ss -tnp

КТО СЛУШАЕТ ЭТОТ ПОРТ?

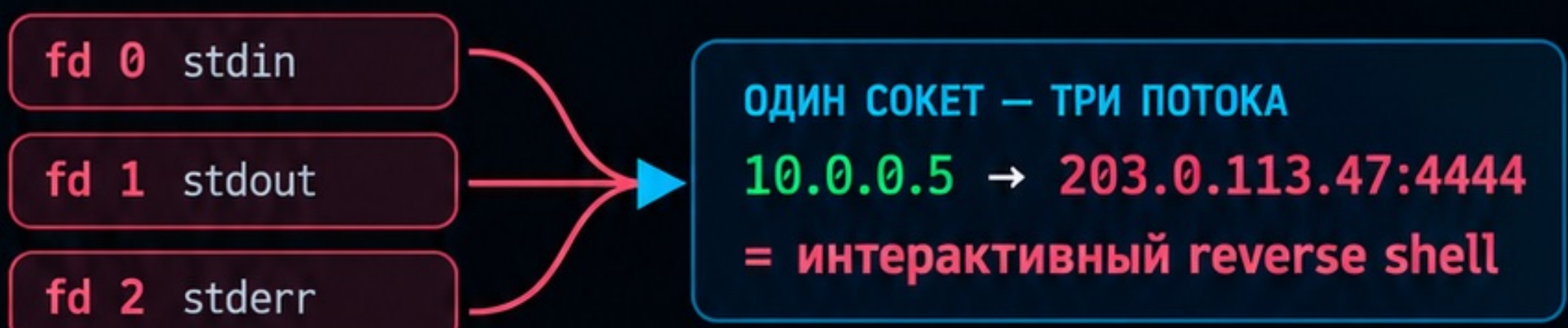
Быстрый способ увидеть, какой процесс принимает соединения.

-t только TCP **-n** без DNS **-p** показать процесс

root@web-01: ~

```
root@web-01:~# ss -Htnp state established
0 0  10.0.0.5:22      198.51.100.8:51422   users:(("sshd",pid=812,fd=4))
0 0  10.0.0.5:443     198.51.100.23:60114  users:(("nginx",pid=1044,fd=12))
0 0  10.0.0.5:48210   203.0.113.47:4444    users:(("bash",pid=2318,fd=2),
("bash",pid=2318,fd=1),("bash",pid=2318,fd=0))

root@web-01:~# ps -o user,cmd -p 2318
USER         CMD
www-data     bash -i
```



ЧТО ПРОИСХОДИТ?

Пользователь **www-data** запустил shell.
Веб-серверы общаются с клиентами, но им не нужен **bash -i**.

ЧТО ДЕЛАТЬ ДАЛЬШЕ?

Не убивай процесс сразу.
Запомни **PID 2318** и проверь, есть ли его бинарный файл на диске (он может быть удалён).

@lionsoulbatyshchikov

02/07

/proc/*/exe

ФАЙЛ ПРОЦЕССА ВСЁ ЕЩЁ ОСТАЁТСЯ НА ДИСКЕ?

Даже если бинарный файл был удалён,
его часто всё ещё можно найти.

ls -l показать, куда ссылается symlink

grep найти удалённые

root@web-01: ~

```
root@web-01:~# ls -l /proc/2318/exe
lrwxrwxrwx 1 www-data www-data 0 /proc/2318/exe ->
  /tmp/.x/kworkerd (deleted)
root@web-01:~# ls -l /proc/*/exe | grep deleted
/proc/2318/exe -> /tmp/.x/kworkerd (deleted)
root@web-01:~# # бинарник уже удалён с диска, но всё ещё в RAM
root@web-01:~# cp /proc/2318/exe /root/evidence.bin
root@web-01:~# sha256sum /root/evidence.bin
86d31f6f...b799e91f  /root/evidence.bin
```

ВАЖНАЯ ДЕТАЛЬ

Обновлённые пакеты, которые не были перезапущены, тоже могут отображаться как **(deleted)**. Необычный путь и необычный пользователь — это повод для проверки.

ПОЧЕМУ ЭТО ВАЖНО?

Процесс, чей файл помечен как **(deleted)**, был запущен, а затем удалил свой бинарник, чтобы скрыться. Перезагрузка не поможет — следы остаются только в памяти.

КАК ЭТО ИСПОЛЬЗОВАТЬ?

Бинарный файл всё ещё жив в памяти. Команда **cp /proc/PID/exe** позволяет вытащить полную копию для анализа.

@lionsoulbatyshchikov

03/07

ld.so.preload

МОЖНО ЛИ ДОВЕРЯТЬ ДАЖЕ КОМАНДАМ ps И ls?

cat прочитать файл

ldd посмотреть, что загружает

root@web-01: ~

```
root@web-01:~# cat /etc/ld.so.preload
/usr/local/lib/libprocesshider.so
root@web-01:~# # этот файл должен быть пустым или отсутствовать
root@web-01:~# ldd /bin/ps
        libprocesshider.so => /usr/local/lib/...(0x00007f..)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
root@web-01:~# # теперь каждый процесс загружает библиотеку атакующего
```

ЧТО ЭТО ТАКОЕ?

Это пользовательский rootkit в userland.
Любая библиотека, указанная в **ld.so.preload**, принудительно загружается в **каждый** запускаемый процесс, поэтому она может скрывать процессы, файлы и сетевые соединения.

КАК ЭТО ОБНАРУЖИТЬ?

Статические бинарники не используют LD_PRELOAD.
Запусти busybox ps и сравни результат — скрытые PID снова появятся.

РЕАЛЬНЫЙ СЛУЧАЙ

На «чистой» Linux-системе файла **ld.so.preload** вообще нет. Если он существует — считай это признаком компрометации, пока не доказано обратное.

@lionsoulbatyshchikov

04/07

kernel/tainted

БЫЛ ЛИ КАКОЙ-ТО МОДУЛЬ ИЗМЕНЁН В ЯДРЕ?

cat прочитать флаг

lsmod список модулей

root@web-01: ~

```
root@web-01:~# cat /proc/sys/kernel/tainted
0     # чистое ядро, ничего подозрительного
root@web-01:~# cat /proc/sys/kernel/tainted
12288   # биты 12 + 13 установлены
   bit 12 (4096)   = загружен неподписанный модуль
   bit 13 (8192)   = модуль вне дерева (out-of-tree)
root@web-01:~# lsmod | head -3
diamorphine   16384  0
```

ЧТО ЭТО ЗНАЧИТ?

Ненулевое значение в **/proc/sys/kernel/tainted** указывает, что ядро было изменено во время работы. **LKM** руткиты, такие как **Diamorphine**, оставляют именно такой след.

КАК ЭТО ОБНАРУЖИТЬ?

Легитимные драйверы (например, **NVIDIA**, **VirtualBox**) тоже могут устанавливать флаг **tainted**. Не паникуй — проверь **lsmod** и изучи неизвестный модуль.

БЫСТРЫЙ АНАЛИЗ

Число — это битовая маска. Запусти скрипт **kernel-chktaint**, чтобы расшифровать, какие биты установлены обычным текстом.

Module	Size	Used
diamorphine	16384	0
nvidia	245760	12
vboxdrv	53248	5
overlay	16384	1
br_netfilter	32768	0

@lionsoulbatyshchikov

05/07

authorized_keys

КАК ЗЛОУМЫШЛЕННИКИ ВОЗВРАЩАЮТ СЕБЕ ДОСТУП?

find найти файлы ключей

-ls показать детали

root@web-01: ~

```
root@web-01:~# find /root /home -name authorized_keys -ls
262 4 -rw------- root   /root/.ssh/authorized_keys
root@web-01:~# cat /root/.ssh/authorized_keys
ssh-ed25519 AAAAC3Nz...tuxJ backup@ops
root@web-01:~# # вы никогда не добавляли этот ключ
root@web-01:~# grep AuthorizedKeysFile /etc/ssh/sshd_config
AuthorizedKeysFile /etc/ssh/keys/%u
```

ЧТО ЭТО ТАКОЕ?

Файл **authorized_keys** содержит публичные **SSH keys**, которым разрешён доступ. Если злоумышленник добавит свой ключ, он сможет заходить на сервер без пароля — тихо и надолго.

ПОЧЕМУ ЭТО ОПАСНО?

Это один из самых незаметных способов сохранить доступ. Новый пользователь не создаётся, пароль не меняется, и доступ переживает любую перезагрузку.

КАК ПРОВЕРИТЬ?

Если файл **AuthorizedKeysFile** был перенесён, реальные ключи могут лежать в другом месте. Проверь настройки в **/etc/ssh/sshd_config**, прежде чем доверять пустому результату.

@lionsoulbatyshchikov

06/07

getcap -r /

ПРИВИЛЕГИЯ, КОТОРУЮ ЧАСТО ПРОПУСКАЮТ ПРИ ПОИСКЕ SUID

`-r` рекурсивно

`2>/dev/null` без лишнего шума

root@web-01: ~

```
root@web-01:~# getcap -r / 2>/dev/null
/usr/bin/ping      cap_net_raw=ep  # обычно
/tmp/py3           cap_setuid=ep    # не обычно
root@web-01:~# /tmp/py3 -c 'import os;os.setuid(0);os.system("id")'
uid=0(root) gid=0(root) groups=0(root)
```

ЧТО ЭТО ПОКАЗЫВАЕТ?

Команда **getcap -r /** ищет файлы с linux capabilities во всей файловой системе. Например, **cap_setuid** позволяет получать **root** без **SUID**-бита, поэтому такой файл не найдётся через **find -perm -4000**.

ПОЧЕМУ ЭТО ВАЖНО?

Многие сканируют только **SUID**, но почти никто не проверяет **linux capabilities**. Один файл с **cap_setuid** может дать полноценный **root shell**.

НА ЧТО ОБРАТИТЬ ВНИМАНИЕ?

Особенно опасны **cap_setuid**, **cap_dac_override** и **cap_sys_admin** на любых файлах, которые может изменять обычный пользователь.

@lionsoulbatyshchikov

07/07

dpkg -V

КАКИЕ СИСТЕМНЫЕ ФАЙЛЫ БЫЛИ ИЗМЕНЕНЫ?

dpkg -V проверить Debian

rpm -Va проверить RHEL

root@web-01: ~

```
root@web-01:~# dpkg -V
# формат: ???? = изменён checksum
??5??????  /usr/sbin/sshd
??5??????  /lib/x86_64-linux-gnu/security/pam_unix.so
missing    /usr/share/man/man1/tset.1.gz  # отсутствует
root@web-01:~#  # RHEL / CentOS эквивалент:
root@web-01:~# rpm -Va
```

ЗАЧЕМ ЭТО НУЖНО?

Злоумышленники подменяют системные бинарники на свои. Изменённый **sshd** может красть пароли, а патченный **pam_unix.so** будет логировать каждый вход в систему.

ГЛАВНАЯ ИДЕЯ

Ничего устанавливать не нужно.
Менеджер пакетов уже знает правильные контрольные суммы — остаётся только сравнить.

КАК ЧИТАТЬ РЕЗУЛЬТАТЫ?

Столбцы означают **MD5 / size / mode / owner**.
Одинокая **5** — изменено содержимое файла.
Игнорируй предупреждение **missing** для man-страниц.

