УДК 004.725.5

Стрельцов Д. С

Бутенко Е. А

студенты специалитета

Российский государственный университет нефти и газа (НИУ) имени И.М Губкина, г. Москва

СБОРКА RPM-ПАКЕТА. ВОПРОСЫ БЕЗОПАСНОСТИ

Аннотация: В статье рассматриваются аспекты безопасности при сборке и использовании RPM-пакетов в Альт Линукс. Анализируются типичные уязвимости и описываются инструменты защиты, такие как hasher, gear, rpmlint и rpm, а также методы проверки цифровых подписей и контрольных сумм. Практическая часть демонстрирует безопасную сборку и проверку RPM-пакета на примере виджета погоды, что помогает минимизировать риски и обеспечить доверенную установку.

Ключевые слова: RPM-пакеты, безопасность, Альт Линукс, hasher, gear, rpmlint, цифровая подпись, контрольные суммы, уязвимости, сборка пакетов.

Abstract: The article discusses the security aspects of building and using RPM packages in Alt Linux. Typical vulnerabilities are analyzed and security tools such as hasher, gear, rpmlint and rpm are described, as well as methods for verifying digital signatures and checksums. The practical part demonstrates the safe assembly and verification of the RPM package using the example of the weather widget, which helps to minimize risks and ensure a trusted installation.

Key words: RPM packages, security, ALT Linux, hasher, gear, rpmlint, digital signature, checksums, vulnerabilities, package building.

Введение

Современные операционные системы на базе Linux активно используют систему управления пакетами RPM (Red Hat Package Manager) для установки, обновления и удаления программного обеспечения. RPM-пакеты позволяют стандартизировать процесс распространения программ, обеспечивая удобство

в установке и сопровождении. RPM была изначально написана в 1997 году Эриком Троаном и Марком Юингом на основе pms, rpp, и pm опыта. Эта система, зародившаяся в недрах Red Hat более четверти века назад, сегодня стала не просто инструментом для установки программ, но настоящим фундаментом для построения целых дистрибутивов. Примечательно, что несмотря на появление современных альтернатив вроде контейнеров и снапшотов, RPM не только выжил, но и продолжает эволюционировать, доказывая жизнеспособность своей архитектуры.

Однако, несмотря на преимущества, сборка и использование RPM-пакетов несет в себе определённые риски, особенно с точки зрения информационной безопасности. Особую актуальность представляет процесс сборки и верификации RPM-пакетов, являющихся основным методом распространения программного обеспечения в ряде дистрибутивов Linux, включая Альт Линукс. Проблема безопасности сборки RPM-пакетов затрагивает как разработчиков, так и системных администраторов, ответственных за развертывание программного обеспечения в корпоративных средах. Исследованием безопасности пакетной сборки занимались многие специалисты, включая команду разработчиков Альт Линукс, которые создали специализированные инструменты hasher и gear для обеспечения воспроизводимой и безопасной сборки. Значительный вклад в развитие методологий безопасной сборки внесли также специалисты Red Hat и SUSE, разработавшие инструменты mock и Open Build Service соответственно. На сегодняшний день хорошо изучены базовые аспекты безопасности RPM-пакетов, включая цифровую подпись, проверку контрольных сумм и базовый аудит зависимостей.

Статья посвящена рассмотрению ключевых аспектов безопасности RPM-пакетов в Альт Линукс.

Объект исследования — процесс сборки и использования RPM-пакета в Альт Линукс.

Предмет исследования — методы и инструменты обеспечения безопасности при сборке RPM-пакетов

Цель исследования — исследовать потенциальные угрозы безопасности, возникающие при сборке RPM-пакетов, а также разработка комплексного подхода к обеспечению безопасности процесса сборки RPM-пакетов в среде Альт Линукс.

RPM Package Manager

RPM (Red Hat Package Manager) представляет собой мощную систему управления пакетами, применяемую в различных дистрибутивах Linux. Её появление стало ключевым этапом в эволюции дистрибутивов, особенно в контексте автоматизации установки, обновления и удаления программного обеспечения.

RPM пакеты поставляются в виде архивов, которые содержат не менее одного файла, а также инструкции по установки этих файлов, включая права доступа. При работе с RPM ключевым моментом является понимание концепции пакета.

Так, а что собственно из себя представляет RPM пакет? RPM-пакеты делятся на два вида пакетов:

1. SRPM (Source RPM) - пакет с исходным кодом, имеет расширение .src.rpm. Пакет src.rpm можно использовать только для сборки двоичных пакетов, но не установки. В src.rpm пакетах содержится исходный тарболл (исходник программы), какие-либо другие исходники, патчи и самый главный spec-файл, который управляет процессом сборки. Все эти файлы упакованы в cpio архив. Когда вы пытаетесь войти в src.rpm пакет при помощи файлового менеджера mc, вы его увидите.
2. Binary RPM -бинарные пакеты с расширением .rpm содержат собранное ПО, которое может быть установлено и использовано. В .rpm-пакетах содержится cpio-архив с файлами, которые после установки разложатся

по соответствующим каталогам, файлы информации и установочные скрипты.

Структура RPM-пакета состоит из четырёх основных секций: свинцового блока (lead), сигнатуры, заголовка и архива payload. В современных версиях RPM роль свинцового блока существенно уменьшилась, так как большая часть его информации дублируется в заголовке для обеспечения более надёжного и гибкого хранения метаданных.

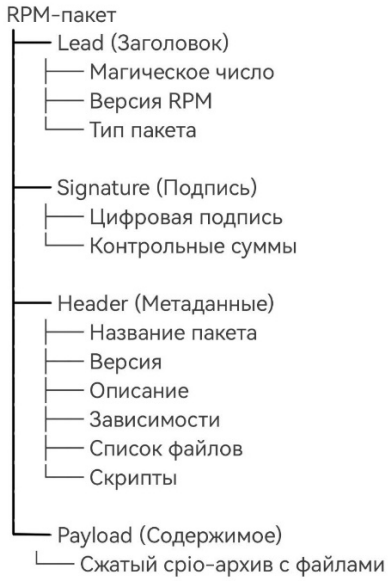

Рисунок 1 – Структура RPM-пакета

Spec-файл — это файл конфигурации, инструкция, по которой происходит сборка пакета при помощи утилиты rpm-build или mock. Spec-файл должен находиться в папке SPECS.

Общая структура SPEC-файла:

1. Заголовок (Header) - обязательные поля:
   - Name — имя пакета.
   - Version — версия программы.
   - Release — номер релиза. Обычно 1%{?dist} (например, 1.el9).
   - Summary — краткое описание.
   - License — тип лицензии.

- URL — URL(сайт) проекта.
- Source0 — имя исходного архива, помещённого в SOURCES/.
- BuildRequires — зависимости для сборки.
- Requires — зависимости для работы.

2. Секции (Sections) - основные блоки spec-файла

Таблица 1. Директивы основной части spec-файла

SPEC Директива	Определение
%description	Описание ПО, входящего в комплект поставки RPM. Длина каждой строки не должна превышать 72 символа. Учитывается при поиске пакета через apt-cache search, выводится во время просмотра информации о пакете при помощи apt-cache show имя_пакета.
%prep	Команда/серия команд для подготовки ПО к сборке, можетсодержать сценарий оболочки (shell скрипт).
%build	Команда/серия команд для фактической сборки программного обеспечения в машинный.
%install	Команды установки/копирования файлов из сборочного каталога в псевдо-корневой каталог. Во время сборки пакета этот раздел эмулирует конечные пути установки файлов в систему.
%check	Команда/серия команд для тестирования ПО.
%files	Список файлов, которые будут установлены в системе конечного пользователя.
%changelog	Запись изменений, произошедших в пакете между сборками разных версий или релизов.

RPM в различных ОС

Практически каждый крупный проект, использующий RPM, имеет свою версию пакетного менеджера, отличающуюся от остальных. Между представителями семейства RPM могут иметься следующие различия:

1. наборы макросов, используемых в spec-файлах;
2. различное поведение RPM при сборке «по умолчанию» – при отсутствии каких-либо указаний в spec-файлах;
3. формат строк зависимостей;
4. мелкие отличия в семантике операций (например, в операциях сравнения версий пакетов);
5. мелкие отличия в формате файлов.

Для пользователя различия чаще всего заключаются в невозможности поставить «неродной» пакет из-за проблем с зависимостями или из-за формата пакета.

Таблица 2. Информация о различных ОС и инструменты для сборки пакетов

ОС	**Сборочная среда**
Fedora / CentOS / RHEL	rpmbuild, mock, koji
openSUSE / SUSE	rpmbuild, osc, Open Build Service (OBS)
ALT Linux	hasher, gear, rpm-build
Mageia / Mandriva	rpmbuild, mga-build
ROSA Linux	rpm-build, abf
Oracle / Amazon Linux	rpmbuild, mock, yum-builddep
Arch / Debian (через chroot)	mock или Docker с rpm-tools

Угрозы безопасности и инструменты для защиты RPM-пакетов в ОС Альт Линукс

Далее в этой статье рассматриваются основные угрозы безопасности RPM-пакетов в ALT Linux и инструменты, которые помогают их минимизировать, а также практическая часть сборки RPM-пакета.

Ниже приведены основные угрозы, с которыми сталкиваются пользователи ALT Linux:

1. Установка вредоносных пакетов из ненадежных источников - установка пакетов из непроверенных источников, таких как сторонние сайты или неподписанные репозитории, может привести к внедрению вредоносного кода, включая трояны, шпионское ПО или программы-вымогатели.
2. Атаки "человек посередине" (MITM) - при загрузке пакетов из репозиториев злоумышленники могут перехватить соединение и подменить пакеты, внедрив в них вредоносный код. Это особенно актуально для незащищенных соединений (HTTP вместо HTTPS).

3. Уязвимости в установленных пакетах - даже пакеты из официальных репозиториев могут содержать уязвимости, которые еще не исправлены. Такие уязвимости могут быть использованы для получения несанкционированного доступа к системе.
4. Несанкционированное изменение файлов установленных пакетов - после установки файлы пакетов могут быть изменены вредоносным ПО или злоумышленниками, что может привести к некорректной работе программ или компрометации системы.
5. Проблемы с зависимостями - неправильное управление зависимостями может привести к установке несовместимых пакетов, что может вызвать нестабильность системы или открыть дополнительные векторы атак.

Инструменты и методы для защиты RPM-пакетов в Альт Линукс

Альт Линукс предоставляет ряд инструментов и методов для защиты RPM-пакетов от вышеуказанных угроз. Эти инструменты интегрированы в систему управления пакетами и могут быть использованы как системными администраторами, так и обычными пользователями.

Ниже приведены основные утилиты и их применение:

1. Synaptic - графический интерфейс для APT-RPM. Позволяет искать, устанавливать, удалять и обновлять пакеты, а также проверять зависимости через удобный интерфейс.
2. Rpm - основной инструмент для работы с RPM-пакетами. Позволяет проверять цифровые подписи, целостность установленных файлов, содержимое, зависимости и скрипты установки.
    - rpm -K package.rpm — проверка цифровой подписи.
    - rpm -V package_name — проверка целостности установленных файлов.
    - rpm -qlp package.rpm — просмотр списка файлов в пакете.

- rpm -qpR package.rpm — анализ зависимостей.
- rpm -qp --scripts package.rpm — проверка скриптов установки.

3. Rpmlint - утилита для статического анализа RPM-пакетов и spec-файлов. Проверяет наличие ошибок, подозрительных конструкций, некорректных разрешений и соответствие стандартам Альт Линукс.
    - rpmlint package.rpm или rpmlint package.spec
4. APT-RPM – менеджер пакетов Альт Линукс, основанный на APT, адаптированный для RPM-пакетов. Автоматически проверяет подписи и зависимости при установке.
    - apt-get install package_name — установка пакета.
    - apt-get check — проверка целостности репозиториев.
5. sha256sum**:** Проверка контрольных сумм исходных файлов и пакетов.
    - sha256sum <название пакета>.src.rpm
6. gpg: Проверка GPG-подписей.
    - gpg –verify <название пакета>.noarch.rpm

Таблица 3. Информация об инструментах, используемых в Альт Линукс

Инструмент	Описание	Команда
rpm	Управление RPM-пакетами: проверка содержимого, зависимостей, подписей, целостности.	rpm -qlp package.rpm rpm -qpR package.rpm rpm -K package.rpm rpm -V package_name
rpmlint	Статический анализ spec-файлов и RPM-пакетов на ошибки и уязвимости (rpmlint GitHub).	rpmlint package.rpm rpmlint package.spec
apt-rpm	Пакетный менеджер для работы с репозиториями и проверки пакетов.	apt-get check apt-cache show package
sha256sum	Проверка контрольных сумм файлов.	sha256sum package.rpm
gpg	Проверка GPG-подписей.	gpg --verify package.rpm

Сборка пакета на ОС Альт Линукс и проверка безопасности

В рамках проекта Сизиф членами ALT Linux Team разработан набор инструментов:

1. **Hasher** — инструмент для безопасной сборки RPM-пакетов в контролируемой среде. Hasher – инструмент для сборки пакетов в «чистой» и контролируемой среде. Это достигается с помощью создания в chroot минимальной сборочной среды, установки туда указанных в source-пакете сборочных зависимостей и сборке пакета в свежесозданной среде. Для сборки каждого пакета сборочная среда создается заново.

2. **Gear** (Get Every Archive from git package Repository) – система для работы с произвольными архивами программ. В качестве хранилища данных gear использует git, что позволяет работать с полной историей проекта. Идея gear заключается в том, чтобы с помощью одного файла с простыми правилами (для обработки которых достаточно sed и git) можно было бы собирать пакеты из произвольно устроенного git-репозитория, по аналогии с hasher, который был задуман как средство для сборки пакетов из произвольных «srpm-пакетов».

3. Alterator — платформа для управления конфигурацией Linux-системы.
4. ALT Linux Installer — инсталлятор, используемый в дистрибутивах ALT Linux.

Для данной статьи установим пакет с официального сайта Альт Линукса: https://packages.altlinux.org/ru/sisyphus/packages/Accessibility/

Для статьи выбрали виджет погоды: https://git.altlinux.org/gears/g/gis-weather.git?a=tree;hb=7be8b08ef90ef4344b8fcb6a0b803d3dc5081554

1. **Подготовка к безопасной сборке**

   1.1 Для начала установим инструменты, необходимые для проверки и сборки пакета:

```
apt-get install hasher rpmlint rpm gpg sha256sum
```

   Эти утилиты позволят нам собрать пакет в изолированной среде и проверить его на безопасность.

   1.2 Создаем рабочую директорию:

```
mkdir ~/gis-weather-work
```

```
cd ~/gis-weather-work
```

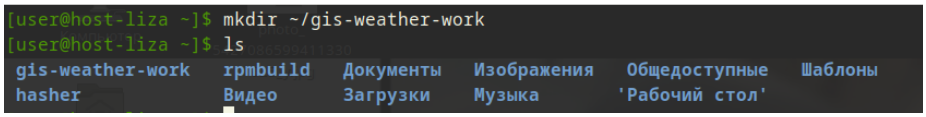

Рисунок 2 – Создание директории

1.3 Затем клонируем репозиторий пакета:

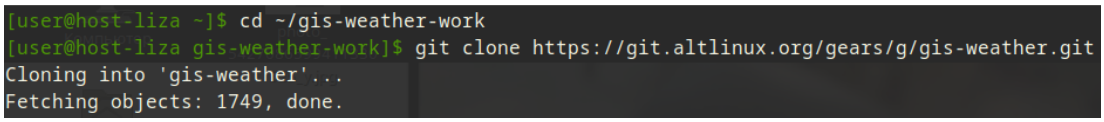

Рисунок 3 – Клонирование репозитория

2. Проверка исходного кода

2.1 Аудит исходников: проверим исходный код на наличие потенциально опасных функций, таких как system() или exec():

```
grep -r "system\|exec"
```

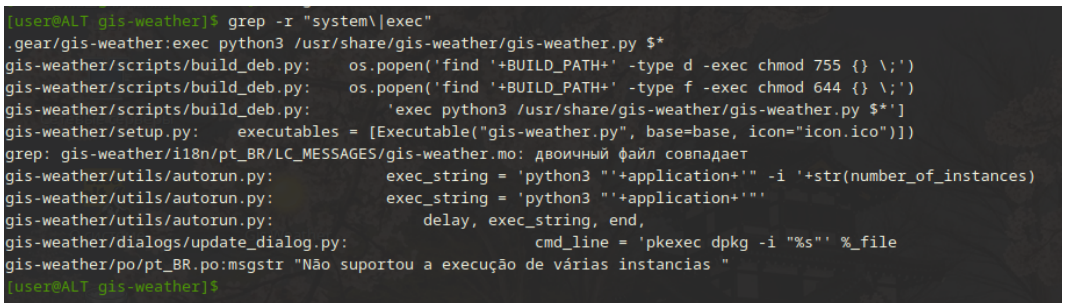

Рисунок 4 – Проверка исходников

2.2 Проверка контрольных сумм

```
sha256sum SOURCES/gis-weather-0.8.4.17.tar.gz
```

3. Проверка spec-файла

3.1 Анализ с помощью rpmlint

Проверим spec-файл на ошибки и потенциальные проблемы:

```
rpmlint gis-weather.spec
```

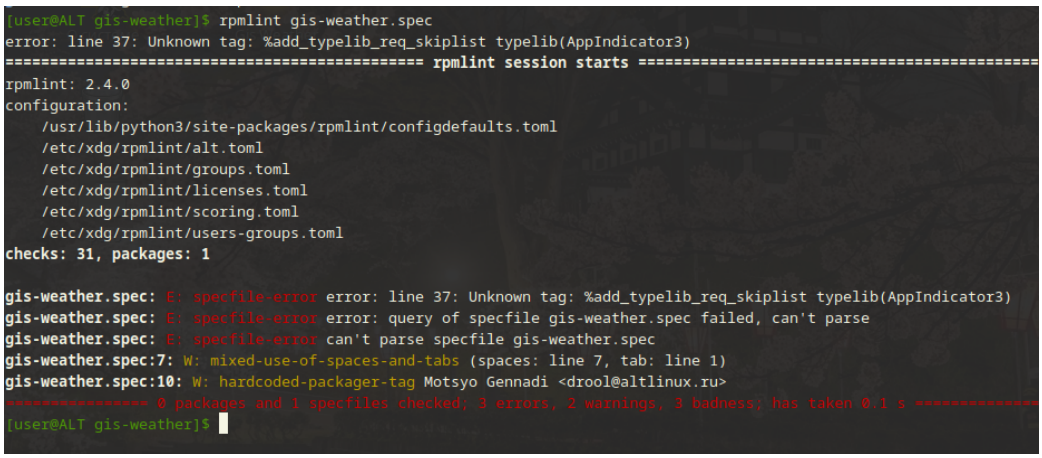

Рисунок 5 – Проверка файла

Разбор ошибок:

- Неизвестный тег %add_typelib_req_skiplist - это специфичный для Альт Линукса тег, который rpmlint не распознал. Ошибка не влияет на сборку или безопасность.
- Невозможность распарсить файл**:** следствие первой ошибки.

Разбор предупреждений:

- Смешивание пробелов и табуляций - стилистическая мелочь, исправляется заменой табуляции на пробелы.
- Жестко заданный упаковщик - не критично, можно оставить или удалить.

3.2 Проверка скриптов %pre и %post

Посмотрим, есть ли в spec-файле скрипты, выполняемые до или после установки:

```
cat gis-weather.spec | grep -A 10 "%pre"
cat gis-weather.spec | grep -A 10 "%post"
```

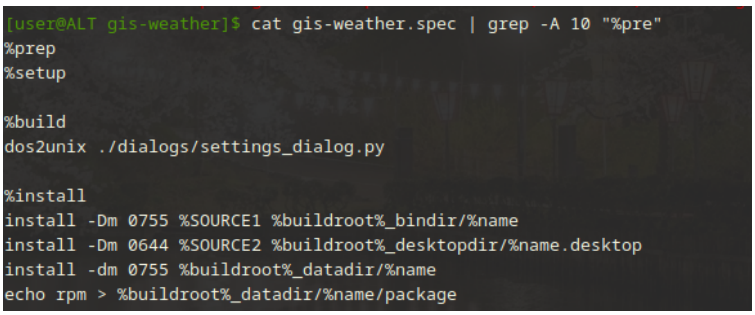

Рисунок 6 – Проверка скриптов

- Команда для %pre показала секцию %prep (подготовка к сборке), но самой секции %pre (скрипт перед установкой) в файле нет.
- Команда для %post не вывела ничего, что означает отсутствие секции %post (скрипт после установки).

4. Сборка пакета в изолированной среде

4.1 Просмотр содержимого:

```
ls -la
```

4.2 Проверка gear-правил:

```
cat .gear/rules
```

Рисунок 7 – Проверка файла перед сборкой

4.3 Установка необходимых зависимостей для сборки

```
apt-get install
```

- dos2unix # Утилита для конвертации текстовых файлов из DOS/Mac формата в UNIX
- python3-module-cairo # Библиотека для работы с графикой
- python3-module-changelog # Модуль для работы с changelog
- python3-module-distro # Модуль для определения дистрибутива
- python3-module-gi # Python-привязки для GObject Introspection

- python3-base # Базовая установка Python3
- rpm-build-gir # Инструменты для сборки GObject Introspection
- rpm-build-python3 # Инструменты для сборки Python3 пакетов

```
[root@host-liza ~]# apt-get install     dos2unix     python3-module-cairo     python3-module-
changelog     python3-module-distro     python3-module-gi     python3-base     rpm-build-gir
    rpm-build-python3
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
```

Рисунок 8 –Установка зависимостей

4.4 Собираем пакет с помощью gear:

Сборка пакета:

```
gear-hsh
```

```
[user@host-liza gis-weather]$ gear-hsh
/.host/cpio: ./usr/bin/cat not created: newer or same age version exists
/.host/cpio: ./usr/bin/cut not created: newer or same age version exists
/.host/cpio: ./usr/bin/du not created: newer or same age version exists
```

Рисунок 9 – Сборка пакета

Если пакет собрался, найдем его точное имя:

```
find ~/hasher/repo -name "gis-weather*.rpm"
```

```
[user@host-liza gis-weather]$ find ~/hasher/repo -name "gis-weather*.rpm"
/home/user/hasher/repo/SRPMS.hasher/gis-weather-0.8.4.17-alt3.src.rpm
/home/user/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-alt3.noarch.rpm
```

Рисунок 10 – Успешная сборка

Теперь мы видим, что пакет успешно собрался, и у нас есть два файла:

1. Исходный пакет (SRPMS): gis-weather-0.8.4.17-alt3.src.rpm

2. Бинарный пакет (RPMS): gis-weather-0.8.4.17-alt3.noarch.rpm

5. Проверка RPM-пакета

5.1 Проверка содержимого

Узнаем, какие файлы будут установлены:

```
rpm -qlp ~/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-
alt3.noarch.rpm
```

5.2 Проверка зависимостей, посмотрим зависимости пакета:

```
rpm -qpR ~/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-alt3.noarch.rpm
```

- Зависимости определяют, какие другие пакеты должны быть установлены для корректной работы данного пакета.
- Проверка зависимостей помогает избежать конфликтов между пакетами. Например, если два пакета требуют разные версии одной и той же библиотеки, это может привести к проблемам.

5.3 Проверка скриптов в пакете

Проверим, какие скрипты выполняются при установке/удалении:

```
rpm -qp --scripts ~/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-alt3.noarch.rpm
```

Скрипты отсутствуют, что повышает безопасность пакета.

5.4 Проверка подписи

Убедимся, что пакет подписан:

```
rpm -K ~/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-alt3.noarch.rpm
```

```
[user@ALT gis-weather]$ rpm -K ~/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-alt3.noarch.rpm
/home/user/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-alt3.noarch.rpm: sha1 md5 OK
```

Рисунок 11 – Проверка подписи пакета

6. Установка пакета

Нам нужен бинарный пакет (noarch.rpm). Теперь установим его, используя полный путь:

```
rpm -Uvh /home/user/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-alt3.noarch.rpm
```

- - rpm - пакетный менеджер
- - -U - обновить пакет (или установить, если не был установлен)
- - -v - подробный вывод
- - -h - показывать прогресс установки

Путь указывает на собранный пакет в директории hasher

```
[root@host-liza ~]# rpm -Uvh /home/user/hasher/repo/x86_64/RPMS.hasher/gis-weather-0.8.4.17-a
lt3.noarch.rpm
Подготовка...                          ########################################### [100%]
Обновление / установка...
1: gis-weather-0.8.4.17-alt3           ########################################### [100%]
Running /usr/lib/rpm/posttrans-filetriggers
```

Рисунок 12 – Установка пакета

7. Проверка целостности после установки

7.1 Убедимся, что установленные файлы не изменены:

```
rpm -V gis-weather
```

```
[root@ALT ~]# rpm -V gis-weather
[root@ALT ~]#
```

Рисунок 13 – Файлы не изменены

8. Проверка репозитория

Убедимся, что пакет из официального источника:

```
apt-cache policy gis-weather
```

```
[root@ALT ~]# apt-cache policy gis-weather
gis-weather:
  Установлен: 0.8.4.17-alt3@1745523487
  Кандидат: 0.8.4.17-alt3@1745523487
  Таблица версий:
 *** 0.8.4.17-alt3@1745523487 0
        100 RPM Database
```

Рисунок 14 – Проверка информации о пакете

Пакет gis-weather актуален, обновлений не требуется, и он установлен из локальной базы RPM.

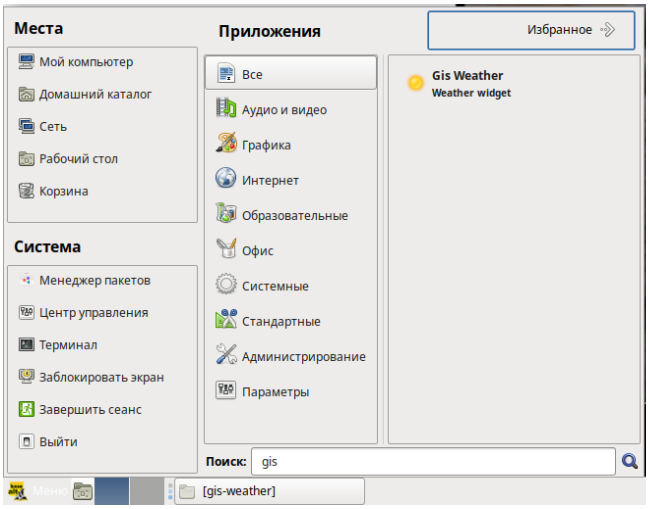

Рисунок 14 – Установленный пакет

9. Результат

После выполнения всех проверок мы убедились, что:

- Исходный код не содержит подозрительных функций.
- Spec-файл не имеет ошибок (rpmlint не выдал предупреждений).
- Пакет подписан корректным GPG-ключом.
- Зависимости корректны и не содержат уязвимых версий.
- Установленные файлы имеют правильные права доступа и не были модифицированы.

Таблица 4. Этапы процесса сборки и проверки пакетов

Этап	Проверка	Команда	Описание
Подготовка	Установка инструментов	apt-get install hasher rpmlint rpm gpg	Установка утилит для сборки и проверки пакетов.
Исходный код	Аудит кода и сумм	grep -r "system|exec" . sha256sum SOURCES/*.tar.gz	Поиск опасных функций и проверка целостности исходников.
Spec-файл	Анализ ошибок	rpmlint package.spec	Проверка spec-файла на ошибки и уязвимости.
Сборка	Изолированная среда	gear-hsh	Сборка пакета в chroot через hasher.
Пакет	Проверка содержимого и подписи	rpm -qlp package.rpm rpm -K package.rpm	Просмотр файлов и проверка цифровой подписи.
Установка	Установка и целостность	rpm -Uvh package.rpm rpm -V package	Установка пакета и проверка файлов после установки.
Репозиторий	Проверка источника	apt-cache policy package	Проверка, что пакет из доверенного репозитория.

Заключение

Исследование процесса сборки и проверки RPM-пакетов в Альт Линукс выявило ключевые аспекты обеспечения безопасности на всех этапах: от подготовки исходного кода до установки и верификации пакета. Анализ показал, что использование специализированных инструментов, таких как hasher, gear, rpmlint и rpm, в сочетании с методами проверки цифровых подписей, контрольных сумм и целостности файлов, позволяет

минимизировать риски, связанные с установкой вредоносного кода, компрометацией репозиториев и уязвимостями в зависимостях.

Практическая часть, включающая сборку и проверку пакета, продемонстрировала, что изолированная среда hasher обеспечивает воспроизводимость и контроль, а утилиты rpmlint и rpm эффективно выявляют потенциальные проблемы в spec-файлах и пакетах.

Список использованных источников и литературы

1. А. Боковой, Д. Левин Глава 3. Управление пакетами / А. Боковой, Д. Левин [Электронный ресурс] // Базальт СПО : [сайт]. — URL: https://docs.altlinux.org/ru-RU/archive/2.2/html-single/master/admin-html/ch03.html (дата обращения: 19.05.2025).
2. Валентин Соколов, Мария Фоканова и другие. Руководство по сборке RPM-пакетов для дистрибутивов Альт / Валентин Соколов, Мария Фоканова и другие. [Электронный ресурс] // alt-packaging-guide.github.io : [сайт]. — URL: https://alt-packaging-guide.github.io/ (дата обращения: 19.05.2025).
3. Команды RPM / [Электронный ресурс] // Alt Linux Wiki : [сайт]. — URL: https://www.altlinux.org/Команды_RPM (дата обращения: 19.05.2025).
4. ООО Базальт СПО Пакеты/Категории / ООО Базальт СПО [Электронный ресурс] // alt linux : [сайт]. — URL: https://packages.altlinux.org/ru/sisyphus/packages/ (дата обращения: 19.05.2025).
5. Разбираем RPM по косточкам: что скрывают свинцовый раздел, сигнатуры и архивы CPIO / [Электронный ресурс] // ДЗЕН : [сайт]. — URL: https://dzen.ru/a/Z-ew6gQQASBgbK8G (дата обращения: 19.05.2025).
6. Сергей Евгеньевич Богомолов Bog BOS: RPM. Управление пакетами программ в Linux от Red Hat / Сергей Евгеньевич Богомолов

[Электронный ресурс] // Bog: bookmarks on steroids (BOS) : [сайт]. — URL: http://www.bog.pp.ru/work/rpm.html (дата обращения: 19.05.2025).

7. Уймин А.Г. Компьютерные сети. L2-технологии : Практикум/ Уймин А.Г. — Москва : Ай Пи Ар Медиа, 2024. — 191 с. — ISBN 978-5- 4497-2539-4. — Текст : электронный.(дата обращения: 19.05.2025)
8. Что внутри RPM? / [Электронный ресурс] // linuxlib.ru : [сайт]. — URL: https://www.linuxlib.ru/inst/inrpm.htm (дата обращения: 19.05.2025).
9. Eric Foster-Johnson, Влад Горелецкий Red Hat RPM Guide - русский перевод / Eric Foster-Johnson, Влад Горелецкий [Электронный ресурс] // UNIX : [сайт]. — URL: https://www.uneex.ru/static/RedHatRPMGuideBook/rpm_guide-linux.html#306_html (дата обращения: 19.05.2025).
10. rpm.org / rpm.org [Электронный ресурс] // RPM : [сайт]. — URL: https://rpm-software-management.github.io/rpm/manual/format_v4.html (дата обращения: 19.05.2025).
11. Valentin Bajrami Как создать RPM-пакет Linux / Valentin Bajrami [Электронный ресурс] // Red Hat : [сайт]. — URL: https://www.redhat.com/en/blog/create-rpm-package (дата обращения: 19.05.2025).