

Лабораторная работа №5.

Блокировка и ограничение доступа пользователей, контроль уязвимостей в системах Linux

Цель работы: Ознакомление с методами блокировки пользовательских учетных записей, ограничением доступа пользователей через PAM и другие инструменты, а также использование Trivy для контроля за уязвимостями.

1. Блокировка пользовательских учетных записей

Блокировка учетной записи может быть необходима в случае подозрительной активности или компрометации.

Способ 1. Блокировка и разблокировка пользователей с помощью команды *passwd*

Команда *passwd* работает с паролями учетной записи пользователя.

Зайдем под учетной записью суперпользователя и создадим пользователя *blockuser* с паролем *P@ssw0rd*:

```
# useradd blockuser  
# passwd blockuser
```

Чтобы заблокировать пользователя *blockuser*, запустим команду *passwd* с опцией *-l* (*--lock*):

```
# passwd -l blockuser
```

Теперь выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью *blockuser*:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser  
Password:  
su: Authentication failure
```

Зайдем под учетной записью суперпользователя.

Чтобы разблокировать пользователя *blockuser*, запустим команду *passwd* с опцией *-u* (*--unlock*):

```
# passwd -u blockuser
```

Выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью *blockuser*:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser  
Password:  
[blockuser@alt-1 ~]$
```

Способ 2. Блокировка и разблокировка пользователей с помощью команды *usermod*

Команда *usermod* используется для изменения учетных записей пользователей.

Чтобы заблокировать пользователя *blockuser*, запустим команду *usermod* с опцией *-L* (*--lock*):

```
# usermod -L blockuser
```

Теперь выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью *blockuser*:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
su: Authentication failure
```

Зайдем под учетной записью суперпользователя.

Чтобы разблокировать пользователя *blockuser*, запустим команду *usermod* с опцией *-U* (*--unlock*):

```
# usermod -U blockuser
```

Выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью *blockuser*:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
[blockuser@alt-1 ~]$
```

Способ 3. Блокировка и разблокировка пользователей с помощью команды *chage*

Команда *chage* используется для изменения информации об истечении срока действия пароля пользователя.

Используем команду *chage* для автоматической блокировки неактивного пользователя после определенного количества дней бездействия.

Заблокируем пользователя *blockuser*:

```
# chage -E 1 blockuser
```

По сути, мы установили дату истечения срока действия 2 января 1970 года.

Выведем информацию о пароле пользователя *blockuser*:

```
# chage -l blockuser
```

```
[root@alt ~]# chage -l blockuser
Последний раз пароль был изменён      : 
Срок действия пароля истекает          : никогда
Пароль будет деактивирован через        : никогда
Срок действия учётной записи истекает   : янв 02, 1970
Минимальное количество дней между сменой пароля : -1
Максимальное количество дней между сменой пароля : -1
Количество дней с предупреждением перед деактивацией пароля : -1
```

Выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью *blockuser*:

```
$ su - blockuser
```

```
[user@alt ~]$ su - blockuser
Password:
Your account has expired; please contact your system administrator.
su: User account has expired
```

Зайдем под учетной записью суперпользователя.

Разблокируем пользователя *blockuser*, удалив дату истечения срока действия учетной записи:

```
# chage -E -1 blockuser
```

Выведем информацию о пароле пользователя *blockuser*:

```
# chage -l blockuser
```

```
[root@alt ~]# chage -l blockuser
Последний раз пароль был изменён      : 
Срок действия пароля истекает          : никогда
Пароль будет деактивирован через        : никогда
Срок действия учётной записи истекает   : никогда
Минимальное количество дней между сменой пароля : -1
Максимальное количество дней между сменой пароля : -1
Количество дней с предупреждением перед деактивацией пароля : -1
```

Выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью *blockuser*:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
[blockuser@alt-1 ~]$
```

2. Ограничение возможности входа пользователей

2.1. Настройка PAM для ограничения доступа пользователей

Настройка PAM для ограничения доступа пользователей может быть выполнена с помощью изменения конфигурационных файлов, обычно расположенных в */etc/pam.d/*.

Несколько модулей PAM были разобраны в предыдущих лабораторных работах. Дополним список модулями *pam_listfile.so* и *pam_time.so*.

Пример 1. Ограничение доступа пользователей с помощью модуля `pam_listfile.so`

Запретим авторизацию пользователя *blockuser*. Для этого воспользуемся модулем *pam_listfile.so*. Данный модуль позволяет ограничить доступ для пользователей на основе файла со списком.

Зайдем под учетной записью суперпользователя.

Отредактируем файл `/etc/pam.d/system-auth-common`:

```
# vim /etc/pam.d/system-auth-common
```

Добавим следующую строку:

```
auth required pam_listfile.so onerr=succeed item=user sense=deny
file=/etc/denyusers
```

```
#%PAM-1.0
#account      required      pam_access.so
auth          required      pam_listfile.so onerr=succeed item=user sense=deny file=/etc/denyusers
session       required      pam_mkttemp.so
session       required      pam_limits.so
```

Модуль *auth* указывает на процесс аутентификации пользователя.

Флаг *required* означает, что данный модуль должен быть выполнен. Если он не пройдет, аутентификация не будет успешной, но другие модули все равно будут выполняться.

Параметр *onerr=succeed* параметр указывает, что если произойдет ошибка (например, файл не найден), то модуль будет считать, что проверка прошла успешно.

Параметр *item=user* указывает, что проверяемым элементом будет имя пользователя.

Параметр *sense=deny* указывает, что если имя пользователя найдено в файле, то доступ будет запрещен.

Параметр *file=/etc/denyusers* указывает на путь к файлу, в котором перечислены пользователи, которым запрещен доступ. Каждый пользователь должен быть записан с новой строки.

Теперь создадим файл `/etc/denyusers`:

```
# vim /etc/denyusers
```

Запишем в созданный файл пользователя *blockuser*:

```
blockuser
```

Выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью *blockuser*:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
su: Authentication failure
```

Зайдем под учетной записью суперпользователя.

Закомментируем строку в файле `/etc/pam.d/system-auth-common`:

```
# auth required pam_listfile.so onerr=succeed item=user sense=deny
file=/etc/denyusers
```

```
##PAM-1.0
#account      required      pam_access.so
#auth         required      pam_listfile.so onerr=succeed item=user sense=deny file=/etc/denyusers
session      required      pam_mktmp.so
session      required      pam_limits.so
```

Выйдем из учетной записи суперпользователя.

Попробуем еще раз зайти под учетной записью `blockuser`:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
[blockuser@alt-1 ~]$
```

Пример 2. Ограничение доступа пользователей с помощью модуля `pam_time.so`

Разрешим доступ пользователю `blockuser` только с 20:00 до 21:00 по будням. Для этого воспользуемся модулем `pam_time.so`. Модуль `pam_time.so` позволяет задавать временные ограничения для доступа пользователей.

Откроем файл `/etc/security/time.conf`:

```
# vim /etc/security/time.conf
```

Добавим следующее правило:

```
*,*;blockusers;MoTuWeThFr2000-2100
```

Отредактируем файл `/etc/pam.d/system-auth-common`:

```
# vim /etc/pam.d/system-auth-common
```

Добавим следующую строку:

```
account      required      pam_time.so
```

```
##PAM-1.0
#account      required      pam_access.so
#auth         required      pam_listfile.so onerr=succeed item=user sense=deny file=/etc/denyusers
account       required      pam_time.so
session      required      pam_mktmp.so
session      required      pam_limits.so
```

Выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью `blockuser`:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
su: Authentication failure
```

Зайдем под учетной записью суперпользователя.

Закомментируем строку в файле `/etc/pam.d/system-auth-common`:

```
#account required pam_time.so
```

```
##PAM-1.0
#account required pam_access.so
#auth required pam_listfile.so onerr=succeed item=user sense=deny file=/etc/denyusers
#account required pam_time.so
session required pam_mkttemp.so
session required pam_limits.so
```

Выйдем из учетной записи суперпользователя.

Попробуем еще раз зайти под учетной записью `blockuser`:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
[blockuser@alt-1 ~]$
```

2.2. Использование `fail2ban` для защиты от брутфорс-атак

`Fail2ban` — локальный сервис, который отслеживает логи служб, работающих на сервере, и при обнаружении подозрительной активности блокирует IP-адреса возможных злоумышленников.

Для защиты от брутфорс-атак (взломов путём перебора паролей) `Fail2ban` можно настроить, например, для защиты службы SSH.

Установим пакет `fail2ban`:

```
# apt-get update && apt-get install fail2ban
```

Запускаем и добавляем в автозагрузку сервис `fail2ban`:

```
# systemctl enable --now fail2ban
```

Защита для SSH начнет работать сразу после установки. По умолчанию `fail2ban` будет на 10 минут блокировать IP-адреса, с которых в течение 10 минут было выполнено 5 неудачных попыток авторизации.

Настройки `fail2ban` хранятся в конфигурационном файле `/etc/fail2ban/jail.conf`.

```
# cat /etc/fail2ban/jail.conf | less
```

Настройки разделены по секциям. В `[DEFAULT]` указываются общие параметры, которые определяют работу `fail2ban` в целом и применяются для большинства служб. Специфические настройки для той или иной службы прописываются внутри соответствующей секции (например, `[sshd]`, `[apache-auth]`, `[vsftpd]`).

Менять какие-либо параметры напрямую в файле `jail.conf` не рекомендуется. Вместо этого необходимо создать новый файл, разместив его в директории `/etc/fail2ban/jail.d/ssh.conf`.

Важно! Перед созданием файла необходимо переключить сбор логов на *systemd*.

Для этого отредактируем файл */etc/fail2ban/jail.conf*:

```
# vim /etc/fail2ban/jail.conf
```

Заменяем следующую строку:

```
before = paths-altlinux.conf
```

На строку:

```
before = paths-altlinux-systemd.conf
```

Теперь можно размещать свои настройки в */etc/fail2ban/jail.d/название.conf*.

Создадим файл */etc/fail2ban/jail.d/ssh.conf* и добавим в него следующие строки:

```
[sshd]
```

```
enabled = true
```

```
findtime = 120
```

```
maxretry = 3
```

```
bantime = 43200
```

Такая настройка означает, что если с какого-либо IP-адреса в течение 2 минут будут выполнены 3 неудачные попытки авторизоваться, *fail2ban* заблокирует этот адрес на 12 часов.

В секции *DEFAULT* могут быть указаны такие параметры, как период блокировки (*bantime*), количество попыток (*maxretry*) и пр.

В файле */etc/fail2ban/jail.d/ssh.conf* указаны следующие параметры:

enabled — обязательный параметр, определяющий включение или отключение секции. Для включения должно быть установлено значение *true*, оно указано по умолчанию;

bantime — продолжительность бана в секундах, то есть период, на который подозрительный IP-адрес будет заблокирован;

maxretry — количество неудачных попыток в течение периода *findtime*, после которых будет выполнена блокировка;

findtime — период в секундах, в течение которого действие (в данном случае — неудачная попытка подключения) должно повториться определенное количество раз (*maxretry*), после чего будет выполнена блокировка.

После создания файла необходимо перезапустить службу *fail2ban*:

```
# systemctl restart fail2ban
```

Информацию о работе модуля *sshd* с помощью следующей команды:

```
# fail2ban-client status sshd
```

```
[root@alt-1 ~]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|   |- Currently failed: 0
|   |- Total failed:     0
|   `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
- Actions
    |- Currently banned: 0
    |- Total banned:     0
    `-- Banned IP list:
```

Проверим работы сервиса *fail2ban*.

Попробуем подключиться к пользователю *user* с *alt-2* на *alt-1*.

Имя хоста	IP-адрес
alt-1	192.168.10.10/24
alt-2	192.168.10.20/24

Примечание! Локальная сеть между машинами настроена по умолчанию.

Важно! Предварительно необходимо проверить статус службы *sshd*:

```
# systemctl status sshd
```

Если служба не запущена, необходимо запустить ее:

```
# systemctl enable --now sshd
```

Зайдем под учетной записью суперпользователя на хосте *alt-2*.

Подключаемся по протоколу *ssh* к пользователю *user* с *alt-2* на *alt-1* и намеренно вводим неверных пароль:

```
# ssh user@192.168.10.10
```

```
[root@alt-2 ~]# ssh user@192.168.10.10
user@192.168.10.10's password:
ssh: Permission denied, please try again.
user@192.168.10.10's password:
ssh: Permission denied, please try again.
user@192.168.10.10's password:
ssh: user@192.168.10.10: Permission denied (publickey,password).
[root@alt-2 ~]# ssh user@192.168.10.10
ssh: connect to host 192.168.10.10 port 22: Connection refused
```

После 3-й неудачной попытки подключиться к пользователю *user* с *alt-2* на *alt-1* возможно будет только через 12 часов.

Дополнительно выведем информацию о работе модуля *sshd* с помощью следующей команды:

```
# fail2ban-client status sshd
```



```
[root@alt-1 ~]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|   |- Currently failed: 0
|   |- Total failed:     3
|   `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
    |- Currently banned: 1
    |- Total banned:     1
    `-- Banned IP list:  192.168.10.20
```

2.3. Использование `/sbin/nologin` для предотвращения входа некоторых пользователей

Скрипт `/sbin/nologin` используется для предотвращения входа пользователей в систему. Он обычно назначается в качестве оболочки для учетных записей, которым не требуется вход в систему. Когда пользователь пытается войти в систему с такой оболочкой, они получают сообщение о том, что вход запрещен, и сессия завершается.

С помощью команды `usermod` с опцией `-s` назначим `/sbin/nologin` в качестве оболочки для пользователя `blockuser`.

```
# usermod -s /sbin/nologin blockuser
```

Проверим изменение оболочки пользователя `blockuser` в файле `/etc/passwd`:

```
# cat /etc/passwd | grep blockuser
```

```
[root@alt-1 ~]# cat /etc/passwd | grep blockuser
blockuser:x:501:501::/home/blockuser:/sbin/nologin
```

Выйдем из учетной записи суперпользователя.

Попробуем зайти под учетной записью `blockuser`:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
This account is currently not available.
```

Зайдем под учетной записью суперпользователя.

Вернем возможность пользователю `blockuser` входить в систему.

Для этого сменим оболочку пользователя `blockuser` на `/bin/bash`:

```
# usermod -s /bin/bash blockuser
```

Проверим изменение оболочки пользователя `blockuser` в файле `/etc/passwd`:

```
# cat /etc/passwd | grep blockuser
```

```
[root@alt-1 ~]# cat /etc/passwd | grep blockuser
blockuser:x:501:501::/home/blockuser:/bin/bash
```

Выйдем из учетной записи суперпользователя.
Попробуем зайти под учетной записью *blockuser*:

```
$ su - blockuser
```

```
[user@alt-1 ~]$ su - blockuser
Password:
[blockuser@alt-1 ~]$
```

3. Контроль за уязвимостями с помощью *Trivy*

Trivy — сканер уязвимостей в образах контейнеров, файловых системах и репозиториях Git. Кроме того, *trivy* может находить ошибки в файлах конфигурации, запрограммированные конфиденциальные данные, использование несовместимых лицензий в проекте.

Установим пакет *trivy*:

```
# apt-get update && apt-get install trivy
```

Синтаксис утилиты *trivy*:

```
trivy <команда> [--scanners <сканер1,сканер2>] <цель>
```

Доступные команды:

image (i) — сканировать образ контейнера;

filesystem (fs) — сканировать локальную файловую систему;

repository (repo) — сканировать git-репозиторий (удаленно);

vm — сканировать образ виртуальной машины;

kubernetes (k8s) — сканировать кластер кubernetes;

aws — сканировать учётную запись AWS;

config — сканировать файлы конфигурации;

rootfs — сканировать rootfs;

sbom — сканировать используемые пакеты ОС и программные зависимости (SBOM);

completion — сгенерировать скрипт автозаполнения для указанной оболочки;

module — управление модулями;

plugin — управление плагинами;

server — режим сервера;

version — вывести версию.

Сканеры:

vuln — известные уязвимости (CVE) (по умолчанию);

config — проблемы с IAC и неправильные настройки;

secret — конфиденциальная информация и секреты (по умолчанию);

license — лицензии на программное обеспечение.

Для получения дополнительной информации по применению *trivy* используем опцию *--help*:

```
# trivy --help
```

Уровни уязвимостей:

UNKNOWN — уязвимости или риски, для которых нет достаточной информации для определения их уровня серьезности.

LOW — уязвимости, которые имеют минимальное влияние на безопасность системы и могут быть легко устранены.

MEDIUM — уязвимости, которые могут представлять некоторую угрозу, но требуют дополнительных условий для эксплуатации или имеют ограниченное воздействие.

HIGH — серьезные уязвимости, которые могут привести к значительным последствиям, если они будут использованы злоумышленниками.

CRITICAL — очень серьезные уязвимости, которые могут позволить злоумышленникам полностью контролировать систему или данные, требующие немедленного внимания.

Разберем несколько примеров использования сканера *Trivy*:

Отсканируем образ контейнера *alt:p10* на уязвимости:

```
# trivy image alt:p10
```

```
[root@alt-1 ~]# trivy image alt:p10
2024-10-07T19:58:55.125+0300 INFO Need to update DB
2024-10-07T19:58:55.125+0300 INFO DB Repository: registry.altlinux.org/alt/trivy-db:2
2024-10-07T19:58:55.125+0300 INFO Downloading DB...
68.07 MiB / 68.07 MiB [-----] 100.00% 1.65 MiB p/s 42s
2024-10-07T19:59:36.846+0300 INFO Vulnerability scanning is enabled
2024-10-07T19:59:36.852+0300 INFO Secret scanning is enabled
2024-10-07T19:59:36.853+0300 INFO If your scanning is slow, please try '--scanners vuln'
to disable secret scanning
2024-10-07T19:59:36.857+0300 INFO Please see also https://aquasecurity.github.io/trivy/v0
.50/docs/scanner/secret/#recommendation for faster secret detection
2024-10-07T20:00:34.977+0300 INFO Detected OS: alt
2024-10-07T20:00:34.980+0300 WARN This OS version is not on the EOL list: alt cpe:/o:alt:
container:10
2024-10-07T20:00:34.981+0300 INFO Detecting ALT vulnerabilities...
123 / 123 [-----] 100.00% 338 p/s
2024-10-07T20:00:35.547+0300 INFO Number of language-specific files: 0

alt:p10 (alt cpe:/o:alt:container:10)

Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

По результатам сканирования образ контейнера *alt:p10* не содержит известных уязвимостей.

Тег — метка, которая используется для обозначения конкретной версии или варианта образа Docker. Теги позволяют пользователям легко управлять и идентифицировать различные версии одного и того же образа.

Теги обычно имеют формат *<имя_образа>:<тег>*.

Отсканируем образ контейнера *alt:p10* на наличие уязвимостей HIGH и CRITICAL с сохранением результата в формате JSON в файл:

```
# trivy image --severity HIGH,CRITICAL -f json -o test.json alt:p10
```

```
[root@alt-1 ~]# trivy image --severity HIGH,CRITICAL -f json -o test.json alt:p10
2024-10-07T20:23:27.040+0300 INFO Vulnerability scanning is enabled
2024-10-07T20:23:27.040+0300 INFO Secret scanning is enabled
2024-10-07T20:23:27.040+0300 INFO If your scanning is slow, please try '--scanners vuln'
to disable secret scanning
2024-10-07T20:23:27.040+0300 INFO Please see also https://aquasecurity.github.io/trivy/v
0.50/docs/scanner/secret/#recommendation for faster secret detection
2024-10-07T20:23:29.373+0300 INFO Detected OS: alt
2024-10-07T20:23:29.374+0300 WARN This OS version is not on the EOL list: alt cpe:/o:alt
:container:10
2024-10-07T20:23:29.374+0300 INFO Detecting ALT vulnerabilities...
123 / 123 [-----] 100.00% ? p/s
2024-10-07T20:23:29.407+0300 INFO Number of language-specific files: 0
```

Выведем содержимое файла *test.json*:

```
# cat test.json | less
```

Отсканируем все файлы по определенному пути (в текущей директории):

```
# trivy fs ./
```

```
[root@alt-1 ~]# trivy fs ./
2024-10-07T20:35:13.465+0300 INFO Vulnerability scanning is enabled
2024-10-07T20:35:13.465+0300 INFO Secret scanning is enabled
2024-10-07T20:35:13.465+0300 INFO If your scanning is slow, please try '--scanners vuln' to disab
le secret scanning
2024-10-07T20:35:13.465+0300 INFO Please see also https://aquasecurity.github.io/trivy/v0.50/docs
/scanner/secret/#recommendation for faster secret detection
2024-10-07T20:35:13.660+0300 INFO Number of language-specific files: 0
```

4. Анализ данных из ALTRepo API

Проект *ALTRepo* предназначен для агрегации информации о дистрибутивах и пакетной базе, а также для предоставления интерфейса к базе данных, используемой в задачах разработки, тестирования и развития системы.

Все данные проекта хранятся в ALTRepo DB, доступ к которой предоставляется посредством REST API: <https://rdb.altlinux.org>.

REST API (Representational State Transfer Application Programming Interface) — архитектурный стиль для разработки сетевых приложений, который использует подход REST для взаимодействия между клиентом и сервером.

Основные характеристики REST API включают:

1. Статус вне сеанса (Stateless).

Каждый запрос от клиента к серверу должен содержать всю необходимую информацию для понимания и обработки запроса. Сервер не сохраняет никакой информации о состоянии клиента между запросами.

2. Система ресурсов (Resources).

В REST API все ключевые элементы представляются как ресурсы, которые могут быть доступны через уникальные идентификаторы (URI).

3. Использование стандартных методов HTTP.

REST API обычно использует стандартные методы HTTP для выполнения операций над ресурсами:

GET: Получение данных.

POST: Создание нового ресурса.

PUT: Обновление существующего ресурса.

DELETE: Удаление ресурса.

4. Представление ресурсов.

Ресурсы могут иметь разные представления (форматы), такие как JSON, XML и другие. Клиент и сервер могут обмениваться данными в выбранном формате.

5. Кэширование.

Ответы от сервера могут быть закешированы клиентом или промежуточными серверами, что помогает снизить нагрузку на сервер и улучшить производительность.

Перейдем в ALTRepo API и разберем его основные разделы.

Раздел	Описание
api	Основные функции API, которые могут предоставлять базовые операции с системами и сущностями ALTRepo.
auth	Методы авторизации и проверки прав доступа пользователей.
task	Методы для работы с задачами, такими как создание, обновление и получение статуса задач.
task/progress	API для отслеживания прогресса выполнения задач, например, выполнения сборок или других длительных операций.
package	API для работы с пакетами, предоставляющее информацию о доступных пакетах в репозиториях.
packageset	Методы для работы с наборами пакетов, позволяющие управлять группами пакетов и их конфигурациями.
acl	API, связанное с системой контроля доступа (ACL), управляет правами доступа к различным объектам системы.
bug	Раздел для работы с ошибками, позволяет запрашивать и отправлять информацию об ошибках, связанных с пакетами.
dependencies	API для работы с зависимостями пакетов, которое

	помогает отслеживать, какие зависимости необходимы для установки или работы пакетов.
file	Методы для работы с файлами пакетов, например, для получения информации о конкретных файлах, входящих в состав пакетов.
image	API для работы с образами, например, с дисковыми образами или другими бинарными объектами, доступными в системе.
export	Раздел, отвечающий за экспорт данных из системы. Позволяет выгружать информацию о пакетах, зависимостях и других сущностях в различных форматах.
errata	API для получения информации об исправлениях и обновлениях, предоставляющее данные о важных исправлениях и изменениях в пакетах.
vuln	Раздел для управления уязвимостями, предоставляющий информацию о найденных уязвимостях в пакетах или системах.
license	API для работы с лицензиями, которое позволяет запрашивать информацию о лицензиях на используемые пакеты.
site	Общие методы для работы с веб-сайтом или интерфейсом системы, возможно, включающие инструменты для управления внешним интерфейсом или настройками системы.

Запросим информацию о пакете *git*. Git — система контроля версий, используемая для управления проектами и совместной работы.

Ссылка на скачивание пакета:

<https://packages.altlinux.org/en/sisyphus/srpms/git/>

В ALTRepo API перейдем в раздел *package* и выберем пункт *GET: /package/package_info*.

Нажмем на кнопку “Try it out”:

GET /package/package_info

Get information for package by parameters from last packages

Parameters

Try it out

Заполним поля *package name* и *package version* и нажмем кнопку “Execute”:

Name	Description
name string (query)	package name <input type="text" value="git"/>
version string (query)	package version <input type="text" value="2.42.2"/>

Результат поиска включает в себя HTTP-запрос к API с ответом в формате JSON:

```
curl -X 'GET' \
'https://rdb.altlinux.org/api/package/package_info?name=git&version=2.42.2&arch=x86_64&source=false&full=false' -H 'accept: application/json'
```

Структура ответа:

1. Request body (тело запроса)

В разделе *request_args* указаны аргументы, с которыми был выполнен запрос. В данном случае:

name: "git" — название пакета, который запрашивался;

version: "2.42.2" — версия пакета;

arch: "x86_64" — архитектура пакета.

Другие параметры, такие как *release*, *source*, *branch*, *packager_email* и т.д., не были указаны в запросе (имеют значение null или false).

2. Response body (тело ответа)

length: 5 — число найденных пакетов (в данном случае 5 версий пакета).

В массиве *packages* содержатся подробные сведения о каждом пакете.

3. Response headers (заголовки ответа)

alt-svc — параметры альтернативного сервиса.

content-length: 6263 — длина тела ответа в байтах.

content-type: application/json — тип возвращаемого контента.

date — дата и время ответа сервера.

server: nginx — сервер, на котором выполняется запрос.

x-cache-status: MISS — кэширование на стороне сервера не было выполнено.

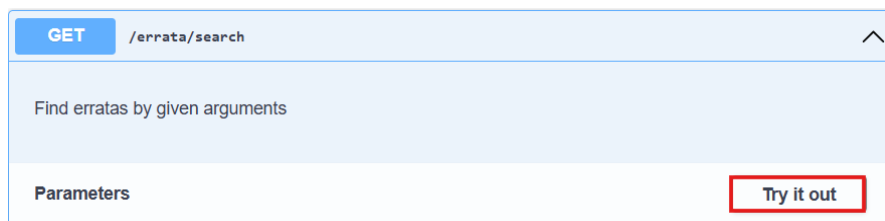
4. Request duration (время выполнения запроса)

1790 ms — время, потраченное сервером на выполнение запроса.

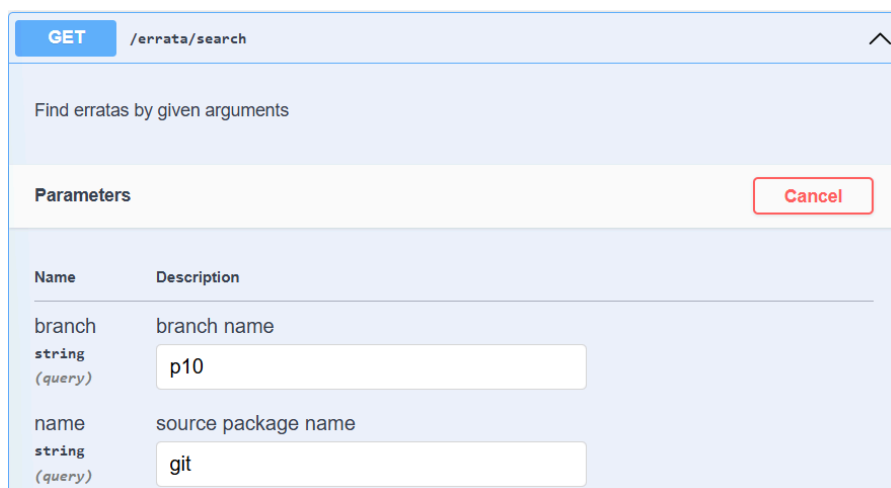
Теперь запросим информацию об об исправлениях и обновлениях пакета *git*.

В ALTRepo API перейдем в раздел *errata* и выберем пункт GET: */errata/search*.

Нажмем на кнопку “Try it out”:



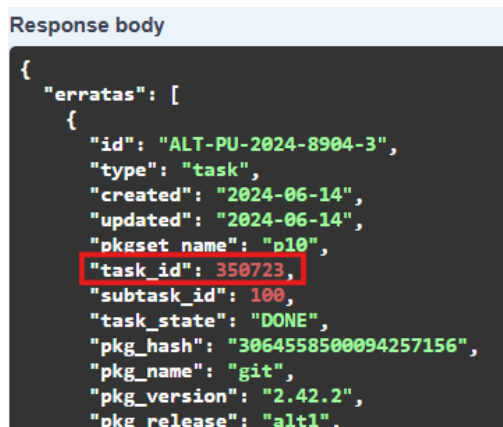
Заполним поля *branch name* и *source package version* и нажмем кнопку “Execute”:



Результат поиска включает в себя HTTP-запрос к API с ответом в формате JSON:

```
curl -X 'GET' \
'https://rdb.altlinux.org/api/errata/search?branch=p10&name=git' \
-H 'accept: application/json'
```

В теле ответа найдем идентификатор последней задачи:



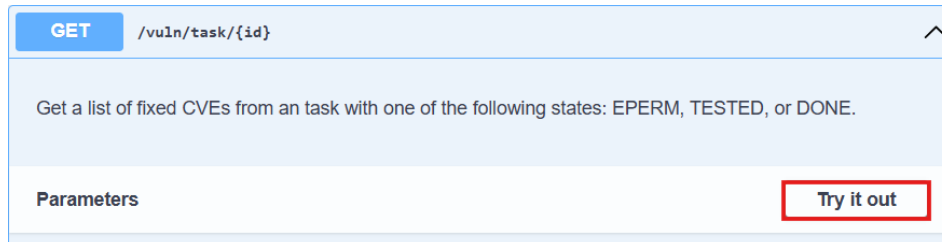
```
{
  "erratas": [
    {
      "id": "ALT-PU-2024-8904-3",
      "type": "task",
      "created": "2024-06-14",
      "updated": "2024-06-14",
      "pkgset name": "p10",
      "task_id": 350723,
      "subtask_id": 100,
      "task_state": "DONE",
      "pkg_hash": "3064558500094257156",
      "pkg_name": "git",
      "pkg_version": "2.42.2",
      "pkg_release": "alt1",
    }
  ]
}
```


task_id = 350723

Используя данный идентификатор, запросим информацию об уязвимостях в пакете *git*.

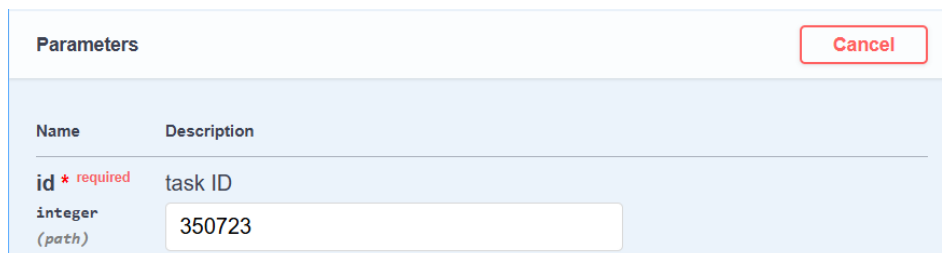
В ALTRepo API перейдем в раздел *vuln* и выберем пункт GET: `/vuln/task/{id}`.

Нажмем на кнопку “Try it out”:



The screenshot shows the ALTRepo API interface. At the top, there is a blue bar with a 'GET' button and the endpoint `/vuln/task/{id}`. Below this, a light blue box contains the text: 'Get a list of fixed CVEs from an task with one of the following states: EPERM, TESTED, or DONE.' At the bottom, there is a 'Parameters' section and a red-bordered button labeled 'Try it out'.

Заполним поле *task ID* и нажмем кнопку “Execute”:



The screenshot shows the 'Parameters' form in the ALTRepo API. It has a 'Cancel' button in the top right. Below the header, there is a table with two columns: 'Name' and 'Description'. The first row is for the 'id' parameter, which is marked as 'required' and has a description of 'task ID'. The 'id' is of type 'integer (path)' and its value is set to '350723' in a text input field.

Результат поиска включает в себя HTTP-запрос к API с ответом в формате JSON:

```
curl -X 'GET' \
'https://rdb.altlinux.org/api/vuln/task/350723' -H 'accept: application/json'
```

В теле ответа найдем идентификаторы уязвимостей, представленные в формате CVE (Common Vulnerabilities and Exposures):

CVE-2022-24975

CVE-2024-32002



```
{
  "id": "CVE-2022-24975",
  "type": "vuln",
  "link": "https://nvd.nist.gov/vuln/detail/CVE-2022-24975"
},
{
  "id": "CVE-2024-32002",
  "type": "vuln",
  "link": "https://nvd.nist.gov/vuln/detail/CVE-2024-32002"
}
```