

Лучшие инструменты аудита безопасности Linux

IVO 4,369



Содержание

- 1 Аудит безопасности Linux собственными силами
- 2 Практические аспекты аудита безопасности Linux
- 3 Инструменты аудита безопасности Linux
 - 3.1 Lynis — auditing system hardening testing
 - 3.2 Lunar — a UNIX security auditing tool
 - 3.3 Nix Auditor — a CIS Audit made easier
 - 3.4 Loki — Simple IOC and Incident Response Scanner
 - 3.5 Linux Security Auditing Tool (LSAT)
- 4 Заключение

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 HP-UX, 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, устанавливайте или нет утилиту, разработчики подготовили маленькое демо, 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 lynis -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
```

```
[root@linuxtechi lynis]# ./lynis audit system --wait

[ Lynis 2.6.4 ]

#####
Lynis comes with ABSOLUTELY NO WARRANTY. This is free software, and you are
welcome to redistribute it under the terms of the GNU General Public License.
See the LICENSE file for details about using this software.

2007-2018, CISOfy - https://cisofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)
#####

[+] Initializing program
-----
- Detecting OS... [ DONE ]
- Checking profiles... [ DONE ]

-----
Program version: 2.6.4
Operating system: Linux
Operating system name: CentOS
Operating system version: CentOS Linux release 7.2.1511 (Core)
Kernel version: 3.10.0
Hardware platform: x86_64
Hostname: linuxtechi
-----
Profiles: /usr/local/lynis/lynis/default.prf
Log file: /var/log/lynis.log
Report file: /var/log/lynis-report.dat
Report version: 1.0
Plugin directory: ./plugins
-----
Auditor: [Not Specified]
Language: en
Test category: all
```

Инициализация тестов

```
Auditor: [Not Specified]
Language: en
Test category: all
Test group: all
-----
- Program update status... [ SKIPPED ]

[+] System Tools
-----
- Scanning available tools...
- Checking system binaries...

[+] Plugins (phase 1)
-----
Note: plugins have more extensive tests and may take several minutes to complete
- Plugins enabled [ NONE ]

[+] Boot and services
-----
- Service Manager [ systemd ]
- Checking UEFI boot [ DISABLED ]
- Checking presence GRUB2 [ FOUND ]
- Checking for password protection [ OK ]
- Check running services (systemctl) [ DONE ]
  Result: found 17 running services
- Check enabled services at boot (systemctl) [ DONE ]
  Result: found 19 enabled services
- Check startup files (permissions) [ OK ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]
```

Результаты тестов из группы System Tool and Boot & Services

```
[+] Kernel
-----
- Checking default runlevel                [ runlevel 3 ]
- Checking CPU support (NX/PAE)            [ FOUND ]
- CPU support: PAE and/or NoeXecute supported [ DONE ]
- Checking kernel version and release      [ DONE ]
- Checking kernel type                     [ DONE ]
- Checking loaded kernel modules
  Found 82 active modules
- Checking Linux kernel configuration file  [ FOUND ]
- Checking default I/O kernel scheduler    [ FOUND ]
- Checking core dumps configuration        [ DISABLED ]
- Checking setuid core dumps configuration [ DEFAULT ]
- Check if reboot is needed                [ NO ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]

[+] Memory and Processes
-----
- Checking /proc/meminfo                    [ FOUND ]
- Searching for dead/zombie processes      [ OK ]
- Searching for IO waiting processes       [ OK ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]
```

Результаты тестов из группы Kernel and Memory & Process auditing

```
[+] Users, Groups and Authentication
-----
- Administrator accounts                  [ OK ]
- Unique UIDs                            [ OK ]
- Consistency of group files (grpck)      [ OK ]
- Unique group IDs                       [ OK ]
- Unique group names                     [ OK ]
- Password file consistency              [ OK ]
- Query system users (non daemons)       [ DONE ]
- NIS+ authentication support             [ NOT ENABLED ]
- NIS authentication support             [ NOT ENABLED ]
- sudoers file                           [ FOUND ]
- Check sudoers file permissions          [ OK ]
- PAM password strength tools             [ OK ]
- PAM configuration file (pam.conf)       [ NOT FOUND ]
- PAM configuration files (pam.d)         [ FOUND ]
- PAM modules                            [ FOUND ]
- Checking user password aging (minimum) [ DISABLED ]
- User password aging (maximum)          [ DISABLED ]
- Checking expired passwords              [ OK ]
- Checking Linux single user mode authentication [ OK ]
- Determining default umask
  - umask (/etc/profile and /etc/profile.d) [ SUGGESTION ]
  - umask (/etc/login.defs)                [ OK ]
  - umask (/etc/init.d/functions)          [ SUGGESTION ]
- LDAP authentication support            [ NOT ENABLED ]
- Logging failed login attempts          [ DISABLED ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]
```

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 сохраняться 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](#)

```
[+] Initializing program
-----

Usage: lynis command [options]

Command:

  audit
    audit system          : Perform local security scan
    audit system remote <host> : Remote security scan
    audit dockerfile <file>  : Analyze Dockerfile

  show
    show                  : Show all commands
    show version          : Show Lynis version
    show help             : Show help

  update
    update info           : Show update details

Options:

  --no-log                : Don't create a log file
  --pentest               : Non-privileged scan (useful for pentest)
  --profile <profile>     : Scan the system with the given profile file
  --quick (-Q)           : Quick mode, don't wait for user input

Layout options
  --no-colors             : Don't use colors in output
  --quiet (-q)            : No output
  --reverse-colors        : Optimize color display for light backgrounds

Misc options
```

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.**

```
[+] Boot and services
-----
- Service Manager [ SysV Init ]
- Checking UEFI boot [ DISABLED ]
- Checking presence GRUB [ OK ]
- Checking presence GRUB2 [ FOUND ]
- Checking for password protection [ WARNING ]
- Check services at startup (rc2.d) [ DONE ]
  Result: found 8 services
- Check startup files (permissions) [ OK ]

[+] Kernel
-----
- Checking default run level [ RUNLEVEL 2 ]
- Checking CPU support (NX/PAE)
  CPU support: PAE and/or NoeXecute supported [ FOUND ]
- Checking kernel version and release [ DONE ]
- Checking kernel type [ DONE ]
- Checking loaded kernel modules [ DONE ]
  Found 50 active modules
- Checking Linux kernel configuration file [ FOUND ]
- Checking default I/O kernel scheduler [ FOUND ]
- Checking for available kernel update [ OK ]
- Checking core dumps configuration [ DISABLED ]
  - Checking setuid core dumps configuration [ PROTECTED ]
- Check if reboot is needed [ YES ]

[+] Memory and Processes
-----
- Checking /proc/meminfo [ FOUND ]
- Searching for dead/zombie processes [ OK ]
- Searching for IO waiting processes [ OK ]

[+] Users, Groups and Authentication
-----
- Administrator accounts [ OK ]
```

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.

```
-[ Lynis 2.6.4 Results ]-  
  
Warnings (1):  
-----  
! Found some information disclosure in SMTP banner (OS or software name) [MAIL-8818]  
  https://cisofy.com/controls/MAIL-8818/  
  
Suggestions (38):  
-----  
* Configure minimum password age in /etc/login.defs [AUTH-9286]  
  https://cisofy.com/controls/AUTH-9286/  
  
* Configure maximum password age in /etc/login.defs [AUTH-9286]  
  https://cisofy.com/controls/AUTH-9286/  
  
* Default umask in /etc/profile or /etc/profile.d/custom.sh could be more strict (e.g. 027) [AUTH-9328]  
  https://cisofy.com/controls/AUTH-9328/  
  
* To decrease the impact of a full /home file system, place /home on a separated partition [FILE-6310]  
  https://cisofy.com/controls/FILE-6310/  
  
* To decrease the impact of a full /tmp file system, place /tmp on a separated partition [FILE-6310]  
  https://cisofy.com/controls/FILE-6310/  
  
* To decrease the impact of a full /var file system, place /var on a separated partition [FILE-6310]  
  https://cisofy.com/controls/FILE-6310/  
  
* Disable drivers like USB storage when not used, to prevent unauthorized storage or data theft [STRG-1840]  
  https://cisofy.com/controls/STRG-1840/  
  
* Disable drivers like firewire storage when not used, to prevent unauthorized storage or data theft [STRG-1846]  
  https://cisofy.com/controls/STRG-1846/  
  
* Check DNS configuration for the dns domain name [NAME-4028]  
  https://cisofy.com/controls/NAME-4028/
```

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
```

```
-[ Lynis 2.6.4 Results ]-

Great, no warnings

No suggestions

=====

Lynis security scan details:

Hardening index : 100 [#####]
Tests performed : 18
Plugins enabled : 0

Components:
- Firewall [X]
- Malware scanner [X]

Lynis Modules:
- Compliance Status [?]
- Security Audit [V]
- Vulnerability Scan [V]

Files:
- Test and debug information : /var/log/lynis.log
- Report data : /var/log/lynis-report.dat

=====
```

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:

```
root@kali:~/Desktop/lunar# ./lunar.sh

Usage: ./lunar.sh -[a|A|s|S|d|p|c|l|h|c|V] -[u]

-a: Run in audit mode (for Operating Systems - no changes made to system)
-A: Run in audit mode (for Operating Systems - no changes made to system)
    [includes filesystem checks which take some time]
-w: Run in audit mode (for AWS - no changes made to system)
-d: Run in audit mode (for Docker - no changes made to system)
-x: Run in recommendations mode (for AWS - no changes made to system)
-s: Run in selective mode (only run tests you want to)
-R: Print information for a specific test
-S: List all UNIX functions available to selective mode
-W: List all AWS functions available to selective mode
-D: List all Docker functions available to selective mode
-l: Run in lockdown mode (for Operating Systems - changes made to system)
-L: Run in lockdown mode (for Operating Systems - changes made to system)
    [includes filesystem checks which take some time]
-c: Show changes previously made to system
-p: Show previously versions of file
-u: Undo lockdown (for Operating Systems - changes made to system)
-h: Display usage
-V: Display version
-v: Verbose mode [used with -a and -A]
    [Provides more information about the audit taking place]
```

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
```

```

root@root-kpf01: ~
./lunar.sh -d audit_apache

# SYSTEM INFORMATION:

Platform: i386
Vendor: Apple
Name: Darwin
Version: 10.12
Update: 3

Checking: If node is managed
Notice: Node is not managed

# Module: audit_apache

# Solaris:

# The action in this section describes disabling the Apache 1.x and 2.x web
# servers provided with Solaris 10. Both services are disabled by default.
# Run control scripts for Apache 1 and the NCA web servers still exist,
# but the services will only be started if the respective configuration
# files have been set up appropriately, and these configuration files do not
# exist by default.
# Even if the system is a Web server, the local site may choose not to use
# the Web server provided with Solaris in favor of a locally developed and
# supported Web environment. If the machine is a Web server, the administrator
# is encouraged to search the Web for additional documentation on Web server
# security.

# Linux:

# HTTP or web servers provide the ability to host web site content.
# The default HTTP server shipped with CentOS Linux is Apache.
# The default HTTP proxy package shipped with CentOS Linux is squid.
# Unless there is a need to run the system as a web server, or a proxy it is
# recommended that the package(s) be deleted.

# Refer to Section(s) 3.11,14 Page(s) 66-9 CIS CentOS Linux 6 Benchmark v1.0.0
# Refer to Section(s) 2.2.10 Page(s) 110 CIS Ubuntu Linux 16.04 Benchmark v1.0.0
# Refer to Section(s) 3.11,14 Page(s) 79-81 CIS RHEL 5 Benchmark v2.1.0
# Refer to Section(s) 3.11,14 Page(s) 69-71 CIS RHEL 6 Benchmark v1.2.0
# Refer to Section(s) 2.2.10,13 Page(s) 110,113 CIS RHEL 7 Benchmark v2.1.0
# Refer to Section(s) 6.10,13 Page(s) 59,61 CIS SLES 11 Benchmark v1.0.0
# Refer to Section(s) 2.4.14.7 Page(s) 56-7 CIS OS X 10.5 Benchmark v1.1.0

```

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
```

```
#####  
# Unofficial CIS Audit Script ^Tested on RHEL 6,7... CentOS 6,7 ^#  
#####  
# For best results, run as ROOT. Always be ROOT. "Evil grin"  
# https://github.com/Xalife/CIS-Audit  
#  
#####  
  
Scan started at:  
Tue Mar 21 18:01:18 EAT 2017  
  
##### Sysinfo #####  
Kernel info:  
Linux Thinkpad 4.4.0-53-generic #74-Ubuntu SMP Fri Dec 2 15:59:10 UTC 2016 x86_64 GNU/Linux  
  
Kernel versions:  
Linux version 4.4.0-53-generic (build@lcy01-28) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4) ) #74-Ubuntu SMP Fri Dec 2 15:59:10 UTC 2016  
  
Specific release information:  
DISTRIB_ID=LinuxMint  
DISTRIB_RELEASE=18.1  
DISTRIB_CODENAME=serena  
DISTRIB_DESCRIPTION="Linux Mint 18.1 Serena"  
NAME="Linux Mint"  
VERSION="18.1 (Serena)"  
ID=linuxmint  
ID_LIKE=ubuntu  
PRETTY_NAME="Linux Mint 18.1"  
VERSION_ID="18.1"  
HOME_URL="http://www.linuxmint.com/"  
SUPPORT_URL="http://forums.linuxmint.com/"  
BUG_REPORT_URL="http://bugs.launchpad.net/linuxmint/"  
VERSION_CODENAME=serena
```

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 —reginfo);
- 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, криптооров, 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
--noprocsan    Skip the process scan
--nofilescan   Skip the file scan
--noindicator  Do not show a progress indicator
--reginfo      Do check for Reginfo virtual file system
--onlyrelevant Only print warnings or alerts
--nolog       Don't write a local log file
```

```
(C) Florian Roth
December 2016
Version 0.18.0

DISCLAIMER - USE AT YOUR OWN RISK

[NOTICE] Starting Loki Scan SYSTEM: PROMETHEUS TIME: 20161210T00:14:37Z PLATFORM: windows
[INFO] File Name Characteristics initialized with 1649 regex patterns
[INFO] C2 server indicators initialized with 19817 elements
[INFO] Malicious MD5 Hashes initialized with 11753 hashes
[INFO] Malicious SHA1 Hashes initialized with 4311 hashes
[INFO] Malicious SHA256 Hashes initialized with 11865 hashes
[INFO] False Positive Hashes initialized with 30 hashes
[INFO] Processing YARA rules folder X:\Workspace\Loki\./signature-base/yara
[INFO] Initializing Yara rule apt_alienspy_rat.yar
[INFO] Initializing Yara rule apt_apt17_malware.yar
[INFO] Initializing Yara rule apt_apt28.yar
[INFO] Initializing Yara rule apt_apt30_backspace.yar
[INFO] Initializing Yara rule apt_apt6_malware.yar
[INFO] Initializing Yara rule apt_backdoor_ssh_python.yar
[INFO] Initializing Yara rule apt_backspace.yar
```

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
```

```
Jan 24 17:51:12 PROMETHEUS LOKI: LOKI - Starting Loki Scan on PROMETHEUS
Jan 24 17:51:12 PROMETHEUS LOKI: File Name Characteristics initialized with 34 regex patterns
Jan 24 17:51:12 PROMETHEUS LOKI: File Name Suspicious Characteristics initialized with 51 regex patterns
Jan 24 17:51:12 PROMETHEUS LOKI: Malware Hashes initialized with 43 hashes
Jan 24 17:51:12 PROMETHEUS LOKI: False Positive Hashes initialized with 8 hashes
Jan 24 17:51:12 PROMETHEUS LOKI: Successfully compiled Yara rules from file thor-hacktools.yar
Jan 24 17:51:12 PROMETHEUS LOKI: Successfully compiled Yara rules from file thor-webshells.yar
Jan 24 17:51:12 PROMETHEUS LOKI: Successfully compiled Yara rules from file yara_rules.yar
Jan 24 17:51:12 PROMETHEUS LOKI: Scanning C:\$Recycle.Bin ...
Jan 24 17:51:15 PROMETHEUS LOKI: Yara Rule MATCH: Amplia_Security_Tool FILE: C:\$Recycle.Bin\S-1-5-21-94966
Jan 24 17:51:15 PROMETHEUS LOKI: Yara Rule MATCH: WindowsCredentialEditor FILE: C:\$Recycle.Bin\S-1-5-21-94
Jan 24 17:51:15 PROMETHEUS LOKI: Yara Rule MATCH: Amplia_Security_Tool FILE: C:\$Recycle.Bin\S-1-5-21-94966
Jan 24 17:51:15 PROMETHEUS LOKI: Yara Rule MATCH: HackTool_Samples FILE: C:\$Recycle.Bin\S-1-5-21-949666807
Jan 24 17:51:15 PROMETHEUS LOKI: Yara Rule MATCH: HackTool_Producers FILE: C:\$Recycle.Bin\S-1-5-21-9496668
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: Amplia_Security_Tool FILE: C:\$Recycle.Bin\S-1-5-21-94966
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: WindowsCredentialEditor FILE: C:\$Recycle.Bin\S-1-5-21-94
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: Amplia_Security_Tool FILE: C:\$Recycle.Bin\S-1-5-21-94966
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: WindowsCredentialEditor FILE: C:\$Recycle.Bin\S-1-5-21-94
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: Amplia_Security_Tool FILE: C:\$Recycle.Bin\S-1-5-21-94966
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: WindowsCredentialEditor FILE: C:\$Recycle.Bin\S-1-5-21-94
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: HackTool_Producers FILE: C:\$Recycle.Bin\S-1-5-21-9496668
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: WindowsCredentialEditor FILE: C:\$Recycle.Bin\S-1-5-21-94
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: Amplia_Security_Tool FILE: C:\$Recycle.Bin\S-1-5-21-94966
Jan 24 17:51:16 PROMETHEUS LOKI: Yara Rule MATCH: WCE_Modified_1_1014 FILE: C:\$Recycle.Bin\S-1-5-21-949666
Jan 24 17:51:17 PROMETHEUS LOKI: Yara Rule MATCH: WindowsCredentialEditor FILE: C:\$Recycle.Bin\S-1-5-21-94
Jan 24 17:51:17 PROMETHEUS LOKI: Yara Rule MATCH: Amplia_Security_Tool FILE: C:\$Recycle.Bin\S-1-5-21-94966
Jan 24 17:51:17 PROMETHEUS LOKI: Yara Rule MATCH: HackTool_Producers FILE: C:\$Recycle.Bin\S-1-5-21-9496668
Jan 24 17:51:17 PROMETHEUS LOKI: Yara Rule MATCH: jsp_reverse.jsp FILE: C:\$Recycle.Bin\S-1-5-21-949666807-
Jan 24 17:51:17 PROMETHEUS LOKI: Yara Rule MATCH: cmdjsp.jsp FILE: C:\$Recycle.Bin\S-1-5-21-949666807-30978
Jan 24 17:51:17 PROMETHEUS LOKI: Yara Rule MATCH: Amplia_Security_Tool FILE: C:\$Recycle.Bin\S-1-5-21-94966
Jan 24 17:51:17 PROMETHEUS LOKI: Yara Rule MATCH: Mimikatz_SampleSet_1 FILE: C:\$Recycle.Bin\S-1-5-21-94966
```

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.

```
DISCLAIMER - USE AT YOUR OWN RISK

[INFO] LOKI - Starting Loki Scan on PROMETHEUS
[INFO] File Name Characteristics initialized with 34 regex patterns
[INFO] File Name Suspicious Characteristics initialized with 51 regex patterns
[INFO] Malware Hashes initialized with 43 hashes
[INFO] False Positive Hashes initialized with 8 hashes
[INFO] Successfully compiled Yara rules from file thor-hacktools.yar
[INFO] Successfully compiled Yara rules from file thor-webshells.yar
[INFO] Successfully compiled Yara rules from file yara_rules.yar
[INFO] Scanning M:\sonstige3 ...
[ALERT] Yara Rule MATCH: Amplia_Security_Tool FILE: M:\sonstige3\getlsasrvaddr.exe
[ALERT] Yara Rule MATCH: Amplia_Security_Tool FILE: M:\sonstige3\md5.csv
[ALERT] Yara Rule MATCH: WindowsCredentialEditor FILE: M:\sonstige3\wce.exe
[ALERT] Yara Rule MATCH: Amplia_Security_Tool FILE: M:\sonstige3\wce.exe
[ALERT] Yara Rule MATCH: WCE_Modified_1_1014 FILE: M:\sonstige3\wce.exe
[ALERT] Yara Rule MATCH: WindowsCredentialEditor FILE: M:\sonstige3\wce64.exe
[ALERT] Yara Rule MATCH: Amplia_Security_Tool FILE: M:\sonstige3\wce64.exe
[ALERT] Yara Rule MATCH: WCE_Modified_1_1014 FILE: M:\sonstige3\wce64.exe
[RESULT] INDICATORS DETECTED!
[RESULT] Loki recommends a forensic analysis and triage with a professional triage tool like THOR APT Scanner.
```

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!