

# Руководство по пентесту и защите от киберугроз на Linux и Kali Linux

**Целевая аудитория:** системные администраторы и опытные пользователи Linux.

## Оглавление

1. Введение
2. Анализ вредоносного ПО (теория, инструменты, примеры)
  3. 2.1 Теоретические основы
  4. 2.2 Инструменты анализа и обнаружения
  5. 2.3 Примеры и практические рекомендации
6. **Защита серверов** (брандмауэры, SELinux/AppArmor, SSH-hardening, Fail2Ban и пр.)
  7. 3.1 Настройка брандмауэра (Firewall)
  8. 3.2 Усиленная конфигурация SSH
  9. 3.3 Системы принудительного контроля доступа (SELinux и AppArmor)
10. 3.4 Защита от брутфорса: Fail2Ban
11. 3.5 Дополнительные меры защиты
12. **Форензика** (поиск следов, анализ логов, сохранение образов и т.д.)
  13. 4.1 Сохранение образов и данных для расследования
  14. 4.2 Поиск следов взлома в системе
  15. 4.3 Анализ логов
  16. 4.4 Анализ образов дисков и памяти

## 1. Введение

Безопасность Linux-систем достигается сочетанием профилактических мер и умения реагировать на инциденты. Данное руководство охватывает **пентест и защиту от киберугроз** – от предотвращения атак путем правильной настройки системы до выявления и анализа вредоносного ПО и следов компрометации. Мы рассмотрим как **инструменты обнаружения вредоносного ПО**, так и **методы укрепления (hardening) серверов**, а также основы **цифровой криминалистики (форензики)** для расследования инцидентов безопасности.

Настройка Linux-сервера должна выполняться с учётом принципов «защиты в глубину» – то есть использования нескольких уровней безопасности. Даже при высокой общей защищённости, необходимо уметь обнаруживать проникновения: Linux подвержен вредоносному ПО не меньше, чем другие системы, хотя его архитектура и модель прав доступа затрудняют массовое распространение вирусов. Тем не менее, существуют *руткиты*, *черви*, *майнеры* и другие угрозы для Linux, поэтому актуальны как **сканирование на наличие вредоносных программ**, так и **регулярное аудирование конфигурации**.

**Kali Linux** – специализированный дистрибутив для пентестинга – предоставляет множество инструментов, которые пригодны и для задач защиты. В этом гайде мы будем опираться на открытые решения, доступные как в Kali, так и в других дистрибутивах.

## 2. Анализ вредоносного ПО

Анализ вредоносного ПО на Linux включает понимание того, как различные типы вредоносных программ работают в этой среде, и использование инструментов для их обнаружения и исследования. В данном разделе сначала рассмотрены общие сведения о вредоносном ПО на Linux, затем перечислены **открытые инструменты** для его обнаружения и анализа, после чего приведены практические примеры использования этих инструментов.

### 2.1 Теоретические основы

**Вредоносное программное обеспечение (malware)** – это общее название для программ, созданных с целью выполнения нежелательных или вредоносных действий. В контексте Linux наиболее распространены следующие виды угроз:

- **Руткиты:** наборы программ, позволяющие злоумышленнику получить привилегированный (root) доступ и скрыть своё присутствие. Руткиты часто модифицируют системные утилиты или ядро с целью маскировки процессов и файлов.
- **Трояны и backdoors:** вредоносные программы, маскирующиеся под легитимные и предоставляющие удалённый доступ или выполняющие вредоносные функции (например, кража данных).
- **Черви и скрипты-вредители:** автономно распространяющиеся угрозы (особенно в среде серверов), эксплуатирующие уязвимости служб. На практике в Linux часто встречаются вредоносные bash-скрипты или Python-скрипты, загруженные злоумышленником после получения доступа.
- **Прочее ПО:** майнеры криптовалют (использующие ресурсы системы скрытно), логиеры, боты для DDoS – любые нежелательные программы.

**Особенности анализа в Linux:** Вредоносное ПО на Linux может распространяться не так массово, как на Windows, но зачастую нацелено на серверы. Для успешного противодействия важно оперативно *выявлять аномалии* (необычные процессы, сетевую активность, новые файлы) и проводить сканирование системы на наличие известных сигнатур. Анализ malware делится на *статический* (исследование файлов вредоносного ПО, например, с помощью дизассемблеров или утилиты `strings`) и *динамический* (запуск подозрительного объекта в изолированной среде – sandbox – либо отслеживание его поведения через системные мониторы). Однако в боевых условиях администратору чаще требуется **быстро просканировать систему** открытыми утилитами на наличие известных угроз и бэкдоров.

### 2.2 Инструменты анализа и обнаружения

Для Linux разработано несколько **открытых (open-source) инструментов**, позволяющих обнаруживать вредоносное ПО и признаки компрометации. Рассмотрим наиболее популярные из них:

- **Lynis** – продвинутый аудит безопасности и поиск rootkit'ов. Lynis выполняет глубокое сканирование системы: проверяет конфигурацию, целостность файлов, права, брандмауэр, наличие известных уязвимостей и руткитов <sup>1</sup>. Он **не исправляет** проблемы автоматически, но дает подробный отчёт и рекомендации по усилению защиты. Пример использования: `lynis audit system` выполнит полный аудит системы. Lynis часто используют для регулярных проверок (его можно запускать по cron), получая отчёты на почту <sup>2</sup>. Эта утилита полезна на этапе выявления проблем (например, открытых портов, слабых настроек), что помогает косвенно обнаружить и следы вредоносной активности.

- **Chkrootkit** – легковесный *детектор руткитов*. Представляет собой набор скриптов и утилит, которые ищут на системе признаки присутствия руткитов <sup>3</sup>. Chkrootkit проверяет целостность важных системных бинарных файлов, наличие скрытых процессов, модулей ядра, а также запускает тесты (например, `chkproc`, `chkdirs`) для обнаружения скрытых вещей. Утилиту можно установить из репозитория (например, `sudo apt install chkrootkit` на Debian/Ubuntu) и затем запустить командой `chkrootkit` от имени root. Вывод утилиты сообщит о найденных подозрительных артефактах или сообщит “not infected” для каждого теста. Например, проверка на наличие известных руткитов LKM может показать предупреждение, если обнаружены аномальные значения в ядре (см. пример на скриншоте ниже). Учтите, что **наличие предупреждений** не всегда означает компрометацию – требуется дальнейший анализ вручную.

```

root@localhost:~# chkrootkit
sniffer v050808 w50808 scalper slapper z2 chutmp OSX RSPUG and basenane hiff
chfn chsh cron crontab date du dirname echo egrep env find fingers gpm
grep hdparm su ifconfig inetd inetdconf identd init killall lddpreload
login ls lsdf mail minigetty netstat named passwd pidof pop2 pop3 ps
pstree rcinfo rlogind rshd slogin sendmail sshd syslogd tar tcpd
tcpdump top telnetd timed traceroute wdir w write
ROOTDIR is '/'
Checking 'allens'... no suspect files
Searching for sniffer's logs, it may take a while... nothing found
Searching for rootkit Hldrootkit's default files... nothing found
...
Checking 'lkm'...
expected 240322 value
chkproc: Warning: Possible LKM Trojan installed
chkdirs: nothing detected
Checking 'resids'... not found
Checking 'sniffer'... ls: not promise
and no packet sniffer sockets eth0: PACKET SNIFFER(/usr/sbin/dhccpd[1414],
/usr/sbin/dhccpd[1414])
Checking 'w50808'... not infected
Checking 'w50808'... 1 deletion(s)
between Sun Nov 15 23:24:31 2020 and Mon Nov 16 11:45:21 2020
1 deletion(s) between Thu Jan 21 11:39:20 2021 and Thu Jan 21 22:55:00 2021
Checking 'scalper'... not infected
Checking 'slapper'... not infected
Checking 'z2'... chklastlog: nothing deleted
The tty of the following
user process(es) were not found
in /var/run/utmp:
PID TTY CMD
1 root 3668 tty8 daemon --foreground --respaun --attempts=20
--delay=10 --name=0 -var -log -mail.log -pidfiles/run/console-log/
Debian-console-log/B -var -log -mail.log -user Debian-console-log-adm
/usr/share/console-log/logpaper --less /var/log/mail.log 7000000
...
chkutmp: nothing deleted
Checking 'OSX.RSPUG'... not tested
Checking 'and'... not found
Checking 'basenane'... not infected
Checking 'hiff'... not found
Checking 'chfn'... not infected
...
Checking 'telnetd'... not found
Checking 'timed'... not found
Checking 'traceroute'... not infected
Checking 'wdir'... not infected
Checking 'w'... not infected
Checking 'write'... not infected
...
root@localhost:~# chkrootkit
sniffer v050808 w50808 scalper slapper z2 chutmp OSX RSPUG and basenane hiff
chfn chsh cron crontab date du dirname echo egrep env find fingers gpm
grep hdparm su ifconfig inetd inetdconf identd init killall lddpreload
login ls lsdf mail minigetty netstat named passwd pidof pop2 pop3 ps
pstree rcinfo rlogind rshd slogin sendmail sshd syslogd tar tcpd
tcpdump top telnetd timed traceroute wdir w write
ROOTDIR is '/'
Checking 'allens'... not infected
Checking 'cron'... not infected
Checking 'su'... not infected
Checking 'ifconfig'... not infected
Checking 'inetd'... not infected
Checking 'inetdconf'... not infected
Checking 'identd'... not found
Checking 'rshd'... not found
Checking 'slogin'... not infected
Checking 'sendmail'... not infected
Checking 'sshd'... not found

```

Вывод примера сканирования с помощью Chkrootkit: утилита проверяет систему на признаки известных руткитов и аномалий (фрагмент отчёта).

- **Rkhunter (Rootkit Hunter)** – ещё один популярный сканер руткитов и бэкдоров. Rkhunter тщательно проверяет систему на наличие скрытых уязвимостей, известных сигнатур руткитов, бэкдоров и локальных эксплойтов <sup>4</sup>. Он сканирует важные бинарные файлы (сравнивая их хэши), смотрит права файлов, ищет подозрительные строки в модулях ядра и скрытые файлы. Установка (через `apt install rkhunter` или аналог) и запуск (`rkhunter -c`) просты. После сканирования утилита выдаёт отчёт с пометками `[ OK ]` для пройденных тестов и `[ Warning ]` для потенциально опасных находок. Все детали также пишутся в лог `/var/log/rkhunter.log` для дальнейшего изучения. **Rkhunter** рекомендуется использовать совместно с Chkrootkit для большего охвата проверок <sup>5</sup>. Примечание: некоторые предупреждения могут оказаться ложными срабатываниями (например, изменённые легитимные файлы), поэтому анализировать результаты нужно внимательно.

```
root@localhost: /home/rtoribek# rkhunter -c
[ Rootkit Hunter version 1.4.6 ]

Checking system commands...
Performing 'strings' command checks
Checking 'strings' command [ OK ]

Performing 'shared libraries' checks
Checking for preloading variables [ None found ]
Checking for preloaded libraries [ None found ]
Checking LD_LIBRARY_PATH variable [ Not found ]

Performing file properties checks
Checking for prerequisites [ OK ]
/usr/sbin/adduser [ OK ]
/usr/sbin/ckroot [ OK ]
/usr/sbin/cron [ OK ]
/usr/sbin/groupadd [ OK ]
/usr/sbin/groupdel [ OK ]
/usr/sbin/groupmod [ OK ]
/usr/sbin/grpck [ OK ]
/usr/sbin/lftstatus [ Warning ]
...
/usr/bin/lwp-request [ Warning ]

Performing malware checks
Checking running processes for suspicious files [ None found ]
Checking for login backdoors [ None found ]
Checking for sniffer log files [ None found ]
Checking for suspicious directories [ None found ]
Checking for suspicious (large) shared memory segments [ Warning ]

Performing filesystem checks
Checking /dev for suspicious file types [ Warning ]
Checking for hidden files and directories [ Warning ]

Rootkit checks...
Rootkits checked : 484
Possible rootkits: 10

Applications checks...
All checks skipped

The system checks took: 2 minutes and 1 second

All results have been written to the log file: /var/log/rkhunter.log

One or more warnings have been found while checking the system.
Please check the log file (/var/log/rkhunter.log)
root@localhost: /home/rtoribek# cat /var/log/rkhunter.log
[08:26:22] Running Rootkit Hunter version 1.4.6 on localhost
[08:26:22] Info: Start date is Thu 18 Feb 2021 08:26:22 AM MSK
[08:26:22]
[08:26:22] Checking configuration file and command-line options...
[08:26:22] Info: Detected operating system is 'Linux'
[08:26:22] Info: Found O/S name: Progress Linux 6+ (fuchur-backports)
[08:26:22] Info: Command line is /usr/bin/rkhunter -c
[08:26:22] Info: Environment shell is /bin/dash, rkhunter is using dash
[08:26:22] Info: Using configuration file '/etc/rkhunter.conf'
[08:26:22]
[08:26:22] The system checks took: 2 minutes and 1 second
[08:26:22]
[08:26:22] Info: End date is Thu 18 Feb 2021 08:28:25 AM MSK
```

Пример запуска Rkhunter: сканирование системы на руткиты и бэкдоры. Видны результаты проверок (OK или Warning) по разным категориям. 4

- **ClamAV** – кроссплатформенный антивирусный движок с открытым исходным кодом 6. Он обнаруживает вирусы, трояны и прочее вредоносное ПО на Linux. ClamAV чаще всего применяют для сканирования файловых хранилищ и почтовых серверов, но его можно использовать и для проверки серверов на наличие заражённых файлов. Установка (`apt install clamav clamav-daemon`) добавляет демон для фоновой сканирования. После установки необходимо обновить базы сигнатур (`freshclam`), затем можно сканировать систему, например, командой `clamscan -r -i /path/to/scan` (рекурсивно, выводить только заражённые файлы) 7. ClamAV поддерживает множество архивных форматов, поэтому способен проверить даже упакованные файлы. Важно понимать, что ClamAV в основном **сигнатурный сканер** – его эффективность зависит от актуальности баз и наличия сигнатур для конкретного вредоноса. Тем не менее, это один из стандартных инструментов антивирусной проверки на Linux. Пример: регулярное ночное сканирование веб-директории на сервере с отчетом на e-mail.

- **Linux Malware Detect (LMD)** – также известен как Maldet, специализированный сканер вредоносных файлов в Linux. Разработан для веб-серверов и хостинговых сред. LMD может работать совместно с ClamAV, дополняя его своими сигнатурами 8. Он ищет специфичные вредоносные веб-скрипты, шеллы (типа `c99.php`), бэкдоры, которые часто загружаются на взломанные сайты. LMD не входит в стандартные репозитории многих дистрибутивов; его устанавливают вручную, скачав архив с официального сайта (скрипт `install.sh`). После установки командой `maldet` можно сканировать систему или настроить мониторинг каталогов в реальном времени. Пример: `maldet -a /var/www` просканирует веб-файлы на известные веб-шеллы. Отчёты LMD можно настроить на отправку администратору.

- **YARA** – не упомянута напрямую, но стоит знать: утилита для описания сигнатур вредоносного ПО в виде правил. Опытные исследователи могут использовать YARA-правила для поиска специфичных образцов malware по характерным строкам/шаблонам. Инструмент open-source, и множество правил находятся в открытом доступе. YARA может

помочь в **поиске семейств вредоносных программ**, если вы получили подозрительный файл или память процесса.

- **Дополнительные инструменты:** В составе Kali Linux присутствуют и другие средства анализа: например, **Radare2** (дизассемблер/отладчик для продвинутого анализа бинарей), **strace/ltrace** (для отслеживания системных вызовов и библиотечных вызовов подозрительного процесса), **binwalk** (для исследования бинарных файлов и прошивок) и т.д. Эти инструменты требуют отдельных навыков и применяются при детальном разборе конкретного образца вредоносного ПО, но для общей безопасности сервера первоочередная задача – выявить факт заражения и очистить систему.

## 2.3 Примеры и практические рекомендации

**Регулярное сканирование сервера:** Администраторам рекомендуется внедрить плановые проверки системы на malware. Например, настроить ежедневный запуск **rkhunter** и **chkrootkit** по крону с отправкой отчёта на почту <sup>9</sup> <sup>10</sup>. Аналогично, можно раз в неделю запускать **lynis audit system** для общего аудита. Такие проактивные меры помогут рано заметить подозрительные изменения (например, появление скрытого пользователя или изменения в конфигурации SSH).

**Обновление баз сигнатур:** Для инструментов, основанных на сигнатурах (ClamAV, LMD), важно регулярно обновлять базы. У ClamAV есть демон **freshclam**, который можно включить для автоматического обновления сигнатур. Без обновлений антивирус пропустит новые угрозы.

**Анализ подозрительных процессов:** Если вы заметили незнакомый процесс, потребляющий ресурсы (CPU/RAM) или установили факт взлома, проведите экспресс-анализ:

- Посмотрите, откуда запущен процесс: **ls -l /proc/<PID>/exe** даст путь до исполняемого файла, **cat /proc/<PID>/cmdline** – командную строку.
- Проверьте сетевую активность: **ss -plntu** покажет порты и соответствующие PID процессов; **lsof -p <PID>** перечислит открытые файлы (модули, сокеты) процесса.
- Если процесс явно вредоносный, *снимите его дамп памяти* для дальнейшей форензики (см. раздел 4) или хотя бы сохраните исполняемый файл на анализ, прежде чем убить процесс. В изолированной среде (например, на другой машине или внутри Docker) можно запустить этот файл через **strace** для наблюдения за его действиями (сеть, файлы).

**Использование песочницы:** Для продвинутого динамического анализа вредоноса можно задействовать песочницу типа **Cuckoo Sandbox** (open-source). Она позволяет запускать образец в виртуализированной среде (включая Linux-гостя) и наблюдать его поведение: системные вызовы, изменения файлов, трафик. Однако развёртывание песочницы – отдельная сложная задача, поэтому на уровне администрирования чаще ограничиваются сбором артефактов и передачей их специалистам по безопасности.

**Доверие к утилитам:** Имейте в виду, что на скомпрометированной системе результаты **chkrootkit/rkhunter** могут быть **скомпрометированы**, если злоумышленник установил продвинутый руткит, скрывающий своё присутствие. В идеале, для полного анализа используют *внешние носители*: например, загрузившись с Live-CD типа Kali/CAINE и запустив сканирование уже на нерабочей системе (offline-сканирование диска на malware). Это исключает влияние активного руткита.

**Реакция на обнаружение:** Если сканеры нашли конкретные заражённые файлы (например, скрипт-шелл в /tmp или модифицированный системный бинарь), необходимо: 1) изолировать систему (отключить от сети, если возможно), 2) как можно скорее обновить/переустановить затронутые пакеты системы из доверенных источников, 3) проанализировать, как malware попал на сервер (чтобы закрыть эту брешь), 4) проверить систему на закладки (backdoors) – например, неизвестных пользователей, задачи cron, автозагрузки (об этом – в разделе форензики).

---

### 3. Защита серверов

В этом разделе описаны практические шаги по **усилению безопасности Linux-сервера**. Цель hardening-a – максимально затруднить компрометацию, внедрив многоуровневую защиту: от базовых настроек (как SSH) до расширенных механизмов ядра (SELinux/AppArmor) и мониторинга. Мы рассмотрим настройку сетевого экрана (firewall), ограничение доступа по SSH, использование **MAC (Mandatory Access Control)** систем и средств предотвращения атак типа **Fail2Ban**, а также другие рекомендации.

Безопасность сервера начинается ещё **до** установки специальных средств: с минимизации доступных сервисов и своевременного применения обновлений. Предполагается, что сервер актуализирован (патчи безопасности установлены) и на нём отключены или удалены все неиспользуемые службы. Ниже перечисленные меры накладываются поверх базовой кибергиены.

#### 3.1 Настройка брандмауэра (Firewall)

**Брандмауэр (сетевой экран)** контролирует сетевые подключения, разрешая или запрещая трафик по определённым портам, протоколам и адресам. В Linux роль фаервола выполняет подсистема **Netfilter** ядра, для конфигурирования которой традиционно используется утилита **iptables** (в современных системах – её наследник **nftables**). Многие дистрибутивы предоставляют упрощённые интерфейсы: например, **UFW (Uncomplicated Firewall)** в Ubuntu/Debian или **Firewalld** в CentOS/Fedora – они позволяют управлять правилами через удобные команды или даже GUI.

Основные задачи при настройке firewall на сервере: - **Заккрытие всех неиспользуемых портов.** По умолчанию политика должна быть «всё запрещено, кроме разрешённого». Правила фаервола обычно настраивают так, чтобы из внешней сети были доступны только необходимые сервисы (например, 22/SSH, 80/443/HTTP(S), 25/SMTP – в зависимости от роли сервера). - **Разграничение входящего и исходящего трафика.** Как минимум, нужно фильтровать входящие соединения. В некоторых случаях имеет смысл ограничить и исходящие (например, серверу не нужно устанавливать внешние соединения на порты, не связанные с его функциями – это может затруднить действия злоумышленника при компрометации). - **Защита от сканирования и DoS.** Netfilter позволяет добавлять ограничение по количеству подключений (rate limiting) для защиты от элементарных DoS-атак. Например, `iptables -A INPUT -p tcp --dport 22 -m connlimit --connlimit-above 5 -j DROP` ограничит число одновременных SSH-соединений от одного IP. - **Логирование подозрительного трафика.** С помощью таргета **LOG** (iptables) или метки **log** (nftables) можно протоколировать пакеты, которые фаервол отклоняет. Анализ этих логов поможет в форензике (см. раздел 4).

**Пример:** Настройка iptables вручную:

```

iptables -P INPUT DROP      # политика по умолчанию - сбрасывать входящие
iptables -P OUTPUT ACCEPT   # исходящий трафик разрешен (можно ограничить
                              при необходимости)
iptables -P FORWARD DROP    # транзит не нужен - запрещаем

# Разрешаем уже установленные соединения и loopback
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT

# Открываем SSH (порт 22) только для определенного IP (пример)
iptables -A INPUT -p tcp --dport 22 -s 203.0.113.5 -m state --state NEW -j
ACCEPT

# Открываем HTTP/HTTPS для всех
iptables -A INPUT -p tcp -m multiport --dports 80,443 -m state --state NEW -
j ACCEPT

# Логируем все остальное (префикс "IPTABLES DROP:")
iptables -A INPUT -j LOG --log-prefix "IPTABLES DROP: " --log-level 4

```

В этом примере все неописанные пакеты будут отфильтрованы. Правило с LOG позволит видеть в `/var/log/syslog` обращения на закрытые порты.

#### UFW пример:

Для тех, кто предпочитает UFW (если он доступен), эквивалентные настройки могли бы быть:

```

ufw default deny incoming
ufw default allow outgoing
ufw allow 22/tcp      # SSH
ufw allow 80/tcp      # HTTP
ufw allow 443/tcp     # HTTPS
ufw enable

```

UFW автоматически применит правила через iptables/nftables. Он удобен простотой, однако для тонкой настройки (например, ограничение по IP или продвинутые опции) администратору нужно знать базовые команды iptables.

**Firewalld (зональная модель):** В RHEL/Fedora используется демон firewalld, который позволяет применять правила к *зонам* (например, `public`, `internal`). Команды `firewall-cmd` позволяют открывать порты в нужной зоне. Концепция схожа: минимизировать поверхность атаки, открыв лишь необходимые направления трафика.

Важно помнить, что **firewall – первая линия обороны**. Правильно настроенный брандмауэр предотвратит множество простых атак (сканирование портов, попытки подключения к службам, которые не должны быть доступны). Однако, если злоумышленник уже запустил вредонос на сервере, firewall не обезвредит его – тут вступают в силу другие средства (IDS, антивирусы, мониторинг).

## 3.2 Усиленная конфигурация SSH

Сервис **SSH** обычно является основным методом удалённого администрирования Linux-серверов, и, соответственно, главной мишенью для атак типа перебора паролей (brute force) и эксплойтов.

**Hardening SSH** включает ряд мер:

- **Запрет входа под root:** Отключить прямую возможность удалённого входа суперпользователя. Вместо этого администраторы должны входить под своими учетными записями и повышать права через `sudo`. В файле конфигурации `/etc/ssh/sshd_config` параметр `PermitRootLogin` следует выставить в `no`<sup>11</sup>. Это предотвращает бесчисленные попытки подобрать пароль к root – даже если пароль теоретически слабый, подключиться напрямую нельзя.
- **Аутентификация по ключам:** Включить вход по паре ключей (SSH public key) и **отключить аутентификацию по паролю**. Параметр `PasswordAuthentication no` в `sshd_config` закрывает возможность логина по паролю – даже если злоумышленник узнает пароль, без приватного ключа он не войдет. Ключи должны быть достаточно длинными (минимум 2048-bit RSA или лучше Ed25519), закрытые ключи хранить в безопасности. Для дополнительной защиты можно защитить закрытый ключ паролем-фразой.
- **Ограничение по IP и пользователям:** Если известно, что в SSH будут входить только с определённых IP-адресов (например, офис, VPN), можно ограничить доступ брандмауэром или самим SSH. С помощью параметра `AllowUsers` / `AllowGroups` можно задать конкретных пользователей или группы, которым разрешён вход<sup>11</sup>. Остальные попытки будут отвергаться сразу при аутентификации. Также можно указать `MaxAuthTries 3` (число попыток ввода пароля) и `LoginGraceTime 60` (таймаут на логин) для усложнения брутфорса.
- **Смена порта SSH:** Нет, это не панацея, но изменение порта по умолчанию (22) на нестандартный (например, 2022) может отсеять часть автоматических бот-сканеров. Однако целенаправленного злоумышленника это не остановит (он просканирует порты). Поэтому на смену порта нельзя полагаться как на серьёзную защиту, но иногда она снижает «шум» в логах. Если порт сменён, обязательно отразите это в правилах firewall (открыть новый порт) и документируйте для команды.
- **SSH протокол и алгоритмы:** Убедитесь, что используется современный протокол SSHv2 (в старых конфигурациях можно встретить опцию `Protocol 2`, SSHv1 давно небезопасен). Также стоит отключить устаревшие алгоритмы шифрования и хэширования. Например, исключить CBC-шифры и слабые MAC: можно настроить `Ciphers` (оставив только сильные, типа `aes256-gcm@openssh.com`) и `MACs` (`hmac-sha2-512` и т.п.). OpenSSH обычно по умолчанию уже достаточно безопасен, но проверить лишним не будет.
- **Двухфакторная аутентификация (2FA):** Для действительно важных систем рассмотрите установку 2FA для SSH. Например, модуль **Google Authenticator** (libpam-google-authenticator) или использование аппаратных ключей (U2F/FIDO). При этом вход по SSH потребует одноразовый код с телефона или наличие USB-ключа. Это существенно повышает безопасность, хотя и добавляет сложности при настройке.



После любых изменений в `/etc/ssh/sshd_config` не забудьте перезапустить `sshd` и протестировать вход. Рекомендуется иметь текущую сессию открытой, пока проверяете новый, чтобы не отрезать себя из-за опечатки.

**Пример hardened-конфигурации** `/etc/ssh/sshd_config`:

```
Port 2022
Protocol 2
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
PermitEmptyPasswords no
ChallengeResponseAuthentication no    # Отключаем keyboard-interactive (если
не используем OTP)
MaxAuthTries 3
LoginGraceTime 30s
AllowUsers alice bob
```

Эти настройки: переносят SSH на порт 2022, отключают root и пароли, разрешают вход только по ключам для пользователей alice и bob, ограничивают попытки и время ожидания.

### 3.3 Системы принудительного контроля доступа (SELinux и AppArmor)

Помимо стандартной модели безопасности Linux (DAC – Discretionary Access Control, основанной на UNIX-разрешениях пользователя/группы), существуют **модели принудительного контроля доступа (MAC)**, реализованные в ядре Linux. Две наиболее распространённые системы – **SELinux** и **AppArmor** – позволяют создавать более строгие политики безопасности, ограничивая действия даже привилегированных процессов.

- **SELinux (Security-Enhanced Linux)**: разработан Агентством нац.безопасности США (NSA) и внедрён в ядро Linux как LSM (Linux Security Module). SELinux вводит понятие **контексты безопасности** для каждого процесса и объекта (файла, порта и т.д.) и политику, которая определяет, какие действия разрешены. По сути, SELinux реализует мандатное разграничение доступа на основе меток. В SELinux существует несколько режимов работы: *Enforcing* (политика применяются, запрещённые операции блокируются), *Permissive* (только логирование нарушений) и *Disabled*. Для боевых серверов рекомендуется Enforcing-режим. Многие дистрибутивы (Red Hat/CentOS, Fedora) идут с SELinux Enforcing по умолчанию.

Политики SELinux довольно сложны; обычно используются преднастроенные политики (типа Targeted Policy, которая защищает ключевые демоны). Например, веб-сервер Apache под SELinux даже с правами root не сможет читать любые файлы, кроме помеченных как `httpd_content_t`. Если злоумышленник скомпрометирует Apache, SELinux не даст ему выйти из “песочницы” веб-сервера (если настроено правильно). **Применение:** убедитесь, что SELinux включён (`sestatus` покажет статус). Если он был выключен, включите (в `/etc/selinux/config` выставить `SELINUX=enforcing` и перезагрузить). Но будьте готовы разрешать законные активности, которые политика блокирует – изучайте логи (`auditd`) и используйте утилиту `audit2allow` для генерации локальных исключений, либо настраивайте Booleans (`setsebool`) для стандартных

случаев. SELinux требует некоторого времени на освоение, но значительно повышает стойкость системы к атакам, изолируя службы друг от друга <sup>12</sup>.

- **AppArmor**: альтернативный MAC фреймворк, используемый по умолчанию в Ubuntu, Debian (и в некоторых других). В отличие от SELinux, который базируется на метках, AppArmor использует профили, привязанные к именам исполняемых файлов (по путям). Профиль AppArmor описывает, какие ресурсы (файлы, сеть, capability и т.д.) разрешено использовать процессу. AppArmor проще в начальном освоении: профили пишутся текстом, часто генерируются утилитой `aa-genprof` на основе наблюдения за реальной работой программы. Например, можно создать профиль для `/usr/sbin/nginx`, который позволит читать `/var/www/html`, открывать сокет 80, но запретит запускать shell или читать чужие файлы.

AppArmor так же имеет режимы Enforce (применение) и Complain (только логирование нарушений). Проверить статус можно командой `aa-status`. В Ubuntu, как правило, уже есть профили для основных сервисов (например, для `sshd`, `apache2`, `mysql` и т.п.). Следует убедиться, что **AppArmor включён** (в `/etc/apparmor.d` есть конфиги, и сервис `apparmor` активен). При необходимости – написать/подправить профили под специфичные приложения, которые вы устанавливаете. Например, если вы разрабатываете собственный демон, разумно ограничить его профиль AppArmor'ом.

**SELinux vs AppArmor**: Оба дополняют друг друга не могут – обычно система использует что-то одно (в ядре SELinux и AppArmor могут быть включены одновременно, но дистрибутивы выбирают). SELinux считается более **тонконастраиваемым и всеобъемлющим**, но сложным. AppArmor – более **простым в управлении**, но чуть менее гибким. Главное – *включить хотя бы один* MAC. Даже в Permissive режиме MAC-системы полезны, т.к. логируют подозрительные действия (например, если вредонос пытается сделать что-то запрещённое политикой, это появится в логах audit). В Enforcing режиме эти действия ещё и блокируются, что может предотвратить эскалацию атаки.

**Практический совет**: При эксплуатации SELinux/AppArmor, регулярно просматривайте логи (`audit.log` для SELinux или `syslog/dmesg` для AppArmor) на предмет *denied* сообщений. Legitimные сервисы иногда блокируются излишне (например, SELinux не дал Nginx подключиться к внешнему API – нужно включить boolean `httpd_can_network_connect`), и эти случаи надо устранять корректно, а не просто отключая SELinux. Но такие логи также могут указать на попытки вредоносной активности (например, malware попытался вскрыть память другого процесса – SELinux заблокировал и залогировал AVC denial). Таким образом MAC системы полезны и для **детектирования** атак.

### 3.4 Защита от брутфорса: Fail2Ban

**Fail2Ban** – популярный open-source инструмент для защиты сервисов от атак методом перебора паролей (brute force) и других типов злоупотреблений. Принцип работы Fail2Ban прост: он **мониторит логи** выбранных сервисов (SSH, FTP, веб-сервер, почтовый сервер и т.д.) и при обнаружении большого числа неудачных попыток или подозрительной активности – *автоматически вносит IP-адрес нарушителя в блокировку* (обычно через правила firewall) на заданное время. Таким образом, Fail2Ban действует как динамический бан-лист, реагируя на проявления атаки.

Основные компоненты Fail2Ban – это *фильтры* (описывают, какие лог-сообщения считать “триггером”) и *“джейлы” (jails)* – сочетание фильтра и действия бана. Например, есть готовый

фильтр для SSH (анализирует `/var/log/auth.log` на предмет «Failed password»), и jail, который при 5 неудачных попытках за 10 минут банит IP на час. Администратор может настраивать параметры: количество попыток (maxretry), период отслеживания (findtime), длительность бана (bantime) и т.д.

**Установка:** Fail2Ban написан на Python и доступен практически во всех дистрибутивах. Для установки на Debian/Ubuntu: `apt install fail2ban`, на CentOS: установить EPEL репозиторий и затем `yum install fail2ban`<sup>13</sup>. После установки – запустить сервис `systemctl enable --now fail2ban`. Fail2Ban сразу начнёт мониторинг базовых сервисов, но по умолчанию часто активирован только SSH. Конфигурация находится в `/etc/fail2ban/`. Основной файл – `jail.conf`, но его лучше не править напрямую; принято создавать `jail.local` с переопределениями.

**Настройка SSH-защиты:** В `jail.local` обычно достаточно включить `[sshd]` jail (если не включён) и задать параметры. Например:

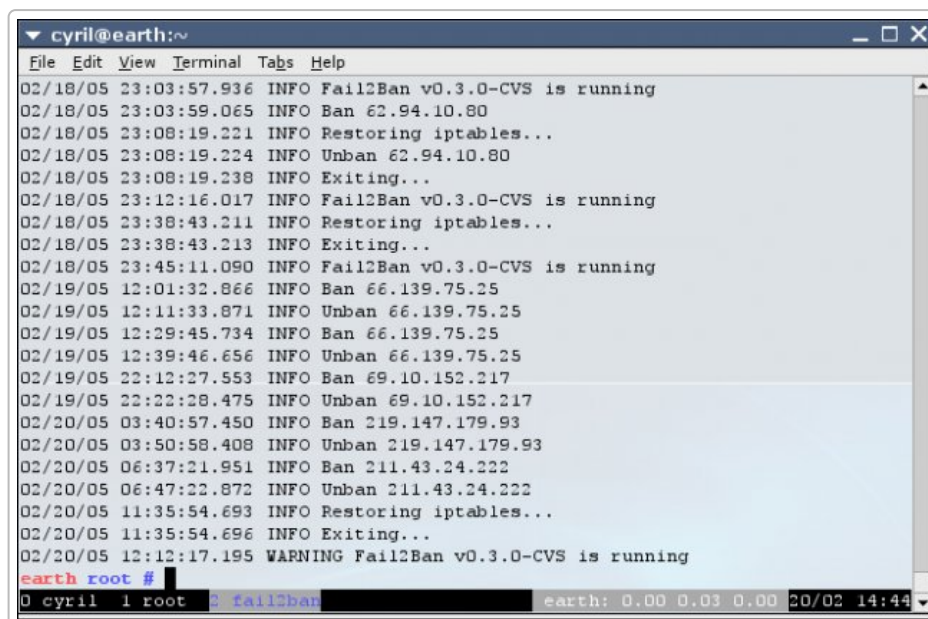
```
[sshd]
enabled = true
port    = 222
filter  = sshd
maxretry = 5
findtime = 10m
bantime = 1h
```

Здесь мы включаем защиту SSH (на порт 222, если вы сменили порт), допускаем 5 ошибок логина в 10-минутный интервал, после чего IP банится на час. Бан осуществляется добавлением правила в iptables (Fail2Ban делает это автоматически через `iptables -action`).

**Защита других сервисов:** Fail2Ban имеет десятки фильтров для различных приложений: `apache-auth` (неудачные BasicAuth на вебе), `nginx-http-auth`, `vsftpd`, `dovecot` (почта), даже `sshd-ddos` (отдельный фильтр для обнаружения подозрительной активности SSH, указывающей на DoS). Чтобы активировать, достаточно в `jail.local` создать секцию с именем нужного jail и `enabled = true`. Например, чтобы банить IP, которые сканируют несуществующие URL на веб-сервере (404), можно включить `apache-badbots`.

**Monitoring и разблокировка:** Чтобы увидеть текущий статус, используйте: `fail2ban-client status` – покажет активные jail'ы и общее число заблокированных IP. А `fail2ban-client status sshd` – детализацию по конкретному jail (сколько попыток, какие IP забанены). Если необходимо разбанить IP до истечения срока, можно выполнить `fail2ban-client set sshd unbanip 203.0.113.45`.

**Пример работы Fail2Ban:** После внедрения Fail2Ban, в логах (`/var/log/auth.log`) при брутфорсе SSH вы сначала увидите привычные "Failed password for user ...", а затем запись Fail2Ban о бане IP. В `/var/log/fail2ban.log` будет строка вида `Ban 203.0.113.45` для jail sshd. Брандмауэрное правило (можно увидеть через `iptables -L`) будет блокировать этот IP на порт 22. Через час (по bantime) Fail2Ban снимет бан автоматически (или не снимет, если bantime было указано как `-1` для бессрочного бана).



```
cyril@earth:~
File Edit View Terminal Tabs Help
02/18/05 23:03:57.936 INFO Fail2Ban v0.3.0-CVS is running
02/18/05 23:03:59.065 INFO Ban 62.94.10.80
02/18/05 23:08:19.221 INFO Restoring iptables...
02/18/05 23:08:19.224 INFO Unban 62.94.10.80
02/18/05 23:08:19.238 INFO Exiting...
02/18/05 23:12:16.017 INFO Fail2Ban v0.3.0-CVS is running
02/18/05 23:38:43.211 INFO Restoring iptables...
02/18/05 23:38:43.213 INFO Exiting...
02/18/05 23:45:11.090 INFO Fail2Ban v0.3.0-CVS is running
02/19/05 12:01:32.866 INFO Ban 66.139.75.25
02/19/05 12:11:33.871 INFO Unban 66.139.75.25
02/19/05 12:29:45.734 INFO Ban 66.139.75.25
02/19/05 12:39:46.656 INFO Unban 66.139.75.25
02/19/05 22:12:27.553 INFO Ban 69.10.152.217
02/19/05 22:22:28.475 INFO Unban 69.10.152.217
02/20/05 03:40:57.450 INFO Ban 219.147.179.93
02/20/05 03:50:58.408 INFO Unban 219.147.179.93
02/20/05 06:37:21.951 INFO Ban 211.43.24.222
02/20/05 06:47:22.872 INFO Unban 211.43.24.222
02/20/05 11:35:54.693 INFO Restoring iptables...
02/20/05 11:35:54.696 INFO Exiting...
02/20/05 12:12:17.195 WARNING Fail2Ban v0.3.0-CVS is running
earth root #
0 cyril 1 root 2 fail2ban earth: 0.00 0.03 0.00 20/02 14:44
```

Пример работы Fail2Ban: фрагмент лог-файла, где видны сообщения о запуске Fail2Ban, бане и разбане IP-адресов с помощью обновления правил iptables.

Fail2Ban существенно повышает **устойчивость сервера к словарным атакам и сканированию на уязвимости**. Например, боты, пытающиеся массово логиниться по SSH, будут быстро отсеяны. Также Fail2Ban можно настроить для защиты от некоторых видов DoS – например, банить IP, которые создают слишком много соединений к веб-серверу (через соответствующий фильтр и count). Однако, Fail2Ban не спасает от распределённых атак (с разных IP) или очень медленных переборов (when attempts are under threshold). Поэтому он должен сочетаться с другими мерами (например, использованием ключей вместо паролей, см. SSH-hardening выше).

### 3.5 Дополнительные меры защиты

Помимо вышеперечисленного, реализуйте на сервере и другие важные практики безопасности:

- **Регулярное обновление системы:** Большинство атак используют известные уязвимости. Устанавливайте патчи безопасности без промедления. Настройте автоматическое обновление безопасности (unattended-upgrades на Ubuntu/Debian или dnf-automatic на RHEL) либо следите вручную.
- **Минимизация сервисов:** Удалите или выключите все демон-сервисы, которые не используются. Каждый запущенный сервис – потенциальная точка атаки. Проверьте `netstat -tulpn` или `ss -tulpn` – какие порты открыты, и убедитесь, что это ожидаемо. Например, если на сервере не используются принтеры, убедитесь, что CUPS не запущен и т.п.
- **Разделение привилегий:** Пользователи и сервисы должны работать с минимально необходимыми правами. Настройте `sudo` таким образом, чтобы администраторы имели нужные привилегии, но не больше (например, использование ограничения команд в `/etc/sudoers` для некоторых пользователей). Сервисы, запускаемые от отдельных системных пользователей, не должны иметь избыточных прав на файловой системе.

- **Файловая целостность (HIDS):** Рассмотрите установку **AIDE** (Advanced Intrusion Detection Environment) или аналогов. AIDE создаёт базу хэшей критических файлов и может по расписанию проверять, не изменились ли они. Если злоумышленник подменит бинарий `ls` или библиотеку, AIDE это зафиксирует. Lynis, о котором говорилось выше, тоже частично выполняет эту задачу, но специализированные HIDS могут быть более подробными. В связке с системами логирования вы быстро узнаете о подозрительных изменениях.
- **IDS/IPS и мониторинг трафика:** Для сетевого уровня можно внедрить IDS (систему обнаружения вторжений), например **Snort** или **Suricata** – они анализируют сетевой трафик на известные сигнатуры атак. Такие системы требуют отдельного сервера или хотя бы выделенного сетевого интерфейса в режиме promisc. Однако на самом сервере можно использовать **OSSEC** (HIDS) или **Wazuh**, которые собирают логи, мониторят целостность и тоже могут обнаруживать атаки. Если ресурсов и навыков достаточно, эти системы добавляют еще один уровень защиты.
- **Шифрование данных:** Если на сервере хранятся конфиденциальные данные, используйте шифрование дисков или отдельных разделов (LUKS/dm-crypt). Тогда даже при физическом доступе злоумышленник не получит данные в открытом виде. Это больше о защите от кражи оборудования, но в целом – хорошая практика. Также убедитесь, что важные данные передаются по сети только в зашифрованном виде (протоколы HTTPS, SSH, TLS для почты и т.д.).
- **Резервное копирование:** Не совсем мера защиты от взлома, но критически важная для восстановления – регулярные бэкапы. В случае ransomware-атаки или разрушительных действий злоумышленника, только наличие свежих резервных копий позволит быстро восстановить систему. Бэкапы должны храниться отдельно и проверяться.

Наконец, **документируйте** все настройки безопасности и регулярно их пересматривайте. Безопасность – процесс непрерывный. Периодически проводите **пентесты** своего же сервера (или поручайте команде безопасности) с использованием Kali Linux и других инструментов, чтобы проверить эффективность ваших мер. Например, попробуйте с помощью nmap и скриптов выяснить, какие порты открыты, уязвим ли SSH к перебору (hydra), нет ли простых ошибок конфигурации. По итогам – укрепляйте систему.

---

## 4. Форензика (анализ инцидентов)

Если произошёл инцидент – взлом или подозрение на компрометацию – важно правильно провести расследование, не разрушив при этом ценные доказательства. **Цифровая форензика** в Linux охватывает сбор и анализ данных с скомпрометированной системы: дисков, оперативной памяти, лог-файлов, и других артефактов. Цель – понять, что произошло, каким образом, и какие изменения произведены злоумышленником. Ниже рассмотрены ключевые шаги: сохранение копий (образов), поиск следов взлома, анализ логов и исследование полученных образов дисков/памяти.

**Важно:** При подозрении на взлом *минимизируйте изменения* на системе. Не перезагружайте её сразу (можно потерять артефакты в памяти), по возможности перейдите в режим только чтения (уменьшить запись на диск), запускайте утилиты форензики из-под Live-CD или по сети. Каждый ваш шаг может затереть улики,

поэтому действуйте взвешенно. Если инцидент серьёзный, рассмотрите привлечение профессиональных форензик-специалистов.

## 4.1 Сохранение образов и данных для расследования

Первый шаг – **сохранение копий ключевых данных** с компрометированной системы, чтобы затем анализировать их в безопасной обстановке, не рискуя поменять исходные улики. Основные объекты для сохранения (дампирования):

- **Образ диска (HDD/SSD):** Сделайте побитовую копию всех важных дисковых устройств. Используйте классическую утилиту `dd` – она позволяет создавать точные “bit-by-bit” копии разделов или всего диска <sup>14</sup> <sup>15</sup>. Предварительно выведите список разделов командой `fdisk -l` или `lsblk` <sup>16</sup>. Затем, например:

```
dd if=/dev/sda of=/mnt/forensics/sda.img bs=4M conv=fsync,noerror
```

Здесь `noerror` указывает `dd` не прерываться на битых секторах (если диск поврежден), а `fsync` записывать данные сразу. **Важно:** `of` должен указывать на место с достаточным объёмом (в идеале, внешний диск). Если диск большой, можно сжимать на лету: `dd if=/dev/sda | gzip > sda.img.gz`. После создания образа вычислите его хэш (например, `sha256sum sda.img`) и запишите – это подтвердит целостность копии. Анализ диска лучше вести на копии, монтируя образ read-only (например, `mount -o loop,ro sda.img /mnt/compromised`). В сложных случаях, вместо `dd` можно использовать `dcfldd` или `Guymager` (с GUI) – они добавляют удобства вроде автоматического подсчёта хэшей.

- **Дамп оперативной памяти (RAM):** В памяти могут оставаться следы, не сохранившиеся на диске – активные процессы злоумышленника, расшифрованные ключи, фрагменты команд. Снять дамп памяти сложнее, т.к. Linux не даёт просто так читать всю память. Один из подходов – использование модуля ядра **LiME (Linux Memory Extractor)** <sup>17</sup>. LiME можно загрузить (`insmod`), он создаст специальное устройство (например, `/dev/pmem`), из которого `dd` сможет вычитать содержание RAM <sup>18</sup> <sup>19</sup>. Также существуют утилиты `avml` (Azure VM Memory scraper) или старый `/dev/fmem` модуль. Если таких инструментов под рукой нет, можно попробовать:

```
dd if=/proc/kcore of=/mnt/forensics/ram.dump bs=1M
```

но `/proc/kcore` – ограничен и может не содержать всего. Некоторые дистрибутивы предоставляют `/dev/crash` или `virsh dump` (если VM). **Важно:** Снимать дамп памяти надо *до выключения* машины. После перезагрузки все данные ОЗУ будут потеряны. Полученный дамп памяти (`.dump` или `.raw`) можно потом анализировать инструментами вроде Volatility (см. раздел 4.4).

- **Сетевые соединения (при необходимости):** Если атака продолжается или есть подозрительный трафик – целесообразно **захватить сетевой трафик**. Например, запустить `tcpdump` на интерфейсе и сохранить пакетный дамп:

```
tcpdump -i eth0 -w /mnt/forensics/network.pcap
```

Это позволит потом, в спокойной обстановке, изучить трафик (например, с помощью Wireshark) – выяснить, к каким серверам подключался malware, что передавал и т.д. Однако, будьте осторожны: tcpdump тоже влияет на систему. Лучше делать это с удалённого хоста (например, span-порт на коммутаторе, либо если это VM – через hypervisor capture).

- **Логи и конфигурации:** Помимо “больших” дампов, сохраните копии ключевых лог-файлов: `/var/log/auth.log`, `/var/log/syslog` (или `/var/log/messages`), логи приложений (например, веб-сервера), логи аудита (`/var/log/audit/audit.log` при SELinux). Также скопируйте файлы конфигураций сервисов – они помогут понять, что мог изменить злоумышленник. Например, `.ssh/authorized_keys` пользователей (вдруг злоумышленник добавил свой ключ), cron-задания (`/etc/crontab`, содержимое `/etc/cron.*` и `crontab -l` от каждого пользователя), список установленных пакетов (`dpkg -l` или `rpm -qa` в файл). Многие из этих данных можно собрать скриптами (существуют готовые сборщики артефактов, например, *Ubuntu IR script*, но их использование выходит за рамки гайда). Главное – **собрать максимум информации, не изменяя её**: лучше скопировать весь `/etc` и `/var/log` на флешку, чем потом жалеть об утерянном.

Все собранные образы (disk.img, ram.dump, network.pcap, logs/) необходимо **хранить в безопасности**. Работайте всегда с копиями, оригиналы отложите как эталон (ideally, сделать их “только для чтения”). Не забудьте про хэши для доказательства, что образы аутентичны.

## 4.2 Поиск следов взлома в системе

При расследовании инцидента полезно иметь чек-лист, что проверять на взломанной системе. Многие артефакты могут указать на действия злоумышленника:

- **Пользователи и группы:** Просмотрите список учетных записей (`/etc/passwd`) – не заведён ли “лишний” пользователь. Особое внимание к UID 0 (должен быть только root). Команда `last` покажет последние логины – есть ли неизвестные вам? Команда `lastb` – неудачные попытки логина (bruteforce). Также `getent group sudo` (или `wheel`) – кто имеет админские права.
- **Запущенные процессы (особенно от root):** Выполните `ps -ef` или `ps aux`. Ищите непривычные процессы. Малварь может маскировать имя, но часто заметны аномалии: странное имя, запущено из `/tmp` или `/opt`, либо от имени пользователя, который обычно такого не запускает. Обратите внимание на процессы с именами похожими на системные (например, “cron” или “rsyslog” запущенный не из стандартных путей – это подозрительно). Используйте `top` / `htop` для выявления процессов, грузящих CPU – майнеры обычно дают о себе знать высоким нагрузком.
- **Сетевые подключения и сервисы:** Запустите `ss -tulnp` (или `netstat -tulnp`); проверьте все прослушиваемые порты и установленные соединения. Не появился ли неизвестный слушающий порт? Например, бекдор может открывать порт 4444 для shell. Также `ss -anp | grep ESTAB` – установленные внешние соединения. Есть ли

соединения к чужим IP от процессов, которые не должны в интернет ходить? Например, `apache` процесс сессия к неизвестному IP – подозрительно.

- **Планировщики заданий:** Очень распространено, что злоумышленник после доступа устанавливает `cron-job` для персистентности (например, каждую минуту качать malware если удалили). Проверьте `crontab -l` для каждого пользователя (в т.ч. root), содержимое `/etc/crontab`, файлы в `/etc/cron.hourly`, `/etc/cron.d`. Ищите незнакомые задания или скрипты. Аналогично, посмотрите на автозапуски сервисов (systemd unit files в `/etc/systemd/system`), `rc.local`, скрипты в `/etc/profile.d` и т.п.
- **SUID-биты:** Выполните поиск файлов с SUID битом, особенно принадлежащих root, кроме стандартных:

```
find / -xdev -perm -4000 -type f -print
```

Злоумышленники иногда оставляют setUID-бинарник для повышения привилегий позже. Если найдете что-то нестандартное (например, `/tmp/susprog` с `suid root`) – это явная красная флажка. Стандартные SUID-файлы – `/usr/bin/passwd`, `/usr/bin/sudo`, `/bin/ping` и несколько еще. Незнакомые – под анализ.

- **Скрытые файлы и директории:** Поиск скрытых элементов:

```
find / -xdev -name ".*" -type f -o -type d
```

Посмотреть содержимое подозрительных. Часто malware прячут в каталогах с точкой или дают имена, напоминающие системные (например, `.ssh` в `/etc`, чего не должно быть).

- **Файлы с необычными правками:** Полезно смотреть по временным меткам. Например, какие файлы изменялись в период, когда предположительно был взлом:

```
find /etc -type f -newermt "2025-05-01"
```

(найдет конфиги, изменённые после 1 мая 2025). Если у вас есть approximate время взлома – ищите изменения вокруг него. Особенно проверять: `/etc/ssh/sshd_config`, `/etc/passwd`, `/etc/sudoers`, `/root/.ssh/authorized_keys`.

- **Логи Fail2Ban, firewall:** Если на сервере был Fail2Ban, посмотрите его отчеты – возможно IP злоумышленника там есть (что покажет время и примерную точку входа) <sup>20</sup>. Логи iptables (если настроено логирование) тоже могут показать попытки сканирования или подключения на нестандартные порты.
- **Состояние SELinux/AppArmor:** Если SELinux был включен, через `audit.log` вы можете найти записи типа “авс: denied” – если malware пытался что-то сделать, SELinux мог заблокировать и залогировать <sup>20</sup>. Например, процесс httpd пытался запустить `/bin/sh` – SELinux denial бы это зафиксировал. Аналогично AppArmor – в `dmesg` видны “audit: ... DENIED” сообщения.



Составив список подозрительных артефактов, можно реконструировать картину. Например: в логах auth.log вы нашли успешный вход по SSH под user 'backup' в 3:00 ночи, хотя этот пользователь не должен входить – значит, вероятно пароль был угадан (или ключ скомпрометирован). Далее, видим что в 3:02 изменен /etc/crontab – добавлена строка запуска /tmp/.script.sh. А в 3:05 Fail2Ban забанил много IP – возможно, злоумышленник сканировал. Соединив факты, вы уже лучше понимаете развитие инцидента.

Здесь важно соблюдать **хронологию**: можно выписать таймлайн событий (по логам и таймстампам файлов). Для комплексных случаев цифровые криминалисты используют инструменты типа plaso/log2timeline для автоматического построения линии времени, но вручную тоже можно многое сделать.

### 4.3 Анализ логов

Логи – главный источник информации при расследовании. В Linux журналы распределены по разным файлам, ключевые из них:

- **Аутентификация и авторизация:** /var/log/auth.log (Debian/Ubuntu) или /var/log/secure (RHEL/CentOS). Здесь записаны все попытки входа (ssh, sudo, su, etc). Ищите в них:  
1) Время и IP входа злоумышленника (успешного или многократные неудачные). 2) Сообщения о создании пользователей, изменении паролей, использовании sudo. Пример: строка "Accepted publickey for alice from 203.0.113.5" – успешный SSH вход, если IP неизвестный – след. "Failed password for root from 198.51.100.10 port ..." – брутфорс.
- **Системный журнал:** /var/log/syslog (Debian) или /var/log/messages (CentOS). Здесь общие системные сообщения: запуск демонов, kernel warnings, Cron задачи. По Cron можно увидеть, выполнялся ли подозрительный скрипт. Kernel части – например, OOM killer, или сообщения audit (SELinux/AppArmor) сюда же могут попадать.
- **Логи сервисов:** веб-сервера (/var/log/apache2/access.log и error.log, либо /var/log/nginx/...), базы данных (MySQL: /var/log/mysql/error.log), FTP, почты и т.д. Если взлом пришел через веб-уязвимость, в access.log часто видны странные запросы (например, длинные URL с инъекциями). По IP в логах веба можно сопоставить с IP в auth.log и понять, что один и тот же источник.
- **Лог ядра и audit:** /var/log/kern.log (для kernel) и /var/log/audit/audit.log (если auditd активен, чаще на SELinux-системах). Audit.log очень полезен, но сложен: он фиксирует системные вызовы и denials. Если SELinux помешал атаке, будет запись типа  
avc: denied { getattr } for pid=1234 comm="malware" name="shadow" dev="sda1" ...  
Это говорит, что процесс malware пытался читать /etc/shadow, но SELinux не дал. Такие вещи – звоночки.
- **Логи Fail2Ban:** /var/log/fail2ban.log – содержит информацию о банах/разбанах IP с таймстампами и jail, сработавшим фильтром. Например:  
fail2ban.actions [12345]: BAN 198.51.100.3. Это сообщает, что IP был заблокирован – а значит ранее он совершил несколько подозрительных действий. Проверьте, какие – через связанный сервисный лог (например, auth.log для ssh jail).

**Методика анализа:** Применяйте фильтрацию и поиск по логам утилитами grep, awk, etc. Например:

- `grep -i "Accepted" auth.log` – все успешные входы.
- `grep "Failure" auth.log | wc -l` – сколько неудачных попыток, грубо.
- `grep "sudo:" auth.log` – кто использовал sudo и когда.
- `grep "sh" /var/log/apache2/access.log` – есть ли обращения к shell через веб (признак веб-шелла).
- `grep -R "uid=0" /var/log` – поиск упоминаний UID 0 (root) в логах, вдруг что-то выдали.

Полезно искать по временным промежуткам: можно вырезать блок логов за интересующий час. Например, `awk '$0 > "May 21 03:00" && $0 < "May 21 04:00"' /var/log/syslog` даст строки между 3:00 и 4:00 21 мая.

Комбинируйте информацию: допустим, вы нашли, что злоумышленник получил root в 03:15 (через sudo или эксплойт). Тогда смотрите, что происходило после 03:15: какие команды запускались (можно посмотреть `.bash_history` файла root, хотя злоумышленники часто его чистят или отключают историю). Проверяйте логи на записи после этого времени – возможно, cron в 03:17 создал задачу.

**Специализированные инструменты:** Для облегчения анализа логов есть утилиты: `ausearch` (для audit.log, фильтр по PID, времени, ключевым словам), `journalctl` (в системах с systemd – объединённый просмотр журналов). Кроме того, в Kali можно использовать Log file analyzer (плагины для Splunk, ElasticStack) – но это за пределами ручного администрирования.

Главное – **не доверять только одному источнику данных**. Логи могли быть подчищены злоумышленником. Например, если `/var/log/auth.log` подозрительно обрывается на время атаки, это плохой признак – возможно, логи удалили. Тогда обратитесь к бэкапам логов (если настроен rsyslog удалённо) или вторичным признакам (например, last показывает логины, даже если auth.log стёрли, у утилиты last свои записи в /var/log/wtmp).

## 4.4 Анализ образов дисков и памяти

После того, как основные симптомы проверены на живой системе, наступает этап глубокой форензики образов – уже **офлайн анализ** ранее сохранённых дампов. Здесь в ход идут специализированные инструменты и методичность:

**Анализ дискового образа:** Если вы создали образ диска (например, `sda.img`), можно использовать инструментарий **The Sleuth Kit (TSK)** или графическую оболочку **Autopsy**. Они позволяют: - Просматривать файловую систему образа, включая удалённые файлы (TSK утилиты `fls`, `icat` могут восстановить удалённые, если не перезаписаны). - Строить временную шкалу (mactime) изменений файлов. - Искать конкретные сигнатуры/шаблоны по образу (например, строки паролей, URLы).

В Kali Linux TSK предустановлен. Пример:

```
fls -r -m / sda.img > timeline.txt
```

создаст список всех файлов и директорий с мета-данными. Затем с помощью `mactime` можно формировать таймлайн. Но требуется сноровка.

Autopsy (также open-source) дает более удобный интерфейс через браузер: можно добавить образ как доказательство, и Autopsy сама вытащит все удалённые файлы, покажет какие файлы менялись в подозрительное время, и т.п.

**Поиск IoC (Indicators of Compromise):** Зная, что искать, можно пройтись по образу. Например, если в логах был упомянут `/tmp/.script.sh`, то на образе можно найти этот файл и открыть (даже если удалили, в образе он может сохраниться). Или поискать по сигнатурам: если знаете хэш вредоносного файла (например, ClamAV выявил "Linux.Backdoor.X"), найти по образу этот хэш (командой `grep -aR <hex> sda.img`) – но лучше монтировать образ и использовать обычные `grep/find` внутри.

**Memory forensics:** Дамп памяти – кладёшь информации о процессах, сетевых подключениях, и даже, возможно, несохранённых на диск данных (например, содержимое файлов, которые malware держал открытыми). Самый известный фреймворк для анализа памяти – **Volatility** (open-source, Python) <sup>21</sup>. Он поддерживает образы памяти Linux, хотя требуется указать правильный профиль (версию ядра). Существуют Volatility 2 (старый) и Volatility 3 (новый, с автодетектом ОС).

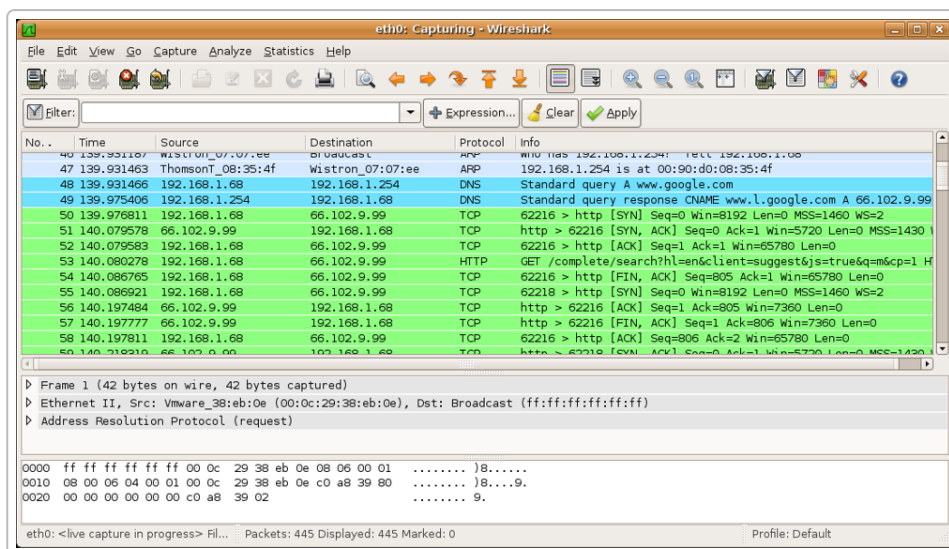
С помощью Volatility можно извлечь: список процессов, дерево процессов, модули ядра, открытые сетевые соединения, содержимое сокетов, команды bash из истории в памяти, и много другого. Примеры команд (Volatility 2 syntax):

```
vol.py -f ram.dump --profile=LinuxUbuntu16_04x64 linux_pslist      # список
процессов
vol.py -f ram.dump --profile=LinuxUbuntu16_04x64 linux_netstat    # сетевые
соединения
vol.py -f ram.dump --profile=LinuxUbuntu16_04x64 linux_bash
# фрагменты bash history из памяти
vol.py -f ram.dump --profile=LinuxUbuntu16_04x64 linux_lsof      # открытые
файлы по процессам
```

Volatility 3 упрощает тем, что не всегда нужно указывать профиль – он сам определит ОС. Команды примерно те же (модуль `linux.pslist`, `linux.netstat`, etc.). Например, используя Volatility, вы можете подтвердить наличие вредоносного процесса, даже если он уже не активен: его остатки могут быть в памяти. Или увидеть скрытый процесс, не отображавшийся в `ps` (rootkit мог скрывать, а в дампе памяти он останется).

Из интересных трюков: Volatility умеет выдирать из дампа *именно* исполняемые области процессов, позволяя вам восстановить вредоносный код. Например, `linux_procdump` плагин сохранит образ процесса на диск для дальнейшего анализа антивирусами или дизассемблером.

**Анализ сетевого трафика:** Файл `network.pcap`, если вы его получили, анализируется в **Wireshark** – мощном сетевом анализаторе с GUI. В Wireshark можно фильтровать по IP, по протоколам. Например, увидеть, какие команды шли по SSH (если расшифровать не выйдет, но можно увидеть длительность сессии), или обнаружить, что сервер внезапно слал данные на какой-то внешний адрес по HTTP (может, утечка). На рисунке ниже – пример окна Wireshark, где отображены захваченные пакеты различных протоколов.



*Пример интерфейса Wireshark: анализ захваченного сетевого трафика, что помогает отследить активность злоумышленника по сети.*

Сетевой трафик особенно полезен, если malicious активно общался с командным сервером – вы сможете извлечь IP/домены, которые затем стоит заблокировать и проверить на них индикаторы (может, они известны в базах злоум.активности).

**Документирование и выводы:** В процессе форензики крайне важно записывать все находки: время события, что произошло, какие файлы затронуты. В конце должно сложиться целостное представление: *вектор атаки* (например, взлом через уязвимый плагин WordPress), *время и длительность компрометации*, *какие данные могли утечь или быть изменены*, *какие действия совершил злоумышленник* (создал пользователя, установил руткит, развернул майнер и т.п.). Эти данные помогут не только устранить текущие последствия (удалить malware, закрыть уязвимость), но и улучшить защиту на будущее.

После окончания расследования, **восстановите систему** из доверенного источника (бэкапа или переустановки) – никогда не продолжайте эксплуатировать систему, в целостности которой есть сомнения. А полученные уроки используйте для улучшения разделов 2 и 3 данного руководства (анализа и защиты), создайте план действий на случай инцидентов.

Каждая из тем, затронутых в гайде, заслуживает еще более подробного изучения, но даже применив описанные меры и инструменты, вы значительно повысите безопасность своего Linux-сервера и будете готовы квалифицированно реагировать на киберугрозы. Помните, что информационная безопасность – это непрерывный процесс: **проактивная защита + мониторинг + своевременное реагирование.**

[t.me/linuxkalii](https://t.me/linuxkalii)

1 2 3 4 6 7 8 9 10 5 инструментов для сканирования Linux-сервера  
<https://wiki.merionet.ru/articles/5-instrumentov-dlya-skanirovaniya-linux-servera>

5 screenshots.debian.net  
<https://screenshots.debian.net/package/rkhunter>

11 Безопасность Linux сервера: полное руководство | Записи Кушнаренко Е.  
<https://evgeny.kushnaren.co/posts/linux-server-security/>

12 Как защитить систему Linux от вредоносных действий с ... - Яндекс  
[https://ya.ru/neurum/c/drugoe/q/kak\\_zaschitit\\_sistemu\\_linux\\_ot\\_vredonosnyh\\_0bef221f](https://ya.ru/neurum/c/drugoe/q/kak_zaschitit_sistemu_linux_ot_vredonosnyh_0bef221f)

13 How to Use Fail2ban to Secure Your Linux Server  
<https://www.tecmint.com/use-fail2ban-to-secure-linux-server/>

14 15 16 17 18 19 20 Некоторые приёмы форензики под Linux | OTUS  
<https://otus.ru/nest/post/1058/>

21 Analyzing Memory Dump with Volatility | by Nishant Sharma | Pentester Academy Blog  
<https://blog.pentesteracademy.com/analyzing-memory-dump-with-volatility-302f30917713>