой по каналу связи информации.

При установке соединения между клиентом и сервером происходит фаза рукопожатия, при которой выполняются следующие действия:

* Клиент генерирует hello-сообщение и отправляет его серверу.
* В ответном сообщении сервер отправляет hello-сообщение клиенту, а также свой сертификат, который содержит публичный ключ.
* Клиент публичным ключом производит расшифровку сертификата сервера и проверяет полученные данные с отпечатком (digest) сертификата, который выдаётся центром сертификации. Если данные не совпадают, значит сертификат сервера был подделан, вследствие чего дальнейшее соединение с сервером прекращается.
* После проверки подлинности сервера, сервер производит шифрование открытым ключом для определения секретного ключа, используемого для обмена информацией.
* Производится взаимная отправка finished-сообщений с целью подтверждения согласия сторон на начало передачи данных.

Во время фазы передачи данных каждая сторона разбивает исходящие сообщения на фрагменты и прикрепляет к ним коды авторизации сообщений **MAC (message authentication code)**. Для получения полноценного SSL пакета каждая из сторон объединяет данные фрагмента, код авторизации сообщения, заголовки сообщения и шифрует их с использованием секретного ключа. При получении пакета, каждая из сторон расшифровывает его и сверяет полученный код авторизации сообщения со своим. Если полученные в итоге проверки коды не совпадают, пакет подделан и дальнейшая обработка запросов прекращается, о чём сигнализируется пользователю.

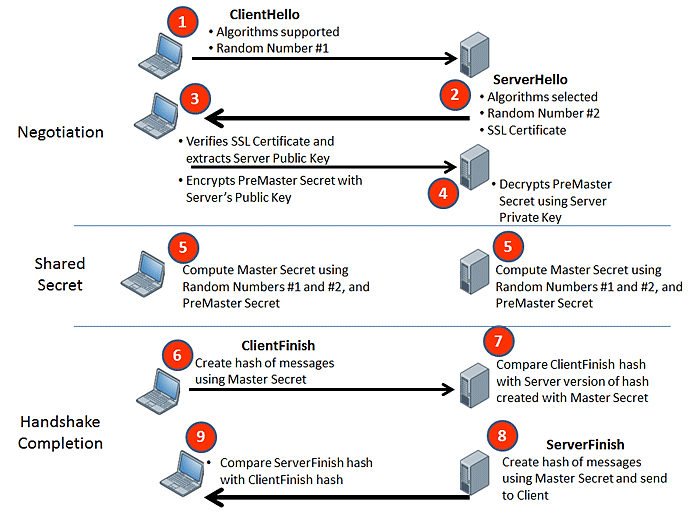


Рисунок 1 – Принцип работы SSL протокола

* 1. Виды цифровых сертификатов и способы их получения

Существуют следующие виды сертификатов, которые могут быть выданы соответствующему серверу:

* Сертификат, выданный Центром сертификации (CA)
* Самоподписанный сертификат – сертификат, который создан самим пользователем. В данном случае издатель сертификата и его владелец совпадают.
* «Пустой» сертификат – сертификат, содержащий фиктивную информацию, используемую в качестве временной для настройки SSL и проверки его функциональности в данной среде.

Второй и третий вид сертификатов используются, в основном, для настройки и тестирования SSL, а также проверки его функциональности в данной среде. Использование самоподписанного протокола не является доверенным действием, а все сайты, которые его используют, например, для Google Chrome, автоматически считаются небезопасными, о чём сообщается пользователю при попытке входа на сайт.

Для обеспечения безопасности работы сервера с использованием протокола шифрования SSL, владельцу необходимо получить соответствующий сертификат, который проверяется и выдаются **Центром сертификации (CA - Certificate Authorities)**. При подаче заявки на получение сертификата генерируется **CSR (Certificate Signing Request - запрос на получение сертификата)** с парой ключей на сервере владельца.

При получении сертификата сервера владельцу необходимо совершить следующие действия:

1. Сгенерировать CSR и пару ключей локально на сервере. Пара ключей состоит из открытого (**public key**) и закрытого (**private key**) ключей.
2. Отправить CSR в Центр сертификации, который проверит личность, а также принадлежность ей домена, которому выдаётся сертификат.
3. После проверки организация получает копию своего сертификата SSL, включающего бизнес данные, а также открытый ключ. Теперь организация может установить сертификат на своем сервере.
4. Когда центр сертификации выдает сертификат, он связывается с сертификатом «доверенного корня» (trusted root) Центра сертификации. Корневые сертификаты встроены в каждый браузер и связаны с индивидуально выданными сертификатами для установления HTTPS-соединения.
   1. Генерирование CSR в OpenSSL

Для демонстрации функционала OpenSSL сгенерируем CSR для получения сертификата. Для этого будем использовать OpenSUSE Leap 15.4 с предустановленным OpenSSL 1.1.1l.

Запустим OpenSSL, после чего сгенерируем запрос на создание приватного ключа и CSR, необходимых для получения сертификата. В процессе создания CSR-формы заполним её тестовыми данными:

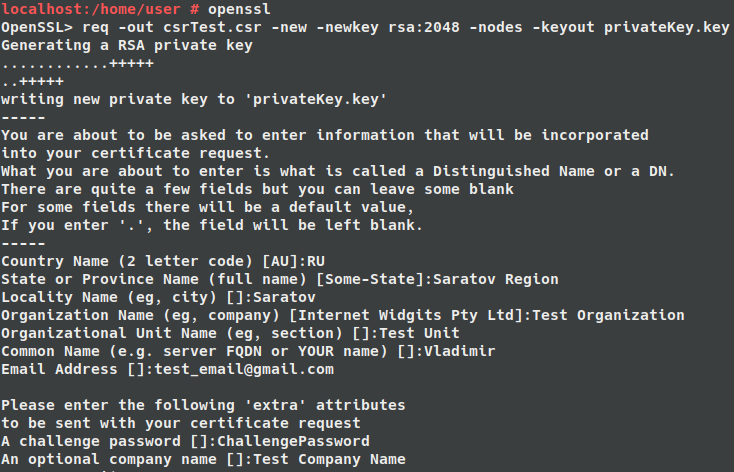


Рисунок 2 – Генерирование CSR запроса

Результатом заполнения являются файлы с приватным ключом и запросом на получение сертификата:

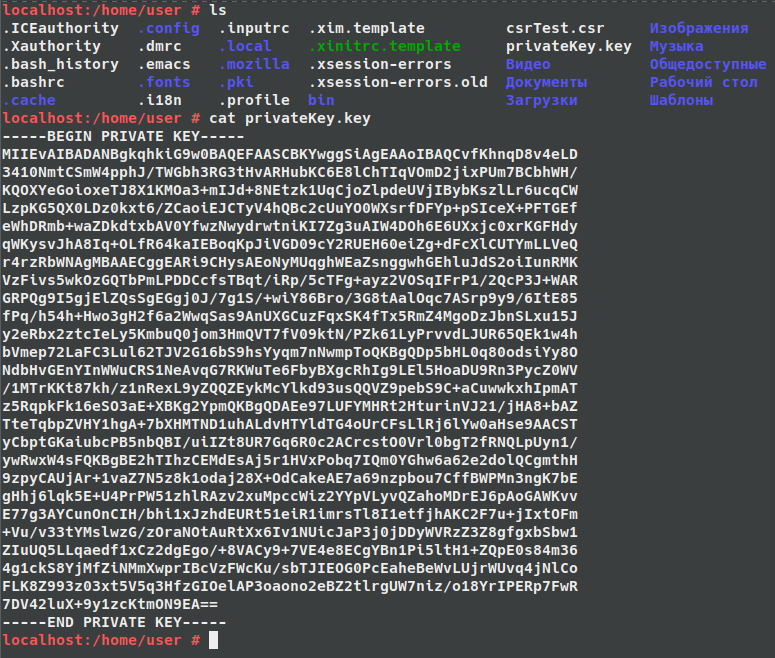


Рисунок 3 – Сгенерированный секретный ключ

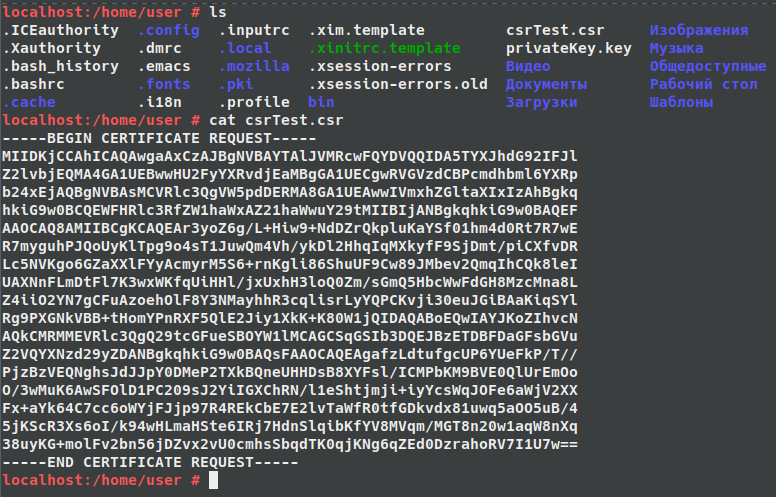


Рисунок 4 – Сгенерированный CSR запрос

В случае с владельцем сервера полученные данные можно передавать в Центр сертификации для получения необходимого сертификата. Для нашего случая можно получить самоподписанный сертификат, используя OpenSSL. Получим данный сертификат, после чего выведем его содержимое в консоль:

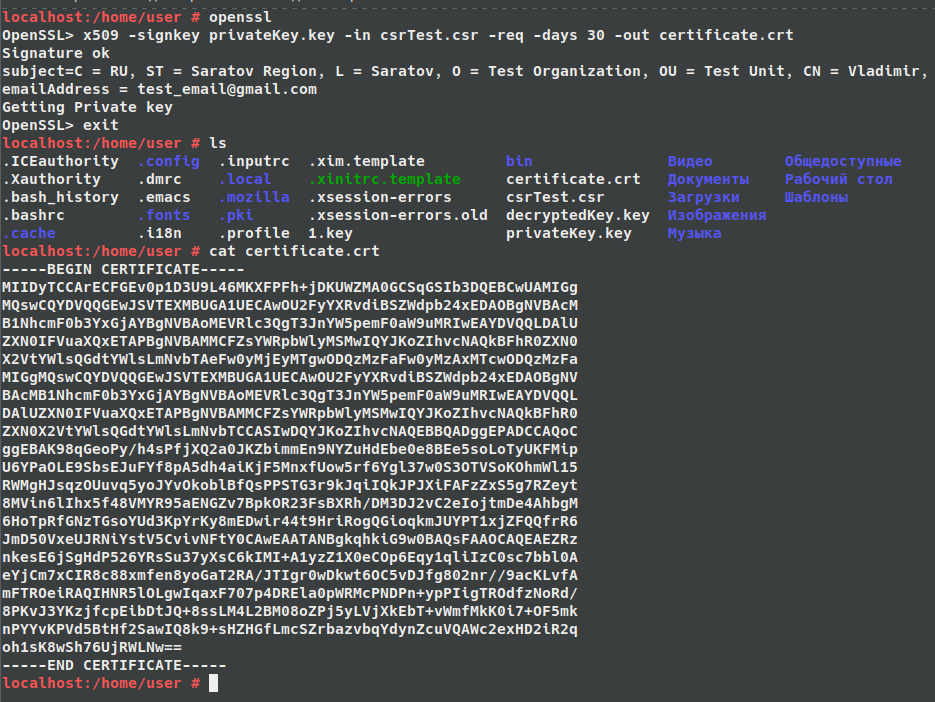


Рисунок 5 – Сгенерированный самоподписанный сертификат

Данный сертификат можно использовать для проведения тестирования работоспособности и функциональности OpenSSL. Однако, для обеспечения непосредственной защиты сервера с использованием SSL/TLS необходимо обратиться в Центр сертификации для получения соответствующего сертификата.

* 1. Преимущества и недостатки SSL

Основными преимуществами использования протокола шифрования SSL являются:

1. Предоставление пользователям конфиденциальности, целостности информации, а также предотвращение утечки персональных данных.
2. Гарантия подлинности и правильности загружаемых данных.
3. Сравнительно низкая стоимость и простота установки на любом хостинге.

Несмотря на приведенные выше преимущества, протокол шифрования SSL также имеет и некоторые недостатки:

1. Смешанные режимы. Если ваша реализация SSL настроена неправильно, и у вас все еще есть файлы, обслуживаемые через HTTP, а не через HTTPS, посетители получат предупреждающее сообщение в своем браузере, сообщающее им, что некоторые данные не защищены. Это может сбивать с толку некоторых посетителей сайта.
2. Кэширование прокси. Еще одна возможная проблема - если на вашем веб-сервере настроена сложная система кеширования прокси. Зашифрованный контент нельзя будет кэшировать. Чтобы обойти это, вам нужно добавить сервер для обработки шифрования, прежде чем оно попадет на сервер кеширования. Это потребует дополнительных затрат, но это хороший способ убедиться, что вы шифруете данные посетителей, когда они заходят на ваш сайт.
3. Обзор уязвимостей криптографической библиотеки OpenSSL

Рассмотрим уязвимости криптографической библиотеки OpenSSL из банка данных угроз (БДУ) ФСТЭК России.

* 1. BDU:2022-02461: Уязвимость библиотеки OpenSSL, связанная с чтением за границами буфера в памяти, позволяющая нарушителю вызвать отказ в обслуживании.

Данная уязвимость связана с чтением за границами буфера в памяти. Эксплуатация уязвимости может позволить нарушителю, действующему удаленно, вызвать отказ в обслуживании.

Распространена данная уязвимость в OpenSSL, используемой в ОС Ubuntu 12.04, Red Hat Enterprise 6 и 7, а также в Debian 9.

Была выявлена 26.09.2016 и на данный момент времени устранена.

По критерию CVSS 2.0 имеет высокий уровень опасности (базовая оценка 7,1); по критерию оценки CVSS 3.0 имеет средний уровень опасности (базовая оценка 5,9).

Для устранения уязвимости необходимо обновить OpenSSL до последней версии.

* 1. BDU:2022-02557: Уязвимость реализации протокола DTLS библиотеки OpenSSL, позволяющая нарушителю вызвать отказ в обслуживании.

Уязвимость связана с ошибками в реализации протокола DTLS и заключается в неправильном управлении ресурсами. Эксплуатация уязвимости может позволить нарушителю, действующему удаленно, вызвать отказ в обслуживании

Данная уязвимость характера для OpenSSL версий 1.0.1n, 1.0.1o, 1.0.2b, 1.0.2c, используемых в ОС Red Hat Enterprise Linux 6 и 7, Ubuntu 12.04.

Была выявлена 16.09.2016 и на данный момент устранена

По критериям CVSS 2.0 и CVSS 3.0 имеет высокий уровень опасности (базовая оценка составляет 7,8 и 7,5 соответственно)

Для ликвидации уязвимости необходимо обновить OpenSSL до последней версии.

* 1. BDU:2022-06608: Уязвимость функционала проверки сертификата X.509 библиотеки OpenSSL, позволяющая нарушителю выполнить произвольный код

Данная уязвимость связана с функционалом проверки сертификата X.509 библиотеки OpenSSL и реализуется в переполнении буфера в стеке. Эксплуатация уязвимости может позволить нарушителю, действующему удалённо, выполнить произвольный код.

Уязвимость распространена в OpenSSL, используемом на ОС семейства Debian GNU/Linux 10 и 11.

Была выявлена 01.11.2022 и на данный момент времени устранена

Эта уязвимость – критическая. Согласно критериям CVSS 2.0 и CVSS 3.0 базовая оценка составляет 10 и 9,8 соответственно.

Для ликвидации уязвимости необходимо обновить OpenSSL до последней версии.

* 1. BDU:2021-01844: Уязвимость реализации протокола TLS библиотеки OpenSSL, позволяющая нарушителю вызвать отказ в обслуживании

Данная уязвимость связана с реализацией протокола TLS библиотеки OpenSSL и заключается в ошибках разыменования указателей. Эксплуатация уязвимости может позволить нарушителю, действующему удалённо, вызвать отказ в обслуживании с помощью специально созданного вредоносного сообщения «ClientHello».

Уязвимость распространена в OpenSSL, используемом на ОС семейства Debian GNU/Linux 8 и 9, а также ОС семейства Astra Linux Common Edition 1.6 «Смоленск» и 2.12 «Орёл».

Была выявлена 11.01.2018, на данный момент устранена

Согласно критериям оценка CVSS 2.0 и CVSS 3.0 имеет средний уровень опасности (базовая оценка составляет 4,3 и 5,9 соответственно)

Для данной уязвимости существует эксплойт, находящийся в открытом доступе.

Для устранения уязвимости необходимо обновить OpenSSL до последней версии.

* 1. BDU:2020-02968: Уязвимость компонента statem/statem.cc библиотеки OpenSSL, позволяющая нарушителю получить несанкционированный доступ к конфиденциальным данным, вызвать отказ в обслуживании или оказать воздействие на целостность данных.

Уязвимость связана с компонентом statem/statem.cc библиотеки OpenSSL и заключается с использованием области памяти после её освобождения. Эксплуатация уязвимости может позволить нарушителю, действующему удаленно, получить несанкционированный доступ к конфиденциальным данным, вызвать отказ в обслуживании или оказать воздействие на целостность данных через специально созданную TLS сессию.

Распространена в OpenSSL версий до 1.1.0a включительно, которые используются на Debian GNU/Linux 8, 9, 10, а также в Astra Linux Common Edition 2.12

Была выявлена 26.09.2016 года, на данный момент устранена

По критериям CVSS 2.0 и CVSS 3.0 имеет критический уровень опасности (базовая оценка составляет 10 и 9,8 соответственно)

Для устранения уязвимости необходимо обновить OpenSSL до версии 1.0.2e или более поздней.

* 1. CVE-2014-0160: OpenSSL TLS Heartbeat Extension - 'Heartbleed'.

Ошибка Heartbleed является серьезной уязвимостью в популярной криптографической библиотеке программного обеспечения OpenSSL. Дефект Heartbleed может приводить к утечке системной информации, если используется уязвимая версия OpenSSL. Речь может идти о такой в высшей степени конфиденциальной информации, как закрытые ключи и имена и пароли пользователей, хранящиеся в памяти.

Причиной Heartbleed является классическое чтение за границей  
буфера в реализации протокола Heartbeat в OpenSSL. Протокол Heartbeat позволяет устройствам проверять, существует ли еще подключение к серверу с поддержкой SSL; для этого серверу отправляется запрос Heartbeat, содержащий произвольную строку символов, заданную отправителем. Если все хорошо, сервер пошлет ответ Heartbeat, содержащий ту же строку.  
Помимо строки символов, запрос Heartbeat содержит поле, в ко-  
тором указана длина строки. Именно некорректная обработка этого поля и стала причиной уязвимости Heartbleed. Уязвимые версии  
OpenSSL позволяют атакующему указать длину, гораздо большую  
фактической длины строки, что заставит сервер отправить дополни-  
тельные байты из памяти при копировании строки в ответ.

1. Практическая реализация уязвимости Heartbleed

Данная уязвимость характерна для OpenSSL версий 1.0.1 – 1.0.1f включительно. При этом версия 1.0.1g уже не содержит данную уязвимость, а также её не содержат OpenSSL до версии 1.0.0 включительно.

Данная ошибка обнаружена в OpenSSL в декабре 2011 года и была обнаружена с момента выпуска OpenSSL 1.0.1 14 марта 2012 года. OpenSSL 1.0.1g, выпущенный 7 апреля 2014 года, исправляет ошибку.

OpenSSL с данной уязвимостью входит в состав следующих операционных систем:

* Debian Wheezy (stable), OpenSSL 1.0.1e-2+deb7u4
* Ubuntu 12.04.4 LTS, OpenSSL 1.0.1-4ubuntu5.11
* CentOS 6.5, OpenSSL 1.0.1e-15
* Fedora 18, OpenSSL 1.0.1e-4
* OpenBSD 5.3 (OpenSSL 1.0.1c 10 May 2012) and 5.4 (OpenSSL 1.0.1c 10 May 2012)
* FreeBSD 10.0 - OpenSSL 1.0.1e 11 Feb 2013
* NetBSD 5.0.2 (OpenSSL 1.0.1e)
* OpenSUSE 12.2 (OpenSSL 1.0.1c)

Для реализации данной уязвимости была использована SEED Ubuntu 12.04, на которой уже запущен сервер на базе Apache 2.2.22 с OpenSSL 1.0.1, содержащей данную уязвимость и Kali Linux 2022.4. Действия по эксплуатации уязвимости производились с использованием фреймворка Metasploit Framework и встроенного модуля auxiliary/scanner/ssl/openssl\_heartbleed

Для реализации угрозы с использованием данной уязвимости выполним следующие действия:

Узнаем IP-адрес виртуальной машины с Ubuntu, после чего пропингуем виртуальную машину с Kali Linux:

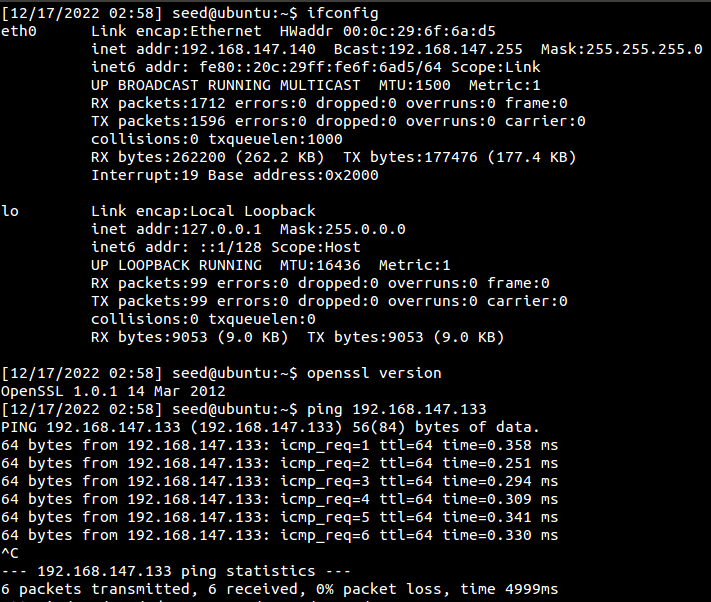
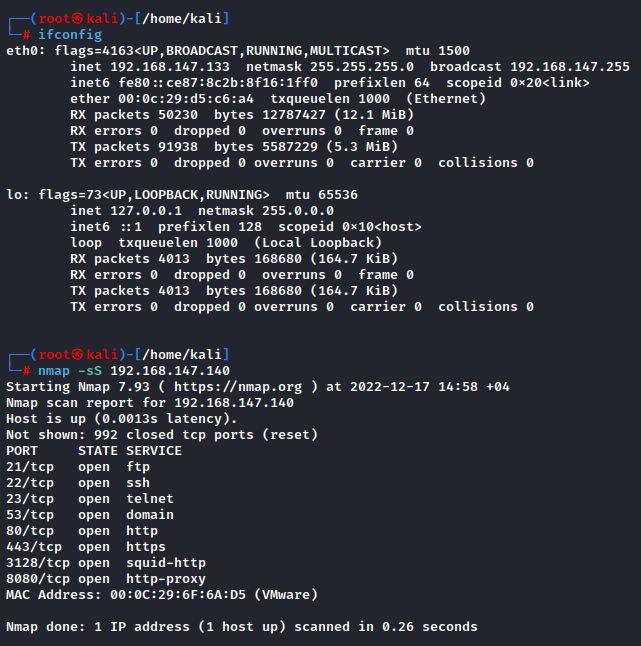
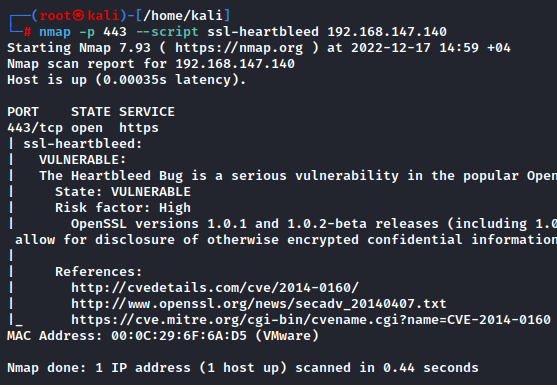


Рисунок 6 – IP-адрес

На виртуальной машине с Kali Linux утилитой nmap произведём сканирование жертвы и убедимся, что 443 порт открыт:

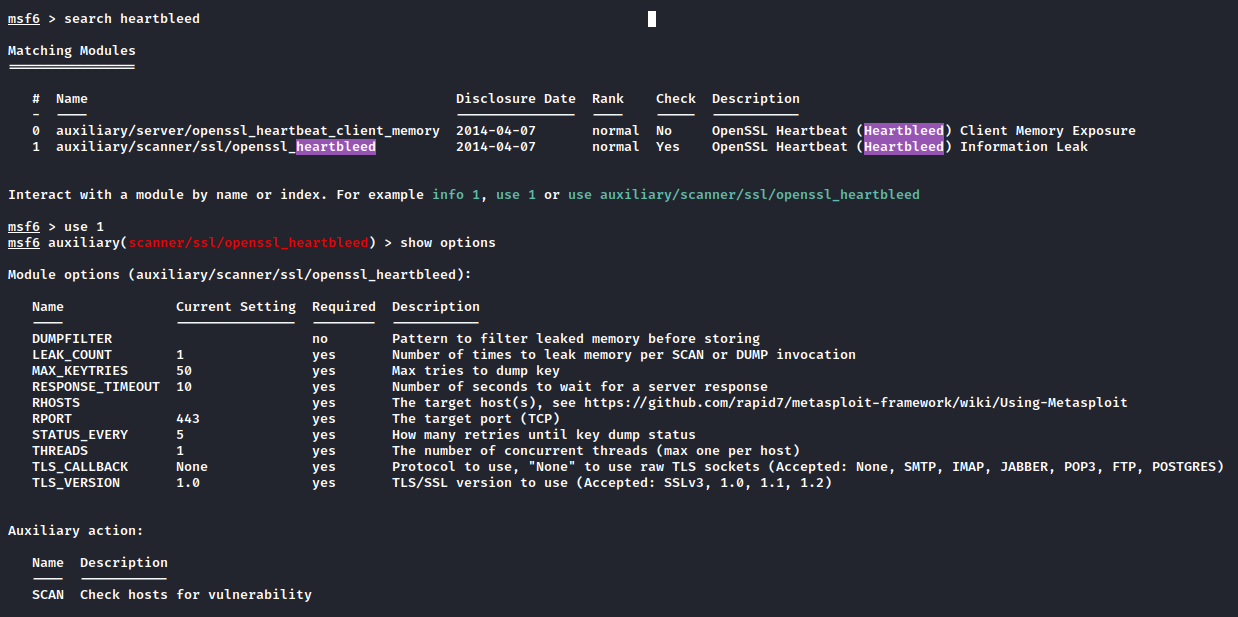


443 порт на виртуальной машине с Ubuntu открыт. С помощью той же утилиты произведём сканирование на наличие уязвимости Heartbleed:

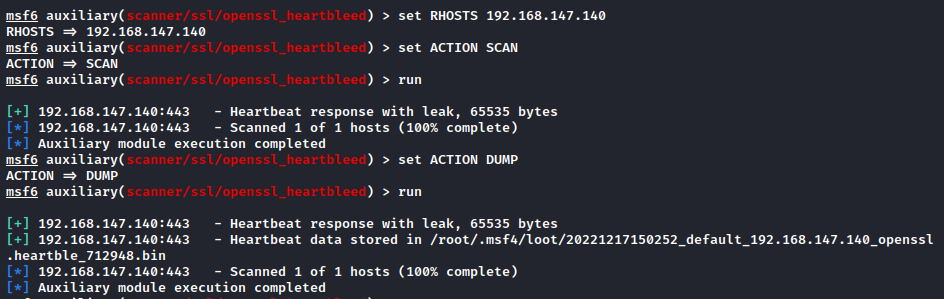


В результате сканирования видно, что OpenSSL, установленная на Ubuntu 12.04, подвержена реализации угрозы.

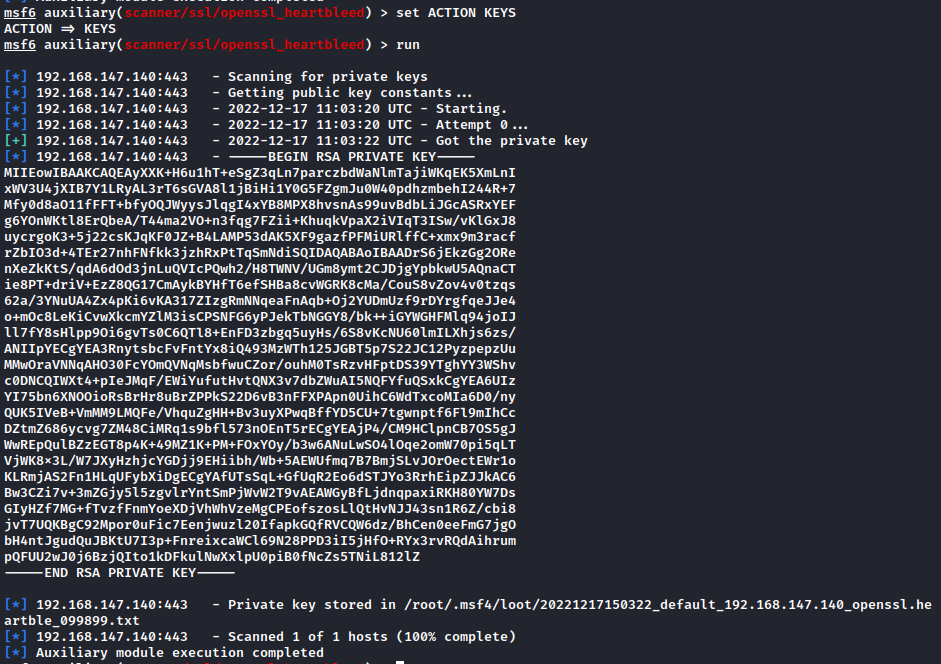
Произведем реализацию данной угрозы. Запустим Metasploit Framework, после чего найдём модуль openssl\_heartbleed, активируем его и определим параметры для конфигурации:



Сконфигурируем модуль: назначим IP-адрес жертвы и действие, после чего произведём сбор необходимого трафика:



После выполнения данных действий из полученных данных достаём приватный ключ шифрования:



Для устранения уязвимости в данном случае необходимо обновить Ubuntu до версии 22.04 LTS или 22.10, в которых установлена одна из последних версий OpenSSL, или скомпилировать из исходников OpenSSL для Ubuntu 12.04. Второй вариант не рекомендуется, так как эта версия Ubuntu – устаревшая, для неё не выпускаются обновления безопасности, что ставит под угрозу общую безопасность системы.

Заключение

Список используемой литературы

1. https://help.ubuntu.com/community/OpenSSL
2. https://hackware.ru/?p=12982#1П
3. https://habr.com/ru/company/otus/blog/666700/П
4. https://wiki.merionet.ru/servernye-resheniya/45/rukovodstvo-po-openssl-ssl-sertifikaty-klyuchi-i-csr/П
5. https://reader.lanbook.com/journalArticle/246715#2П
6. https://reader.lanbook.com/book/241139#293П
7. https://phoenixnap.com/kb/openssl-tutorial-ssl-certificates-private-keys-csrsП
8. https://bdu.fstec.ru/vulП
9. https://heartbleed.com/П
10. ГОСТ 34.10-2015
11. <https://habr.com/ru/company/aktiv-company/blog/467707/>
12. 3. Exploit database [Электронный ресурс]. – Режим доступа: https://www.exploit-db.com/?platform=linux&port=21

ЛИСТ ЗАМЕЧАНИЙ