Активные сетевые атаки

DOS, DDOS, arp spoofing, dhcp spoofing, dns spoofing, sslsplit, sslstrip, sslsplit.

Активные сетевые атаки

Активное воздействие на сеть оказывает непосредственное влияние на работу системы: изменяет конфигурацию, нарушает работоспособность и принятую политику безопасности. Практически все типы удаленных атак — это активные воздействия.

Они отличаются от пассивных тем, что их принципиально можно обнаружить — с большими или меньшими усилиями. Дело в том, что в результате активных воздействий в системе происходят изменения.

DoS и DDoS

DoS-атака направлена на то, чтобы жертва вошла в состояние «отказа в обслуживании» (Denial of Service). Так мы не компрометируем и не повреждаем конфиденциальную информацию, но делаем ее недоступной на время.

Иногда выделяют DDoS-атаки (Distributed Denial of Service) — распределенные атаки типа «отказ в обслуживании». Воздействие производится не одним узлом, а массированным числом «зомби»-компьютеров, ботнетом.

DoS-атаки можно условно поделить на три группы:

1. Атаки большим числом формально корректных, но, возможно, сфальсифицированных пакетов. Такие воздействия направлены на истощение ресурсов узла или сети.

2. Атаки специально сконструированными пакетами, вызывающие общий сбой системы из-за ошибок в программах.

3. Атаки фальсифицированными пакетами, вызывающими изменения в конфигурации или состоянии системы. Блокируют передачу данных, сбрасывают соединение или резко снижают его эффективность.

DoS-атаки:

1. Шторм ложных TCP-запросов на создание соединения.

2. UDP flood.

3. Ложные DHCP-клиенты.

4. Smurf-атака.

5. Изменение конфигурации или состояния узла

6. Атаки, использующие ошибки реализации сетевых служб.

Для защиты от DDoS-атак используют специальные сервисы. Наиболее известные:

- QRATOR — используется сервисом Habrahabr.ru;

- Cloudflare.

В этом случае в DNS-домены защищаемых ресурсов анонсируются на DNS-сервера соответствующего сервиса. Эти IP-адреса — **anycast**: один «белый» IP-адрес разделяется множеством машин. Это происходит благодаря протоколу маршрутизации **BGP**, который выбирает ближайший anycast-узел.

Далее трафик приходит на машину сервиса-защитника, который анализирует, является ли данный запрос DDoS-атакой. Если все хорошо, он пропускает на серверы клиента.

MITM. Спуфинг

Man In The Middle — «человек посередине». Прослушать чужой трафик легко, если мы находимся в том же сегменте сети, а сеть построена на базе концентратора (хаба). То же можно сделать, если сеть работает с коммутатором. В обоих случаях можно внушить жертве, что злоумышленник — это маршрутизатор, а последнего заставить считать таковым клиента.

Если используется коммутатор, тоже можем сделать так, что выбранный абонент будет пересылать информацию через компьютер злоумышленника. Это позволяет и прослушивать, и подменять трафик. Это возможно благодаря архитектурным особенностям ряда протоколов стека TCP/IP, прежде всего **ARP**, **DHCP** и **DNS**.

Разберем основные атаки:

- ARP-Spoofing;
- DHCP-Spoofing;
- DNS-Spoofing.

На базе одной из них выстраивается фундамент атаки на

SSL: ● sslstrip; ● sslsplit.

ARP Spoofing

Для адресации IP-пакетов в Internet кроме IP-адреса хоста необходим:

- либо Ethernet-адрес его сетевого адаптера — в случае адресации внутри одной подсети;
- либо Ethernet-адрес маршрутизатора — в случае межсетевой адресации.

Чтобы хост смог найти эти адреса, применяя алгоритмы удаленного поиска в сети, используется протокол **ARP** (Address Resolution Protocol).

Анализ безопасности протокола **ARP** показывает: перехватив на атакующем хосте внутри данного сегмента сети широковещательный ARP-запрос, можно послать ложный ARP-ответ. Если в нем объявить себя искомым хостом (например, маршрутизатором), в дальнейшем можно активно контролировать сетевой трафик дезинформированного хоста.

Arp-spoofing — это атака L2-уровня. Разберем ее схему атаки:

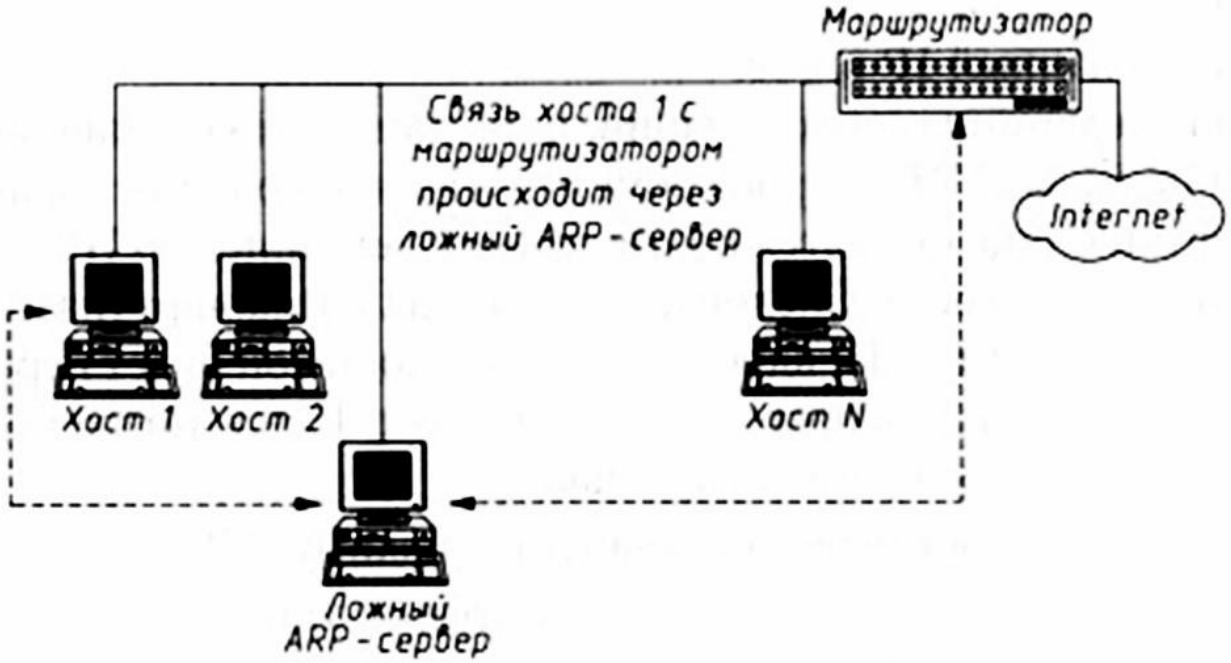
1. **Ожидание ARP-запроса от жертвы.**

Если получен такой запрос, по сети на запросивший хост передается ложный ARP-ответ. В нем указывается адрес сетевого адаптера атакующей станции (ложного ARP-сервера) или тот Ethernet-адрес, на котором будет принимать пакеты ложный ARP-сервер.

2. **Прием, анализ, воздействие на пакеты обмена и передача их между взаимодействующими хостами.**

arp

2. Фаза атаки.



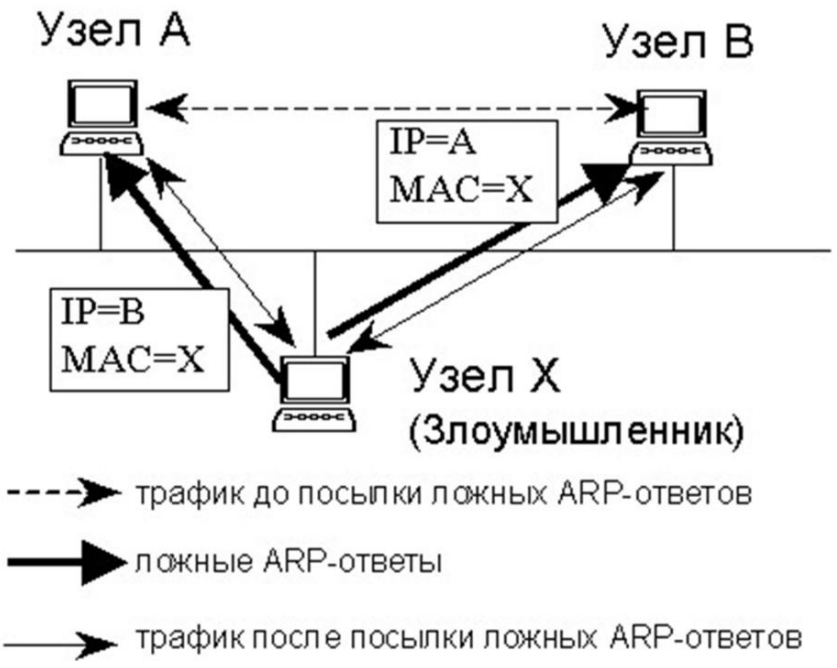
3. Фаза приема, анализа, воздействия и передачи перехваченной информации на ложном ARP-сервере.

3. **Атакованный хост передает пакеты на ложный ARP-сервер.**

Ложный ARP-сервер посылает принятые от атакованного хоста пакеты на маршрутизатор.

Если пришел ответ на запрос, маршрутизатор адресует его непосредственно на атакованный хост, минуя ложный ARP-сервер.

Но можно посылать ложные ARP-ответы в обе стороны. Так делают, чтобы реализовать большой класс MITM-атак.



Рассмотрим, как используют **ARP Spoofing** с помощью утилиты **arpspoof**. Отравим arp-таблицу маршрутизатору:

```
arpspoof -i eth0 10.0.2.6 -t 10.0.2.1
```

Отправим arp-таблицу для жертвы (DVL):

```
arpspoof -i eth0 10.0.2.1 -t 10.0.2.6
```

Посмотрим arp-таблицу на DVL:

```
arp -a
```

В обе стороны:

```
arpspoof -i eth0 -c both -t 10.0.2.1 -r 10.0.2.6
```

Запустим на жертве:

```
ping ya.ru
```

Понаблюдаем трафик в Wireshark на машине злоумышленника.

Что помогает защититься:

- использовать статические **arp**;
- отслеживать **arpwatch**;
- разбивать сеть на **vlan**;
- применять **Access lis**t на управляемом коммутаторе.

Вариант — использовать «железо» для обнаружения вторжений.

На практике почти никто так не поступает: **vlan** делают по другим соображениям.

Чтобы конструкция заработала, нужно в Kali Linux включить **ip forwarding**. Вместо **arpspoof** можно использовать **ettercap**:

```
ettercap -i eth0 -T -q -M ARP /10.0.2.1/10.0.2.6/
```

Выполним **arpspoof** для всех:

```
ettercap -i eth0 -T -q -M ARP /10.0.2.1//
```

etter

Или использовать графическую оболочку:

```
ettercap -G
```

Чтобы написать фильтры:

```
etterfilter
```

Чтобы добавить плагин, используем ключ **-P autoadd**:

```
ettercap -i eth0 -P autoadd  -T -q -M ARP /10.0.2.1//


ettercap -i eth0 -T -q -w ettercap.pcap -M ARP /10.0.2.1/10.0.2.6/
```

Посмотрим фильтры:

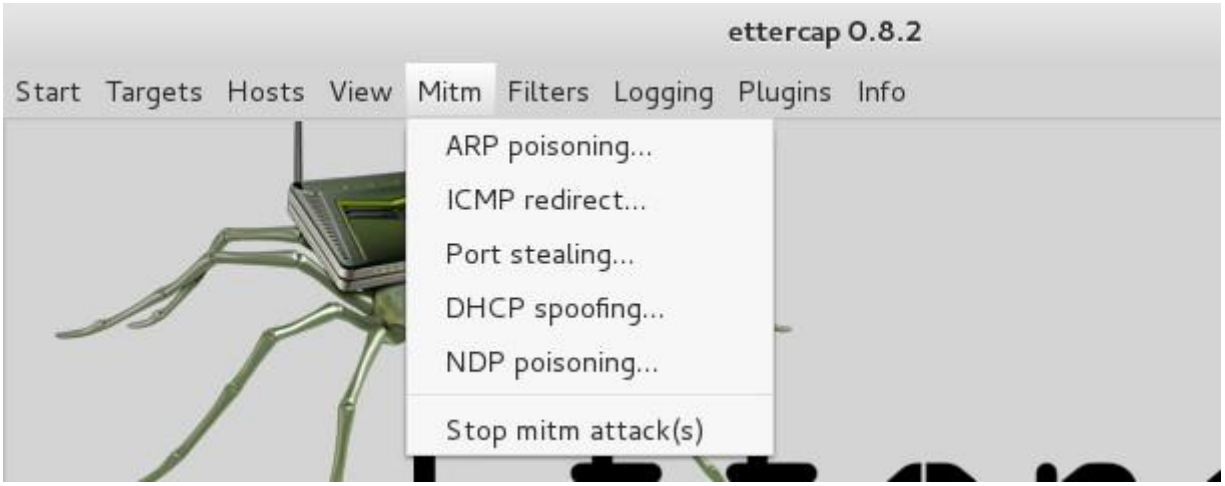
```
ls /usr/share/ettercap
cat /usr/share/ettercap/etter.filter.example
```

DHCP-spoofing

Рассмотрим алгоритм работы **DHCP**. С порта 68/UDP на 67/UDP компьютер отправляет запрос обнаружения DHCP-сервера на 255.255.255.255 с адреса 0.0.0.0. Сервер откликается, предлагая IP-адрес, маску, настройки DNS и шлюза по умолчанию. Клиент отправляет запрос адреса, сервер его подтверждает, и клиент использует настройки, сообщенные dhcp-сервером.

Ettercap позволяет не рассылать постоянно arp-запросы, а один раз заставить жертву добровольно использовать в качестве шлюза машину злоумышленника. Срабатывать это будет не всегда. Все зависит от того, кто ответит первым: настоящий DHCP-сервер или ложный.

Чтобы устранить эту проблему, нужно чуть заDoSить подлинный DHCP-сервер.



DNS-spoofing

Система DNS обеспечивает поиск для доменного имени соответствующего значения ресурса — как правило, IP-адреса.

Рассмотрим алгоритм удаленного поиска IP-адреса по имени в сети Internet.

Ближайший DNS-сервер, как правило, устанавливается при настройке сетевой ОС. Хост посылает на его IP-адрес DNS-запрос и указывает в нем имя сервера, IP-адрес которого необходимо найти.

DNS-сервер, получив такое сообщение, ищет указанное в запросе имя в своей базе. Если находит, отправляет на хост DNS-ответ с соответствующим IP-адресом. Если не обнаруживает такого имени, пересылает DNS-запрос на один из ответственных за домены верхнего уровня DNS-серверов. Их адреса содержатся в файле настроек DNS-сервера. Тогда процедура повторяется, пока имя не будет найдено или окончательно не обнаружено.

Три варианта удаленной атаки на DNS:

1. Перехват DNS-запроса.

2. Направленный шторм ложных DNS-ответов на атакуемый хост.

3. Перехват DNS-запроса или создание направленного шторма ложных DNS-ответов на DNS-сервер.

Перехват DNS-запроса (схема атаки):

1. Ожидают DNS-запроса.

2. Из полученного сообщения извлекают сведения. На запросивший хост передают по сети ложный DNS-ответ от имени (с IP-адреса) настоящего DNS-сервера. Здесь указывают IP-адрес ложного DNS-сервера.

3. Если приходит пакет от хоста — прописывают в его IP-заголовке IP-адрес ложного DNS-сервера и передают пакет на сервер. Ложный DNS-сервер ведет работу с сервером от своего имени.

4. Если приходит пакет от сервера — прописывают в его IP-заголовке IP-адрес ложного DNS-сервера и передают пакет на хост. Так для атакованного хоста ложный DNS-сервер выглядит как настоящий.

```
dnsspoof
```

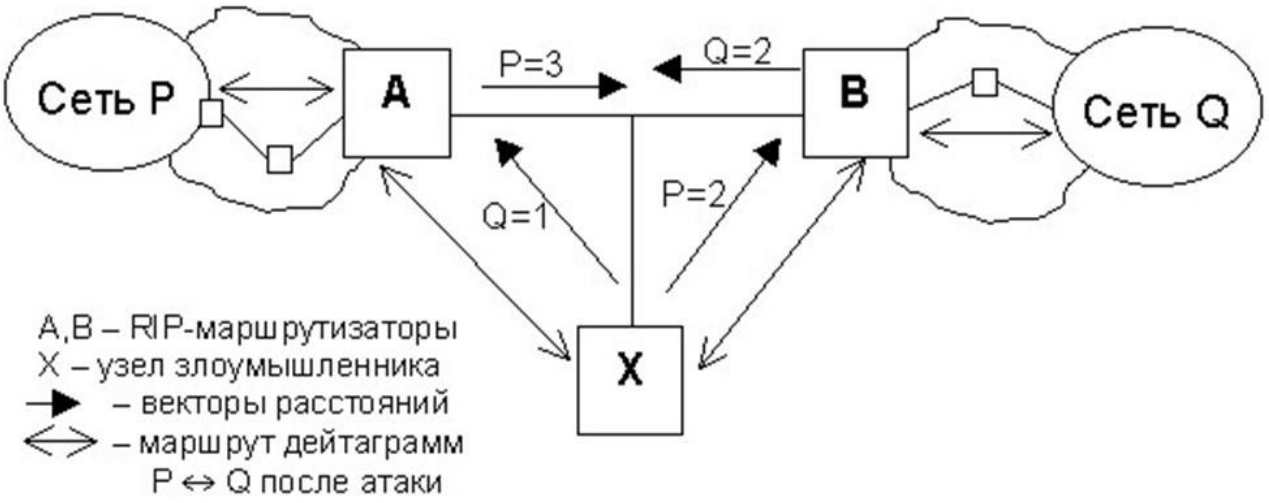
hostfile — файл, похожий на **/etc/hosts**, но служит для того, чтобы навязывать жертвам ответы:

```
dnsspoof -f hostfile
```

Если утилита **dnsspoof** не сработает, попробуйте использовать **ettercap**.

Атаки на протоколы маршрутизации

Разберем пример атаки на протокол маршрутизации RIP. Он довольно простой: маршрутизаторы рассылают метрики — информацию о количестве хопов. Если злоумышленник будет отправлять в обе стороны сигнал о том, что маршрут доступен через него с меньшей метрикой — трафик пойдет через ложный маршрутизатор.



Атаки на SSL

Многие протоколы изначально не были шифрованными. Выполнить **arpspoof** уже было достаточно, чтобы с помощью Wireshark проанализировать трафик жертвы, получить доступ к конфиденциальным данным и паролям. HTTP, SMTP, FTP изначально передают данные в нешифрованном виде.

Даже сейчас продолжают работать сайты, которые передают данные по HTTP, и хостинги, которые предлагают заливать данные на сервер по FTP. Таких веб-ресурсов и сервисов становится все меньше, но они еще есть.

HTTPS — протокол HTTP, инкапсулированный в TLS (дальнейшее развитие протокола SSL). Он проверяет подлинность ресурса и шифрует данные, а потом инкапсулируется в сессию TCP — она уже не подразумевает шифрование.

Но это не значит, что HTTPS — панацея. Рассмотрим две атаки, которые позволяют получить доступ даже к шифрованной информации.

sslstrip

Sslstrip базируется на том, что если в браузере вбить адрес ресурса — например, **site.ru**, — он сначала попытается подключиться по протоколу **http**. И только когда клиент подключится к серверу на порт 80/TCP, получит редирект 302 с указанием адреса **https** в заголовке **Location**.

Злоумышленник может воспользоваться этим: продолжить отвечать по **http** с порта 80/TCP, а к серверу подключаться по HTTPS на 443 TCP-порт от имени клиента. При этом утилита **sslstrip** вынуждена анализировать и **html**, заменяя **https** в схеме на **http** — и это не всегда корректно работает.

Для убедительности **sslstrip** может добавить и иконку замка. Когда она размещалась слева от схемы, это дезинформировало пользователей. Сейчас там находится надпись «Защищено» и значок замка, а иконки браузеры вынесли в заголовок таба.

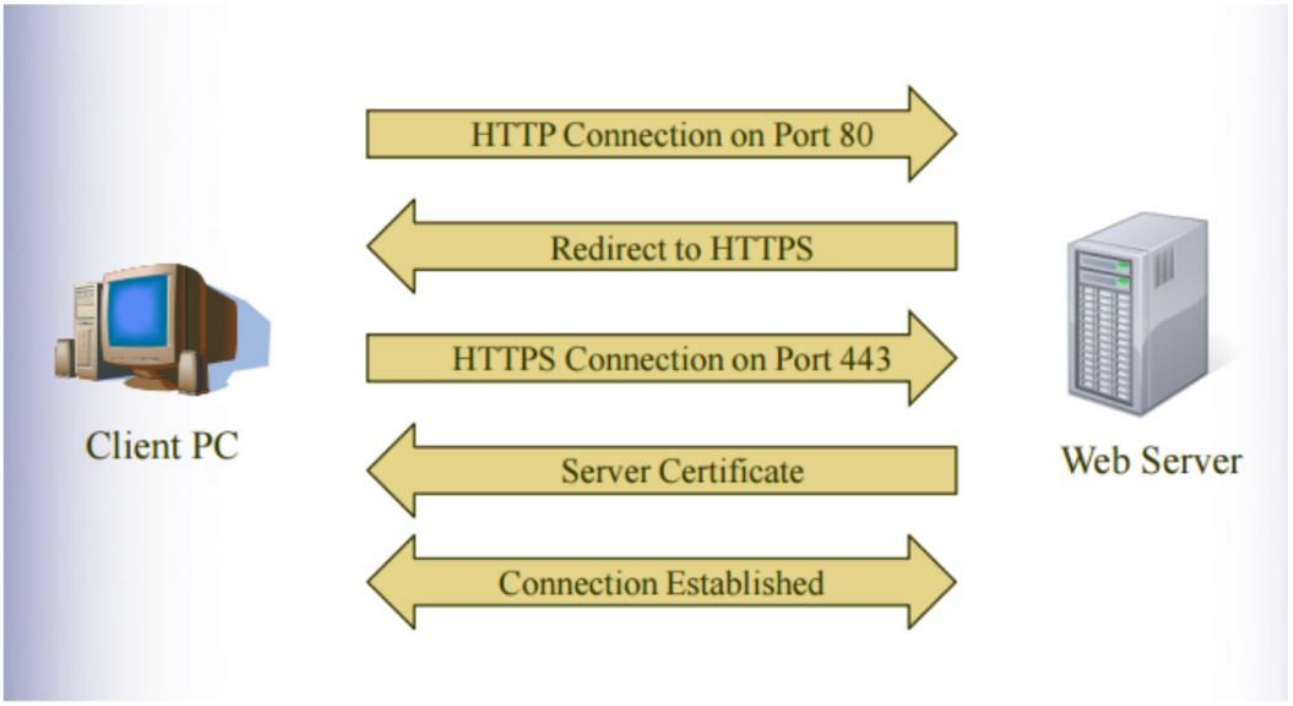


Схема подключения к https

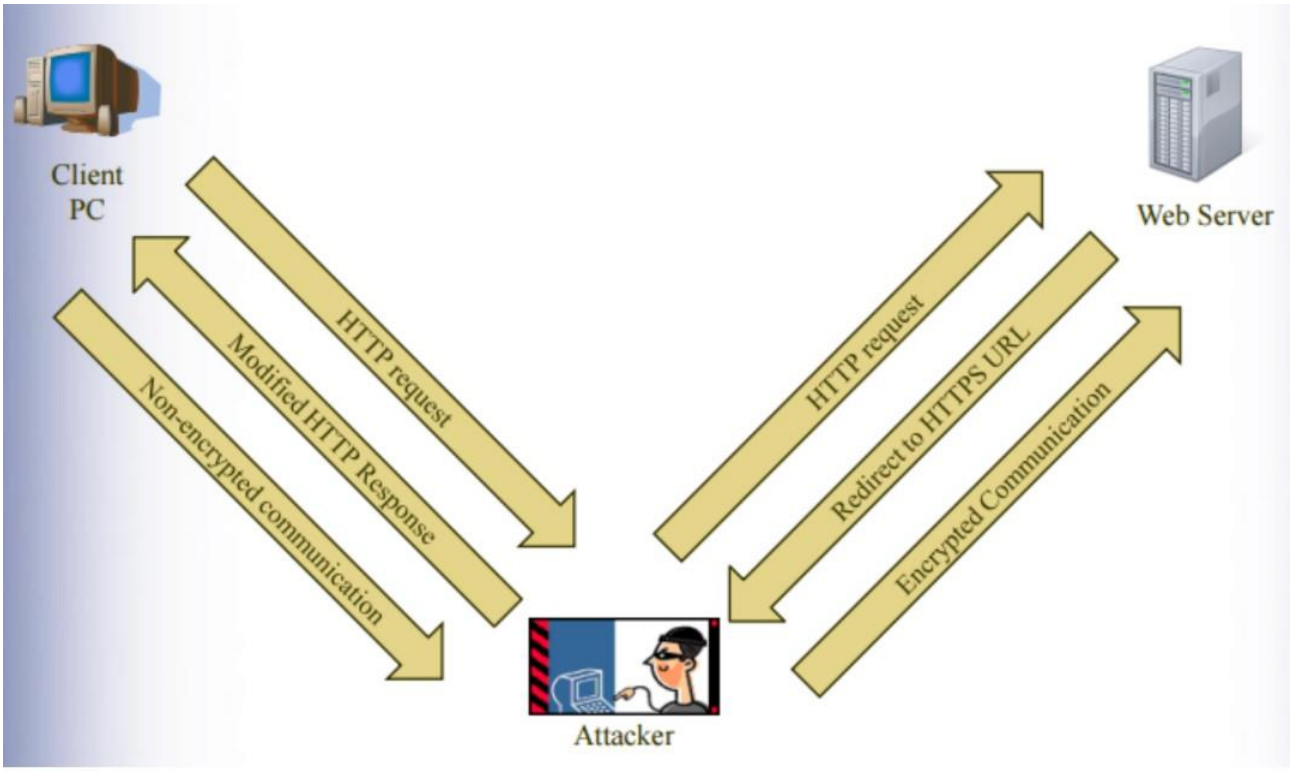


Схема с злоумышленником

Чтобы защититься от **sslstrip**, используются механизмы https everywhere и HSTS — последний тоже не панацея, можно его обойти.

Еще можно применять **arpspoof** или запущенный **ettercap**.

Следует включить IP Forwarding:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Настроить пересылку с порта 80 на 8080:

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080
```

Запустить, записывая запросы и ответы в **log**, слушать на порту 8ыыдыекш 080:

```
sslstrip -w sslstrip.log -l 8080
```

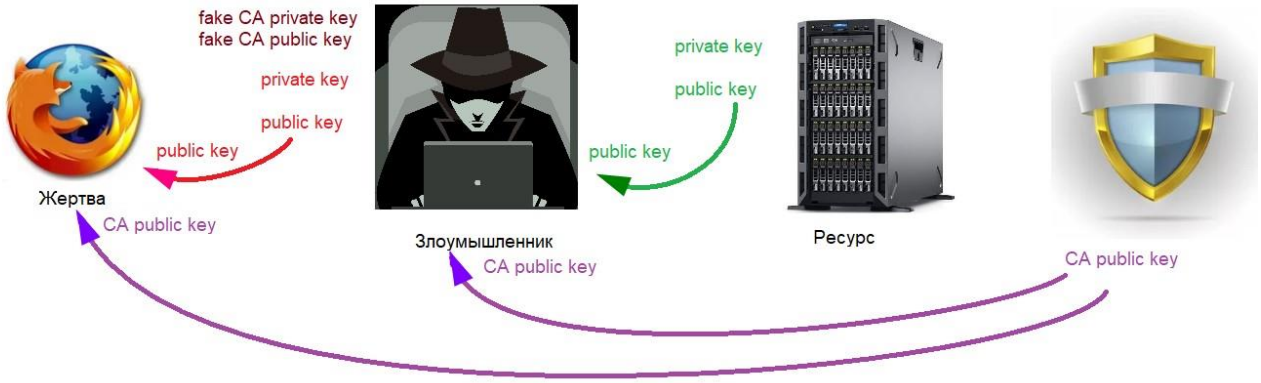
SSLSPLIT

SSLSPLIT позволяет заменить сертификат. Злоумышленник выпускает пару «закрытый-открытый ключ» и подписывает ее своим сертификатом. Если он не проверяется, хакер получает доступ к конфиденциальным данным.

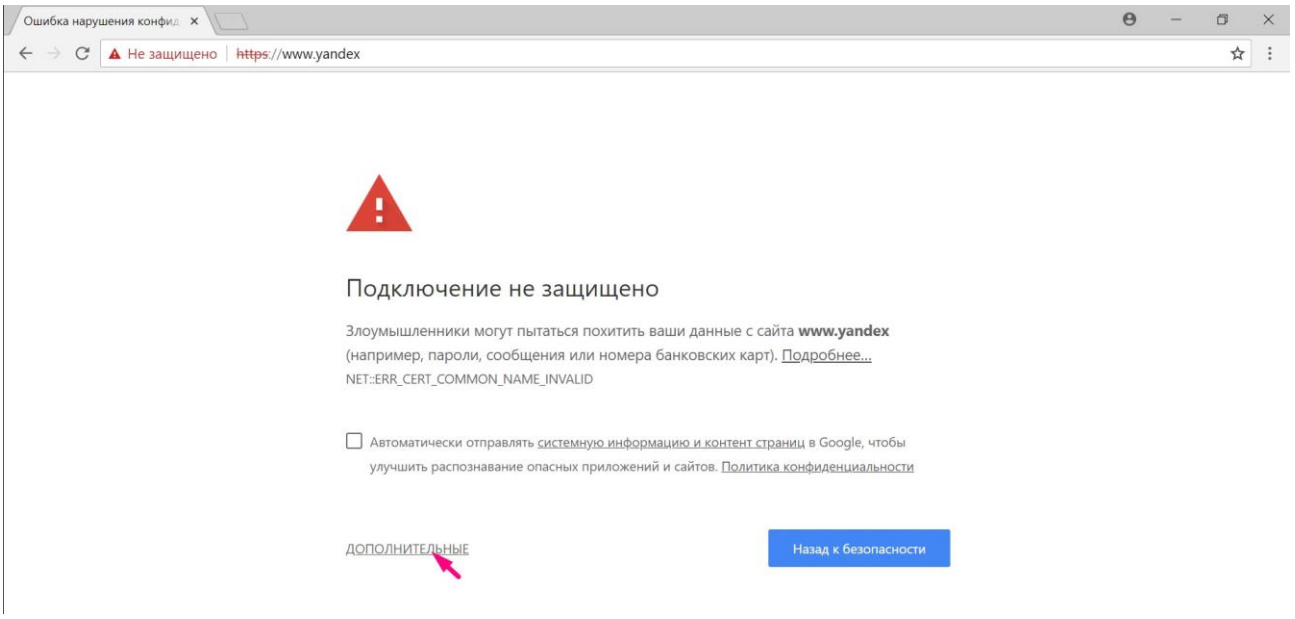
Даже если браузер видит, что сертификат не валиден, он может дать возможность нажать «Далее» и работать по незащищенному соединению.

Но сертификат можно и импортировать. Это может сделать провайдер, якобы чтобы иметь доступ к локальным ресурсам **.localnet**. Или работодатель, который уже встроил сертификат в браузер. Так может поступить злоумышленник, используя методы социальной инженерии. Заходит жертва на сайт казино, а он предлагает: чтобы деньги и пароли не утекли, нужно установить в браузер сертификат и использовать защищенное соединение. Пользователь та и делает, и браузер больше не предупреждает, что сертификат не валидный.

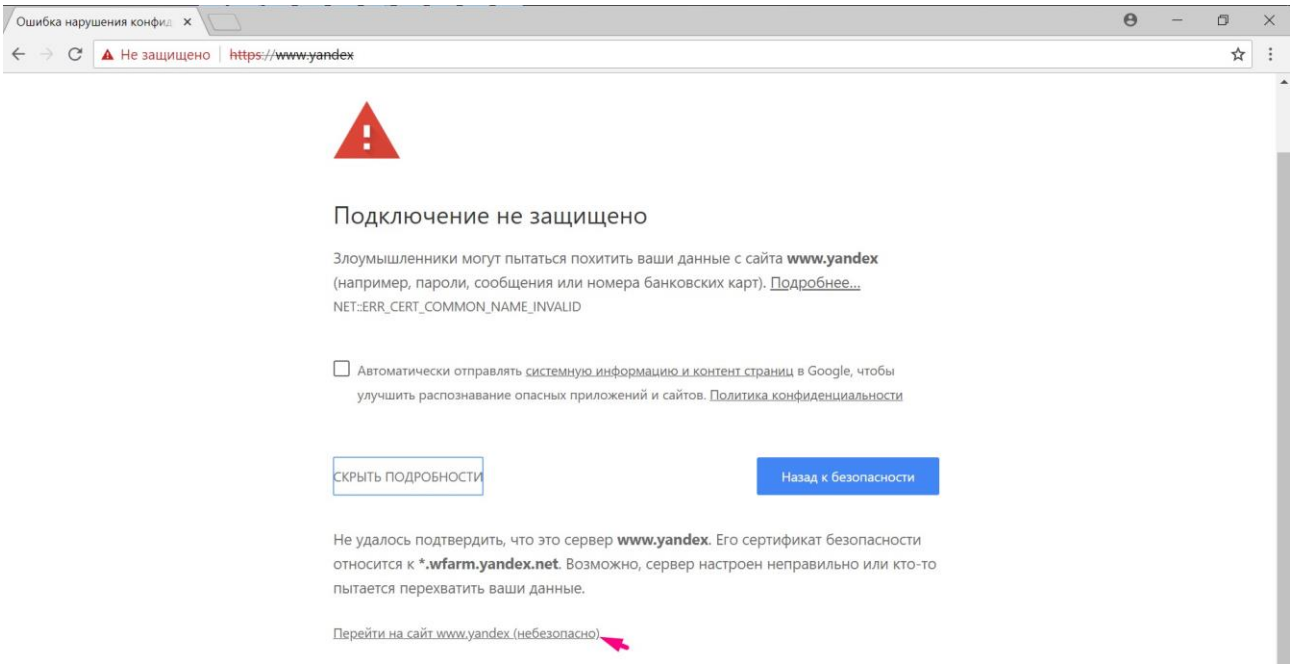
Чтобы защититься, надо повышать компьютерную и информационную грамотность сотрудников: учить не скачивать расширения, не устанавливать приложения, не открывать сомнительные аттачи в почте.



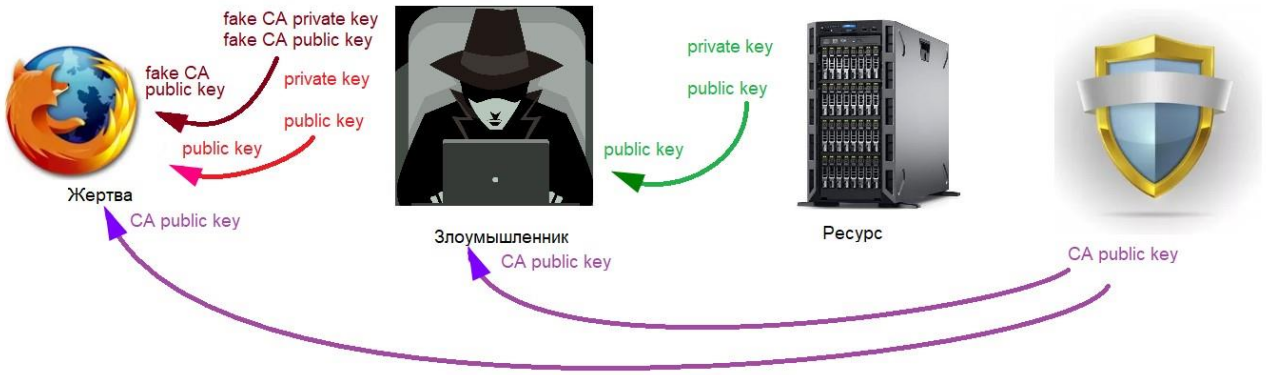
Злоумышленник представляется клиентом, а для жертвы — ресурсом. Если жертва (например, браузер) проверяет, валиден ли сертификат, эта схема сработает, только если пользователь нажмет «далее»



Только для умных



На свой страх и риск



Если злоумышленник убедит пользователя импортировать свой сертификат ложного CA, то предупреждений уже не будет

Включаем **IP Forwarding**:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Настраиваем пересылку с 80 TCP-порта на 8080:

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080
```

Настраиваем пересылку с 443 TCP-порта на 8443:

```
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8443
```

Генерируем сертификат:

```
openssl genrsa -des3 -out server.key 2048 openssl rsa -in
server.key -out server.key.insecure mv server.key
server.key.secure mv server.key.insecure server.key openssl req
-new -key server.key -out server.csr
openssl x509 -req -days 365 -in server.csr -sign key.key -out
server.crt
```

Запускаем **sslsplit**:

```
sslsplit -D -l sslsplit.log -j /tmp/sslsplit/ -S /tmp/ -k server.key -c
server.crt ssl 0.0.0.0 8443 tcp 0.0.0.0 8080
```

Далее надо запустить в отладочном режиме (**-D**), вести лог подключений (**-l sslsplit.log**), установить **chroot jail (-j /tmp/sslsplit/**), сохранить файлы на диск (**-S /tmp/**), указать ключ (**-k server.key**) и сертификат (**-c server.crt**), настроить прокси (**0.0.0.0 8443** для **ssl**, **0.0.0.0 8080** для **tcp**).

Чтобы у жертвы страница открылась сразу, пользователь должен сам дать разрешение браузеру на подключение в небезопасном режиме или заранее импортировать сертификат.

В мобильных приложениях цепочку сертификатов могут вовсе не проверять:

```
private static bool CertificateValidationCallBack(
                object sender,
          System.Security.Cryptography.X509Certificates.X509Certificate
certificate,
                    System.Security.Cryptography.X509Certificates.X509Chain
chain,
```

```
System.Net.Security.SslPolicyErrors sslPolicyErrors)
    { return true;
    }
```

Рекомендуем проводить аудит кода и исправлять такие уязвимости. Защитить приложение от **sslsplit** может **Certificate Pinning** — сертификат встраивается в код продукта.

Дополнительные материалы

1. http://ettercap.github.io/ettercap/.
2. http://bit.ly/1C9ge9U (ettercap filters sample).
3. https://ru.wikipedia.org/wiki/HTTPS_Everywhere.

Использованная литература

1. https://kali.tools/?p=177.
2. https://kali.tools/?p=1232.
3. http://blog.regolit.com/2010/02/16/personal-ca-and-self-signed-certificates.