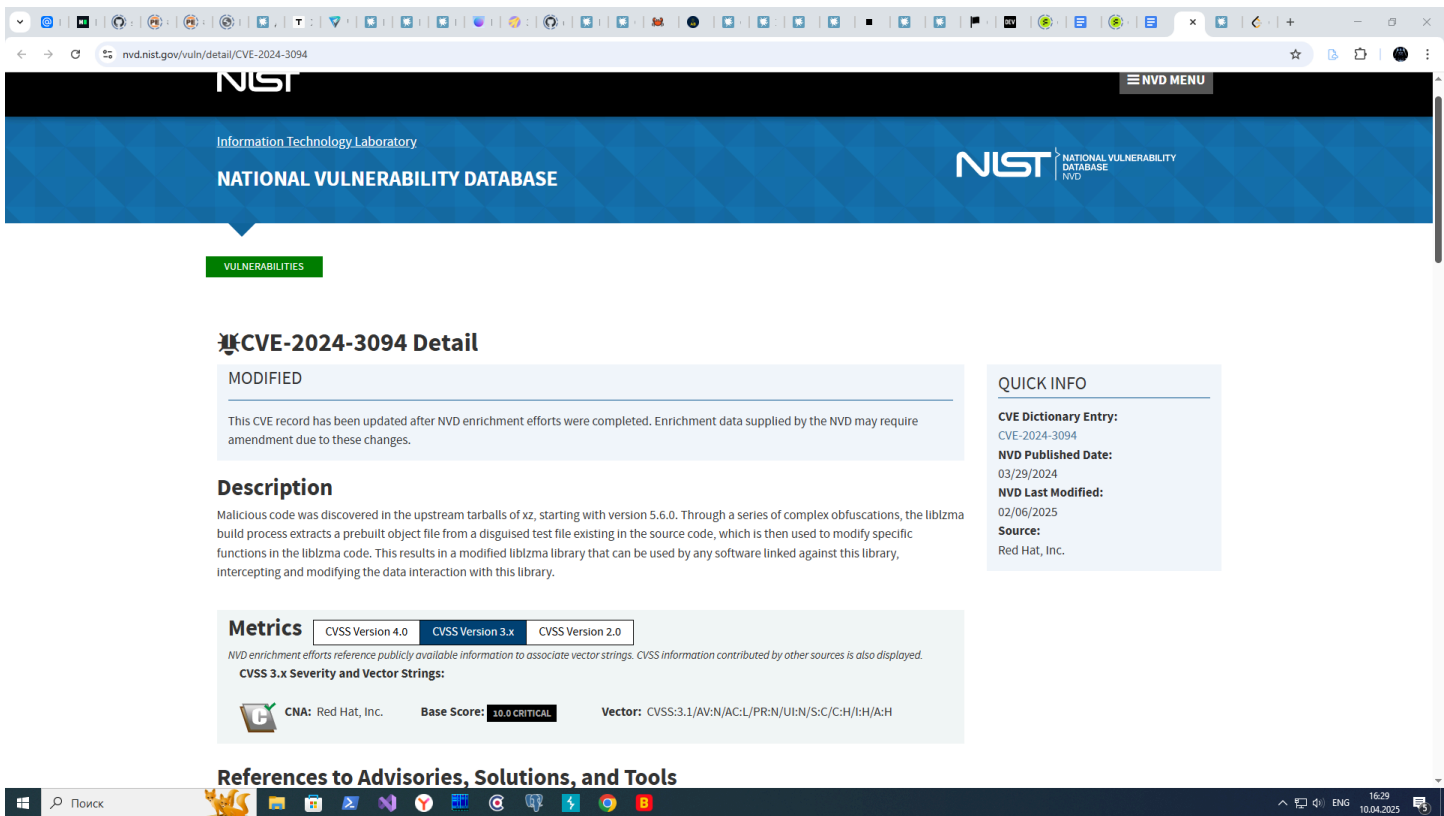


Часть 1

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 обфускаций 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).



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

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

OpenNET

новости (+) контент wiki ман'ы форум Поиск (теги)

Профиль: [Аноним](#) (выход | регистрация)

В библиотеке xz/liblzma выявлен бэкдор, организующий вход через sshd

29.03.2024 22:18

В пакете **XZ Utils**, включающем библиотеку liblzma и утилиты для работы со сжатыми данными в формате "xz", **выявлен бэкдор (CVE-2024-3094)**, позволяющий перехватывать и модифицировать данные, обрабатываемые приложениями, связанными с библиотекой liblzma. Основной целью бэкдора является сервер OpenSSH, в некоторых дистрибутивах связанный с библиотекой libsystemd, которая, в свою очередь, использует liblzma. Связывание sshd с уязвимой библиотекой позволяет злоумышленникам получить доступ к SSH-серверу без аутентификации.

Бэкдор присутствовал в официальных выпусках xz 5.6.0 и 5.6.1, опубликованных 24 февраля и 9 марта, которые успели попасть в состав некоторых дистрибутивов и репозиториях, например, **Gentoo, Arch Linux, Debian sid/unstable, Fedora Rawhide и 40-beta, openSUSE factory и tumbleweed, LibreELEC, Alpine edge, Solus, CRUX, Cygwin, MSYS2 mingw, HP-UX, Homebrew, KaOS, NixOS unstable, OpenIndiana, Parabola, PCLinuxOS, OpenMandriva cookier и rolling, pkgsrc current, Slackware current, Manjaro, Void Linux.** Всем пользователям выпусков xz 5.6.0 и 5.6.1 рекомендуется срочно откатиться как минимум на версию **5.4.6**, но желательно на **5.4.1**, как последний выпуск, опубликованный прошлым сопровождающим.

Из спланировавших проблему факторов можно отметить то, что версия liblzma с бэкдором не успела войти в состав стабильных выпусков крупных дистрибутивов, но затронула **openSUSE Tumbleweed, Arch Linux и Gentoo** использовали уязвимую версию xz, но не подвержены атаке, так как не применяют к OpenSSH патч для поддержки systemd-notify, приводящий к связыванию sshd к liblzma. Для Fedora 40-beta был опубликован пакет с xz 5.6.0, но из-за проблем с valgrind он был **собран** без поддержки IFUNC ("configure --disable-ifuncs"), т.е. бэкдор в нем был неработоспособен. Ubuntu и RHEL/CentOS не импортировали проблемные версии в свои репозитории. Бэкдор затрагивает только системы x86_64 на базе ядра Linux и Си-библиотеки Glibc, в которых sshd **связывается** с libsystemd для поддержки механизма sd_notify.

Код активации бэкдора был спрятан в m4-макросах из файла build-to-host.m4, используемого инструментарием automake при сборке. При сборке в ходе выполнения запутанных обфусцированных операций на основе архивов bad-3-corrupt_lzma2.xz и good-large_compressed.lzma, добавленных под предлогом использования в тестах корректности работы распаковщика, формировался объектный файл с вредоносным кодом, который включался в состав библиотеки liblzma и изменял логику работы некоторых ее функций.

Активирующие бэкдор m4-макросы входили в состав tar-архивов релизов, но отсутствовали в Git-репозитории (были **добавлены** в .gitignore). При этом вредоносные тестовые архивы присутствовали в репозитории, т.е. внедривший бэкдор имел доступ как к репозиторию, так и к процессам формирования релизов. Более того, изначально файлы bad-3-corrupt_lzma2.xz и good-large_compressed.lzma были **загружены** 23 февраля перед релизом xz 5.6.0, а 9 марта **обновлены** перед релизом xz 5.6.1.

При использовании liblzma в приложениях вредоносные изменения могли использоваться для перехвата или модификации данных, а также для влияния на работу sshd. В частности, вредоносный код подменял функцию RSA_public_decrypt для обхода процесса аутентификации в sshd. Для неявного вызова и перехвата функций в бэкдоре применялся механизм **IFUNC**, присутствующий в Glibc. Бэкдор включал защиту от обнаружения и не проявлял себя при выставленных переменных окружения LANG и TERM (т.е. при запуске процесса в терминале) и не выставленных переменных окружения LD_DEBUG и LD_PROFILE, а активировался только при выполнении исполняемого файла /usr/sbin/sshd. В бэкдоре также имелись средства обнаружения запуска в отладочных окружениях.

В файле m4/build-to-host.m4 использовались конструкции

Памятные даты

8 апреля RFC исполнилось 56 лет

Навигация

Фильтры

Каналы: [XMI](#) [u](#) [f](#) [t](#) [b](#)

Разделы новостей | Сводные

Что нового на OpenNet

Поиск в новостях

Новые комментарии

Добавить свою новость

Важное

- 03.04 Mozilla развивает Thunderbird Pro и сервис Thundermail в стиле Gmail и Office365 (169 +19)
- 26.03 Google переходит к разработке Android за закрытыми дверями с открытием кода после релизов (245 +31)
- 25.03 Уязвимости в ingress-nginx, позволяющие выполнить код и захватить управление кластером Kubernetes (87 +19 L)
- 24.03 Релиз ядра Linux 6.14 (121 +40)
- 22.03 Уязвимости в Pajure и OBS, допускавшие компрометацию пакетов в репозиториях Fedora и openSUSE (40 +11 L)
- 21.03 Перепрошивка инфраструктуры KDE, GNOME, Fedora, Codeberg и SourceHut из-за ИИ-индикаторов (212 +35)
- 17.03 Релиз графического редактора GIMP 3.0.0 (328 +98)
- 15.03 Компрометация GitHub Actions-обработчика changed-files, примененного в 23 тысячах репозиториях (40 +17)
- Сводка событий за 2024 год

3. Определяю CVE:

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

4. Смотрю, к каким CVE относится данная CVE:

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

nvdc.nist.gov/vuln/detail/CVE-2024-3094

https://security.archlinux.org/CVE-2024-3094	Third Party Advisory
https://secunty.netapp.com/advisory/ntap-20240402-0001/	
https://tukaani.org/xz-backdoor/	Issue Tracking Vendor Advisory
https://twitter.com/LetsDefendIO/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 US Government Resource
https://www.darkreading.com/vulnerabilities-threats/are-you-affected-by-the-backdoor-in-xz-utils	Third Party Advisory
https://www.kali.org/blog/about-the-xz-backdoor/	
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	
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.

cwe.mitre.org/data/definitions/506.html

CWE Common Weakness Enumeration

A community-developed list of SW & HW weaknesses that can become vulnerabilities

Home > CWE List > CWE-506: Embedded Malicious Code (4.17)

Home About Learn Access Content Community Search

CWE-506: Embedded Malicious Code

Weakness ID: 506

Vulnerability Mapping: ALLOWED (with careful review of mapping notes)

Abstraction: Class

View customized information: Conceptual Operational Mapping Friendly Complete Custom

Description

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

Extended Description

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.

Common Consequences

Impact	Details
Execute Unauthorized Code or Commands	Scope: Confidentiality, Integrity, Availability

Potential Mitigations

Phase(s)	Mitigation
Testing	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

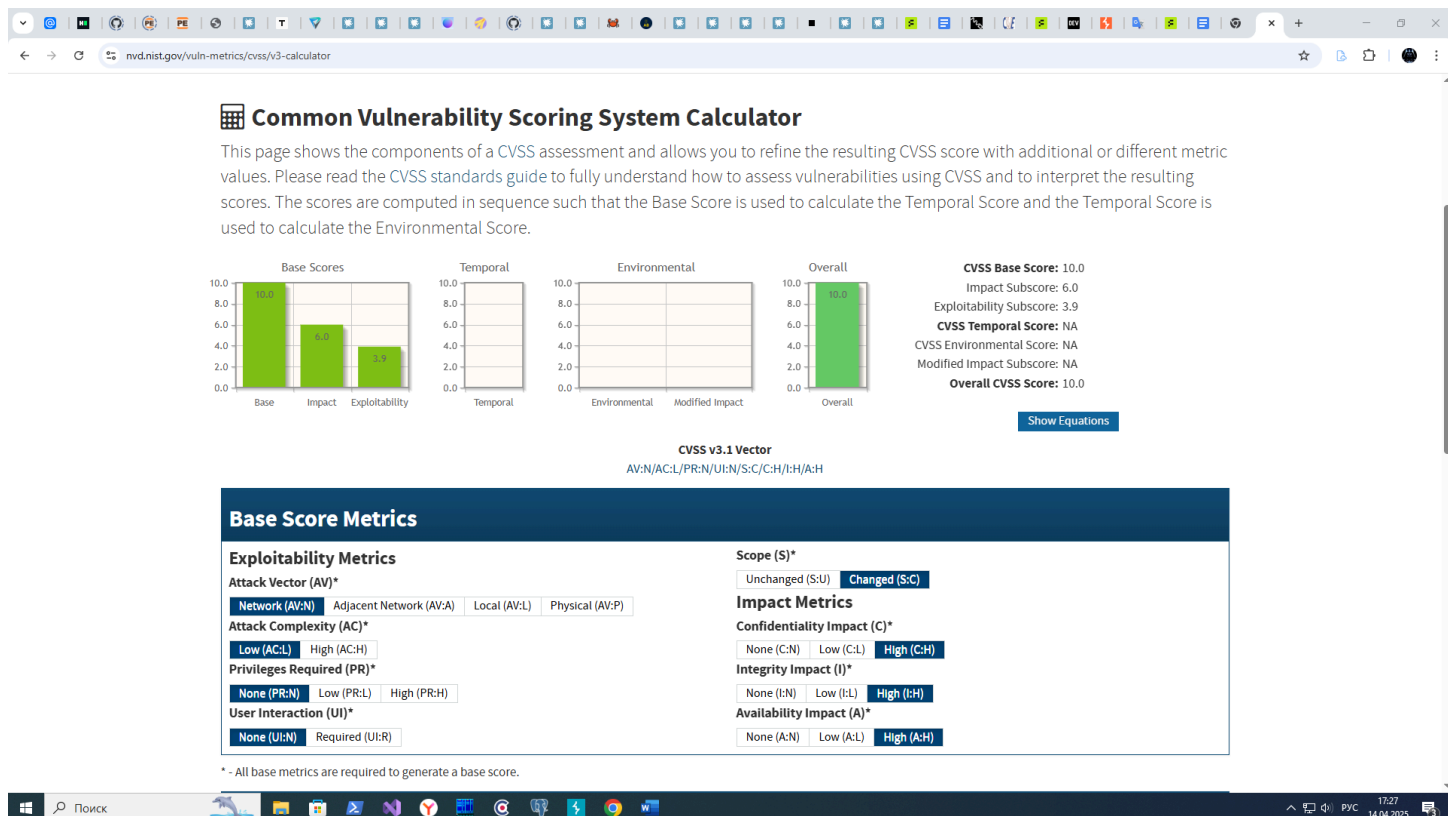
Relevant to the view "Research Concepts" (View-1000)

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 liblzm 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:

1. 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).
2. 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	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. https://owasp.org/Top10/A04_2021-Insecure_Design/
Использование стойких средств для аутентификации и авторизации	Необходимо использовать правильные (стойкие) средства для аутентификации и авторизации. Это относится к целому ряду подходов: <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. Так, например, явно не стоит использовать basic access authentication в качестве подхода к аутентификации. https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
Внедрение ролевой модели. Использование принципа минимальных привилегий	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. PS. Если ролевой модели будет недостаточно по каким-либо причинам, то следует использовать разграничение доступа на основе атрибутов. https://owasp.org/Top10/A01_2021-Broken_Access_Control/
Разделение API (на backend) с учетом ролевой модели	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. https://owasp.org/Top10/A01_2021-Broken_Access_Control/
Использование правильных средств работы с БД	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</p> <p>https://owasp.org/Top10/A03_2021-Injection/</p>
Использование правильного подхода при работе с пользовательским вводом	<p>При обработке пользовательского ввода следует относиться 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.</p> <p>https://owasp.org/Top10/A03_2021-Injection/</p>
Использование логирования	<p>Необходимо использовать логирование для всех ошибок и внештатных ситуаций, происходящих в приложении. Также необходимо логировать все действия, которые выполняются с повышенными привилегиями.</p> <p>https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/</p>
Создание и использование автотестов	<p>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.</p>
Использование CI/CD	<p>Необходимо использовать CI/CD для автоматизации сборки, выполнения автотестов, а также тестирования SAST, DAST, SCA на регулярной основе</p>
Использование https протокола	<p>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.</p>
Использование SAST для тестирования исходного кода	<p>Необходимо использовать SAST для тестирования исходного кода с помощью некоторого набор 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).</p>
Использование DAST для тестирования готового приложения	<p>Необходимо использовать DAST для тестирования создаваемого приложения с помощью некоторого набор 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).</p>
Использование SCA для	<p>Необходимо использовать SCA для анализа (проверки)</p>

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.