

Научная статья

DOI 10.66424/2071-8217-2026-3-6

УДК 004.056.53

МЕТОДИКА ВЫЯВЛЕНИЯ СКРЫТЫХ КАНАЛОВ УТЕЧКИ ИНФОРМАЦИИ ПРИ НЕКОРРЕКТНЫХ ОБНОВЛЕНИЯХ ПРОГРАММ ЯДРА СИСТЕМЫ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ

В. Ю. Руссу*, Д. Ю. Казьмин, Д. Н. Бирюков

Военно-космическая академия имени А. Ф. Можайского, Санкт-Петербург, Россия

✉ *russu_valeriy@mail.ru

ДЛЯ ЦИТИРОВАНИЯ

Руссу В. Ю., Казьмин Д. Ю.,
Бирюков Д. Н. Методика выявления
скрытых каналов утечки
информации при некорректных
обновлениях программ ядра
системы управления базой данных //
Проблемы информационной
безопасности. Компьютерные
системы. 2026. № 3. С. 84–95.
DOI: 10.66424/2071-8217-2026-3-6

ПОСТУПИЛА 11.02.2026

ПРИНЯТА 02.06.2026

ОПУБЛИКОВАНА 13.10.2026

© Руссу В. Ю., Казьмин Д. Ю.,
Бирюков Д. Н.

Издатель: Санкт-Петербургский
политехнический университет
Петра Великого

АННОТАЦИЯ

Рассмотрена проблема уязвимостей, возникающих при обновлении программного обеспечения в изолированных корпоративных системах на базе ОС Astra Linux. Продемонстрирован возможный метод реализации атаки с получением root-доступа через модифицированный пакет обновления программного обеспечения. Предложена многоуровневая методика обнаружения фактов модификации пакетов обновлений, включающая статический, поведенческий и сигнатурный анализ с использованием методов и алгоритмов машинного обучения, позволяющая существенно повысить защищенность процесса обновления ПО в условиях сетевой изоляции.

КЛЮЧЕВЫЕ СЛОВА

Защита информации, автоматизация поиска уязвимостей, автоматизация распаковки прошивки, информационная безопасность

Original article

DOI 10.66424/2071-8217-2026-3-6

A METHOD FOR IDENTIFYING HIDDEN INFORMATION LEAK CHANNELS DUE TO INCORRECT UPDATES OF DATABASE MANAGEMENT SYSTEM KERNEL PROGRAMS

V. Yu. Russu*, D. Yu. Kazmin, D. N. Biryukov

Military Space Academy named after A. F. Mozhaisky, St. Petersburg, Russia

✉ *russu_valeriy@mail.ru

FOR CITATION

Russu V. Yu., Kazmin D. Yu.,
Biryukov D. N. A method for identifying
hidden information leak channels
due to incorrect updates of database
management system kernel programs.
Problems of information security.
Computer systems. 2026. No. 3, pp. 84–95.
DOI: 10.66424/2071-8217-2026-3-6
(In Russian)

ABSTRACT

This article examines the problem of vulnerabilities that arise during software updates in isolated corporate systems running Astra Linux. A possible method for implementing an attack to gain root access through a modified software update package is demonstrated. A multi-layered method for detecting update package modifications is proposed, including static, behavioral, and signature-based analysis using machine learning methods and algorithms. This method significantly improves the security of the software update process in network isolation.

RECEIVED 11.02.2026
ACCEPTED 02.06.2026
PUBLICATION 13.10.2026

KEYWORDS

Information security, automated vulnerability detection, automated firmware unpacking, information security

1. ВВЕДЕНИЕ

При построении автоматизированных систем обработки информации (АСОИ) на объектах с повышенными требованиями к безопасности (критическая инфраструктура, военные системы, объекты энергетики) широко применяется архитектура «воздушного зазора» (air-gapped), исключая прямое сетевое взаимодействие элементов АСОИ с внешними ресурсами. Обновление программного обеспечения (ПО) таких АСОИ осуществляется исключительно через локальные репозитории или физические носители (CD/DVD/USB). Парадоксально, но сама изоляция, призванная минимизировать угрозы информационной безопасности, создает уникальные уязвимости в цепочке обновления ПО, обусловленные ослаблением криптографического контроля при ручном переносе артефактов, накоплением устаревших компонентов в локальных репозиториях и рисками компрометации на этапах производства/транспортировки носителей [1].

Известен инцидент с дистрибутивами операционной системы (ОС) Red Hat Enterprise Linux, в котором несанкционированная модификация пакета openssh-server на этапе тиражирования машинных носителей информации (компакт-дисков) привела к внедрению stealth-трояна [2]. Имеются сведения, что эксплуатация уязвимости CVE-2019-3689 в SUSE Manager позволила разместить вредоносные RPM-пакеты на корпоративных репозиториях [3]. Уязвимость CVE-2015-0235 в библиотеке *glibc* сохранялась в офлайн-репозиториях на 37 % дольше, чем в интернет-связанных системах (среднее время обновления 14,8 мес против 9,3) [4]. В 2016 г. скомпрометированный администратор внедрил модифицированный пакет *systemd* в локальный репозиторий ОС CentOS, принадлежащей энергетической компании, обеспечив персистентный доступ [5]. Исследования, проведенные в 2022–2024 гг., показали: 68,2 % организаций, эксплуати-

рующих АСОИ с архитектурой «воздушного зазора», фиксировали инциденты, связанные с некорректной верификацией офлайн-обновлений [6]; латентность обнаружения компрометации локальных репозиторий достигала 312 дней [7]; 41 % уязвимостей в изолированных системах обусловлен устаревшими зависимостями в «замороженных» репозиториях [8].

Перечисленные факты подтверждают актуальность задачи повышения защищенности процесса обновления ПО в изолированных от глобальной компьютерной сети Интернет сегментах государственных информационных систем, информационных систем персональных данных и значимых объектов критической информационной инфраструктуры. Особую актуальность имеет эта задача для АСОИ на базе ОС Astra Linux как критически важной для государственных органов Российской Федерации [9, 10].

2. АНАЛИЗ СЦЕНАРИЕВ КОМПРОМЕТАЦИИ ПРОЦЕССА ОБНОВЛЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В ОС ASTRA LINUX

Для обновления приложений в ОС Astra Linux используется менеджер пакетов Synaptic. Рекомендации по обновлению ОС Astra Linux и сами обновления размещены на официальном информационном ресурсе, который является источником получения оперативных обновлений операционных систем специального назначения Astra Linux Special Edition РУСБ.10015-01 [11]. Схема процесса обновления программного обеспечения в ОС Astra Linux включает в себя пять этапов: добавление компакт-диска или репозитория; добавление открытого gpg-ключа репозитория; верификация открытого gpg-ключа репозитория; обновление информации о пакетах; установка приложения.

Наиболее уязвимым местом в схеме данного процесса является этап верификации открытого ключа репозитория, поскольку

администратор сервера АСОИ должен разбираться в алгоритмах инфраструктуры открытых ключей PKI и самостоятельно сверить добавляемый ключ с данными производителя операционной системы (в случае ОС Astra Linux – сайт АО «Научно-производственное объединение Русские базовые информационные технологии» [12]). Если обновляемая ОС находится в закрытом сегменте АСОИ без доступа к глобальной компьютерной сети Интернет, то сделать это практически невозможно. Особенно это касается обновлений, выпускаемых сторонними компаниями, подписываемыми распространяемое обнов-

ление пакетов своим собственным закрытым ключом, при этом не предоставляя администраторам доступ к открытому ключу. В этом случае администратору приходится «вслепую» обновлять пакеты, что может привести к добавлению модифицированных злоумышленником пакетов программного обеспечения, которые могут нанести вред операционной системе или всей инфраструктуре АСОИ в целом.

Структура пакета обновления программного обеспечения в ОС Astra Linux и обобщенная схема алгоритм его модификации злоумышленником представлены на рис. 1.

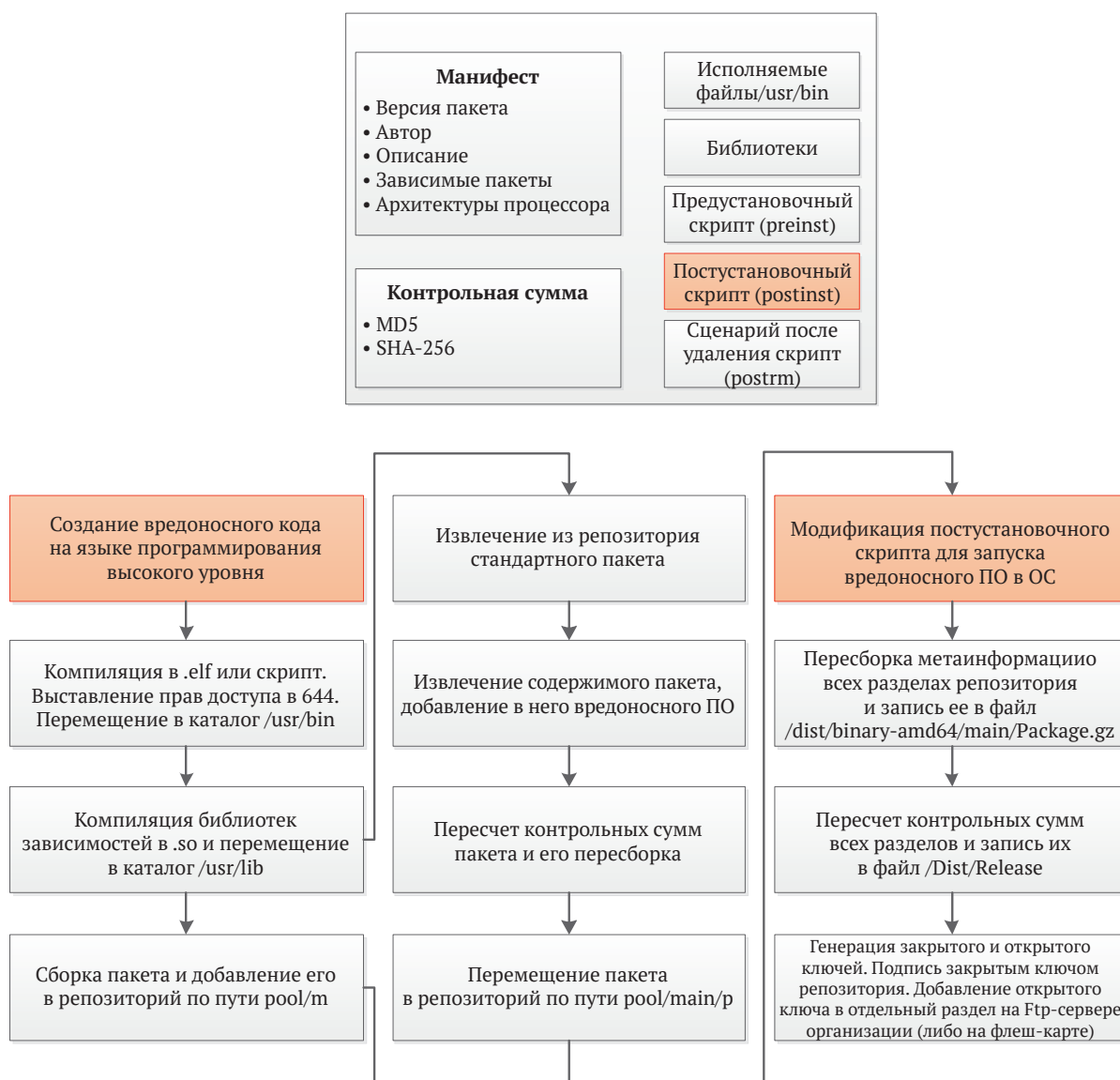


Рис. 1 | Структура пакета обновления ПО и схема алгоритма его модификации

Fig. 1 | Software update package structure and modification algorithm scheme

Из рис. 1 видно, что для компрометации системы злоумышленнику достаточно обладать навыками написания программ на языке С или С++, создания скриптов на языке сценариев Bash, знанием основ инфраструктуры открытых ключей PKI и элементарными навыками администрирования Linux-подобных систем. Перечисленными навыками обычно обладает разработчик средней квалификации, что делает представленный тип атаки опасным.

Для экспериментальной оценки устойчивости механизма обновления реализован и встроен в зависимости пакета обновления системы управления базами данных (СУБД) Postgresql 9.6 исследовательский программный модуль, который позволял получать доступ к оболочке `/bin/bash` целевого компьютера (так называемый *reverse shell*) с правами суперпользователя *root*. Схема процесса возможной

атаки на систему обновления ОС Astra Linux и получения несанкционированного доступа к корпоративным данным изображена на рис. 2.

Запуск разработанного программного модуля на исполнение с наивысшими привилегиями осуществлялся при обновлении СУБД Postgresql через пакетный менеджер *Synaptic* при вызове постустановочного скрипта (*postinst*). Получение наивысших привилегий происходило за счет того, что процесс установки пакета являлся дочерним от процесса *Synaptic*, который всегда запускается только с правами суперпользователя (*root*). Средства антивирусной защиты Kaspersky Endpoint Security Linux и Dr. Web для Linux не могли блокировать установку и работу разработанного программного модуля, поскольку в отличие от Windows систем (где они имеют уровень доступа выше, чем у пользователя Administrator) они имеют

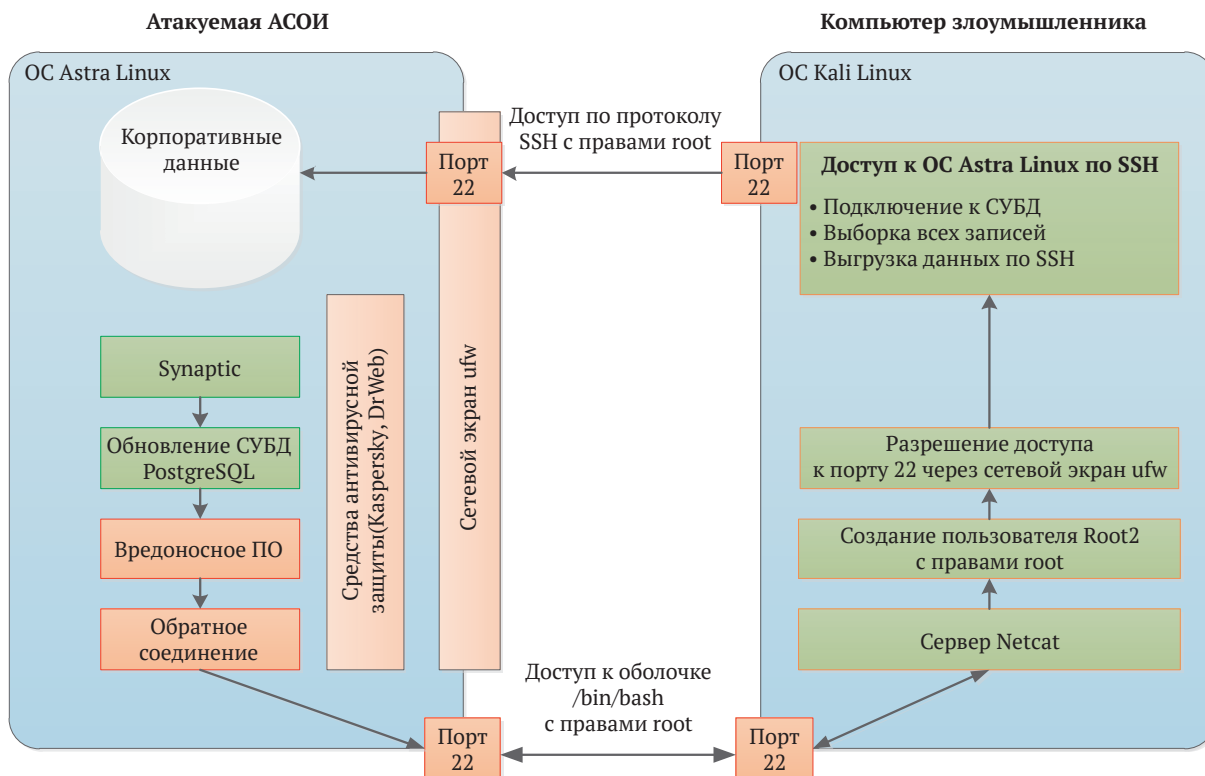


Рис. 2 | Схема потенциального сценария компрометации на систему обновления ОС Astra Linux

Fig. 2 | Diagram of a potential compromise scenario for the update system

```
DO $$
DECLARE
    sql_query TEXT := '';
BEGIN
    -- Собираем запрос UNION ALL для всех таблиц
    SELECT string_agg(
        format(
            'SELECT %L AS table_schema, %L AS table_name, to_jsonb(t) AS row_data FROM %I.
            %I',
            table_schema,
            table_name,
            table_schema,
            table_name
        ),
        ' UNION ALL '
    ) INTO sql_query
    FROM information_schema.tables
    WHERE table_type = 'BASE TABLE'
        AND table_schema NOT IN ('pg_catalog', 'information_schema');

    -- Создаём временную таблицу и вставляем данные
    DROP TABLE IF EXISTS all_tables_data;
    EXECUTE 'CREATE TEMP TABLE all_tables_data AS ' || sql_query;
END $$;
```

Рис. 3 | Вариант SQL-скрипта с выборкой по всем таблицам

Fig. 3 | A SQL script variant with a selection across all tables

ограниченные возможности контроля процессов с правами суперпользователя, инициализированных постустановочными скриптами. Кроме того, организованный скрытый канал не блокировался средствами стандартного сетевого экрана *ufw*, поскольку соединение с внешней средой инициализировалось со стороны самой ОС Astra Linux и, в соответствии с политикой безопасности, не могло быть заблокировано ограничительными правилами сетевого экрана.

После получения обратного доступа (*reverse shell*) нарушитель с правами суперпользователя *root* может выполнять различные сценарии сбора информации об АСОИ. Например, если под управлением СУБД PostgreSQL функционирует база данных, содержащая важную корпоративную информацию, то схема действий нарушителя может быть следующей: создать дополнительного суперпользователя (*root2*); изменить правила сетевого экрана (разрешить доступ к 22 порту, если он заблокирован), запустить SSH-сервер и получить доступ к серверу АСОИ по протоколу *ssh*; от имени пользователя

postgres с использованием утилиты *su* через основной терминальный клиент *psql* сделать выборку по всем таблицам (вариант SQL-скрипта представлен на рис. 3). Либо можно полностью скопировать все файлы БД на внешний компьютер, но процесс перемещения больших объемов данных может вызвать срабатывание IDS/IPS систем.

Определим потенциальную угрозу от реализации описанной атаки, воспользовавшись стандартом количественной оценки уязвимостей в программном обеспечении (Common Vulnerability Scoring System) [13]. Для этого определим значение вектора атаки (*attack vector*) и рассчитаем числовой рейтинг (*CVSS score*) по методике, приведенной в спецификации CVSS v.3.1. Оценки метрик CVSS для предложенной атаки с использованием механизма обновления ПО представлены в табл. 1.

Таким образом, вектор атаки будет иметь вид CVSS:3.1 / AV:N / AC:L / PR:H / UI:N / S:C / C:H / I:H / A:N.

Вычислим значения вспомогательных оценок: подоценки воздействия (ISS),

Таблица 1 | Оценки метрик CVSS**Table 1** | CVSS metrics scores

Метрика	Значение	Числовая константа	Пояснение
Путь внедрения (<i>attack vector</i> – AV)	Network (сеть)	0,85	Атака выполняется через загрузку пакета из локального репозитория
Сложность использования (<i>attack complexity</i> – AC)	Low (низкий)	0,77	Нет специальных условий – только публикация пакета
Необходимые привилегии (<i>privileges required</i> – PR)	High (высокий)	0,5	Нужны права администратора
Взаимодействие с пользователем (<i>user interaction</i> – UI)	None (нет)	0,85	Взаимодействие не требуется – код запускается при установке
Область действия (<i>scope</i> – S)	Changed (изменяющаяся)	–	Возможен выход за рамки прав процесса (например, доступ к ОС)
Влияние на конфиденциальность данных (<i>confidentiality</i> – C)	High (высокая)	0,56	Утечка/доступ к данным
Влияние на целостность данных (<i>integrity</i> – I)	High (высокая)	0,56	Изменение данных, внедрение кода
Влияние на доступность данных (<i>availability</i> – A)	High (высокая)	0,56	Возможен отказ в обслуживании, сбой или полная недоступность

эксплуатационной оценки (Exploitability) и воздействия (Impact) при изменяющейся области воздействия:

$$ISS = 1 - [(1 - Confidentiality)(1 - Integrity)(1 - Availability)] = 1 - [(1 - 0,56)(1 - 0,56)(1 - 0,56)] \approx 0,9148;$$

$$Exploitability = 8,22AVACPRUI = 8,22 \cdot 0,85 \cdot 0,77 \cdot 0,5 \cdot 0,85 \approx 2,2865;$$

$$Impact = 7,52(ISS - 0,029) - 3,25(ISS - 0,02)^{15} = 7,52(0,9148 - 0,029) - 3,25(0,9148 - 0,02)^{15} \approx 6,0478.$$

С учетом вычисленных значений вспомогательных оценок базовая оценка будет равна $BaseScore = 1,08(Exploitability + Impact) \approx 9$. Таким образом, предложенная атака с использованием механизма обновления ПО, относящаяся к классу «удаленное выполнение кода» (*remote code execution*), претендует на 9 из 10 баллов по стандарту CVSS v.3.1 и имеет критический уровень опасности.

3. МЕТОДИКА ЗАЩИТЫ ОТ АТАК НА ОБНОВЛЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОС ASTRA LINUX

Существующие решения обеспечения безопасности цепочки поставки ПО в Linux-системах условно можно разделить на четыре класса: системы управления локальными репозиториями; средства сигнатурного анализа; платформы анализа зависимостей; системы поведенческого анализа.

Сравнение существующих решений с предлагаемой методикой представлено в табл. 2.

Системы управления локальными репозиториями, такие как Pulp и Katello, обеспечивают централизованное хранение и распространение пакетов, однако не выполняют комплексную проверку их внутреннего содержимого и не ориентированы на выявление модифицированных

Таблица 2 | Сравнение существующих решений и предлагаемой методики**Table 2** | Comparison of existing solutions and the proposed methodology

Решение	Контроль зависимостей	ML-анализ	Поведенческий анализ	Работа в изолированной среде	Адаптация под Astra Linux
Pulp	Частично	Нет	Нет	Да	Нет
Katello	Частично	Нет	Нет	Да	Нет
OSSGadget	Да	Нет	Нет	Частично	Нет
ClamAV	Нет	Нет	Нет	Да	Частично
Cuckoo Sandbox	Нет	Нет	Да	Частично	Нет
Предлагаемая методика	Да	Да	Да	Да	Да

postinst-скриптов либо вредоносных зависимостей.

Инструменты анализа зависимостей (например, OSSGadget) предназначены преимущественно для выявления известных уязвимостей в open-source компонентах и требуют постоянного доступа к внешним базам данных уязвимостей, что затрудняет их использование в изолированных сегментах АСОИ.

Антивирусные средства (ClamAV, Linux Malware Detect, RKNHunter) основаны преимущественно на сигнатурном подходе и демонстрируют неадекватную эффективность при обнаружении модифицированного вредоносного кода, встроенного в postinst-скрипты или зависимости Linux-пакетов.

Платформы динамического анализа (например, Cuckoo Sandbox) позволяют выявлять аномальное поведение приложений, однако требуют значительных вычислительных ресурсов и не интегрированы в процесс обновления Astra Linux.

Отличительной особенностью предлагаемой методики является объединение контроля дерева зависимостей, ML-классификации содержимого пакетов, поведенческого анализа и адаптации к изолированным сегментам Astra Linux.

Операционная система Astra Linux имеет встроенные механизмы защиты информации, включая мандатное разграничение доступа, замкнутую программную

среду, контроль целостности, поддержку доверенной загрузки. Однако перечисленные механизмы ориентированы преимущественно на защиту уже функционирующей системы и не обеспечивают комплексный контроль безопасности пакетов обновления до их установки.

Особенность Astra Linux заключается в широком использовании изолированных сегментов АСОИ, в которых отсутствует оперативная синхронизация с внешними репозиториями, обновления переносятся через физические носители, невозможна онлайн-проверка репутации пакетов и ключей подписи и существенно увеличивается время жизни уязвимых зависимостей.

Кроме того, подсистема замкнутой программной среды Astra Linux контролирует запуск приложений, однако postinst-скрипты Debian-пакетов исполняются в доверенном контексте пакетного менеджера Synaptic от имени суперпользователя root. Это создает дополнительный риск внедрения вредоносной функциональности через механизм обновления. В отличие от существующих средств защиты, предлагаемая методика ориентирована именно на предварительную проверку пакетов обновления до их установки в доверенную среду Astra Linux.

Для предотвращения типа атак, рассмотренного ранее, может быть использована методика, включающая в себя три

основных этапа (рис. 4): сравнение с эталонной моделью системы; сигнатурный анализ пакета; поведенческий анализ пакета.

На этапе сравнения с эталонной моделью системы предлагается использовать дерево зависимостей пакетов друг от друга, которое разработчик ПО должен составлять и рассылать вместе пакетом обновлений. В рамках исследования разработан скрипт, который строит дерево зависимостей эталонного компьютера и проводит сверку. Если при сверке обнаруживаются пакеты, существующие в репозитории, но которые не должны там находиться, то возможно это вредоносное ПО.

На втором этапе методики (сигнатурный анализ) могут быть использованы стандартные средства детектирования наличия вредоносного ПО в Linux-пакетах. Проведенный анализ существующих средств (ClamAV [14], Linux Malware Detect (LMD) [15], RKNHunter [16], Kaspersky [17])

показал, что изменяя названия переменных, функций, порядок формирования массивов, списков, коллекций и прочего при создании вредоносного ПО несложно повлиять на результативность их работы.

Для повышения результативности этапа сигнатурного анализа в рамках исследования на основе проекта Kiinitix [18] разработан программный модуль, реализующий алгоритм машинного обучения на основе архитектуры Random Forest, осуществляющий бинарную классификацию файлов на вредоносные и доброкачественные. Для тренировки алгоритма использовался общедоступный набор данных, содержащий 70,1 % вредоносных программ. Точность обнаружения составила 96 %.

На третьем этапе методики проводится поведенческий анализ обновляемого ПО. Перед обновлением ПО в реальной АСОИ обновляемый пакет устанавливается на виртуальной машине, после чего

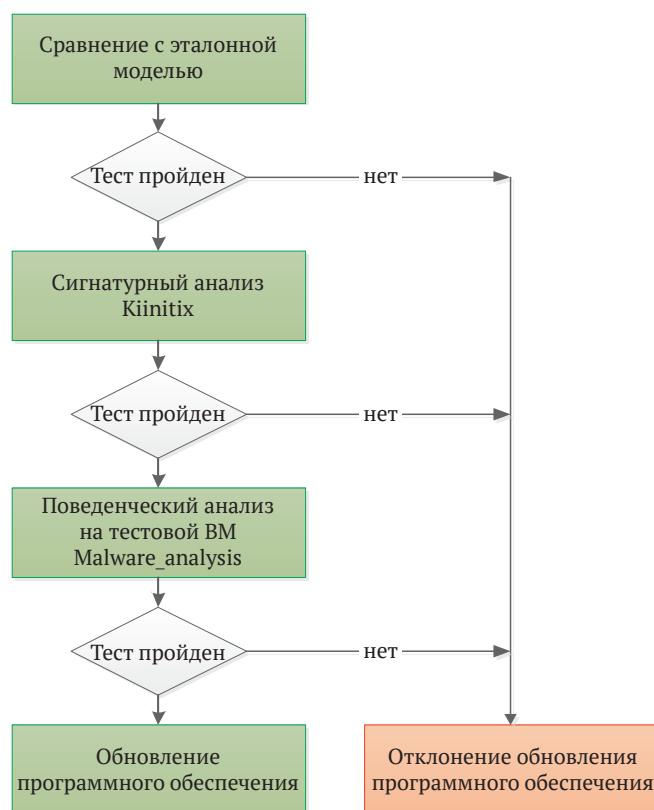


Рис. 4 | Схема методики защиты от атак на обновление ПО Astra Linux

Fig. 4 | A diagram of the Astra Linux software update protection method

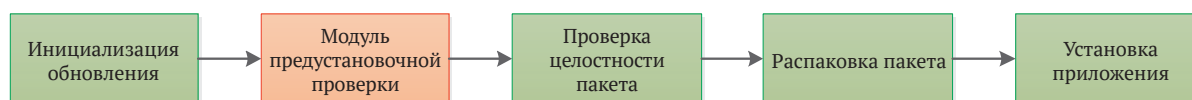


Рис. 5 | Модифицированная схема работы менеджера пакетов Synaptic

Fig. 5 | Modified operation scheme of the Synaptic package manager

осуществляется сборка и анализ профиля его работы. Для решения задач этого этапа методики предлагаем использовать общедоступный проект *Malware_analysis* [20], разработчики которого собрали 1000 образцов вредоносного ПО с сайта *VirusShare* и получили их поведенческие профили, запустив в «песочнице» *Cuckoo* [21]. На основе полученных данных с помощью метода *k*-ближайших соседей построен классификатор, точность которого составила 92,12 %.

Для реализации предложенной методики в схему работы менеджера пакетов *Synaptic* необходимо добавить дополнительный программный модуль, реализующий описанные этапы и обеспечивающий проверку пакетов на наличие возможно внедренного вредоносного ПО до их установки в систему (рис. 5).

4. ЗАКЛЮЧЕНИЕ

Проведенный анализ показал, что изолированные среды, такие как корпоративные сегменты с ОС Astra Linux, подвержены специфическим уязвимостям в процессе обновления ПО. Основными угрозами являются недостаточная верификация пакетов, устаревшие локальные репозитории и возможность внедрения вредоносного кода через модифицированные пакеты. Демонстрация атаки через

пакет обновления СУБД PostgreSQL показала, что злоумышленник может получить доступ уровня *root* и реализовать устойчивое управление системой. Классические средства защиты, основанные на сигнатурах, оказываются недостаточно эффективными из-за легкости обхода используемых в большинстве из них сигнатурных методов.

Научная новизна работы заключается в разработке комплексной методики предварительного контроля пакетов обновления для изолированных АСОИ на базе Astra Linux, объединяющей анализ дерева зависимостей, методы машинного обучения для статической классификации пакетов и поведенческий анализ в изолированной виртуальной среде. В отличие от существующих решений, методика ориентирована на применение в условиях сетевой изоляции и учитывает особенности механизма обновления Astra Linux.

Для повышения безопасности процесса обновления в изолированных системах необходимо внедрять многоуровневую защиту: анализ дерева зависимостей, статическую проверку пакетов с помощью моделей машинного обучения и поведенческое тестирование в «песочнице». Такая комплексная методика позволит существенно снизить риски компрометации критических систем и может стать важным шагом к построению безопасной инфраструктуры в условиях сетевой изоляции.

КОНФЛИКТ ИНТЕРЕСОВ / CONFLICT OF INTERESTS

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflict of interests.

СПИСОК ИСТОЧНИКОВ

1. **Enck W., Williams L.** Top Five Challenges in Software Supply Chain Security: Observations From 30 Industry and Government Organizations // *IEEE Security & Privacy*. 2022. Vol. 20. № 2. P. 8–9.
2. Red Hat Security Response Team. Lessons from a Physical Media Compromise. 2008. URL: <https://access.redhat.com/articles/20283> (дата обращения: 15.05.2025).
3. MITRE CVE-2019-3689: Command Injection in SUSE Manager // Национальная база уязвимостей (NVD). 2019. URL: <https://nvd.nist.gov/vuln/detail/CVE-2019-3689> (дата обращения: 15.05.2025).
4. **Shahzad M., Shafiq M. Z., Liu A. X.** Large Scale Exploratory Analysis of Software Vulnerability Life Cycles // *Proceedings of the 34th International Conference on Software Engineering (ICSE)*. 2012. P. 7–9.
5. CERT-EU. Case Study: Insider Threat in Critical Infrastructure. 2020. URL: https://cert.europa.eu/static/WhitePapers/CERT-EU_WhitePaper_Insider_Threats.pdf (дата обращения: 15.05.2025).
6. Ponemon Institute. Cybersecurity of Air-Gapped Industrial Control Systems. 2023. URL: <https://www.ponemon.org/library/air-gapped-ics-security> (дата обращения: 15.05.2025).
7. FireEye. M-Trends 2024: The Air-Gap Dilemma. 2024. URL: <https://www.fireeye.com/content/dam/fireeye-www/current-threats/pdfs/mtrends-2024.pdf> (дата обращения: 15.05.2025).
8. SANS Institute. Offline Patching Risks: A Quantitative Analysis. 2022. URL: <https://www.sans.org/white-papers/offline-patching-risks/> (дата обращения: 15.05.2025).
9. Минцифры России. Отчет о внедрении отечественного ПО в органах государственной власти (2023 г.). URL: <https://digital.gov.ru/uploaded/files/otchet-2023.pdf> (дата обращения: 15.05.2025).
10. Федеральная служба безопасности Российской Федерации. Заявление для СМИ от 14.06.2025 «О реализации проектов импортозамещения» // Официальный сайт ФСБ России. URL: <https://fsb.ru/fsb/press/message/single.htm> (дата обращения: 15.05.2024).
11. Справочный центр Astra Linux: Оперативные обновления для Astra Linux Special Edition. 2025 URL: <https://wiki.astralinux.ru/pages/viewpage.action?pageId=36077663> (дата обращения: 09.06.2025).
12. Программная инфраструктура собственной разработки АО «НПО РусБИТех». 2025. URL: <https://rusbitech.ru/products/zptk/ptk-karkas-s> (дата обращения: 09.06.2025).
13. CVSS v3.1 Specification Document. Revision 1. 2019. URL: <https://www.first.org/cvss/specification-document> (дата обращения 09.06.2025).
14. ClamAV: документация // Cisco Systems. 2025. URL: <https://docs.clamav.net/> (дата обращения 09.06.2025).
15. **Решетилов А.** Linux Malware Detect: обнаружение вредоносного ПО на веб-серверах // *Habr*. 2013. URL: <https://habr.com/ru/articles/194346/> (дата обращения 09.06.2025).
16. Rootkit Hunter: руководство администратора // Linode LLC. 2023. URL: <https://www.linode.com/docs/guides/anti-virus-and-rootkit-protection-with-clamav-and-rkhunter/> (дата обращения 09.06.2025).
17. Kaspersky XDR: соответствие ГОСТ Р 59712-2022 // АО «Лаборатория Касперского». 2025. URL: <https://support.kaspersky.com/help/XDR/1.3/ru-RU/283206.htm> (дата обращения 09.06.2025).
18. Kiinitix. Malware-Detection-using-Machine-learning. 2024. URL: <https://github.com/Kiinitix/Malware-Detection-using-Machine-learning> (дата обращения 09.06.2025).
19. Anushthakalia. Malware_analysis. URL: https://github.com/anushthakalia/Malware_analysis (дата обращения 09.06.2025).
20. Cuckoo Sandbox: Automated Malware Analysis System // Cuckoo Foundation. URL: <https://www.cuckoosandbox.org/> (дата обращения 09.06.2025).

REFERENCES

1. **Enck W., Williams L.** Top Five Challenges in Software Supply Chain Security: Observations from 30 Industry and Government Organizations. *IEEE Security & Privacy*. 2022. Vol. 20. No. 2. pp. 8–9.

2. Red Hat Security Response Team. Lessons from a Physical Media Compromise: official bulletin. 2008. URL: <https://access.redhat.com/articles/20283> (accessed: 15.05.2025).
3. MITRE CVE-2019-3689: Command Injection in SUSE Manager. *National Vulnerability Database (NVD)*. 2019. URL: <https://nvd.nist.gov/vuln/detail/CVE-2019-3689> (accessed: 15.05.2025). (In Russian)
4. **Shahzad M., Shafiq M. Z., Liu A. X.** Large Scale Exploratory Analysis of Software Vulnerability Life Cycles. Proceedings of the 34th International Conference on Software Engineering (ICSE). 2012, pp. 7–9.
5. CERT-EU. Case Study: Insider Threat in Critical Infrastructure: technical report VF-2020-067. 2020. URL: https://cert.europa.eu/static/WhitePapers/CERT-EU_WhitePaper_Insider_Threats.pdf (accessed: 15.05.2025).
6. Ponemon Institute. Cybersecurity of Air-Gapped Industrial Control Systems: research report. 2023. URL: <https://www.ponemon.org/library/air-gapped-ics-security> (accessed: 15.05.2025).
7. FireEye. M-Trends 2024: The Air-Gap Dilemma : analytical review. 2024. URL: <https://www.fireeye.com/content/dam/fireeye-www/current-threats/pdfs/mtrends-2024.pdf> (accessed: 15.05.2025).
8. SANS Institute. Offline Patching Risks: A Quantitative Analysis : scientific report WP-2022-187. 2022. URL: <https://www.sans.org/white-papers/offline-patching-risks/> (accessed: 15.05.2025).
9. Ministry of Digital Development, Communications and Mass Media of the Russian Federation. Report on the Implementation of Domestic Software in Government Agencies. URL: <https://digital.gov.ru/uploaded/files/otchet-2023.pdf> (accessed: 15.05.2025). pp. 34–35.
10. Federal Security Service of the Russian Federation. Statement for the Media June 14 2025 “On the Implementation of Import Substitution Projects”. *Official Website of the FSB of Russia*. URL: <https://fsb.ru/fsb/press/message/single.htm> (accessed: 15.05.2024).
11. Astra Linux Help Center: Latest Updates for Astra Linux Special Edition. 2025. URL: <https://wiki.astralinux.ru/pages/viewpage.action?pageId=36077663> (accessed: 09.06.2025).
12. Software infrastructure of JSC NPO RusBI-Tech’s own development. 2025. URL: <https://rusbitech.ru/products/zptk/ptk-karkas-s> (accessed: 09.06.2025).
13. CVSS v3.1 Specification Document. Revision 1. 2019. URL: <https://www.first.org/cvss/specification-document> (accessed: 09.06.2025).
14. ClamAV: documentation. *Cisco Systems*. 2025. URL: <https://docs.clamav.net/> (accessed: 09.06.2025).
15. **Reshetilov A.** Linux Malware Detect: detecting malware on web servers. *Habr*. 2013. URL: <https://habr.com/ru/articles/194346/> (accessed: 09.06.2025).
16. Rootkit Hunter: administrator’s guide. *Linode LLC*. 2023. URL: <https://www.linode.com/docs/guides/anti-virus-and-rootkit-protection-with-clamav-and-rkhunter/> (accessed: 09.06.2025).
17. Kaspersky XDR: compliance with GOST R 59712-2022. *AO Kaspersky Lab*. 2025. URL: <https://support.kaspersky.com/help/XDR/1.3/ru-RU/283206.htm> (accessed: 09.06.2025).
18. Kiinitix. Malware-Detection-using-Machine-learning. 2024. URL: <https://github.com/Kiinitix/Malware-Detection-using-Machine-learning> (accessed: 09.06.2025).
19. Anushthakalia. Malware_analysis. URL: https://github.com/anushthakalia/Malware_analysis (accessed: 09.06.2025).
20. Cuckoo Sandbox: Automated Malware Analysis System. *Cuckoo Foundation*. URL: <https://www.cuckoosandbox.org/> (accessed: 09.06.2025).

СВЕДЕНИЯ ОБ АВТОРАХ / INFORMATION ABOUT AUTHORS

РУССУ Валерий Юрьевич – адъюнкт, Военно-космическая академия имени А. Ф. Можайского, Россия, 197198, Санкт-Петербург, Ждановская ул., д. 13
E-mail: russu_valeriy@mail.ru

RUSSU Valery Yu. – Adjunct, A. F. Mozhaysky Military Space Academy, Russia, 197198, St. Petersburg, Zhdanovskaya str., 13

КАЗЬМИН Денис Юрьевич – адъюнкт, Военно-космическая академия имени А. Ф. Можайского, Россия, 197198, Санкт-Петербург, Ждановская ул., д. 13
E-mail: nemaskin3112@mail.ru

KAZMIN Denis Yu. – Adjunct, A. F. Mozhaysky Military Space Academy, Russia, 197198, St. Petersburg, Zhdanovskaya str., 13

БИРЮКОВ Денис Николаевич – д-р техн. наук, профессор, начальник кафедры, Военно-космическая академия имени А. Ф. Можайского, Россия, 197198, Санкт-Петербург, Ждановская ул., д. 13
E-mail: Biryukov.D.N@yandex.ru
ORCID: 0000-0003-1300-2470

BIRYUKOV Denis N. – Doctor of Engineering Sciences, Professor, Head of the Department, A. F. Mozhaysky Military Space Academy, Russia, 197198, St. Petersburg, Zhdanovskaya str., 13