0.1. Оситов Е.А. Устранение уязвимостей в библиотеках Java

В мире современной IT-разработки активно используются сторонние библиотеки как помощь в решении поставленных задач. Такие библиотеки позволяют применять оптимально реализованные подходы, не изобретая собственных решений для второстепенных задач. Вместе с интегрированными библиотеками в исходный код попадают уязвимые компоненты, которые повышают риск взлома системы или отдельных ее частей. Поиск и устранение этих уязвимостей является важной частью разработки коммерческих продуктов.

В ходе исследования поставленной задачи были выявлены следующие способы поиска уязвимостей в сторонних библиотеках:

1. CVE — наиболее обширная бесплатная база данных об уязвимостях, хранит информацию для конкретных версий ПО [1];
2. NVD — национальная база данных уязвимостей США, основана на CVE, содержит информацию про CVSS (открытый стандарт, разработанный для определения уровня опасности обнаруженной уязвимости по десятибалльной шкале) [2];
3. OSS Index — это бесплатный каталог компонентов с открытым исходным кодом и инструментов сканирования, основана на информации об уязвимостях из открытых источников [3];
4. VulnDB — платная база данных, которая содержит более полную информацию по уязвимостям, чем CVE;
5. Анализ исходного кода сторонних библиотек статическими анализаторами — SAST (Static Application Security Testing) — тестирование «белого ящика», позволяет разработчикам находить уязвимости безопасности в исходном коде приложения.

Стоит отметить, что вышеперечисленные способы позволяют найти уязвимость в отдельно взятых библиотеках. Для анализа всех зависимостей большого проекта можно использовать такие инструменты как OWASP Dependency-Check [4], Black Duck, JFrog Xray и другие.

Методы устранения уязвимостей достаточно известны, но при этом разрознены. Автором был разработан алгоритм, который полностью описывает возможные способы устранения уязвимостей, включающий в себя примеры авторского кода. В общем виде описать алгоритм устранения можно следующим образом:

1. Изменение версии — повышение или понижение версии используемой библиотеки;
2. Изменение исходного кода библиотеки с целью использования собственной версии, не содержащей уязвимостей;
3. Обоснование неэксплуатируемости уязвимости:

(a) Ошибки инструмента: артефакт определен некорректно, уязвимость отсутствует в данной версии библиотеки;

(b) Сценарии, при которых код стороннего продукта может быть уязвим, не применяются нашим кодом, а так же другими используемыми библиотеками;

4. Изменение библиотеки в процессе сборки — удаление классов/директорий/shaded зависимостей из стороннего компонента, а так же замена частей библиотек собственными классами или замена транзитивных shaded зависимостей на те версии, которые не содержат уязвимостей;

Таким образом, в процессе исследования поставленной задачи было выявлено отсутствие полноценного описания доступных решений проблемы поиска и устранения уязвимостей в сторонних библиотеках Java, что повышает актуальность дальнейших разработок.

Список литературы

- [1] SAYAR I., ET AL. An in-depth study of java deserialization remote-code execution exploits and vulnerabilities // ACM Transactions on Software Engineering and Methodology. 2023. Vol. 32. N. 1. P. 1–45.
- [2] VIETTEL F. ET AL. Are third-party libraries secure? a software library checker for java // Risks and Security of Internet and Systems: 13th International Conference. Location: Springer International Publishing, 2019. P. 18–34.
- [3] DANN A., ET AL. Identifying challenges for oss vulnerability scanners—a study and test suite // IEEE Transactions on Software Engineering. 2021. Vol. 48. N. 9. P. 3613–3625.
- [4] PONTA S., PLATE H., SABETTA A. Detection, assessment and mitigation of vulnerabilities in open source dependencies // Empirical Software Engineering. 2020. Vol. 25. N. 5. P. 3175–3215.