Текстовый редактор Sublime Text 3

Внезапно открыл для себя **Sublime Text 3**.

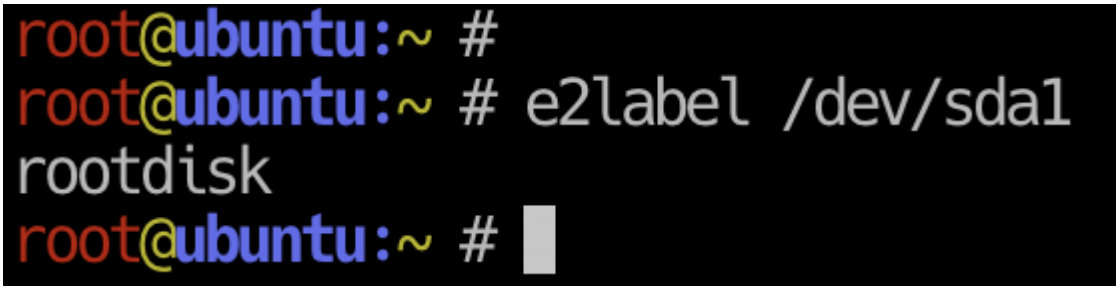
Слышал о нём раньше, и даже пробовал пару раз — как-то не впечатлялся. А вот в этот раз, когда каждый день почти разрабатываю что-то на Ansible или скрипты на Python, так очень даже нравится!

Буду обновлять этот пост в ближайшем будущем, расскажу про свои настройки и плагины, которыми я решил пользоваться.

Ссылки по теме

- Сайт Sublime Text
- Основной сайт **Unix Tutorial**
- Ansilbe

Проверяем и меняем метки файловых систем с e2label



Команда e2label проверяет метки файловых систем ext2/ext3/ext4

Некоторые (особенно не самые свежие) дистрибутивы Linux предпочитают использовать метки для инентификации файловых систем – примерно так же, как и уникальные идентификаторы (UUIDs) или имена устройств.

Когда-то давно я уже рассказывал, что команда tune2fs может менять метки, но для файловых сисетм ext2/ext3/ext4 есть более простой способ: **e2label command**.

Как узнать метку файловой системы ext2/ext3/ext4 с e2label

Просто запускаем команду **tune2fs** и указываем имя диска:

root@ubuntu:~ # **e2label /dev/sda1**

Как сменить метку с помощью e2label

Если запустить ту же команду и указать метку, то она будет присвоена этому диску:

```
root@ubuntu:~ # e2label /dev/sda1 rootdisk
root@ubuntu:~ # e2label /dev/sda1
rootdisk
```

Я завтра я покажу, **как использовать метки файловых систем в /etc/fstab**.

Ссылки

- Команда tune2fs
- **Как использовать метки файловых систем в /etc/fstab**
- Освобождаем резервное пространство с помощью tune2fs

Вышел Tower 4.0 для Mac

Вышел Tower 4.0

Отлично! Вышла новая версия графического клиента Tower 4.0 для работы с git. Куча багфиксов и основное улучшение релиза — умная функция отмены (исправления) ошибок.

Я — не разработчик софта, но всё чаще пишу и храню в git системах код. Обычно это инфраструктура (Infrastructure As A Code), но и скрипты на Python стали появляться.

Я уже где-то два года изредка пользуюсь **Tower**, и каждый раз это сплошное удовольствие. Пробовал несколько других клиентов, и по-прежнему активно сижу в командной строке git, но на macOS у Tower просто нет конкуренции, на мой взгляд.

Улучшения в Tower 4.0

Самое главное улучшение в этом релизе — **умная и удобная отмена ошибки по Cmd+Z** — это значит, что Tower выполнит цепочк из нативных git команд, чтобы отменить какое-то из последних ваших действий.

Для моего (всё ещё начального) уровня git, самые частые и

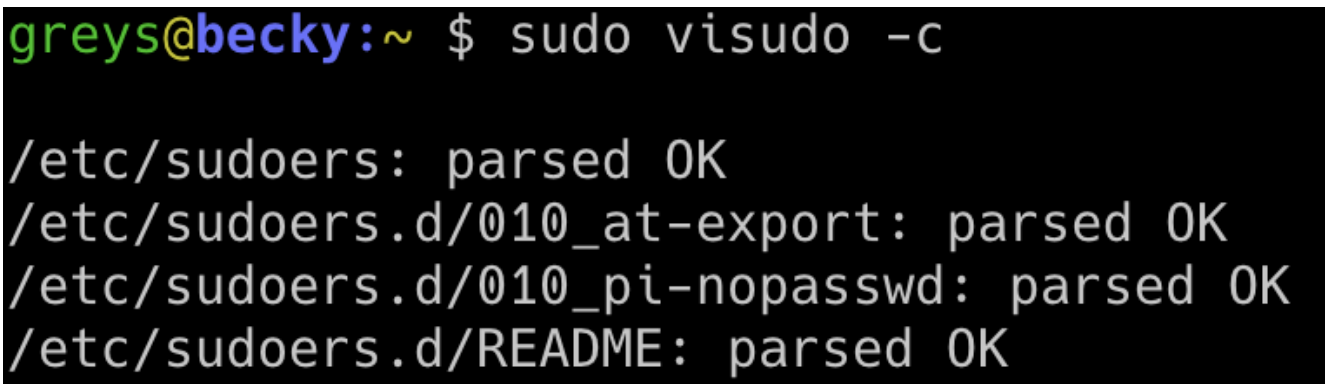
полезные (если их отменить) ошибки будут вот эти:

- Отмена удаления ветки или тэга
- Отмена удаления файлов
- Отмета коммитов
- Отмена публикации ветки в remote
- Отмена добавления/исключения локальных изменений (staging)

Ссылки

- Tower 4.0 announcement — на сайте издателя Tower

Как проверить конфиг sudo с помощью visudo



Проверка правильности конфигов sudo с помощью visudo

Я как раз скоро опубликую более подробный взгляд на решение и

отладку проблем с sudo, а пока вот такая небольшая заметка про полезный функционал команды visudo.

ВАЖНО: всегда старайтесь редактировать sudoers из интерактивной оболочки – то есть у вас есть root шелл, и если даже sudoers испытывает проблемы – не нужно на него полагаться, чтобы что-то выполнить или исправить с правами супер-пользователя.

Два основных метода работы visudo

Основной метод работы **visudo** – интерактивный: запускаем команду, она открывает редактор файла **/etc/sudoers**, потом применяет наши изменения в основную конфигурацию sudo.

А во второй метод – тоже очень полезный! Файлы мы не редактируем, а только проверяем на синтаксическую правильность – вот об этом я сегодня и расскажу.

Проверка /etc/sudoers с помощью visudo

Запускаем команду visudo с опцией -c и получаем отчёт о главном конфиге – **/etc/sudoers** и всех других файлах, которые он может подгружать из **/etc/sudoers.d**:

```
greys@becky:~ # visudo -c
/etc/sudoers: parsed OK
```

```
/etc/sudoers.d/010_at-export: parsed OK
/etc/sudoers.d/010_pi-nopasswd: parsed OK
/etc/sudoers.d/README: parsed OK
```

Как выглядит ошибка, найденная visudo

Вот так она выглядит:

```
root@becky:~ # visudo -c
/etc/sudoers: syntax error near line 10 <<<
       parse error in /etc/sudoers near line 10
```

Как видно из этого примера, я запускаю visudo уже из активной сессии root. Так что, видя ошибку, я могу её попытаться исправить прямым редактированием файла /etc/sudoers (или какой там ещё файл найдёт visudo) — именно потому, что мы уже являемся супер-пользователем и не зависим от работоспособности **sudo**.

В этом примере, что-то неправильно в основном файле **/etc/sudoers**, в 10й строке.

ВАЖНО: после исправления ошибок, снова запустите visudo -c, чтобы убедиться: больше ошибок нет. Финальной проверкой должен стать отдельный логин в систему и попытка использования sudo. НИ В КОЕМ СЛУЧАЕ не закрывайте текущую сессию root, потому что она по-прежнему может быть единственным способом что-то исправлять.

Ссылки

- sudo command (пока на английском)
- sudo meaning (тоже на английском)
- visudo command (увы — не успел ещё перевести)
- How To Use visudo (как пользоваться командой visudo)

Проект: Защищаем вебсайт паролем в nginx

Логотип веб-сервера nginx

Нужно было оставить один из старых сайтов в онлайне, но ограничить к нему доступ, чтобы уменьшить вероятность взлома (старая CMS, и т.д.) — так что я решил закрыть сайт паролем.

Что это за nginx?

nginx (произносится Энжин-Икс) это веб-сервер, обратный прокси и мощный кэш для поддержки и раздачи трафика, он испольщуется в одних из самых крупных сайтов сегодняшнего Интернета. **nginx** — очень легковесный сервер, может раздавать внушительные объёмы данных при массивной параллельности запросов и минимальных требованиях к севреру, где **nginx** запущен.

Компания **nginx** была приобретена компанией **F5** в 2019 году.

Я буду очень много рассказывать про **nginx** в 2020м году — уже давно им пользуюсь в своих серверах и решениях, так что буду использовать каждую возможность задокументировать мои настройки и поделиться опытом.

Защита сайтов паролем в nginx

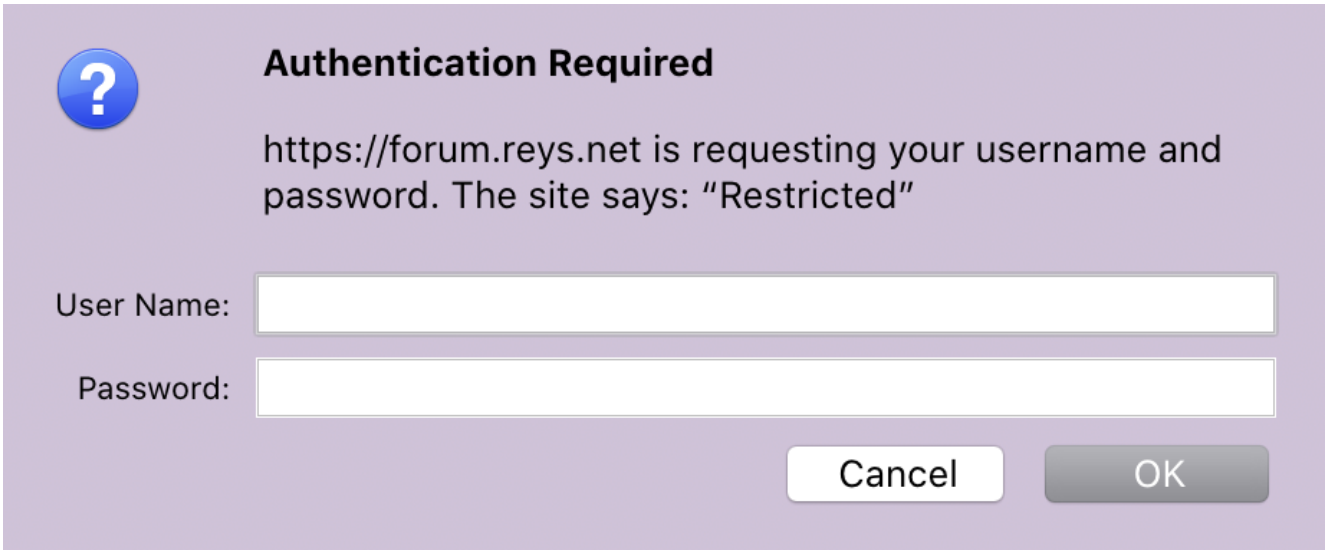
Есть несколько простых шагов для защиты сайта паролем в **nginx** (очень похожие шаги, но с чуть иным синтаксисом, используются в другом веб-сервере — Apache):

1. Решить, будем мы создавать или обновлять файл с паролями. И как он будет называться.
2. Решить, какие имя и пароль мы будем использовать для защиты
3. Сгенерировать шивроанный хэш на основе пароля, добавать его в файл с паролями
4. Обновить конфигурацию вебсервера, чтобы активировать

защиту

Сайты – это всего лишь папки с файлами на каких-то серверах. Поэтому при включении защиты паролем, можно с одинаковой лёгкостью закрыть весь сайт – например, www.unixtutorial.org или только под-раздел типа www.unixtutorial.org/images.

ИНТЕРЕСНО: хотя и говорят обычно “закрыть сайт паролем”, на самом деле защита требует двух элементов: имени пользователя и правильного к нему пароля. Так что при успешной защите сайта мы видим диалоговое окно вроде такого, спрашивающего имя пользователя и пароль:



Запрос имени пользователя и пароля для доступа к сайту

Файл с паролями и настройка логина

Большинство доступа контролируется файлами паролей, обычно они называются htpasswd. Можно создать общий файл **/etc/nginx/htpasswd** или конкретный файл для каждого из сайтов, типа **/etc/nginx/unixtutorial.htpasswd**.

Файл с паролями – обычный текстовый. Можно создать даже так:

```
# touch /etc/nginx/unixtutorial.htpasswd
```

Но ещё лучше – воспользоваться командой htpasswd. Она входит в состав Apache tools, так что может понадобиться установить соответствующий пакет:

```
$ sudo yum install httpd-tools
```

Когда мы запускаем команду htpasswd, нужно указывать два параметра: имя файла с паролями и имя пользователя, для коготорого мы хотим создать пароль.

Если указан несуществующий файл, мы получим ошибку и подсказку:

```
$ sudo htpasswd /etc/nginx/htpasswd unixtutorial
htpasswd: cannot modify file /etc/nginx/htpasswd; use '-c' to create it.
```

И да, используя опцию -c (от английского create – создавать), мы можем создать файл и добавить в него пароль для пользователя **unixtutorial**:

```
$ sudo htpasswd -c /etc/nginx/htpasswd unixtutorial
New password:
Re-type new password:
Adding password for user unixtutorial
```

Если посмотреть теперь в файл /etc/nginx/passwd, там окажется наш пользователь unixtutorial и хэш его пароля:

```
$ cat /etc/nginx/htpasswd
unixtutorial:$apr1$bExTryjo/$uxRop/uv5UwXvWl4EM5gv0
```

ВАЖНО: самих паролей в этом файле нет, но хэши могут быть использованы для подбора паролей, так что рекомендую защищать эти файлы, как и любую другую важную информацию.

Активация защиты вебсайта паролем

Мой сайт настраивается вот таким фрагментом конфигурации **nginx**:

```
server {
     listen      *:80;
     server_name forum.reys.net;
     keepalive_timeout    60;

       access_log  /var/log/nginx/forum.reys.net/access.log
multi_vhost;
     error_log /var/log/nginx/forum.reys.net/error.log;

location / {
     include "/etc/nginx/includes/gzip.conf";
     proxy_pass  http://172.31.31.47:80;

     include "/etc/nginx/includes/proxy.conf";
     include "/etc/nginx/includes/headers.conf";
     }
}
```

Защита паролем делается на уровне подраздела (location). В моём случае location / означает, что весь сайт (а не подраздел) будет защищён паролем.

Так что прямо перед элементом **proxy_pass** я добавляю защиту паролем:

```
auth_basic "Restricted";
auth_basic_user_file /etc/nginx/htpasswd;
```

Как видно, мы указали имя файла с паролями, который только что создали. Есть ещё параметр для auth_basic, можно заменить слово Restricted на любое другое текстовое сообщение – его будет видно в диалоге запроса имени-пароля при доступе к сайту.

Вот как будет выглядеть это дело:

```
location / {
     include "/etc/nginx/includes/gzip.conf";
     proxy_pass  http://172.31.31.47:80;

     auth_basic "Restricted";
     auth_basic_user_file /etc/nginx/htpasswd;

     include "/etc/nginx/includes/proxy.conf";
     include "/etc/nginx/includes/headers.conf";
     }
```

Сохраняем файл и перезапускаем **nginx**:

```
$ sudo service restart nginx
```

Порядок! Теперь сайт **https://forum.reys.net** защищён паролем!

Ссылки

- Официальный сайт nginx
- **Проекты Unix Tutorial**
- Install Ubuntu on Dell XPS laptop (моя статья на английском)
- Project: NAS storage with Helios 4 (моя статья на анлийском)

Запуск задач в Ansible под конкретную версию операционки

Red Hat Ansible

Я уже рассказывал как-то, что в Ansible довольно легко указать, что конкретная задача должна выполняться только в нужном нам дистрибутиве – Debian/Ubuntu или RedHat/Centos). Но недавно мне довелось решить вопрос поинтереснее: как запускать задачи в рамках дистрибутива, но для разных релизов – RHEL 7, но не RHEL 8, например.

Как запускать задачи Ansible только для RedHat или Debian

Для начала, вспомним как запускать задачи для разных дистрибутивов:

```
 - name: Disable IPv6
 template:
 src: templates/disable-ipv6.conf.j2
 dest: /etc/sysctl.d/disable-ipv6.conf
 mode: 0660
 backup: yes
 notify:
 disable ipv6
 tags:
 fix
 sysctl
 noipv6
 when: ansible_os_family == 'RedHat'

 name: Debian | blacklist ipv6 in modprobe
    lineinfile:    "dest=/etc/modprobe.d/blacklist.conf
line='blacklist ipv6' create=yes"
 tags:
 fix
```

```
sysctl
noipv6
when: ansible_os_family == 'Debian'
```

В этом примере, первая задача (Disable IPv6) только сработает в системах семейства RedHat (CentOS, Fedora).

А вторая задача из примера будет запускаться только на системах Debian/Ubuntu.

Как запускать задачу в Ansible для конкретного релиза RedHat

А вот с какой задачей я столкнулся: IPv6 можно было выключить в RHEL 6 и RHEL 7, но теперь в RHEL 8/CentOS 8 модуль IPv6 является встроенным – и его нельзя отключить. Стало быть, и пытаться не стоит.

Вот как я изменил задачу Ansible из предыдущего примера, чтобы она выполнялась только на релизах RedHat/CentOS старше RHEL 8/CentOS 8:

```
- name: Disable IPv6
template:
src: templates/disable-ipv6.conf.j2
dest: /etc/sysctl.d/disable-ipv6.conf
mode: 0660
backup: yes
notify:
disable ipv6
tags:
```

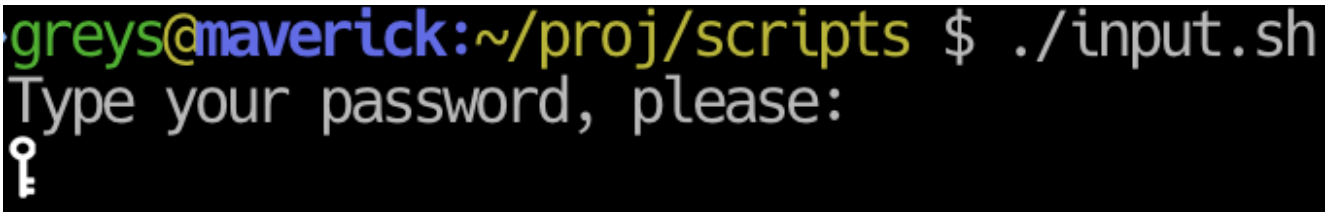
```
 fix
 sysctl
 noipv6
   when:   ansible_os_family   ==   'RedHat'   and
ansible_distribution_major_version|int <= 7
```

Вот и всё на сегодня!

See Also

- Ansible
- Бэкап файлов в задачах Ansible

Безопасный ввод данных в скриптах bash



Мне тут нужно было написать простенький скрипт на bash, и чтобы он спрашивал пароль с клавиатуры и потом дальше передавал его другим командам по ходу дела. Оказывается, с такой задачей отлично справляется встроенная в **bash** функция **read**.

Стандартный ввод в скриптах bash

Вот как работает обычный ввод данных с клавиатуры: мы запускаем функцию read, указываем ей в в качестве параметра имя переменной. В консоли запрашивается ввод текста и когда мы его вводим, то отображается каждая буква ввода – мы видим, что печатаем.

Создаём скрипт:

$ **vi input.sh**

пока это вот такой контент:

```
#!/bin/bash
echo "Type your password, please:"
read PASS
echo "You just typed: $PASS"
```

Сохраняем это дело (жмём Esc, потом вводим **:wq**) и делаем скрипт исполняемым:

$ **chmod a+rx input.sh**

Теперь запускаем скрипт и тестируем:

```
$ ./input.sh
Type your password, please:
mypass
```

```
You just typed: mypass
```

Работает на ура, но отображать секретные пароли прямо на стадии ввода с клавиатуры — не очень секьюрно. В настоящей ситуации я и подтверждать пароль, печатая его в конце скрипта, тоже бы не стал.

Безопасный ввод в bash

Просто указываем параметр **-s** для функции **read** (правим скрипт):

```
read -s PASS
```

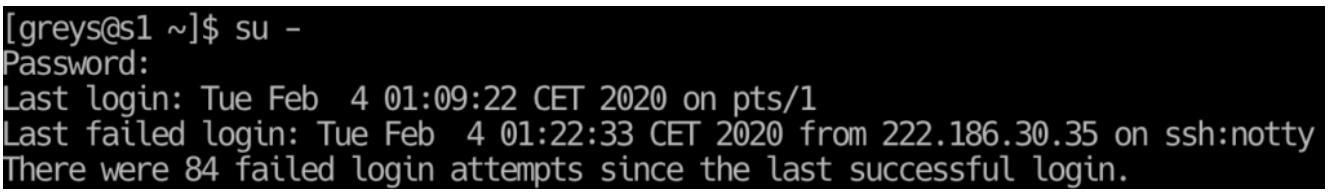
-s это от слова secure.

Сохраняем и запускаем скрипт, и теперь во время ввода данных ничего отображаться не будет — как и полагается:

```
$ ./input.sh
Type your password, please:
You just typed: mypass
```

Здорово, правда?

Неудачные попытки взлома root



Попытки неудачного взлома подбором пароля root

Надо же! Всё обстоит гораздо хуже, чем казалось!

Я тут переустанавливаю один из выделенных серверов для Tech Stack. Запустил установку **CentOS 8** на новый сервер и спустя несколько часов залогинился посмотреть:

```
[greys@s1 ~]$ su -
Password:
Last login: Tue Feb  4 01:09:22 CET 2020 on pts/1
Last failed login: Tue Feb  4 01:22:33 CET 2020 from 222.186.30.35 on ssh:notty
There were 84 failed login attempts since the last successful login.
```

То есть в среднем 6 несанкционированных попыток зайти на новый сервер по SSH под пользователем root. Это несколько сотен попыток взлома в час!

Очевидно, нужно включать key-based SSH logins и настраивать защиту SSH через fail2ban как можно скорее.

Ссылки

- Защищаем SSH с помощью fail2ban
- **SSH reference**
- Generate SSH key

Как распаковать архив XZ

Архивы XZ могут быть распакованы командами из пакета xz-utils

Мне уже не раз попадались архивы XZ — в последний раз при скачивании **Kali Linux**. В общем-то, ничего сложного в работе с ними не должно быть, но в некоторых операционках (macOS, я на тебя смотрю!) всё не совсем очевидно — так что я решил исследовать и задокументировать этот вопрос.

Что это за XZ файлы?

XZ это современный алгоритм сжатия без потери данных, даже более мощный, чем известные в Unix/Linux кругах gzip и bzip2. Оказывается, многие Linux дистрибутивы уже давно используют этот формат для компрессии данных в своих пакетах или для сжатия готовых ISO образов. XZ имеет открытый исходный код и распространяется в рамках проекта Tukaani Project – XZ.

Поддержка формата XZ в команде tar

Современные Linux дистрибутивы (уж точно Ubuntu 19.x и CentOS 7.x, которые я проверил сегодня) уже имеют пакет **xz utils** установленным по умолчанию, так что команда **tar** запросто распаковывает архивы XZ.

Если в вашей системе пакет **xz-utils** ещё не установлен, то в Ubuntu/Debian это можно исправить таким образом:

```
# apt install xz-utils
```

а в CentOS/Fedora/RedHat этот пакет ставится вот так:

```
# yum install xz
```

Поддержка XZ в командах gzip/gunzip на macOS

Хотя tar в macOS и не поддерживает формата XZ:

```
greys@mcfly:~/Downloads $ tar xzvf kali-linux-2020.1-rpi3-nexmon-64.img.xz
 tar: Error opening archive: Unrecognized archive format
```

… для распаковки прекрасно подойдёт **gunzip** (тоже есть в macOS):

```
greys@mcfly:~/Downloads $ ls -al kali-linux-2020*
-rw-r--r--@ 1 greys  staff      259647 30 Jan 22:57 kali-linux-2020-1.png
-rw-r--r--@ 1 greys  staff  1048328860 30 Jan 23:06 kali-linux-2020.1-rpi3-nexmon-64.img.xz
greys@mcfly:~/Downloads $ gunzip kali-linux-2020.1-rpi3-nexmon-64.img.xz
greys@mcfly:~/Downloads $ ls -ald kali-linux-2020.1-rpi3-nexmon-64.img
-rw-r--r--  1 greys  staff  6999999488 30 Jan 23:06 kali-linux-2020.1-rpi3-nexmon-64.img
```

xz-utils на macOS

Если уж прям очень надо именно пакет XZ utils, то поставить его на macOS можно с помощью Homebrew:

```
$ brew install xz
```

… и тогда появляются сражу же несколько команд XZ:

```
xz
xzcat
xzcmp
xzdec
xzdiff
xzegrep
xzfgrep
xzgrep
xzless
xzmore
```

Ссылки

- XZ utils website

Kali Linux 2020.1

Kali Linux 2020.1

Окей, в этот раз я точно найду время поставить Kali Linux на мою новую модель Raspberry Pi 4!

Kali Linux только что выпустили первый релиз в 2020м году, со следующими улучшениями:

- **Единый установщик** – качаешь один образ диска, и он уже предлагает выбрать окружение десктопа (Gnome, KDE, Mate или LXDE) – намного удобнее, чем качать отдельный ISO образ для каждого десктопа
- пользователь **kali** теперь полностью заменил пользователя **root** по умолчанию
- Python 2 больше не поддерживается и не включается (End Of Life)
- Улучшения тем оформления (**kali-undercover** теперь ещё больше похожа на десктоп Windows)

Просто напоминаю: **Kali Linux** это дистрибутив Linux на основе Debian, разработанный специалистами по компьютерной безопасности для проверок безопасности, тестирования устойчивости компьютерных систем (penetration testing) и анализа систем после взлома (digital forensics).

Ссылки в тему

- Kali Linux website
- Kali Linux 2019.4 release
- Дайджест Unix Tutorial за 2019й год