

Часть 1

Выбор CVE

1. Для рассмотрения беру CVE-2024-3094 (CVE от 2024 года под номером 3094)

2. Описание уязвимости из базы NVD:

Вредоносный код был обнаружен в tarballs xz, начиная с версии 5.6.0. С помощью серии сложных обfuscаций в процессе сборки liblzma извлекается уже собранный объектный файл из замаскированного тестового файла, существующего в исходном коде, который затем используется для изменения определенных функций в коде liblzma. Это приводит к изменению библиотеки liblzma, которые могут быть использованы любым программным обеспечением, связанным с этой библиотекой, перехватывая и изменяя взаимодействие данных с этой библиотекой.

Это позволяет выполнить произвольный код при использовании уязвимой версии в таких сервисах, как sshd (через зависимость liblzma).

The screenshot shows the NIST National Vulnerability Database (NVD) interface. At the top, there's a navigation bar with various icons and a search bar containing the URL 'nvd.nist.gov/vuln/detail/CVE-2024-3094'. Below the header, the NIST logo and 'NATIONAL VULNERABILITY DATABASE' text are visible. A green 'VULNERABILITIES' button is highlighted. The main content area displays the details for 'CVE-2024-3094 Detail'. It includes sections for 'MODIFIED' (with a note about enrichment), 'Description' (describing the vulnerability in liblzma), 'QUICK INFO' (with fields for CVE Dictionary Entry, NVD Published Date, NVD Last Modified, and Source), 'Metrics' (CVSS Version 4.0, CVSS Version 3.x, CVSS Version 2.0), and 'References to Advisories, Solutions, and Tools'. The bottom of the screen shows a Windows taskbar with various pinned icons and the date/time '16:29 10.04.2025'.

Вот, например, описание этой уязвимости на OpenNET (<https://www.opennet.ru/opennews/art.shtml?num=60877>):

opennet.ru/opennews/art.shtml?num=60877

Профиль: Аноним (вход | регистрация)

OpenNET

www.opennet.ru

НОВОСТИ (+) КОНТЕНТ ВІКІ МАН'Я ФОРУМ Поиск (теги)

В бібліотеке xz/liblzma виявлен бэкдор, організуючий вход через sshd

29.03.2024 22:18

В пакеті XZ Utils, включаючим бібліотеку liblzma і утиліти для роботи з сжатими даними в форматі "xz", виявлен бэкдор (CVE-2024-3094), позволяючий перехоплювати і модифікувати дані, оброблюваними приложениями, пов'язаними з бібліотекою liblzma. Основною цією бэкдора є сервер OpenSSH, в якому дистрибутивах використовується бібліотека libsystemd, яка, в свою очірдь, використовує liblzma. Сполучення з SSH-сервером без аутентифікації.

Бэкдор присутствував в офіційних випусках xz 5.6.0 і 5.6.1, опублікованих 24 лютого та 9 березня, які успішили попасті в склад некількох дистрибутивів та репозиторіїв, например, Gentoo, Arch Linux, Debian sid/unstable, Fedora Rawhide і 40-beta, openSUSE factory та tumbleweed, LibreELEC, Alpine edge, Solus, CRUX, Cygwin, MSYS2 mingw, HP-UX, Homebrew, KaOS, NixOS unstable, OpenIndiana, Parabola, PCLinuxOS, OpenMandriva cooker і rolling, pkgsr current, Slackware current, Manjaro, Void Linux. Всім користувачам випусків xz 5.6.0 і 5.6.1 рекомендується срочно откатитися до версії 5.4.6, якій вже відсутні проблеми з бекдором.

Із стягнувачами проблему фактором можна відзначити те, що версія liblzma з бэкдором не успішила вйті в склад стабільних випусків великих дистрибутивів, натомість openSUSE Tumbleweed, Arch Linux та Gentoo використовували уязвиму версію xz, яка не підтримується. Для Fedora 40-beta був опублікований пакет з xz 5.6.0, який відрізняється від пакета з xz 5.6.1 тим, що він не містить проблемного коду. Бэкдор застосовується тільки системі x86_64 на базі ядра Linux та Системах з бібліотекою Glibc, в яких sshd сполучується з libsystemd для підтримки механізму sd_notify.

Код активування бэкдора був спрятаний в m4-макросах з файла build-to-host.m4, використовуваних інструментарієм automake при зборці. При зборці в ході виконання запущених обфускованих операцій на основі архівів bad-3-corrupt_lzma2.xz та good-large_compressed.lzma, додавлених під підозру в проблеми з valgrind, бекдор був виключений. Для Fedora 40-beta бекдор не був виключений, оскільки він був виключений в ході зборки відповідної версії xz 5.6.1.

Активуючі бэкдор m4-макроси входили в склад tar-архівів релізів, які не використовувалися в Git-репозиторіях (багато додавлені в .gitignore). При цьому вредоносні тестові архіви присутні в репозиторіях, таємничий бэкдор має доступ до репозиторію, так і до процесів формування релізів. Боліше того, ізначально файли bad-3-corrupt_lzma2.xz та good-large_compressed.lzma були заряджені 23 лютого перед релізом xz 5.6.0, а 9 березня обновлені перед релізом xz 5.6.1.

При використанні liblzma в приложениях вредоносні зміни могли використовуватися для перехвата або модифікації даних, а також для впливу на роботу sshd. В частности, вредоносний код подміняв функцію RSA_public_decrypt для обходу процесу аутентифікації в sshd. Для невідомого виклику та перехвата функції в бэкдорі використовувався механізм IFUNC, присутній в Glibc. Бэкдор включав захисту від обнаруження та не проявляв себе при використанні змінних окружень LANG та TERM (т.е. при запуску процеса в терміналі) та не використовував змінних окружень LD_DEBUG та LD_PROFILE, які активувались тільки при виконанні виконуваного файла /usr/sbin/sshd. В бэкдорі також були використані методи обнаружения запуску в різних окруженнях.

В файлі m4/build-to-host.m4 використовувалися конструкції

Пам'ятні дати
8 квітня RFC виконилось 56 років

Навігація
Фільтри
Канали: XML, RSS, Facebook, Twitter, LinkedIn, YouTube
Розділи новостей | Світ
Что нового на OpenNet
Поиск в новостях
Нові коментарі
Додати свою новість

Важное
- 03.04 Mozilla розвиває Thunderbird Pro та сервіс Thundermail в стилі Gmail та Office365 (169 +19)
- 26.03 Google переходить до розробки Android за закритими дверима з отворенім кодом після релізу (243 -31)
- 25.03 Уязвимості в ingress-nginx, позволяючі виконувати код та захватити управління кластерами Kubernetes (87 +19 б.)
- 24.03 Реліз ядра Linux 6.14 (121 +0)
- 22.03 Уязвимості в Pagure та OBS, допускавши компрометацію пакетів в репозиторіях Fedora та openSUSE (40 +11 б.)
- 21.03 Перегрузка інфраструктури KDE, GNOME, Fedora, Codeberg та SourceHut из-за ІН-індексатора (212 +33)
- 17.03 Реліз графічного редактора GIMP 3.0.0 (328 +98)
- 15.03 Компрометація GitHub Actions-обробочика changed-files, використовуваного в 23 тисячах репозиторіїв (40 +17)
- Сводка подій за 2024 рік

Пошук 16:33 10.04.2024

3. Определяю CWE:

CWE-506: Embedded Malicious Code (Встроенный вредоносный код)

4. Смотрю, к каким CWE относится данная CVE:

CWE-506: Embedded Malicious Code (Встроенный вредоносный код)

Screenshot of the NVD (National Vulnerability Database) page for CVE-2024-3094. The page lists various sources of information about the vulnerability, including third-party advisories, issue tracking, vendor advisories, press/media coverage, mailing lists, and vendor advisories from Red Hat, Inc. and other organizations.

Source	
https://security.archlinux.org/CVE-2024-3094	Third Party Advisory
https://security.netapp.com/advisory/ntap-20240402-0001/	Issue Tracking
https://tukaani.org/xz-backdoor/	Vendor Advisory
https://twitter.com/LetsDefendO/status/1774804387417751958	Third Party Advisory
https://twitter.com/debian/status/1774219194638409898	Press/Media Coverage
https://twitter.com/infosecb/status/1774595540233167206	Press/Media Coverage
https://twitter.com/infosecb/status/1774597228864139400	Press/Media Coverage
https://ubuntu.com/security/CVE-2024-3094	Third Party Advisory
https://www.cisa.gov/news-events/alerts/2024/03/29/reported-supply-chain-compromise-affecting-xz-utils-data-compression-library-cve-2024-3094	Third Party Advisory
https://www.darkreading.com/vulnerabilities-threats/are-you-affected-by-the-backdoor-in-xz-utils	US Government Resource
https://www.kali.org/blog/about-the-xz-backdoor/	Third Party Advisory
https://www.openwall.com/lists/oss-security/2024/03/29/4	Mailing List
https://www.openwall.com/lists/oss-security/2024/03/29/4	Mailing List
https://www.redhat.com/en/blog/urgent-security-alert-fedora-41-and-rawhide-users	Vendor Advisory
https://www.redhat.com/en/blog/urgent-security-alert-fedora-41-and-rawhide-users	Vendor Advisory
https://www.tenable.com/blog/frequently-asked-questions-cve-2024-3094-supply-chain-backdoor-in-xz-utils	Third Party Advisory
https://www.theregister.com/2024/03/29/malicious_backdoor_xz/	Press/Media Coverage
https://www.vicarius.io/vsociety/vulnerabilities/cve-2024-3094	Third Party Advisory
https://xeiaso.net/notes/2024/xz-vuln/	Third Party Advisory

Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-506	Embedded Malicious Code	Red Hat, Inc.

Known Affected Software Configurations

Switch to CPE 2.2

Configuration 1 (hide)

- cpe:2.3:a:tukaani:xz:5.6.0*:***:***:*
- Show Matching CPE(s)▼
- cpe:2.3:a:tukaani:xz:5.6.1*:***:***:*
- Show Matching CPE(s)▼

* Denotes Vulnerable Software

5. Краткое описание CWE, связанный с этой уязвимостью:
Продукт содержит код, который по своей природе кажется вредоносным. Намеренное внедрение вредоносного кода в легитимное ПО.

Screenshot of the CWE (Common Weakness Enumeration) page for CWE-506: Embedded Malicious Code. The page provides detailed information about the weakness, including its description, common consequences, potential mitigations, and relationships to other research concepts.

Description: The product contains code that appears to be malicious in nature.

Common Consequences: Malicious flaws have acquired colorful names, including Trojan horse, trapdoor, timebomb, and logic-bomb. A developer might insert malicious code with the intent to subvert the security of a product or its host system at some time in the future. It generally refers to a program that performs a useful service but exploits rights of the program's user in a way the user does not intend.

Potential Mitigations: Remove the malicious code and start an effort to ensure that no more malicious code exists. This may require a detailed review of all code, as it is possible to hide a serious attack in only one or two lines of code. These lines may be located almost anywhere in an application and may have been intentionally obfuscated by the attacker.

Relationships: Relevant to the view "Research Concepts" (View-1000)

Nature	Type	ID	Name
ChildOf	⊕	912	Hidden Functionality
ParentOf	⊖	507	Trojan Horse
ParentOf	⊖	510	Trapdoor
ParentOf	⊖	511	Logic/Time Bomb
ParentOf	⊖	512	Spyware

Оценка по CVSS

1. Оценка с помощью калькулятора CVSS v3

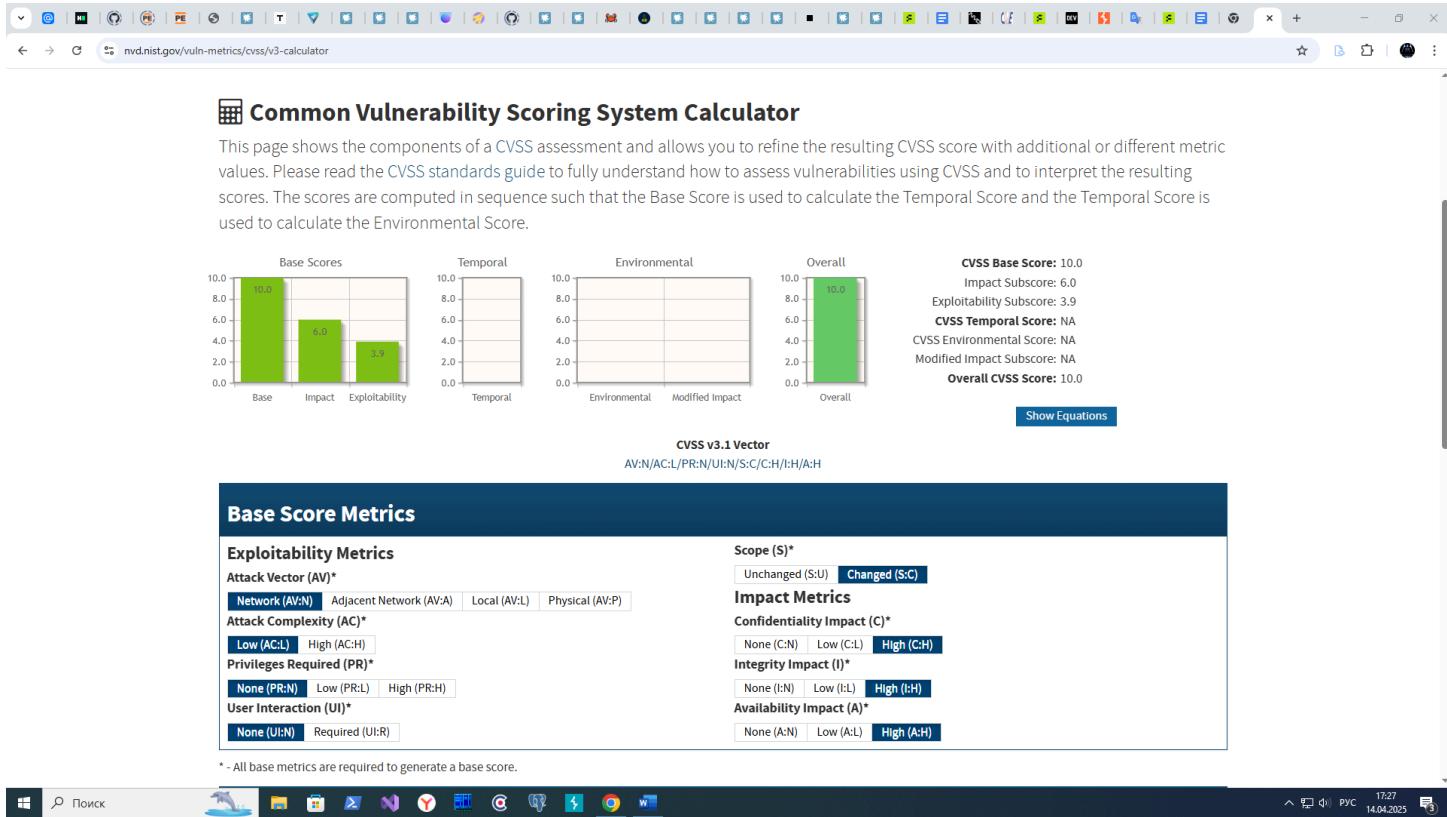
У CVE-2024-3094 (на <https://nvd.nist.gov/vuln/detail/CVE-2024-3094>) следующая оценка:

Base Score: 10.0 CRITICAL

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Мой расчет с помощью калькулятора Common Vulnerability Scoring System Calculator:

Метрика	Значение	Пояснение
Attack Vector (AV)	Network (AV:N)	Уязвимость может эксплуатироваться удаленно - через sshd (сетевой сервис).
Attack Complexity (AC)	Low (AC:L)	Для эксплуатации не требуется сложных условий — достаточно использовать liblzma с внедренным вредоносным кодом.
Privileges Required (PR)	None (PR:N)	Эксплуатация не требует никаких привилегий.
User Interaction (UI)	None (UI:N)	Эксплуатация не требует никаких действий пользователя.
Scope (S)	Changes (S:C)	Для эксплуатации уязвимости требуется наличие liblzma с внедренным вредоносным кодом.
Confidentiality Impact (C)	High (C:H)	При эксплуатации уязвимости злоумышленник получает доступ к любым данным в системе
Integrity Impact (I)	High (I:H)	При эксплуатации уязвимости злоумышленник получает возможность модифицировать любые данные в системе
Availability Impact (A)	High (A:H)	При эксплуатации уязвимости злоумышленник получает возможность выполнения любых действий в системе



2. Разбор метрик CVSS:3.1:

- Attack Vector = Network: уязвимость может эксплуатироваться удаленно, через сетевой сервис - например, через sshd;
- Attack Complexity = Low: для эксплуатации не требуется сложных условий — достаточно использовать liblzma с внедренным вредоносным кодом;
- Privileges Required = None: для эксплуатации уязвимости не требуются никакие привилегии;
- User Interaction = None: для эксплуатации уязвимости не требуется никаких действий пользователя;
- Scope = Changes: для эксплуатации уязвимости требуется наличие liblzma с внедренным вредоносным кодом;
- Confidentiality Impact = High: при эксплуатации уязвимости злоумышленник получает доступ к любым данным в системе;
- Integrity Impact = High: при эксплуатации уязвимости злоумышленник получает возможность модифицировать любые данные в системе;
- Availability Impact = High: при эксплуатации уязвимости злоумышленник получает возможность выполнения любых действий в системе (RCE).

3. Итоговый балл и уровень риска:

10.0 CRITICAL

Причины:

- Возможность атаки через сеть;
- Простота использования уязвимости;
- Отсутствие дополнительных требований (к привилегиям, к взаимодействию с пользователем);
- Максимальное воздействие на всю систему (конфиденциальность, целостность, доступность).

Описание последствий:

Угроза	Описание	Последствия
RCE	Злоумышленник может удаленно выполнить любые команды на уязвимом сервере через liblzma с внедренным вредоносным кодом - например, через sshd сервис	Полный доступ к уязвимому серверу (root-права), возможность выполнения произвольных команд (произвольных действий)
Нарушение конфиденциальности	Злоумышленник может читать любые файлы на сервере	Могут быть украдены пароли, SSH-ключи, токены API, базы данных, платежные реквизиты, персональные данные, конфиги облачных сервисов
Нарушение целостности	Злоумышленник может изменять любые файлы на сервере	Могут быть изменены данные в базах данных, содержимое произвольных файлов, а также подписи для этих данных (при наличии соответствующих ключей)
Нарушение доступности	Злоумышленник может изменять доступность любых сервисов на сервере	Доступность любых сервисов на сервере может быть изменена из-за их включения/отключения, изменений правил firewall и т.д.
Распространение атаки на другие узлы в локальной сети	Злоумышленник может использовать скомпрометированный (захваченный) узел, как плацдарм для атаки на другие узлы в сети	Частичная или полная компрометация инфраструктуры локальной сети. Частичная или полная компрометации криптографических ключей
Использование устройства в качестве узла ботнета	Злоумышленник может установить дополнительно скрытое ПО для дальнейшего использования скомпрометированного (захваченного) узла в своих целях	Использование скомпрометированного (захваченного) узла, в качестве узла ботнета или для майнинга криптовалют

Вывод:

- Согласно блогу Лаборатории Касперского, проблемными являются XZ Utils версий 5.6.0 и 5.6.1 (<https://www.kaspersky.ru/blog/cve-2024-3094-vulnerability-backdoor/37222/>). Поэтому самым простым и логичным вариантом является использование более новых версий XZ Utils (сейчас это версия 5.8.1). В крайнем случае, если нет возможности по каким-либо причинам использовать более новую версию XZ Utils, можно использовать XZ Utils версии до 5.6.0 (хотя там тоже есть проблемы).
- Если игнорировать наличие CVE-2024-3094 в XZ Utils, то легко можно прийти к ситуации, когда сам узел и данные на нем полностью будут под контролем злоумышленника

Часть 2

Безопасная разработка веб-приложений (общие принципы):

Действие	Описание
Создание архитектуры с учетом требований DevSecOp	<p>Необходимо изначально создавать архитектуру приложения исходя из наличия системы безопасности и требований DevSecOp (даже на стадии MVP). Иначе в противном случае можно получить ситуацию, когда система безопасности работает некорректно или неэффективно в каких-то случаях.</p> <p>https://owasp.org/Top10/A04_2021-Insecure_Design/</p>
Использование стойких средств для аутентификации и авторизации	<p>Необходимо использовать правильные (стойкие) средства для аутентификации и авторизации. Это относится к целому ряду подходов:</p> <ul style="list-style-type: none">• управление учетными записями• управление паролями (сложность, время жизни, повторное использование)• хранение паролей (отказ от хранения паролей в открытом виде)• 2FA• использование защищенных токенов (например, JWT токенов с подписью)• ограничение времени жизни пользовательской сессии• использование oauth для доступа к нескольким ресурсам в рамках одной сессии• и т.д. <p>Так, например, явно не стоит использовать basic access authentication в качестве подхода к аутентификации.</p> <p>https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/</p>
Внедрение ролевой модели. Использование принципа минимальных привилегий	<p>Необходимо использовать ролевую модель в приложении: описать роли и необходимые им права (с учетом доменной модели). При этом права для каждой роли должны быть минимально возможны. Необходимо запретить выдачу прав напрямую учетным записям - права у учетных записей должны появляться за счет членства в той или иной роли.</p> <p>PS. Если ролевой модели будет недостаточно по каким-либо причинам, то следует использовать разграничение доступа на основе атрибутов.</p> <p>https://owasp.org/Top10/A01_2021-Broken_Access_Control/</p>
Разделение API (на backend) с учетом ролевой модели	<p>Необходимо методы API (на backend) сгруппировать вместе в зависимости от роли, необходимой для их выполнения (на уровне путей в url). Реализация методов API (на backend), относящихся к разным ролям может быть выполнена в разных микросервисах для физического разделения таких методов API.</p> <p>https://owasp.org/Top10/A01_2021-Broken_Access_Control/</p>
Использование правильных средств работы с БД	<p>Правильный подход означает, что мы должны использовать такие средства (такие библиотеки), которые исключают возможность SQL инъекций. Так, например, вместо формирования запроса вручную</p>

	на основе пользовательского ввода, необходимо использовать параметризованные запросы https://owasp.org/Top10/A03_2021-Injection/
Использование правильного подхода при работе с пользовательским вводом	При обработке пользовательского ввода следует относится к нему, как если он сформирован злоумышленником: пользовательский ввод всегда должен проверяться и обрабатываться так, чтобы исключить возможность какого-либо несанкционированного воздействия - например, SQL инъекции, XSS инъекции, инъекции команд и т.д. В качестве примера такой обработки можно привести использование параметризованных запросов для работы с БД, а не ручное формирование запросов на основе пользовательского ввода. https://owasp.org/Top10/A03_2021-Injection/
Использование логирования	Необходимо использовать логирование для всех ошибок и внештатных ситуаций, происходящих в приложении. Также необходимо логировать все действия, которые выполняются с повышенными привилегиями. https://owasp.org/Top10/A09_2021-SecurityLogging_and_Monitoring_Failures/
Создание и использование автотестов	Автотесты - это очень мощный подход для автоматической проверки работы той или иной функциональности, в том числе и для выявления регрессий. Очевидно, что автотесты должны быть созданы не только на основную функциональность, но и на аспекты, связанные с безопасностью. Очень важно, чтобы тесты создавались (когда это возможно) на все выявленные проблемы с безопасностью, чтобы в будущем не столкнуться с регрессией этих проблем.
Использование CI/CD	Необходимо использовать CI/CD для автоматизации сборки, выполнения автотестов, а также тестирования SAST, DAST, SCA на регулярной основе
Использование https протокола	На самом деле это относится больше к конфигурированию веб приложения (т.к. никак не отражается на подходах к разработке), но тем не менее - обязательно использовать протокол https (TLS 1.3) для защиты передаваемой информации.
Использование SAST для тестирования исходного кода	Необходимо использовать SAST для тестирования исходного кода с помощью некоторого набора заданных правил/метрик, в том числе и связанных с безопасностью. Необходимо выполнять это тестирование на регулярной основе (если добавить этот шаг, как шаг в CI/CD, то данное тестирование будет происходить автоматически).
Использование DAST для тестирования готового приложения	Необходимо использовать DAST для тестирования создаваемого приложения с помощью некоторого набора заданных правил/метрик, в том числе и связанных с безопасностью. Необходимо выполнять это тестирование на регулярной основе (если добавить этот шаг, как шаг в CI/CD, то данное тестирование будет происходить автоматически).
Использование SCA для	Необходимо использовать SCA для анализа (проверки)

анализа зависимостей	используемых зависимостей. Необходимо выполнять это тестирование на регулярной основе (если добавить этот шаг, как шаг в CI/CD, то данное тестирование будет происходить автоматически).
Использование открытых источников (например, NVD) для поиска информации о новых уязвимостях	Необходимо на регулярной основе делать поиск по открытым источникам (например, NVD) информации о новых обнаруженных CVE; после чего, необходимо проанализировать, возможны ли эти CVE в создаваемом веб-приложении. Если да, необходимо их исправлять согласно рекомендациям.