DOI 10.36622/1682-7813.2024.27.4.010
УДК 004.056

РАЗРАБОТКА И РЕАЛИЗАЦИЯ ПОДХОДА АВТОМАТИЗИРОВАННОГО СОПОСТАВЛЕНИЯ ВЕРСИЙ И НАИМЕНОВАНИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МЕЖДУ ФОРМАТАМИ COMMON PLATFORM ENUMERATION И БАНКА ДАННЫХ УГРОЗ БЕЗОПАСНОСТИ ИНФОРМАЦИИ ФСТЭК РОССИИ

А.А. Гончаров, М.А. Тарелкин, Д.А. Нархов

В данной статье представлен разработанный и программно-реализованный подход, позволяющий автоматизировать сопоставление (трансляцию) версий и наименований программного обеспечения между форматами представления Common Platform Enumeration и форматом Банка данных угроз безопасности информации ФСТЭК России. Рассмотрены структуры и форматы атрибутов этих баз данных, проведен их сравнительный анализ. Разработан алгоритм сопоставления наименований программного обеспечения. Разработана схема сопоставления версий программного обеспечения. Программно реализовани трансляция версий и названия программного обеспечения. Актуальность исследования обусловлена необходимостью повышения защищённости отечественных информационных систем, путём обогащения сведений в БДУ ФСТЭК России через расширение номенклатуры программного обеспечения, детектируемого отечественными сканерами уязвимостей.

Ключевые слова: уязвимости, номенклатура программного обеспечения, сканеры уязвимостей, Common Platform Enumeration, Банк данных угроз безопасности информации

В условиях ограничения использования зарубежных средств обеспечения информационной безопасности и анализа защищенности возникла необходимость формирования (совершенствования) отечественных импортозамещающих программных продуктов. Одним из видов такого программного обеспечения являются сканеры уязвимостей. В зарубежных сканерах уязвимостей отсутствует возможность детектирования уязвимостей в большинстве видов отечественного программного обеспечения (ПО), поэтому их применение не позволяет адекватно оценить наличие уязвимостей в проверяемой инфраструктуре, построенной на отечественном программном и программно-аппаратном обеспечении. Отечественные сканеры (Сканер-ВС, MaxPatrol VM и др.), в свою очередь, опираются, как на отечественные, так и на зарубежные источники сведений об уязвимостях, но при таком подходе возникает проблема, связанная с отсутствием соотнесения наименований и версий ПО, что в рамках одного продукта (сканера) порождает сложности при разработке правил детектирования и соотнесении результатов обнаружений (детектов).

Банк данных угроз безопасности информации ФСТЭК России (БДУ) [1] представляет номенклатуру зарубежного ПО более полно, чем база данных (БД) National Vulnerability Database (NVD) [2]. Записи в БД NVD с идентификатором CVE (Common Vulnerabilities and Exposures) и БДУ с идентификатором BDU представляют собой совокупность параметров, однозначно идентифицирующих конкретную уязвимость в программных или программно-аппаратных продуктах. Однако, количество таких параметров и их формат существенно различаются.

Например, описание уязвимости в БДУ имеет строго определённый формат, который может быть представлен в следующем виде:

Уязвимость [компонент] [тип программного обеспечения] [наименование программного обеспечения] связана с [причина возникновения уязвимости]. Эксплуатация уязвимости может позволить нарушителю, [действующему удаленно], [последствия эксплуатации уязвимости].

Указанный формат (используемый в БДУ) является универсальным и позволяет перейти к логико-лингвистическому описанию уязвимости, однако, это является другим направлением исследований авторов.

Описание уязвимости в NVD носит произвольный характер и зачастую берётся в исходном виде и формате с сайта, являющегося первоисточником сведений об уязвимости. Преимуществом базы данных NVD является её центральная роль, она выступает единым агрегатором сведений об уязвимостях. У конечного пользователя отсутствует необходимость отслеживания уведомлений безопасности десятков разработчиков, а достаточно отслеживания появления сведений на сайте NVD. Одними из ключевых недостатков базы NVD является временная задержка при заполнении поля CPE и отсутствие подробной информации, необходимой для проведения анализа сведений об уязвимости и выделения интересующих пользователя аспектов.

Наибольшую сложность для сопоставления и объединения сведений БДУ и NVD составляет корреляция версии и названия программного обеспечения. Для решения этой задачи приведём описания каждого из форматов баз данных отдельно.

Common Platform Enumeration.

Common Platform Enumeration (CPE) представляет собой структурированную схему именования систем информационных технологий, программного обеспечения, пакетов и библиотек в базе уязвимостей NVD. Основное назначение CPE заключается в стандартизации и унификации описания различных информационных продуктов, что позволяет эффективно идентифицировать и сопоставлять уязвимости [3].

Формат записи CPE выглядит следующим образом:

cpe:<cpe_version>:<part>:<vendor>:
<product>:<version>:<update>: <edition>:<language>:<sw_edition>:
<target_sw>:<target_hw>:<other>

Опишем по отдельности каждый элемент:

• cpe_version: Версия спецификации CPE, используемая в записи. Последняя версия CPE на момент написания статьи - 2.3.

• part: Тип ПО: «a» (Applications) для приложений, «o» (Operating Systems) для операционных систем, «h» (Hardware) для аппаратного обеспечения.

• vendor: Имя поставщика программного или аппаратного обеспечения. Значения этого атрибута должны описывать или идентифицировать человека, или организацию, изготовившую или создавшую продукт (например, «microsoft», «apple» и др.).

• product: Название системы, пакета или компонента. Значения для атрибутов vendor и product иногда бывают идентичны. Название не должно содержать пробелов, косых черт и специальных символов. Подчёркивание должно использоваться вместо пробелов. Например, Check Point Zone Labs ZoneAlarm Internet Security Suite 6.1.737.000 будет обозначен следующим образом:

cpe:2.3:a:zonelabs:zonealarm_security_suite:6.1.737.000::*:*:*:*:*:**

• version: Версия должна быть записана в том же формате, что и название ПО. Это означает, что разделители, такие как: точки и дефисы, должны использоваться так же, как и в названии ПО. Например, версия Adobe

Reader 9.0 будет обозначена следующим образом:

cpe:2.3:a:adobe:acrobat_reader:9.0::*:*:*:*:*:**

• update: Информация об обновлении или сервис-паке. Используется для обозначения «точечных» релизов или минорных версий. Технические различия между version и update могут отличаться в зависимости от поставщика и продукта. Общие примеры включают: beta, update4, SP1 или GA (General Availability), но чаще всего данное поле остается пустым. Например, Adobe ColdFusion 2016 update8 будет записан следующим образом:

cpe:2.3:a:adobe:coldfusion:2016:update8::*:*:*:*:*:**

• edition: Дополнительный атрибут, описывающий сборку системы, пакета или компонента, помимо version. На данный момент практически не используется за исключением случаев, когда требуется обратная совместимость с предыдущими версиями CPE. Например, все версии Microsoft Windows NT 2000 Service Pack 4 Professional Edition будут записаны следующим образом:

cpe:2.3:o:microsoft:windows-nt:2000:sp4:pro::*:*:*:**

• language: Допустимый языковой тег, может иметь значения, определённые в стандарте IETF RFC 4646 «Tags for Identifying Languages». Например, японская версия Microsoft Word версии 97 будет обозначена следующим образом:

cpe:2.3:a:microsoft:word:97::*:ja:*:*:*:**

• sw_edition: Значения для атрибута sw_edition должны характеризовать, каким образом продукт адаптирован для конкретного рынка или класса конечных пользователей. Пример: Если продукт предназначен для образовательного сектора, значение атрибута sw_edition может быть education. В таком случае строка CPE для данной версии программного обеспечения может выглядеть следующим образом:

cpe:2.3:a:vendor:product:1.0:update:
education::*:*:*:*:**

• target_sw: Операционная среда продукта. Используется редко.

• target_hw: Целевая аппаратная платформа (например, «x64» для 64-битной системы).

• other: Дополнительная информация, которая может быть полезна для идентификации продукта.

Формат Well-Formed Name (WFN).

В качестве записи уязвимых версий в CPE существует специальный формат: Well-Formed Name (WFN) [3]. WFN представляет собой набор ключ-значение, позволяющий детализировать каждую характеристику ПО.

Атрибуты WFN должны иметь одно из следующих значений:

1. Логическое значение.

2. Строка символов, удовлетворяющая одновременно требованиям к строковым значениям и ограничениям на значения по каждому атрибуту, которые будут указаны далее.

Логические значения. Атрибуту WFN может быть присвоено одно из следующих логических значений:

• ANY («любое значение»). Логическое значение ANY должно быть присвоено атрибуту, когда для этого атрибута нет ограничений на допустимые значения.

• NA («не применимо/не используется»). Логическое значение NA должно быть присвоено при отсутствии верного значения, или когда этот атрибут не используется в описании, включая ситуацию, когда у атрибута есть допустимое значение, равное *null*.

В WFN эти логические значения пишутся заглавными буквами, без кавычек, справа от знака равенства, как в следующих примерах:

wfn:[…,update=ANY,…]
wfn:[…,update=NA,…]

Ограничения на строки значений атрибутов.

Строки значений, присваиваемые атрибутам WFN, должны быть непустыми, непрерывными строками байтов, закодированными с использованием печатных символов в формате Unicode Transformation Format-8 (UTF-8) с шестнадцатеричными значениями от x00 до x7F [RFC3629].

В WFN строки значений атрибутов заключаются в двойные кавычки, как в примерах ниже. Кавычки не считаются частью самих строк значений.

wfn:[…,update=”sr1”,…]
wfn:[…,target_hw=”x64”,update=”sp2”]

Для формирования строк значений в CPE используются буквы верхнего и нижнего регистра, цифры и символ подчеркивания. Пробелы заменяются символом подчеркивания. Обратная косая черта «\» используется как символ экранирования для изменения интерпретации следующих за ней символов. Специальные символы «*» и «?» могут использоваться в начале или в конце строки значений, но не внутри нее. Для предотвращения специальной интерпретации этих символов перед ними обязана присутствовать обратная косая черта.

Специальные символы «*» и «?» используются без экранирования, когда они располагаются в начале или в конце строки значений. Например, строка *version или version? будет корректной без экранирования. Такие строки без экранирования имеют специальные значения в контексте CPE:

• *version: Звездочка «*» в начале строки (также, как и в конце строки) обозначает, что значение соответствует любому значению, которое заканчивается на version. Это полезно для указания шаблона, который охватывает множество значений с одинаковым окончанием.

• version?: Вопросительный знак «?» в конце строки (также, как и в начале строки) означает, что значение соответствует version плюс любой одиночный символ. Это полезно для указания конкретного значения с вариативностью только в одном символе.

Знаки препинания и другие специальные символы (кроме «*» и «?») экранированы при включении в строки значений. Например, строка version!1.0 должна быть записана как version\!1.0, а update#2 как update\#2. Эти правила в спецификации CPE обеспечивают однозначность и корректность идентификации компонентов.

Пример WFN:

wfn:[part=”o”, vendor=”microsoft”, product=”windows_server_2008”, version=”r2”, update=”sp1”, edition=ANY, language=ANY, sw_edition=ANY, target_sw=ANY, target_hw=”itanium”, other=ANY]

Помимо данного представления спецификация CPE определяет две дополнительные машиночитаемые формы:

• URI представление, пример:

cpe:/o:microsoft:windows_server_2008:r2:sp1:~~~~itanium~

• Представление в виде форматированной строки, пример:

cpe:2.3:o:microsoft:windows_server_2008:r2:sp1::*:*:*:itanium:**

Формат БДУ

В БДУ отсутствует строго формализованное стандартное представление уязвимых версий программного обеспечения, аналогичное спецификации CPE. Однако, в паспорте уязвимости можно выделить следующие компоненты, которые соответствуют устоявшимся правилам ведения БДУ [1]:

• производитель,

• тип уязвимого программного обеспечения,

• наименование программного обеспечения,

• диапазон версий,

• программно-аппаратная платформа.

Введем удобный формат представления БДУ для трансляции версий:

<Тип уязвимого программного обеспечения>,<Производитель>,
<Наименование программного обеспечения>,<Диапазон версий>,
<Программно-аппаратная платформа>.

В отдельном поле паспорта уязвимости указывается операционная система, однако, поскольку это поле является независимым, отсутствует возможность прямого связывания конкретных уязвимых версий программного обеспечения с конкретными версиями операционных систем.

Компоненты «Производитель» и «Наименование программного обеспечения» могут содержать любые символы, без ограничений на их заполнение. В отличие от спецификации CPE, нет необходимости использовать специальные символы или следовать строгим правилам форматирования. Это обеспечивает гибкость при заполнении этих полей, но также может усложнить автоматическую обработку и сопоставление данных.

Тип уязвимого ПО может иметь одно или несколько значений:

• прикладное ПО информационных систем,

• ПО сетевого программно-аппаратного средства,

• сетевое программное средство,

• операционная система,

• программное средство АСУ ТП,

• программное средство защиты,

• ПО программно-аппаратных средств защиты,

• ПО программно-аппаратного средства АСУ ТП,

• ПО виртуализации/ПО виртуального программно-аппаратного средства,

• СУБД,

• микропрограммный код,

• ПО виртуализации и ПО виртуальных аппаратных средств.

Поле «Версия программного обеспечения» имеет один из следующих форматов:

• «*версия*» – если уязвимость выявлена в конкретной версии программного обеспечения,

• «**от** *версия1* **до** *версия2*» – если уязвимость выявлена в нескольких версиях программного обеспечения, лежащих в некотором диапазоне, исключая версию 2,

• «**до** *версии*» – если уязвимость выявлена во всех предыдущих версиях программного обеспечения до указанной,

• «**до** *версии* **включительно**» – если уязвимость выявлена в конкретной и во всех предыдущих версиях программного обеспечения.

Пример введённого формата БДУ:

Прикладное ПО информационных систем,
Сообщество свободного программного обеспечения, Gradio, до 4.20, Не указана.

Сравнительный анализ форматов представления данных об уязвимых версиях ПО в БДУ и CPE.

Описания форматов, принятых в БДУ и CPE были представлены 10 аналитикам уязвимостей БДУ ФСТЭК России. Среди указанных экспертов проводился опрос, который включал в себя наименование компонента формата БДУ и CPE. От эксперта требовалось соотнести форматы и установить соответствия, если это возможно. Итоговые результаты опроса приведены в табл. 1.

Таблица 1

Сравнительный анализ формата БДУ и CPE

Компоненты		Сравнение
Формат БДУ	Формат CPE	
Тип уязвимого программного обеспечения	part	CPE предполагает 3 вида типов программного обеспечения, в то время как формат БДУ 12. Предлагается провести следующее соответствие между ними: «a» (Applications): Прикладное ПО информационных систем, Сетевое программное средство, Программное средство защиты, СУБД. «o» (Operating Systems): Операционная система. «h» (Hardware): ПО сетевого программно-аппаратного средства, Программное средство АСУ ТП, ПО программно-аппаратных средств защиты, ПО программно-аппаратного средства АСУ ТП, ПО виртуализации/ПО виртуального программно-аппаратного средства, Микропрограммный код, ПО виртуализации и ПО виртуальных аппаратных средств.
Производитель	vendor	Для корректной трансляции данного поля паспорта уязвимости необходимо учитывать использование специальных символов («*» и «?»), символа экранирования, а также нижнего подчеркивания вместо пробелов, применяемых в формате CPE. В процессе трансляции данных из CPE в БДУ необходимо осуществлять очистку строк от этих символов, чтобы обеспечить правильное и однозначное отображение данных в формате БДУ.
Наименование программного обеспечения	product	
Диапазон версий	version	Трансляция версий из формата CPE в формат БДУ и наоборот представляет собой сложный процесс, учитывающий несколько ключевых факторов: • Отсутствие поддержки диапазонов версий в CPE. В спецификации CPE указаны только одиночные версии, в то время как в БДУ можно указывать диапазоны версий. Формат версий в БДУ фактически соответствует формату JSON, используемому для представления уязвимостей в NVD, где также поддерживаются диапазоны версий. • Различия в структуре версий. В CPE версия может состоять из трех частей: version, update и edition. В БДУ эти компоненты объединяются и заполняются на основе интерпретации эксперта, что может привести к различиям в отображении данных. Эти аспекты делают процесс трансляции данных между форматами сложным и требуют более внимательного подхода для обеспечения корректного отображения информации. Эти и другие проблемные вопросы будут рассмотрены в следующих разделах текста..
	update	
	edition	

Продолжение табл. 1

-	language	Аналоги данных полей CPE в формате БДУ отсутствуют.
-	sw_edition	
Операционная система	target_sw	Поле «Операционная система» в CPE представляет собой текстовое значение, тогда как в БДУ данное поле — это отдельная запись, включающая в себя информацию о производителе, версии и программно-аппаратной платформе. Такой формат БДУ обладает большей гибкостью, поскольку позволяет более детально описывать операционную систему. Однако, отсутствие прямого сопоставления конкретной операционной системы и уязвимого программного обеспечения с версией нивелирует это преимущество. В случаях, когда имеется более одного уязвимого программного средства и более одной версии операционной системы, это может привести к неоднозначности и затруднениям в анализе и интерпретации данных.
Программно-аппаратная платформа	target_hw	Программно-аппаратная платформа как в БДУ, так и в CPE представлена в виде строкового значения, что упрощает их взаимную трансляцию. Однако для обеспечения корректного и точного сопоставления необходимо учитывать некоторые важные аспекты единообразия и стандартизации. Строковые значения должны быть стандартизированы в обоих форматах. Это включает использование одинаковых наименований для одних и тех же аппаратных и программных компонентов, что исключит двусмысленность при трансляции данных.
-	other	Аналог данного поля CPE в формате БДУ отсутствует.

Трансляция программного обеспечения между форматами CPE и БДУ

Рассмотрим трансляцию уязвимостей из CPE (NVD) в БДУ и обратно поэтапно. Начнем с трансляции типов ПО (табл. 2).

Таблица 2

Трансляция типов ПО

CPE (NVD)	БДУ
«a» (Applications)	Прикладное ПО информационных систем
	Сетевое программное средство
	Программное средство защиты
	СУБД
«o» (Operating Systems)	Операционная система

Продолжение табл. 2

«h» (Hardware)	ПО сетевого программно-аппаратного средства
	Программное средство АСУ ТП
	ПО программно-аппаратных средств защиты
	ПО программно-аппаратного средства АСУ ТП
	ПО виртуализации/ПО виртуального программно-аппаратного средства
	Микропрограммный код
	ПО виртуализации и ПО виртуальных аппаратных средств

В ходе анализа баз данных NVD и БДУ было установлено, что названия производителей (БДУ: Производитель / CPE: vendor) ПО в двух базах данных значительно отличаются. Это контрастирует с названиями самих программных продуктов (БДУ: Наименование ПО / CPE: product), которые, как правило, более схожи.

В связи с этим, первоочередное внимание было уделено вопросу сопоставления названий ПО. Далее уже по известному сопоставлению названий ПО будет произведено сопоставление названий производителей.

Для решения задачи сопоставления названий ПО была применена библиотека языка программирования Python datasketch, которая позволяет эффективно рассчитывать сходство строковых значений с использованием алгоритма MinHash и структуры MinHashLSH. MinHash является методом приближенного вычисления коэффициента Жаккара, что позволяет быстро и точно оценивать степень схожести между множествами биграмм строк. MinHashLSH (Locality-Sensitive Hashing) используется для эффективного поиска схожих элементов в больших объемах данных [3-5].

Алгоритм сопоставления ПО

Алгоритм, используемый для сопоставления названий программного обеспечения, включает несколько ключевых этапов:

1. Извлечение наименований ПО.

Извлечение данных из json-файлов Feed NVD [2]:

• Скачивание и обработка json -файлов, публикуемых в Feed NVD, содержащих информацию о конкретных уязвимостях CVE по годам.

• Разбор json-файлов и выделение информации об уязвимых программных продуктах для конкретных CVE. Для этого используется функция, которая анализирует структуры json и извлекает поле cpe23Uri, содержащее сведения о продукте.

Извлечение данных из xlsx-файлов с сайта БДУ:

• Скачивание xlsx-файла с сайта БДУ и выделение двух основных колонок: внешние идентификаторы (CVE) и наименование ПО.

• Формирование двух списков: один из них содержит наименования ПО из БДУ, другой — из NVD. Эти списки представляют собой уникальные пары для каждой уязвимости CVE.

2. Сравнение строк с использованием MinHash:

• Каждое наименование программного обеспечения преобразуется в набор биграмм, что позволяет более точно анализировать схожесть между строками.

• Для каждой строки вычисляется MinHash, что позволяет оценить степень схожести между строками с использованием коэффициента Жаккара. MinHash предоставляет эффективный способ вычисления этого коэффициента.

• Специальная функция MinHash используется для определения степени схожести между наименованиями программного обеспечения из разных списков. Это позволяет сравнивать строки и определять их схожесть на основе расчетов MinHash.

• Использование специальной функции MinHash для определения степени схожести между названиями программного обеспечения (в качестве значения порога схожести выбрано число 0,35).

3. Сохранение результатов в Excel-файл:

• Попарное сравнение элементов из списков БДУ и NVD с использованием рассчитанных MinHash значений для определения степени их схожести.

• Результаты сравнения, включая схожие наименования программного обеспечения и коэффициент их схожести, сохраняются в Excel-файл. Этот файл будет использован для дальнейшего анализа и обработки данных.

Выбор порога схожести в 0.35 для алгоритма MinHash был обоснован на основе экспериментальных исследований и теоретических основ метода MinHash [3-5].

Были проведены эксперименты для оценки эффективности различных порогов схожести при сравнении наименований ПО из баз данных NVD и БДУ. Были протестированы различные значения порога,

чтобы определить оптимальный уровень, при котором достигается максимальная точность и полнота сопоставления анализируемых данных (табл. 3, рис. 1).

Опишем формулу (1) расчета точности (Precision):

$$recision = \frac{TP}{TP + FP}, \quad (1)$$

где *TP* (True Positives) – количество правильно сопоставленных пар наименований,

FP (False Positives) – количество пар наименований, которые были сопоставлены ошибочно.

Опишем формулу (2) расчета полноты (Recall):

$$Recall = \frac{TP}{TP + FN}, \quad (2)$$

где *TP* (True Positives) – количество правильно сопоставленных пар наименований,

FN (False Negatives) – количество пар наименований, которые должны были бы быть сопоставлены программой.

Точность показывает долю правильно сопоставленных наименований среди всех сопоставлений.

Полнота показывает долю правильно сопоставленных наименований среди всех возможных правильных сопоставлений.

Таблица 3

Соотношение между точностью и полнотой

Порог	Точность	Полнота
0.25	0.68	0.58
0.30	0.73	0.63
0.35	0.8	0.75
0.40	0.76	0.72
0.45	0.7	0.68

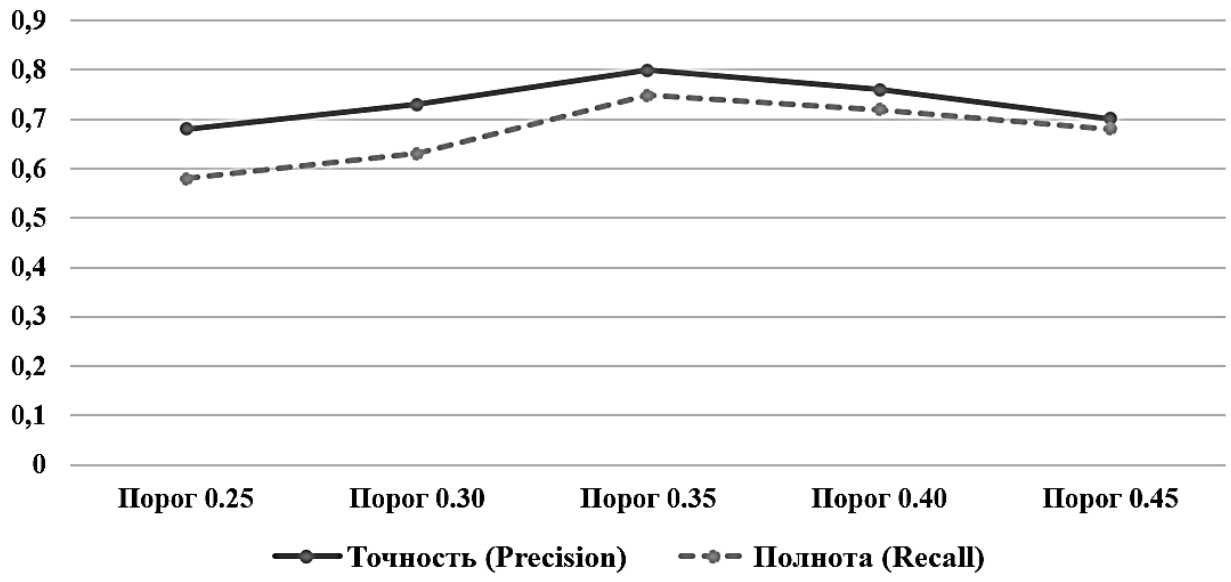

Рис. 1. Зависимость точности и полноты от порога схожести

Результаты показали, что порог в 0.35 обеспечивает наиболее сбалансированное соотношение между точностью и полнотой.

Таким образом, в контексте сопоставления наименований ПО из баз данных: NVD и БДУ, порог 0.35 оказался достаточно высоким, чтобы учитывать вариативность в написании наименований, и в то же время достаточно низким, чтобы не пропустить действительно схожие наименования.

Применение библиотеки datasketch и алгоритма MinHash для сравнения строк позволяет эффективно и точно сопоставлять наименования ПО.

Был получен предварительный необработанный выходной файл сравнения наименований. Данный Excel файл включает три столбца: первый столбец содержит наименования ПО из базы данных БДУ, второй столбец — наименования из спецификации CPE, а третий столбец — мера схожести, вычисленная с использованием метода MinHash, при условии, что она превышает пороговое значение 0.35. Данный файл содержит 5369 строк.

Несмотря на то, что файл формировался без учета дублирующихся записей, когда пары «БДУ: Наименование ПО / CPE: product» повторяются, он нуждается в дальнейшей обработке по следующим причинам:

1. В некоторых случаях одному наименованию ПО из БДУ соответствуют несколько различных наименований из CPE. Например, наименование «Windows» из БДУ может соответствовать различным версиям в CPE, таким как «windows_server_2016», «windows_10», «windows_rt_8.1», «windows_8.1», «windows_server_2008» и т.д. В CPE версия указывается в названии через нижнее подчеркивание, тогда как в БДУ версия указывается в отдельном поле. Такие случаи требуют отдельного анализа и внесения исключений в процессе трансляции наименований (или вынесения таких случаев в отдельную задачу для будущих исследований).

2. В других случаях одному наименованию из CPE может соответствовать несколько различных наименований из БДУ. Например, одна запись в CPE может быть сопоставлена с несколькими записями в БДУ, что также требует ручной фильтрации ложных срабатываний для корректного сопоставления.

После удаления ложных срабатываний было оставлено **3586** сопоставлений ПО. Также были выделены **30** пар, которые должны транслироваться особым образом, как это показано на примере с «Windows».

Далее к каждому наименованию ПО были добавлены соответствующие наименования производителей и тип программного обеспечения как для БДУ, так и для CPE, путем автоматического поиска и сопоставления в исследуемых базах данных. Таким образом, был подготовлен словарь, содержащий наименования производителей, наименования программных средств и их типов для последующего применения при трансляции данных между форматами БДУ и CPE (фрагмент данного словаря представлен на рис. 2)

3580	Прикладное ПО информаı	cbeust	TestNG	a	testng_projec	testng
3581	Прикладное ПО информаı	Сообщество свободнс	ZeroMQ	a	zeromq	zeromq
3582	ПО программно-аппаратн	Broadcom Inc.	Symantec Identity (	a	broadcom	symantec_identity_governance_and_administration
3583	ПО сетевого программно-	Fortinet Inc.	FortiAnalyzer-BigD	a	fortinet	fortianalyzer_bigdata
3584	Прикладное ПО информаı	Сообщество свободнс	uClibc	a	uclibc	uclibc
3585	Прикладное ПО информаı	VideoLan Technologies	dav1d	a	videolan	dav1d

Рис. 2. Фрагмент словаря сопоставлений наименований производителей, программных средств и их типов в БДУ и CPE

На рис. 2:
1-3 столбцы – БДУ:
- тип программного обеспечения,
- наименование производителя,
- наименование программного обеспечения;

4-6 столбцы – CPE:
- тип программного обеспечения, наименование производителя,
- наименование программного обеспечения;
- 3586 строк-сопоставлений

Составление *словаря версий* программного обеспечения признано нецелесообразным по следующим причинам:

1. Большое количество версий и их постоянные изменения: Количество версий программного обеспечения большое и они постоянно обновляются, что делает отслеживание изменений крайне сложным и затратным.

2. Различие в форматах версий в БДУ и CPE: Форматы версий в БДУ и CPE значительно отличаются. Версии в БДУ ближе к формату NVD, где представлены как диапазоны версий, так и отдельные версии. В CPE, напротив, все уязвимые версии представлены отдельно.

Для трансляции версий программного обеспечения между форматами БДУ и NVD предлагается следующая схема сопоставления (табл. 4).

Таблица 4

Трансляция версий программного обеспечения между форматами БДУ и NVD

Формат NVD	Формат БДУ
version	***version***
From (including) ***version***	от включительно ***version****
From (excluding) ***version***	от ***version***
From (excluding) ***version_1*** Up to (excluding) ***version_2***	от ***version_1*** до ***version_2***
From (including) ***version_1*** Up to (excluding) ***version_2***	от включительно ***version_1*** до ***version_2****
From (excluding) ***version_1*** Up to (including) ***version_2***	от ***version_1*** до ***version_2*** включительно
From (including) ***version_1*** Up to (including) ***version_2***	от ***version_1*** включительно до ***version_2*** включительно****
Up to (excluding) ***version***	до ***version***
Up to (including) ***version***	до ***version*** включительно

** Эти форматы версий отсутствуют в БДУ, поэтому целесообразно добавить их для повышения точности трансляции уязвимых версий.*

Предложенные изменения позволят улучшить точность и полноту сопоставления версий программного обеспечения между различными форматами.

Программная реализация трансляции версии и названия программного обеспечения

Трансляция формата CPE в БДУ

Для начала рассмотрим трансляцию данных из форматов CPE в формат БДУ. Этот процесс включает преобразование всех трех форматов представления CPE в формат БДУ. Он был реализован в виде алгоритма на языке программирования Python. Алгоритм использует библиотеку pandas для работы с таблицами, а также регулярные выражения и запросы для обработки данных.

Весь алгоритм представлен на рис.3. Основные этапы алгоритма следующие:

1. Парсинг названий версий и программного обеспечения. Алгоритм начинает с разбора входных данных для определения версий и названия ПО.

2. Определение формата представления данных. Алгоритм проверяет, к какому из трех форматов CPE относятся представленные данные и разбирает их на составные части в соответствии с соответствующим форматом.

3. Трансляция наименований.

3.1. Проверка наличия совпадений в таблице соответствий (рис. 2). Алгоритм (рис. 3) проверяет, существует ли точное соответствие для названий ПО и производителей в формате CPE в таблице соответствий. Если такие соответствия найдены, извлекаются соответствующие названия для формата БДУ.

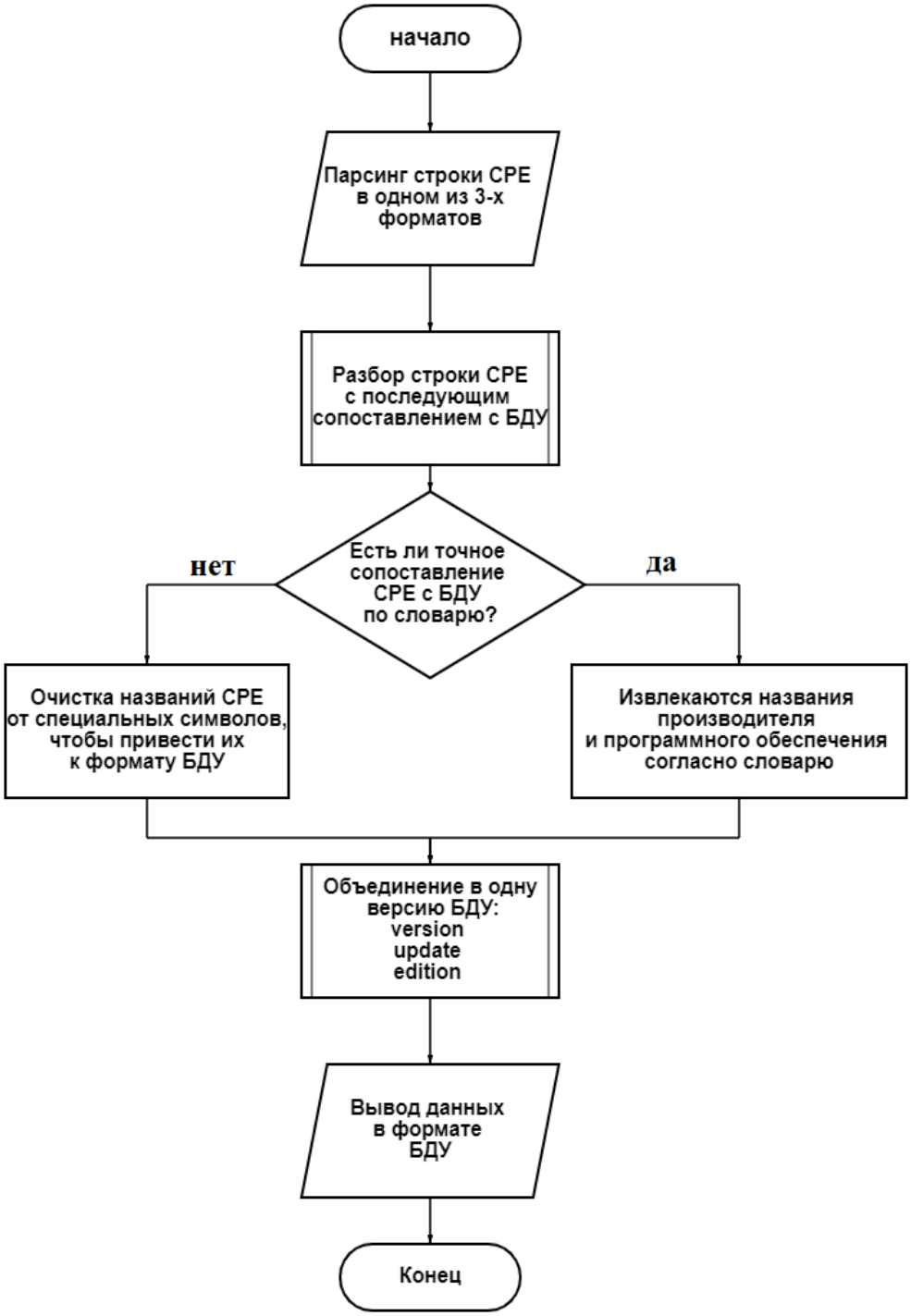

Рис. 3. Алгоритм трансляции версии и названия программного обеспечения из CPE в БДУ

```
Входные данные:
wfn:[part="o", vendor="linux", product="linux_kernel", version="r2", update="sp1", edition=ANY,
language=ANY, sw_edition=ANY, target_sw=ANY, target_hw="itanium", other=ANY]
Выходные данные:
Операционная система,Сообщество свободного программного обеспечения,Linux,r2 sp1,Не указана
[Finished in 2.6s]
```

Рис. 4. Пример применения алгоритма трансляции версий и ПО из формата WFN CPE в формат БДУ

3.2. Очистка строк с наименованиями ПО и производителя формата CPE. В случае отсутствия точного совпадения в таблице соответствий, алгоритм очищает наименования ПО, производителя формата CPE от специальных символов, чтобы привести их к формату БДУ.

4. Объединение некоторых полей CPE в одно поле версии БДУ. Поля CPE <version>, <update>, <edition> объединяются в одно поле для формирования версии БДУ. При этом также происходит очистка данных от специальных символов.

5. Обработка специальных символов. Программа учитывает, что в поле <version> могут присутствовать специальные символы («*» и «?»). Эти символы сохраняются и выводится специальное предупреждение, поскольку в формате БДУ такие обозначения не предусмотрены. *Внедрение подобных символов в БДУ может повысить эффективность и гибкость представления уязвимых версий и программного обеспечения.*

Таким образом, данный алгоритм обеспечивает автоматизированную трансляцию данных между форматами CPE и БДУ.

Далее рассмотрим примеры применения данного алгоритма (рис. 4-6).

На рис. 4 представлен пример трансляции версий и программного обеспечения из формата WFN CPE в формат БДУ.

На рис. 5 представлен пример трансляции версий и ПО из формата CPE в виде форматированной строки в формат БДУ.

На рис. 6 представлен пример трансляции версий и ПО из формата CPE в виде URI представления в формат БДУ.

```
Входные данные:
cpe:2.3:o:linux:linux_kernel:r2:sp1:*:*:*:*:*:*:*
Выходные данные:
Операционная система,Сообщество свободного программного обеспечения,Linux,r2 sp1,Не указана
[Finished in 2.4s]
```

Рис. 5. Пример применения алгоритма трансляции версий и ПО из формата CPE в виде форматированной строки в формат БДУ

```
Входные данные:
cpe:/o:linux:linux_kernel:r2:sp1:~~~~~~
Выходные данные:
Операционная система,Сообщество свободного программного обеспечения,Linux,r2 sp1,Не указана
[Finished in 1.7s]
```

Рис. 6. Пример применения алгоритма трансляции версий и ПО из формата CPE в виде URI представления в формат БДУ

Трансляция формата NVD в БДУ

Рассмотрим процесс трансляции данных из формата NVD в формат БДУ. Этот процесс включает извлечение данных об уязвимостях, представленных на платформе NVD в формате json, которые можно получить по запросу через API.

Целью данного алгоритма является конвертация данных о программном обеспечении из формата CPE в формат БДУ. Для реализации алгоритма используются библиотеки requests и pandas, а также преобразование данных в формате json.

Основные этапы алгоритма следующие:

1. Получение данных об уязвимости. Посредством API запрашиваются данные об уязвимости по идентификатору CVE. Запрос возвращает данные в формате json, содержащие подробную информацию об уязвимости.

2. Парсинг данных. Алгоритм анализирует полученные данные, выделяя версии программного обеспечения. Важно отметить, что в NVD формат CPE может содержать как единичные версии, так и диапазоны версий. Единичные версии представлены в стандартном формате CPE, тогда как диапазоны версий указаны в формате json за пределами представления CPE.

3. Обработка данных. Алгоритм извлекает и обрабатывает диапазоны версий CPE, транслируя их в формат БДУ. Для этого анализируются поля json, содержащие информацию о диапазонах версий, и формируются соответствующие записи в формате БДУ.

4. Проверка соответствий. На следующем этапе алгоритм проверяет наличие названий ПО в таблице соответствий (рис. 2). Если соответствие найдено, то извлекаются необходимые данные в формате БДУ. Если соответствие не найдено, названия программного обеспечения из CPE очищаются от специальных символов для приведения их к формату БДУ.

5. Трансляция данных. Затем производится трансляция диапазонов версий из NVD в формат БДУ (в соответствии с таблицей 2).

Данный алгоритм, реализованный в виде программы на языке Python, можно представить в виде следующей блок-схемы (рис. 7).

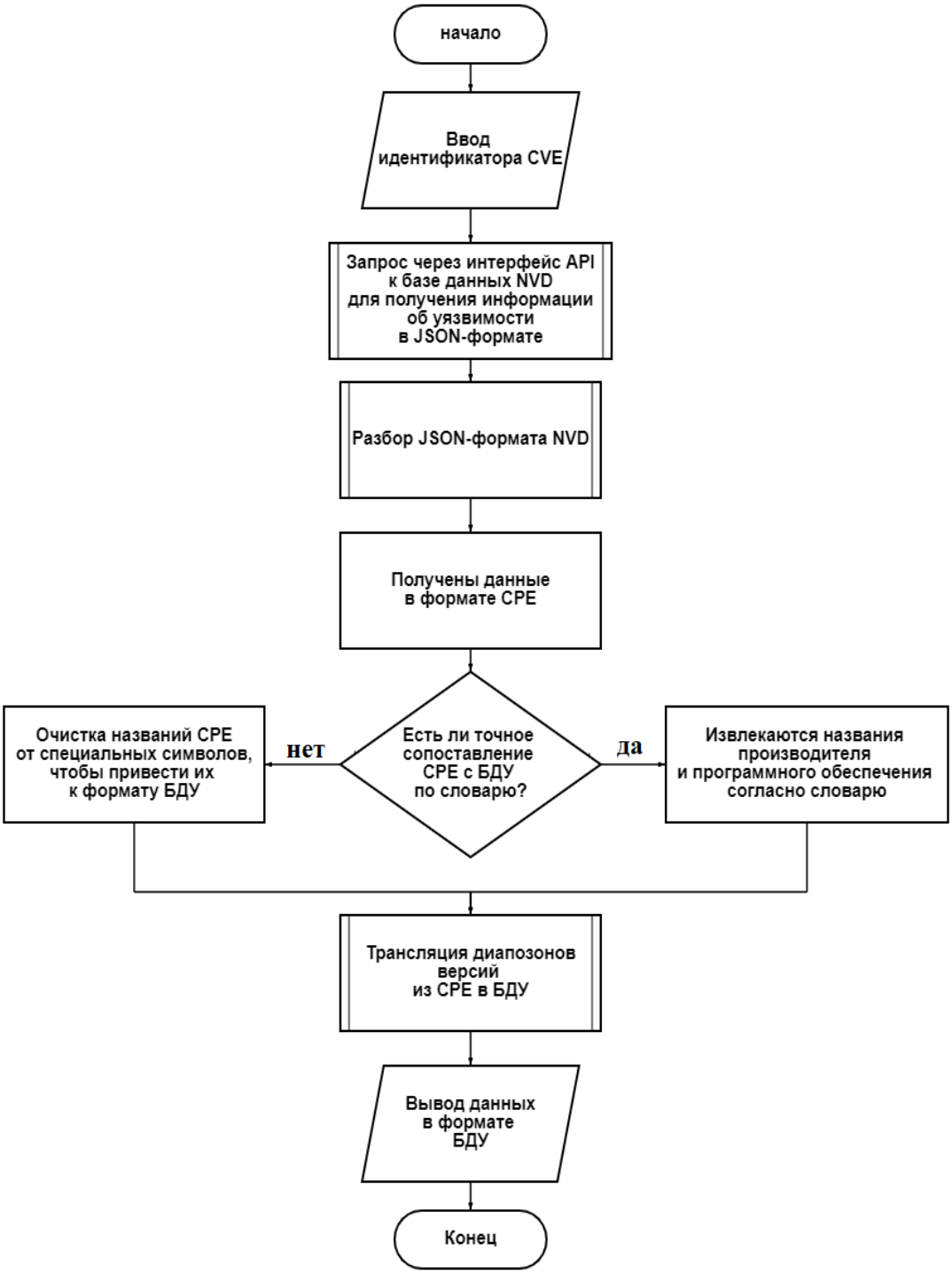

Рис. 7. Алгоритм трансляции версии и названия ПО из NVD в БДУ

В качестве примера может рассматриваться уязвимость CVE-2023-2163 (BDU:2023-03785). Версии представлены на NVD следующим образом – рис. 8.

На рис. 9 представлен пример трансляции версий и ПО из формата NVD в формат БДУ для данной уязвимости.

cpe:2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:* Show Matching CPE(s)	From (including) 5.3	Up to (excluding) 5.4.242
cpe:2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:* Show Matching CPE(s)	From (including) 5.5	Up to (excluding) 5.10.179
cpe:2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:* Show Matching CPE(s)	From (including) 5.11	Up to (excluding) 5.15.109
cpe:2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:* Show Matching CPE(s)	From (including) 5.16	Up to (excluding) 6.1.26
cpe:2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:* Show Matching CPE(s)	From (including) 6.2	Up to (excluding) 6.2.13

Рис. 8. Версии уязвимости с идентификатором CVE-2023-2163 представленные на сайте NVD

```
Операционная система,Сообщество свободного программного обеспечения,Linux,от 5.3 до 5.4.242,Не указана
Операционная система,Сообщество свободного программного обеспечения,Linux,от 5.5 до 5.10.179,Не указана
Операционная система,Сообщество свободного программного обеспечения,Linux,от 5.11 до 5.15.109,Не указана
Операционная система,Сообщество свободного программного обеспечения,Linux,от 5.16 до 6.1.26,Не указана
Операционная система,Сообщество свободного программного обеспечения,Linux,от 6.2 до 6.2.13,Не указана
[Finished in 19.2s]
```

Рис. 9. Пример применения алгоритма трансляции версий и ПО из формата NVD в формат БДУ для уязвимости с идентификатором CVE-2023-2163

Трансляция формата БДУ в CPE

Рассмотрим процесс трансляции данных из формата БДУ в формат CPE. Данный процесс включает преобразование информации о ПО, содержащейся в БДУ, в формат CPE. В отличие от трансляции данных из БДУ в формат NVD, которая сводится к простому переводу терминов, таких как «включительно», «от» и «до», в соответствующие диапазоны версий, трансляция в формат CPE требует более сложного подхода.

Процесс трансляции в формат NVD не представляет значительного интереса, так как он достаточно прямолинейный. Важно также отметить, что формат версий в NVD представляет собой сложную структуру json, в то время как в БДУ версии представлены в виде обычных строковых значений.

Одной из особенностей формата CPE является невозможность использования диапазонов версий, таких как «From (including) version_1 Up to (excluding) version_2» и т.п. Вместо этого требуется явное перечисление всех возможных версий ПО. Для этого необходимо использовать официальный словарь CPE, который доступен в формате XML и может быть скачан по соответствующей ссылке [2].

Конечный результат работы алгоритма должен быть представлен в виде списка версий в формате CPE, если исходные данные включали диапазоны версий в формате БДУ, либо одной версии в формате CPE для одной версии в формате БДУ, если диапазоны не использовались.

Алгоритм трансляции данных из формата БДУ в формат CPE можно разделить на следующие основные этапы:

1. Процесс валидации данных. На данном этапе алгоритм проверяет корректность ввода данных, включая валидацию форматов диапазонов версий. Валидация осуществляется на основе предопределенных форматов, представленных в табл. 4. Если данные не соответствуют требованиям формата, выводится сообщение об ошибке, что позволяет предотвратить дальнейшую обработку некорректной информации.

2. Проверка соответствий. Алгоритм осуществляет проверку наличия соответствий между названиями ПО и производителей в таблице сопоставлений (рис. 2). Если обнаружено полное соответствие по названиям производителя и ПО, из таблицы извлекаются необходимые данные, соответствующие формату CPE. В случае отсутствия соответствия алгоритм выводит сообщение об ошибке, что указывает на невозможность корректной трансляции данных.

3. Трансляция наименований.

3.1. Если указана одна версия ПО, она преобразуется в формат CPE путем создания форматированной строки, соответствующей структуре версий CPE.

3.2. Если указан диапазон версий, алгоритм обращается к словарю CPE и выбирает все версии, попадающие в указанный диапазон. Если в диапазоне указано «включительно» для какой-либо версии, она автоматически включается в итоговый список, даже если её нет в словаре CPE, так как предполагается её существование. В данном процессе важную роль играет встроенная функциональность языка Python, которая позволяет эффективно сравнивать версии, разбивая их на составляющие (числовые и буквенные значения).

Например, Python корректно определяет, что «123.abc.345» < «123.abr.001», что облегчает обработку сложных версий без необходимости написания дополнительного кода.

4. На заключительном этапе алгоритм формирует и выводит список версий в формате CPE. Этот список представляет собой конечный результат трансляции, готовый к использованию в дальнейших процессах или для интеграции с другими системами.

Данный алгоритм, реализованный в виде программы на языке Python, можно представить в виде следующей блок-схемы (рис. 10). Далее рассмотрим пример применения данного алгоритма.

Для примера рассматривается уязвимость CVE-2023-5972 (BDU:2023-08295). Уязвимые версии ядра операционной системы Linux согласно БДУ: от 6.2 до 6.5.8 включительно. Пример трансляции данного диапазона версий представлен на рис. 11.

Как видно из рис. 11 данные выводятся корректно в виде списка форматированных строк.

В ходе данного исследования был разработан и успешно реализован подход, автоматизирующий процесс сопоставления (трансляции) версий и наименований программного обеспечения между форматом CPE) и форматом БДУ. Применение данного подхода позволило расширить базу точных сопоставлений до 3586 уникальных комбинаций, включающих типы ПО, наименования производителей и программных продуктов. Это представляет собой значительный прогресс по сравнению с предыдущей локальной базой, используемой в деятельности БДУ ФСТЭК России содержавшей лишь 500 подобных сопоставлений.

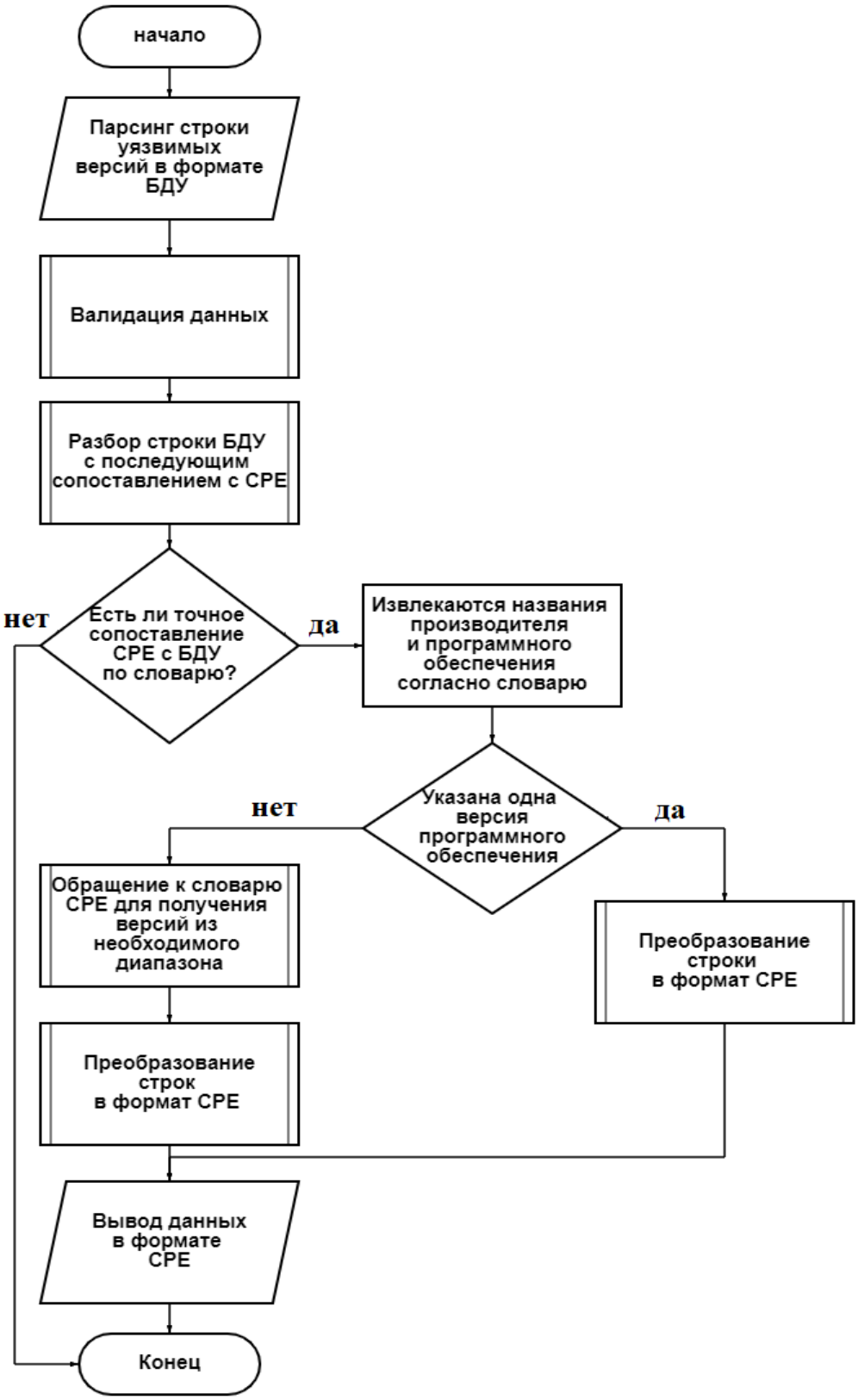

Рис. 10. Алгоритм трансляции версии и названия ПО из формата БДУ в CPE

```
Входные данные: Операционная система,
Сообщество свободного программного обеспечения,
Linux, от 6.2 до 6.5.8 включительно, Не указана
Количество найденных версий: 61
cpe:2.3:o:linux:linux_kernel:6.2.11:*:*:*:*:*:*:*
cpe:2.3:o:linux:linux_kernel:6.2.16:*:*:*:*:*:*:*
cpe:2.3:o:linux:linux_kernel:6.2.12:*:*:*:*:*:*:*
cpe:2.3:o:linux:linux_kernel:6.2.9:*:*:*:*:*:*:*
cpe:2.3:o:linux:linux_kernel:6.4.13:*:*:*:*:*:*:*
cpe:2.3:o:linux:linux_kernel:6.4.9:*:*:*:*:*:*:*
cpe:2.3:o:linux:linux_kernel: 6.3.1:*:*:*:*:*:*:*
....
```

Рис. 11. Пример применения алгоритма трансляции версий и ПО из формата БДУ в формат CPE

Данный результат имеет важное прикладное значение, особенно учитывая, что БДУ ФСТЭК России активно используется различными отечественными сканерами безопасности. В условиях, когда детектирование уязвимого ПО по его названиям и версиям является ключевым элементом в процессе оперативного управления уязвимостями в контексте взаимодействия с базами данных БДУ и NVD, увеличение базы сопоставлений в 7 раз существенно расширяет возможности для эффективного управления уязвимостями, а значит и повышения защищённости отечественных информационных систем.

В ходе исследования также были подробно описаны форматы представления уязвимых версий как в формате БДУ, так и в формате CPE. Это включало детальное рассмотрение структуры данных и специфики каждого формата, что сыграло ключевую роль в разработке и реализации метода трансляции версий ПО между этими форматами.

Кроме того, код и таблица сопоставлений были опубликованы в открытом доступе [6], что способствует повышению защищенности отечественных информационных систем за счет возможности их использования и доработки другими организациями и специалистами в области информационной безопасности.

Таким образом, внедрение разработанного подхода позволяет повысить эффективность и точность идентификации уязвимостей в ПО, а также представленный подход автоматизированного сопоставления может быть использовано в качестве основы для дальнейших улучшений и расширений в области защиты информации, в частности, процедур управления уязвимостями, способствуя общей безопасности информационных систем в Российской Федерации.

Список литературы

1. Банк данных угроз безопасности информации ФСТЭК России // БДУ URL: https://bdu.fstec.ru/ (дата обращения: 29.07.2024).

2. National Vulnerability Database // NVD URL: https://nvd.nist.gov/ (дата обращения: 29.07.2024).

3. Cheikes B. A. et al. Common platform enumeration: Naming specification version 2.3. – Gaithersburg, MD, USA : US Department of Commerce, National Institute of Standards and Technology, 2011.

3. MinHash LSH // github URL: https://github.com/ekzhu/datasketch (дата обращения: 29.07.2024).

4. Wu W. et al. A review for weighted minhash algorithms //IEEE Transactions on Knowledge and Data Engineering. 2020. Т. 34. №. 6. С. 2553-2573.

5. Andoni A., Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions //Communications of the ACM. 2008. Т. 51. №. 1. С. 117-122.

6. Конвертер форматов CPE и БДУ // GitHub. URL: https://github.com/DreyDreyv/MAPPING_CPE_BDU (дата обращения: 13.08.2024).

Государственный научно-исследовательский испытательный институт
проблем технической защиты информации ФСТЭК России
State science research experimental institute of technical information protection problem
of Federal service of technical and export control

Воронежский государственный технический университет
Voronezh State Technical University

Поступила в редакцию 05.09.2024

Информация об авторах

Гончаров Андрей Андреевич –старший научный сотрудник ФАУ «Государственный научно-исследовательский испытательный институт проблем технической защиты информации Федеральной службы по техническому и экспортному контролю», г. Воронеж, Россия. e-mail: zzzsuprema@gmail.com

Тарелкин Михаил Андреевич – начальник лаборатории ФАУ «Государственный научно-исследовательский испытательный институт проблем технической защиты информации Федеральной службы по техническому и экспортному контролю», г. Воронеж, Россия, e-mail: alexanderostapenkoias@gmail.com

Нархов Дмитрий Андреевич – аспирант, Воронежский государственный технический университет, e-mail: alexanderostapenkoias@gmail.com

DEVELOPMENT AND IMPLEMENTATION OF AN APPROACH FOR AUTOMATED MAPPING OF SOFTWARE VERSIONS AND NAMES BETWEEN COMMON PLATFORM ENUMERATION AND THE RUSSIAN FSTEC INFORMATION SECURITY THREAT DATABASE FORMATS

A.A. Goncharov, M.A. Tarelkin, D.A. Narhov

This article presents a developed and software-implemented approach that enables the automation of mapping (translation) of software versions and names between the Common Platform Enumeration format and the format used by the Information Security Threat Database of the Russian Federal Service for Technical and Export Control (FSTEC). The structures and formats of the attributes of these databases are considered, and their comparative analysis is carried out. An algorithm for matching software names has been developed. A scheme for comparing software versions has been developed. Software implementation of translation of software versions and names. The relevance of this research is driven by the need to enhance the security of domestic information systems by enriching the data in the FSTEC Information Security Threat Database through expanding the software nomenclature detected by domestic vulnerability scanners.

Keywords: vulnerabilities, software nomenclature, vulnerability scanners, Common Platform Enumeration, Information Security Threat Database

Submitted 05.09.2024

Information about the authors

Andrey A. Goncharov – Senior Research Scientist at the State Scientific Research and Testing Institute of Technical Information Protection Issues of the Federal Service for Technical and Export Control (FSTEC), Voronezh, Russia. e-mail: zzzsuprema@gmail.com

Mikhail A. Tarelkin – head of laboratory at the State Scientific Research and Testing Institute of Technical Information Protection Issues of the Federal Service for Technical and Export Control (FSTEC), Voronezh, Russia, e-mail: alexanderostapenkoias@gmail.com

Dmitry A. Narhov – graduate student, Voronezh State Technical University, e-mail: alexanderostapenkoias@gmail.com