Лучшие инструменты аудита безопасности Linux

IVO 4,369

Содержание

В сегодняшней статье мы познакомим вас с **лучшими утилитами аудита безопасности Linux** или как говорят наши англоязычные коллеги — **Hardening Linux**. Итак, тема статьи проверка уровня защищенности Linux-систем и оценка корректности конфигов с точки зрения информационной безопасности. Разумеется, мы не только сделаем обзор программ, но и приведем примеры их использования.

Еще по теме: Защита ноутбука с Linux

Аудит безопасности Linux собственными силами

Перед системными администраторами, а уж тем более перед аудиторами информационной безопасности часто встают задачи проверить защищенность большого количества хостов за очень короткое время. И конечно, для решения этих задач в Enterprise-сегменте есть специализированные инструменты, к примеру такие, как сетевые сканеры безопасности. Уверен, что все они — от open

sources движка OpenVAS до коммерческих продуктов типа Nessus или Nexpose — известны нашему читателю. Но данный софт обычно используется, чтобы искать устаревшее и потому уязвимое программное обеспечение и затем запустить патч-менеджмент. Кроме этого не все сканеры безопасности учитывают определенные специфические особенности встроенных механизмов защиты ОС Linux и других продуктов с открытым исходным кодом. Ну и не в последнюю очередь значение имеет цена вопроса, ведь платные продукты могут позволить себе разве что компании, выделяющие под это дело какие-то бюджеты.

Вот поэтому сегодня речь пойдет о специализированном наборе бесплатных утилит, которые могут диагностировать текущий уровень защищенности системы, оценить потенциальные риски, к примеру «лишние сервисы», торчащие в интернет, или небезопасный конфиг по умолчанию, и в определенных случаях предложить варианты исправления найденных аудитом проблем. Еще одно преимущество использования этих инструментов заключается в возможности тиражировать типовые сценарии проверки фермы из любого количества Linux-систем и создавать документально подтвержденную базу тестов в виде логов и отдельных отчетов.

Практические аспекты аудита безопасности Linux

Если посмотреть глазами аудитора, то подход к тестированию можно разделить на два типа.

Первый — это соответствие так называемым compliance-требованиям, здесь проверяется наличие обязательных элементов защиты, прописанных в каком-либо международном стандарте или «best practice». Классический пример — требования PCI DSS для платежных ИТ-систем, SOX404, NIST-800 series, MITRE.

Второй — это сугубо рациональный подход, основанный на вопросе «А что еще можно сделать, чтобы усилить защищенность?». Тут нет обязательных требований — только твои знания, светлая голова и умелые руки. К примеру, это обновление версии ядра и/или пакетов приложений, включение шифрования томов, форсирование SELinux, настройка файрвола iptables.

Все, что относится ко второму подходу, принято называть специальным термином **Linux hardening**, что еще можно определить как «действия, направленные на усиление уровня исходной защищенности ОС (или ПО) преимущественно штатными средствами».

Соответствие compliance-требованиям, как правило, проверяют при подготовке к прохождению обязательного аудита типа PCI DSS или другого сертификационного аудита. Мы же больше уделим внимание Hardening-составляющей. Все крупные вендоры предлагают для своих продуктов *Hardening Guidelines* — руководства, содержащие советы и рекомендации, как усилить

защищенность, учитывая штатные механизмы безопасности и специфику ПО. Так, подобные руководства есть у Red Hat, Debian, Oracle, Cisco.

> Hardening — это термин из мира информационной безопасности, который обозначает процесс обеспечения безопасности системы (программы) за счет снижения ее уязвимости и, как правило, с использованием исключительно штатных утилит или механизмов защиты.

Кстати, у нас уже когда-то была статья про настройку опций Hardening, но в той статье речь шла именно о настройке. Мы же сначала проверим нашу систему с помощью специализированных утилит, то есть проведем аудит безопасности, оценим текущий уровень защиты, а потом уже накрутим туда security option, если в этом будет необходимость. Ну или еще как вариант: если сервер уже настроен с точки зрения безопасности, наши инструменты смогут это проверить и, возможно, подсказать, что же можно сделать еще.

Инструменты аудита безопасности Linux

Lynis — auditing system hardening testing

Lynis — первая в нашем списке инструментов и, пожалуй, самая навороченная тулза для аудита Linux-систем. При этом она довольно простая в использовании и очень наглядная — все тесты и их отчеты выводятся на экран. Утилита сканирует настройки текущего уровня безопасности и определяет состояние защищенности (hardening state) хоста. Найденные тревожные сигналы и важные алерты безопасности, сгруппированные по блокам, выводятся в консоль терминала и отдельно в файл отчета. Кроме сведений о безопасности, Lynis также поможет получить общую информацию о системе, информацию об установленных пакетах и возможных ошибках конфигурации, обновлениях ядра и т.п.

Разработчиками Lynis заявлена поддержка огромного числа операционных систем: от Arch, BackTrack, Kali до разновидностей Debian/Ubuntu, RHEL/CentOS, SuSE, BSD-семейства (FreeBSD, NetBSD, OpenBSD, DragonFly BSD), а также более редких HPUX, Solaris 10+, TrueOS и macOS.

Вся документация с более подробным описанием и примерами использования доступна в разделе Lynis Documentation на официальном сайте CISOfy. Если не хотите ограничиваться предлагаемыми тестами, есть возможность разработки собственных. Более подробно об этом написано в разделе Lynis Software Development Kit. Ну и для тех, кто еще сомневается, устанавливать или нет утилиту, разработчики подготовили маленькое demo, показывающее, как происходит процесс установки и первичный запуск.

Кроме бесплатной версии, которую мы и будем чуть ниже использовать, разработчики предлагают решение Enterprise-уровня. В этом случае к стандартной поставке добавляется веб-интерфейс для администрирования, опциональные дашборды, дополнительная документация (hardening snippets) и развернутый план корректировки выявленных нарушений. И это еще не все, данное решение ко всему прочему вы можете получить в виде облачного сервиса (Software-as-a-Service).

Lynis выполняет огромное количество отдельных тестов, чтобы определить состояние защищенности системы. Сама проверка защищенности заключается в выполнении набора шагов от инициализации программы до генерации отчета.

Поскольку Lynis весьма гибкий и многопрофильный инструмент, он используется для различных целей. К примеру, типовые варианты использования Lynis включают:

- аудит безопасности (типовой сценарий, задаваемый пользователем);
- тестирование на соответствие требованиям (например, PCI DSS, HIPAA, SOX404, OpenSCAP, NSA);
- обнаружение уязвимостей (устаревшее ПО);
- режим Penetration testing (попытка эскалации привилегий);
- улучшение системы (незадействованные твики ядра, демонов и прочего).

Установить утилиту вы можете несколькими способами — как с помощью загрузки из хранилища GitHub:

```
git clone https://github.com/CISOfy/lynis
cd lynis
./lynis
```

так и установкой из репозитория Debian/Ubuntu:

```
sudo apt-get update
sudo apt-get install lynis
```

И для RPM-ориентированных дистрибутивов (предварительно добавив соответствующие репозитории):

```
yum install linus -y
```

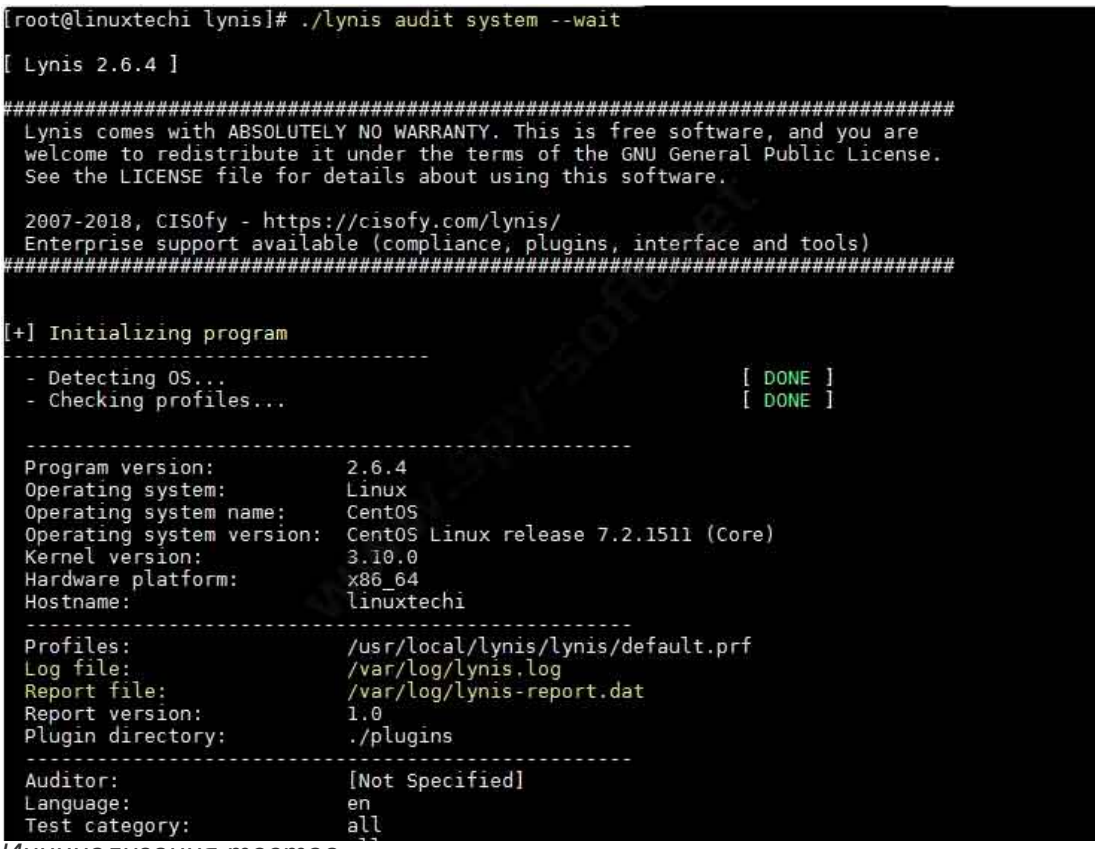
Установка на macOS:

```
$ brew search lynis
$ brew install lynis
```

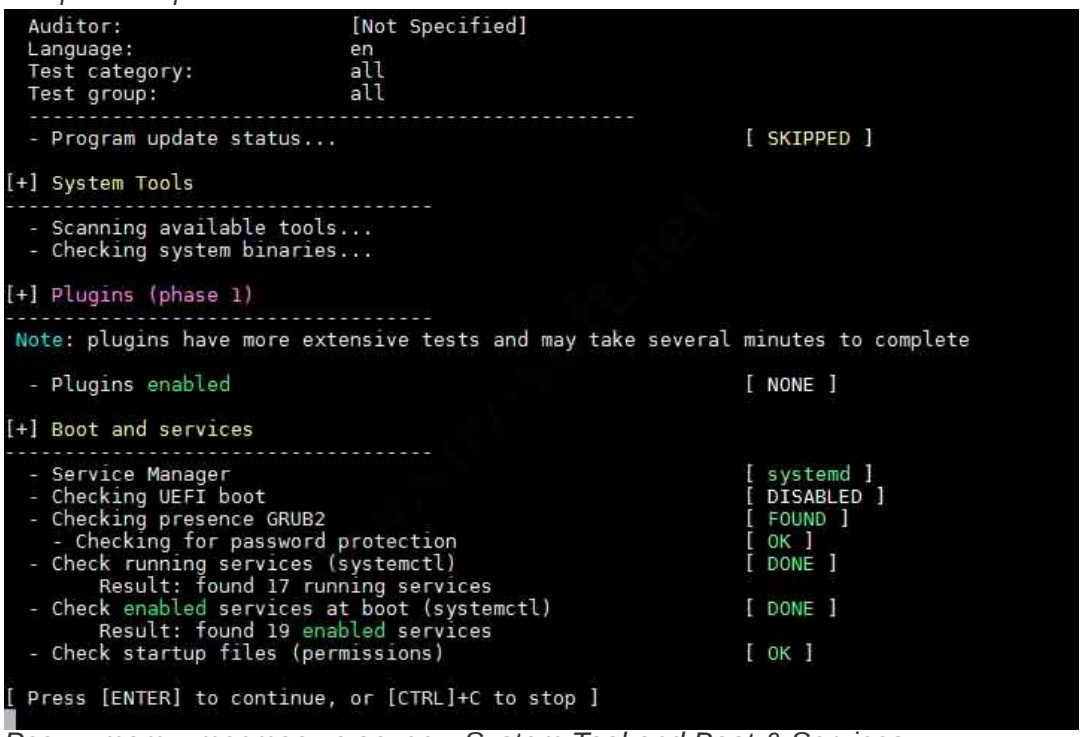
Чтобы запустить Lynis, достаточно указать хотя бы один ключ. К примеру, для запуска всех имеющихся тестов следует указать ключ -c (check all, проверить все):

```
# Полный набор тестов
sudo lynis audit system -c
# Типовой набор тестов
sudo lynis audit system -c
# Сканирование удаленного хоста
```

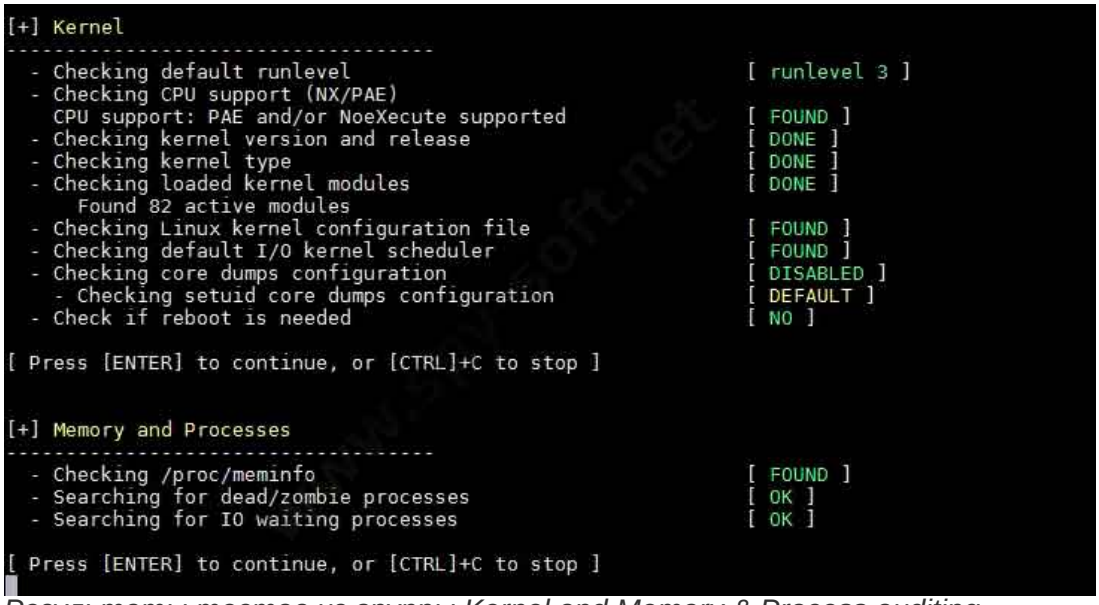
audit system remote <host>



Инициализация тестов



Результаты тестов из группы System Tool and Boot & Services



Результаты тестов из группы Kernel and Memory & Process auditing

Результаты тестов из группы User and Group & Authentication

Перед аудитом всегда полезно проверить, доступна ли новая версия Lynis:

```
lynis update info && lynis update check
```

У утилиты Lynis, помимо стандартного, существует еще один режим — **непривилегированного запуска**:

```
lynis audit --pentest
```

Если же вы хотите поместить имя аудитора, запустившего тестирование, просто добавьте параметр -auditor <name>:

```
sudo lynis audit system -c -auditor Daddy
```

На любом этапе аудита безопасности Linux процесс проверки может быть продолжен (Enter) или принудительно прекращен (Ctrl+C). Результаты выполненных тестов будут писаться в журнал Lynis в каталог /var/log/lynis.log. Учтите, что лог-файл будет перезаписываться при каждом следующем запуске утилиты.

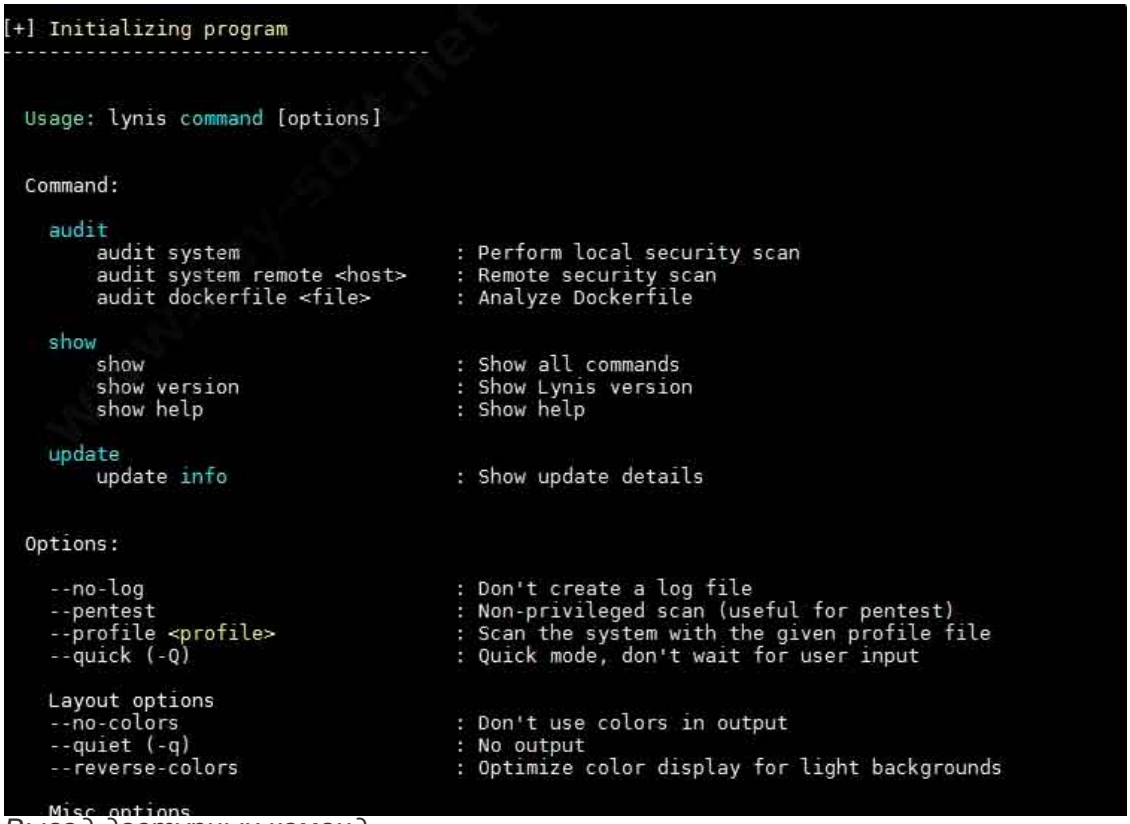
Для тестирования на постоянной основе в автоматическом режиме можно назначить соответствующее задание в планировщик Cron с помощью ключа -cronjob. В таком случае Lynis будет запускаться по заданному шаблону (конфигу) и не будет выводить никаких интерактивных лишних сообщений, вопросов и предупреждений. Все результаты будут сохранятся в лог. К примеру, вот скрипт запуска утилиты с настройками по умолчанию раз в месяц:

```
#!/bin/sh
AUDITOR="automated"
DATE=$(date +%Y%m%d)
HOST=$(hostname)
LOG_DIR="/var/log/lynis"
REPORT="$LOG_DIR/report-${HOST}.${DATE}"
DATA="$LOG_DIR/report-data-${HOST}.${DATE}.txt"
cd /usr/local/lynis
./lynis -c –auditor "${AUDITOR}" –cronjob > ${REPORT}
mv /var/log/lynis-report.dat ${DATA}
# End
```

Сохраните этот скрипт в каталог /etc/cron.monthly/lynis. И не забудьте добавить пути для сохранения логов (/usr/local/lynis and /var/log/lynis), иначе возможна некорректная работа.

Можно посмотреть список всех доступных для вызова команд:

```
lynis show commands
```



Вывод доступных команд

Можно также взглянуть на настройки из конфига по умолчанию:

```
lynis show settings
```

Краткая инструкция по работе с утилитой:

```
man lynis
```

Варианты возможных статусов по результатам проверки ограничиваются следующим списком: **NONE, WEAK, DONE, FOUND, NOT_FOUND, OK, WARNING**.

Пример вывода статусов

Запуск отдельных тестов в Lynis

На практике бывает необходимо провести лишь некоторые определенные тесты. К примеру, если ваш сервер выполняет только функции Mail Server или Apache. Для этого мы можем использовать параметр -tests. Синтаксис команды выглядит таким образом:

```
lynis -tests "Test-IDs"
```

Если вам сложно разобраться из-за большого количества идентификаторов тестов, то вы можете использовать групповой параметр -test-category. С помощью данной опции Lynis запускает идентификаторы только тех тестов, которые входят в определенную категорию. Например, мы планируем запустить тесты брандмауэра и ядра:

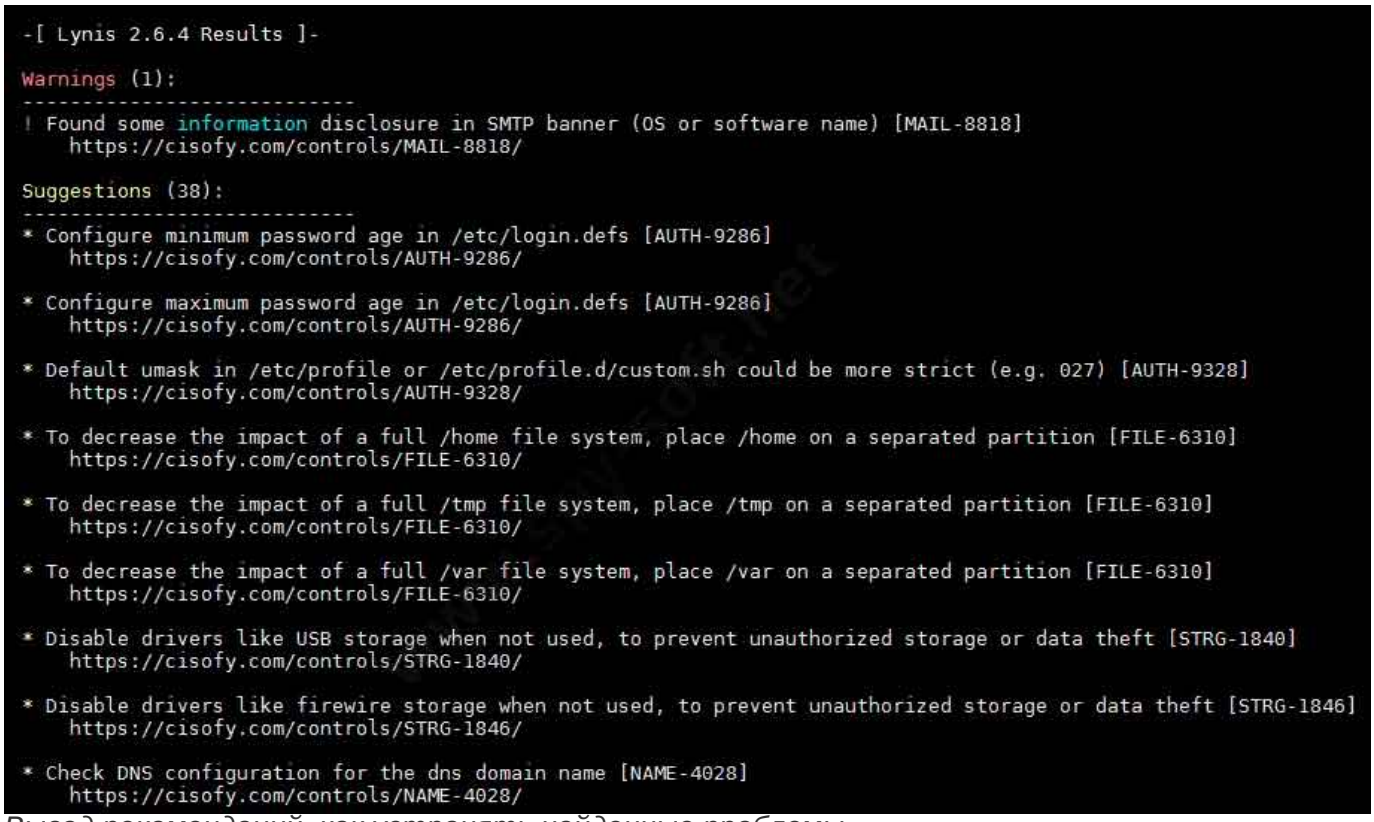
```
./lynis -tests-category "firewalls kernel"
```

Список всех доступных тестов можно посмотреть в разделе Controls.

Помимо этого, функциональность Lynis расширяют различные дополнения, которые можно дописывать самостоятельно, а можно подкладывать новые в существующую директорию.

Все предупреждения (Warnings) будут перечислены после результатов. Каждое начинается с текста предупреждения, потом рядом в скобках указывается тест, который его сгенерировал. В следующей

строке предлагается решение проблемы, если оно конечно существует. По факту последняя строка — это URL-адрес, по которому вы сможете посмотреть подробности и найти дополнительные рекомендации, как устранить возникшую проблему.



Вывод рекомендаций, как устранять найденные проблемы

Профили

Профили, которые управляют аудитом, определяются в файлах с расширением .prf, расположенных в каталоге /etc/lynis. Профиль по умолчанию называется, предсказуемо, default.prf. Разработчики не советуют править его напрямую: любые изменения, которые вы хотите внести в аудит, лучше добавлять в файл custom.prf, находящийся в том же каталоге.

Создаем и редактируем кастомный профиль:

```
touch /etc/lynis/custom.prf
sudo nano /etc/lynis/custom.prf
```

В данном файле можно определить список тестов, которые необходимо исключить из аудита Lynis. Например:

- FILE-6310: проверка разделов;
- HTTP-6622: тест установки nginx;
- HTTP-6702: тест установки Apache.

Чтобы исключить какой-то определенный тест, воспользуйтесь директивой skip-test и укажите ID теста. Например, так:

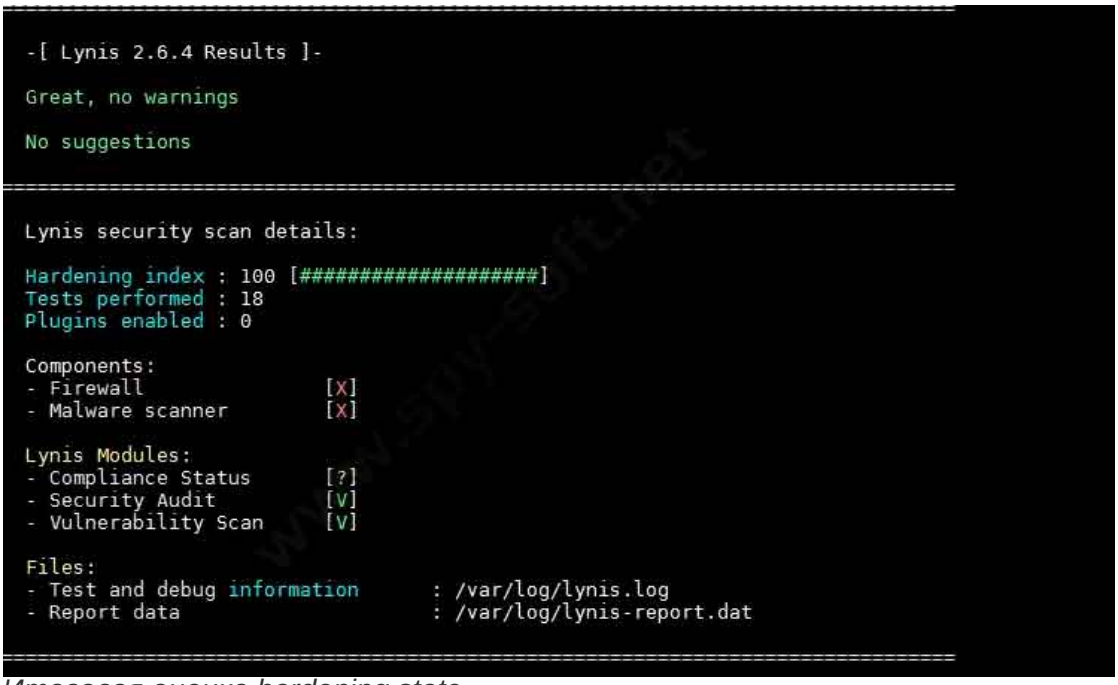
```
# Is nginx installed?
```

```
skip-test=HTTP-6622
# Is Apache installed?
skip-test=HTTP-6702
```

Оценка hardening state

По результатам выполнения всех тестов в конце каждого вывода аудита утилиты (чуть ниже раздела предложений) вы найдете раздел, который будет выглядеть примерно так:

```
Lynis security scan details:
Hardening index : 57 [############.....]
Tests performed : 216
Plugins enabled : 0
```



Итоговая оценка hardening state

Данный результат, выраженный числом, показывает количество пройденных тестов и индекс безопасности системы, то есть **hardening index** — конечное число, с помощью которого Lynis дает оценку общего уровня безопасности сервера. И очень важно не забывать, что индекс безопасности изменяется в зависимости от количества исправленных предупреждений и выполненных рекомендаций Lynis. Поэтому после исправлений найденных проблем повторный аудит безопасности может показать совсем другое число!

> Все манипуляции с системой в режиме суперпользователя требуют пристального внимания и повышенной ответственности. Выполняйте только те действия, которые осознаете и в которых уверены. Не забывайте делать резервные копии и снапшоты.

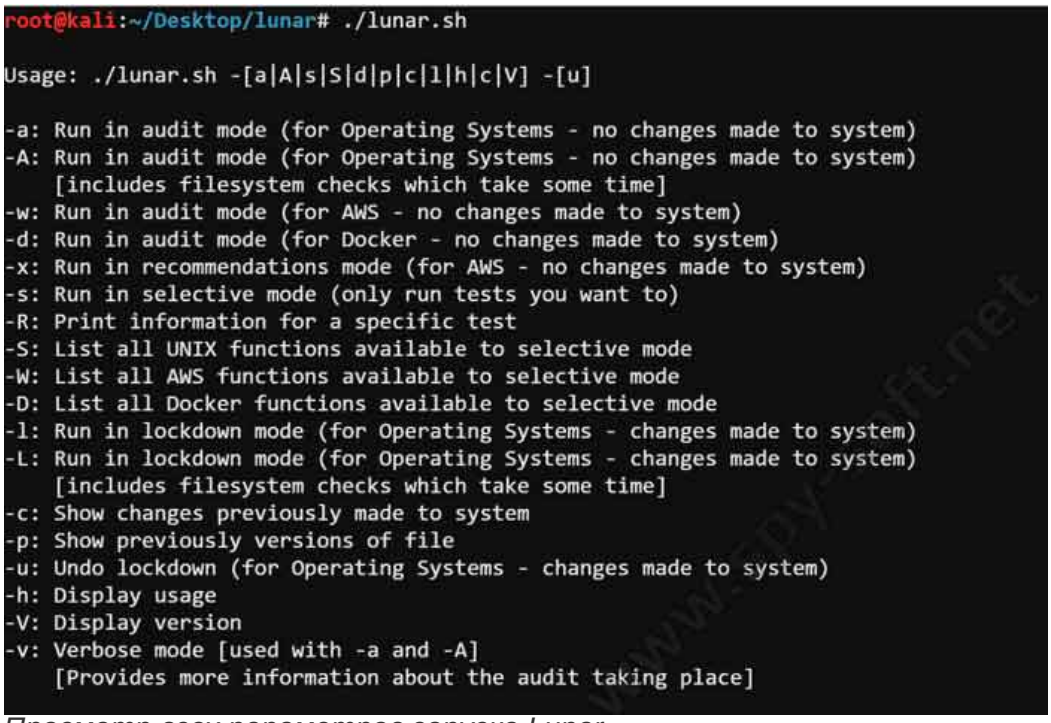
Lunar — a UNIX security auditing tool

Lunar — это набор нативных скриптов, написанных на языке командной оболочки bash, которые тестируют целевую Linux-машину и генерируют по результатам проверки заключение аудита безопасности. Инструмент основан на стандартах CIS и других мировых фреймворках по безопасности.

Заявлена поддержка всех популярных систем: Linux — RHEL и CentOS с версии 5, SLES начиная с версии 10, Debian/Ubuntu, Amazon Linux, Solaris с версии 6, macOS (последние билды), FreeBSD (частично), AIX (частично) и даже ESXi.

Кроме прочего, инструмент поддерживает облачную платформу Amazon Web Services (AWS) и контейнеры Docker. Подробное описание все функций, а также примеры запуска Lunar и инициации тестов приведены в подробной документации Wiki на GitHub.

Примеры запуска команд из CLI:



Просмотр всех параметров запуска Lunar

Запуск Lunar в режиме аудита безопасности, то есть **без внесения изменений в систему**:

```
./lunar.sh -a
```

Запуск Lunar в режиме аудита безопасности и предоставления большей информации:

```
./lunar.sh -a -v
```

Перечисление тестов:

```
./lunar.sh -S
```

Выполнение только тестов на основе оболочки:

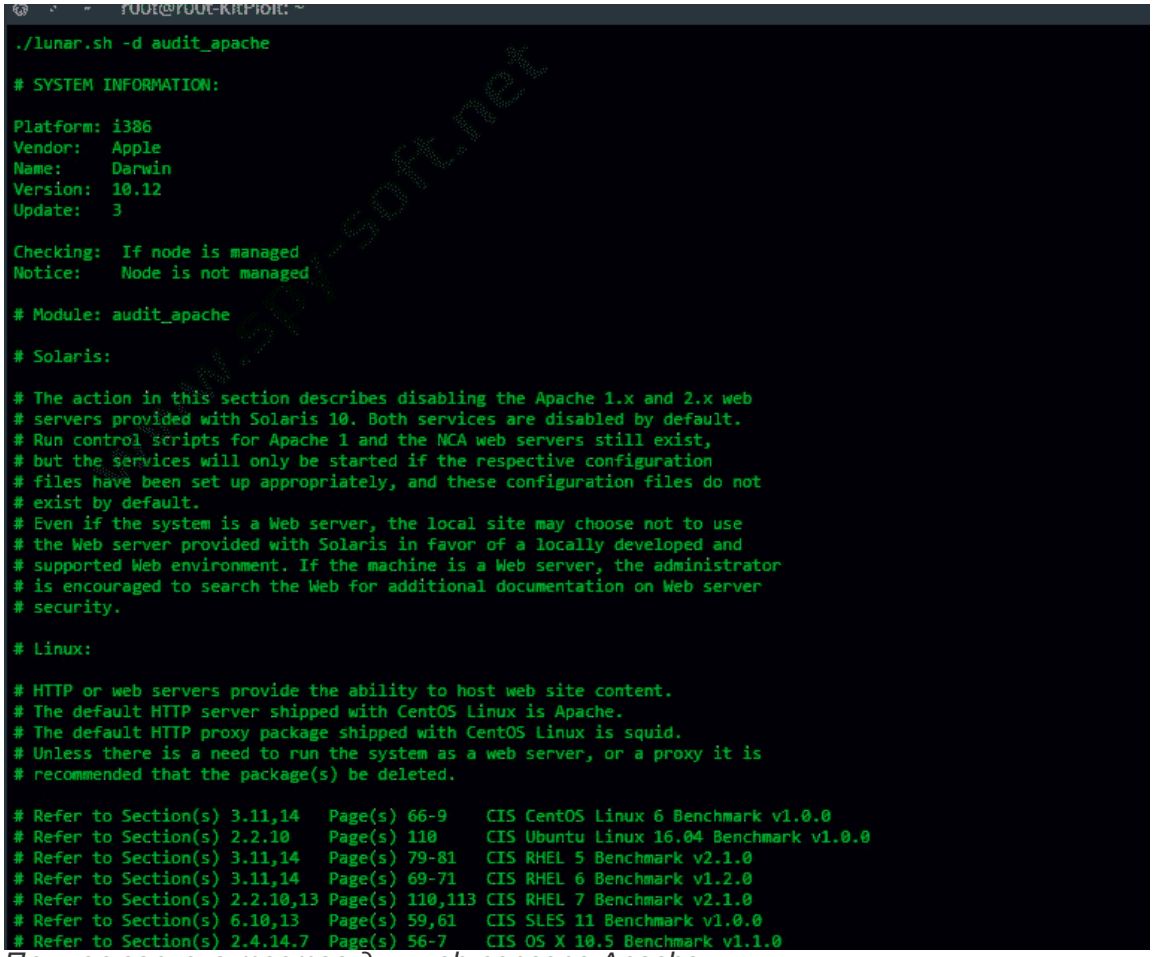
```
./lunar.sh -s audit_shell_services
```

Запуск в режиме исправления, то есть **с внесением изменений в систему**:

```
./lunar.sh -l
```

Просмотр предлагаемых изменений (твиков) системы до их внесения в конфиги:

```
./lunar.sh -d
```



Пример запуска тестов для web-сервера Apache

Nix Auditor — a CIS Audit made easier

Nix Auditor — это очередной скрипт для проверки, соответствует ли безопасность Linux-систем требованиям показателя CIS. Ориентирован на RHEL, CentOS и прочие RPM-дистрибутивы.

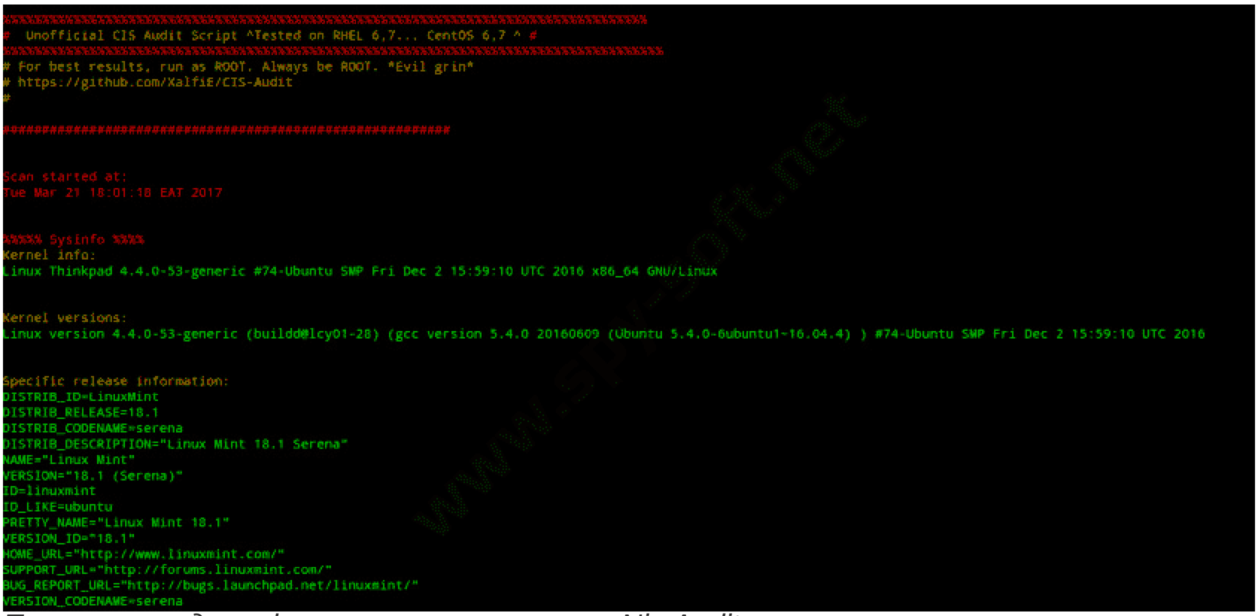
Разработчики заявляют о таких преимуществах Nix Auditor:

- *скорость сканирования* — провести базовую проверку ОС можно менее чем за 120 секунд и тут же получить отчет;
- *точность проверки* — работа Nix Auditor проверена на разных версиях дистрибутивов CentOS и Red Hat;
- *настраиваемость* — исходники с документацией к программе лежат на GitHub, поэтому код легко настраивается в соответствии с типом ОС и набором элементов системы, которые необходимо проверить;
- *простота использования* — достаточно сделать стартовый скрипт исполняемым, и тот уже готов к проверке.

Пример выполнения команд для загрузки утилиты с GitHub-хранилища и последующего запуска скрипта:

```
git clone https://github.com/XalfiE/Nix-Auditor.git
cd Nix-Auditor
```

```
chmod +x nixauditor
./nixauditor
```



Пример вывода информации после запуска Nix Auditor

Loki — Simple IOC and Incident Response Scanner

Утилита Loki — не совсем классический инструмент для проведения аудита Linux, но он отлично подходит для поиска следов взлома, что является форензикой, но отчасти можно отнести и к практике аудита.

По заверениям разработчиков, вот такие возможности дает нам Loki — Simple IOC and Incident Response Scanner:

I. Четыре способа выявить взлом:

- имена файлов (соответствие регулярному выражению полного пути файла);
- проверка в соответствии с правилами Yara (поиск на соответствие сигнатурам Yara по содержимому файлов и памяти процессов);
- проверка и анализ хешей (сравнение просканированных файлов с хешами (MD5, SHA-1, SHA-256) известных вредоносных файлов);
- проверка обратной связи C2 (сравнивает конечные точки технологического соединения с C2 IOC).

II. Дополнительные проверки:

- проверка файловой системы Regin (через —reginfs);
- проверка аномалий системных и пользовательских процессов;
- сканирование распакованных SWF;
- проверка дампа SAM;

- проверка DoublePulsar — попытка выявить бэкдор DoublePulsar, слушающий порты 445/tcp и 3389/tcp.

Чуть-чуть коснемся того, как программа определяет факт взлома. Типовыми признаками (Indicators of Compromise), свидетельствующими, что компьютер был скомпрометирован (то есть взломан), могут быть:

- появление на компьютере вредоноса (вирусов, майнеров, бэкдоров, троянов, крипторов, кейлоггеров, и так далее), а также хакерских утилит (например, для исследования сети, эксплуатации уязвимостей, сбора учетных данных);
- появление неизвестных новых исполнимых и других файлов, даже если они не детектируются антивирусным движком как malware-код;
- аномальная сетевая активность (подключение к удаленным хостам, открытие для прослушивания портов неизвестными программами и прочее);
- аномальная активность на дисковых устройствах (I/O) и повышенное потребление ресурсов системы (CPU, RAM, Swap).

Перед началом инсталляции нужно доустановить несколько зависимых пакетов. Это colorama (дает расцветку строк в консоли), psutil (утилита проверки процессов) и, если еще не установлен, пакет Yara.

Итак, приступаем. Установка в Kali Linux (предварительно должен быть установлен пакет Yara, который по умолчанию уже установлен в Kali Linux):

```
sudo pip2 install psutil netaddr pylzma colorama
git clone https://github.com/Neo23x0/Loki
cd Loki/
python2 loki-upgrader.py
python2 loki.py -h
```

Установка в Ubuntu/Debian:

```
sudo apt-get install yara python-yara python-pip python-setuptools python-dev git
sudo pip2 install --upgrade pip
sudo pip2 install -U setuptools
sudo pip2 install psutil netaddr pylzma colorama
git clone https://github.com/Neo23x0/Loki
cd /home/download/Loki
python2 loki-upgrader.py
python2 loki.py -h
```

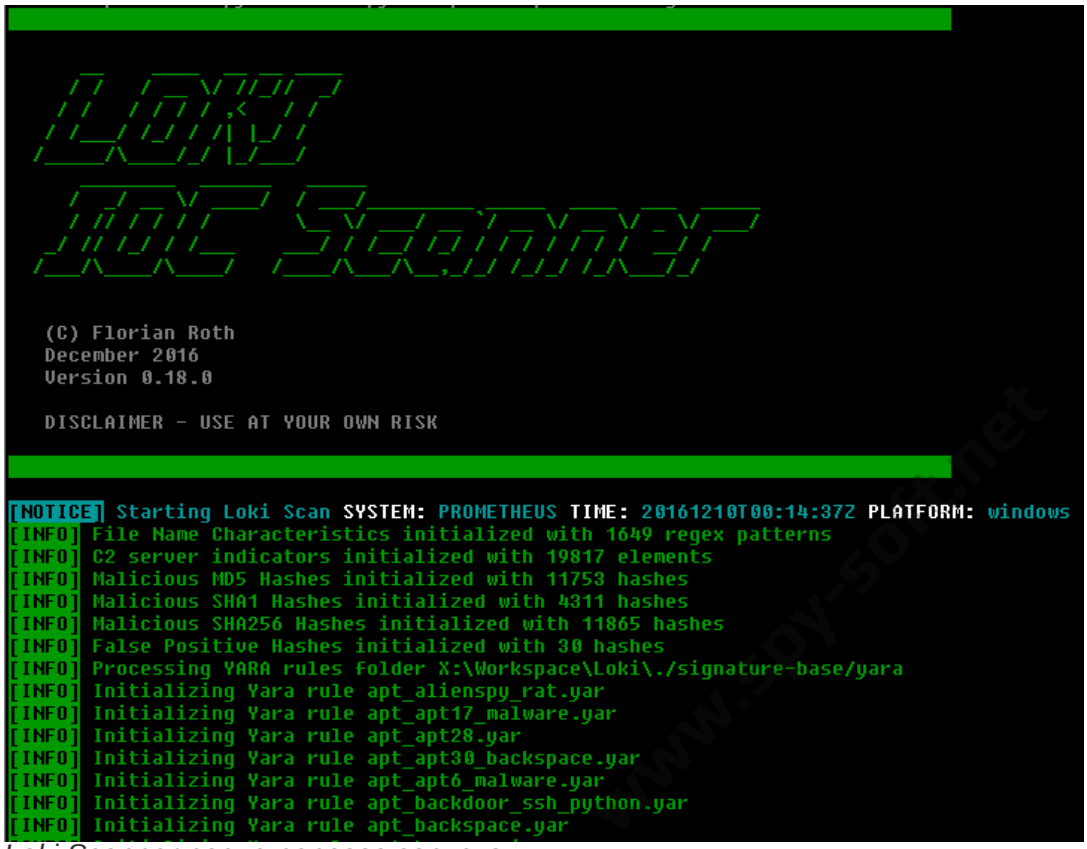
Установка в BlackArch:

```
sudo pacman -S yara python2-pip python2-yara
sudo pip2 install psutil netaddr pylzma colorama
git clone https://github.com/Neo23x0/Loki
cd /home/download/Loki
python2 loki-upgrader.py
python2 loki.py -h
```

Пример использования

Некоторые опции запуска:

```
optional arguments:
  -h, --help      show this help message and exit
  -p path         Path to scan
  -l log-file     Log file
  --printAll      Print all files that are scanned
  --noprocscan    Skip the process scan
  --nofilescan    Skip the file scan
  --noindicator   Do not show a progress indicator
  --reginfs       Do check for Regin virtual file system
  --onlyrelevant  Only print warnings or alerts
  --nolog         Don't write a local log file
```

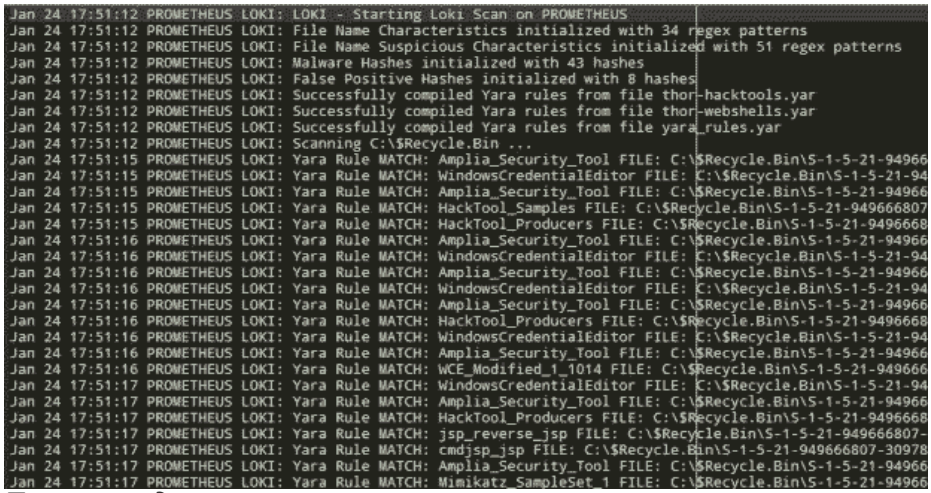


Loki Scanner после первого запуска

Кстати, после установки Loki Scanner неплохо бы проверить локальную базу IoC на обновления, сделать это можете с помощью команды Upgrader:

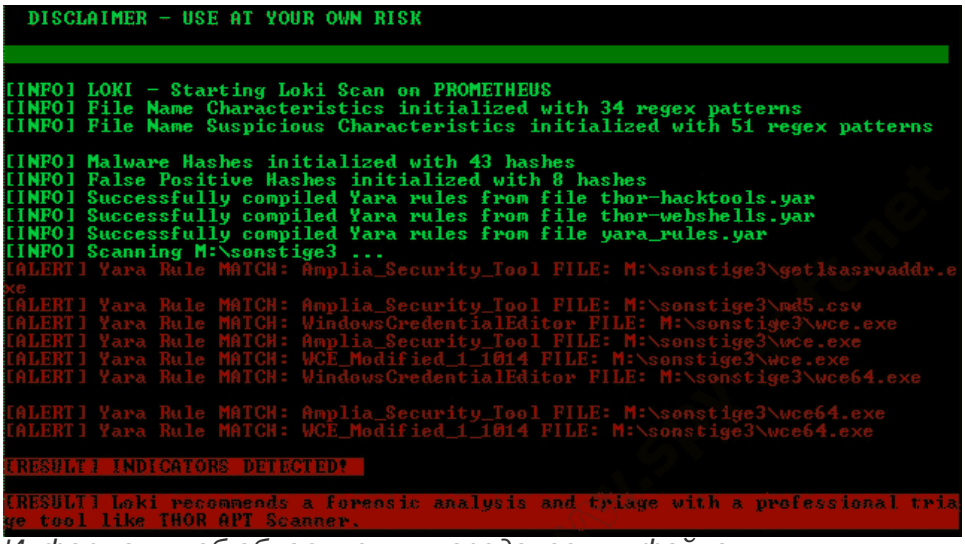
```
Loki - Upgrader

optional arguments:
  -h, --help    show this help message and exit
  --sigsonly    Update the signatures only
  --progonly    Update the program files only
  --nolog       Don't write a local log file
  --debug       Debug output
```



Пример ведения лога при сканировании

В первую очередь пристальное внимание обращаем на сообщения, выделенные красным. В поле *DESCRIPTION* дано описание файла и причины его подозрительности. Обычно это вирусы, бэкдоры и другие подобные им программы.



Информация об обнаруженных вредоносных файлах

Linux Security Auditing Tool (LSAT)

LSAT — заключительный в нашей подборке инструмент аудита безопасности Linux. Особенность данной утилиты в ее модульном дизайне, который, по заверениям разработчика, умеет добавлять новые функции проверки очень оперативно. На данный момент в утилите заявлена поддержка всех самых распространенных ОС: Linux — Gentoo, Red Hat, Debian, Mandrake на архитектуре x86; SunOS (2.x), Red Hat, Mandrake на архитектуре Sparc; а также Apple macOS.

LSAT устанавливается с помощью сборки из исходников и имеет заранее заготовленный автоконфиг — autoconf. Если вы не собираетесь его править на свой вкус, то можно сразу запустить компиляцию:

```
./configure
make
# Также можно установить LSAT в систему, путь расположения /usr/local/bin
make install
# и очистить постинсталляционные файлы
make clean
```

Либо для Debian/Ubuntu-дистрибутивов установить пакет можно прямо из репозитория:

```
sudo apt-get install lsat
```

Запускается LSAT с помощью команды /lsat и добавленными опциями:

```
/lsat [OPTIONS]

Options:
   -d       diff current and old md5, output in lsatmd5.diff
   -f       Force a specific distribution test. Distro names are:
               redhat
               debian
               mandrake
               solaris
               gentoo
               macosx
               If no -f option, lsat will guess. If lsat can
               not guess the distribution, default is redhat.
   -a       Show this (advanced) help page
   -o       Output file name -- default is lsat.out
   -r       Check rpm integrity -- redhat or mandrake only
```

```
  -s      Silent mode
  -v      Verbose output
  -w       Output file in html format
  -x      eXclude module(s) in filelist from checks..
```

Заключение

Мы рассмотрели с вами самые ходовые и в то же время очень крутые и функциональные инструменты и программы для аудита безопасности Linux-серверов. Теперь вы сможете основательно подготовиться к сертификационному или какому-либо другому compliance-аудиту. Это также позволит вам объективно оценить текущий уровень защищенности и в автоматическом или полуавтоматическом режиме настроить свою армию Linux-машин на максимальный показатель hardening index!