УДК 004.056.5

Зубелевич Д.І., м.н.с.
Вандалович В.П., наук. співробітник
Житомирський військовий інститут імені С.П. Корольова

МЕТОДИ ВИЯВЛЕННЯ ТА ПРОТИДІЇ АТАКАМ НА ЛАНЦЮЖКИ ПОСТАЧАННЯ У ВІДКРИТОМУ ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ

Сучасні підходи до розробки програмного забезпечення (далі - ПЗ) значною мірою базуються на використанні компонентів з відкритим вихідним кодом та вільними ліцензіями. Частка сторонніх компонентів у сучасних програмних продуктах може сягати до 90 % загального обсягу коду.

Інтеграція сторонніх бібліотек та фреймворків дозволяє суттєво прискорити розробку програм, проте водночас створює додаткові ризики для безпеки інформаційних систем - компрометація навіть одного використаного елемента відкритого ПЗ створює загрозу для великої кількості інших проєктів, в яких цей елемент використано як залежність.

Атаки на ланцюжки постачання ПЗ (англ. Software Supply Chain Attacks) упродовж останніх років стали поширеним видом кіберзагроз – відомими є випадки компрометації програмних продуктів та бібліотек, що використовуються великою кількістю користувачів. Одним із найбільш відомих прикладів є атака на ПЗ Orion від компанії SolarWinds у 2020 році, коли зловмисники (котрих розслідувачі пов'язують з рф) інтегрували шкідливий код у легітимні оновлення системи моніторингу мереж. У результаті заражені оновлення завантажили близько 18 000 організацій, включно з державними установами. Іншим показовим прикладом є інцидент із бекдором у бібліотеці XZ Utils версій 5.6.0 та 5.6.1 у 2024 році (CVE-2024-3094), який призвів до можливості несанкціонованого доступу до великої кількості Linux-систем по протоколу SSH в обхід механізму аутентифікації.

Зазначені інциденти демонструють недостатність традиційних підходів до гарантування кібербезпеки (аналіз відомих вразливостей та сигнатурне виявлення шкідливого коду) для виявлення скомпрометованих ланцюжків постачання. Одним із причин є складність сучасних програмних екосистем, у яких широко використовуються транзитивні залежності. У багатьох випадках ПЗ може містити десятки або сотні бібліотек, кожна з яких має власні залежності, котрі часто оновлюються. Це значно ускладнює контроль походження та безпечності усіх компонентів системи.

Для підвищення рівня безпеки програмних ланцюжків постачання дедалі ширше застосовуються автоматизовані засоби аналізу складу ПЗ (англ. Software Composition Analysis, SCA). Одним із важливих інструментів у цій сфері є використання Software Bill of Materials (SBOM) - структурованого опису всіх компонентів та залежностей в ПЗ. SBOM дозволяє ідентифікувати використання вразливих бібліотек та оцінювати потенційний вплив нових вразливостей.

Подальшим розвитком цього підходу є використання механізму VEX (англ. Vulnerability Exploitability eXchange), який дозволяє уточнювати, чи може певна вразливість бути фактично експлуатована в конкретному програмному продукті. Це дає змогу зменшити кількість хибних спрацювань під час автоматизованого аналізу безпеки та підвищити ефективність управління вразливостями. Важливу роль також відіграють підходи до забезпечення цілісності програмних артефактів на всіх етапах життєвого циклу розробки. З цією метою активно впроваджуються стандарти SLSA (англ. Supply-chain Levels for Software Artifacts), які визначають рівні довіри до залежностей, використаних в процесі створення ПЗ. Практична реалізація цих підходів передбачає використання криптографічних механізмів підпису програмних компонентів, зокрема інфраструктури Sigstore, для перевірки їх походження та незмінності.

Перспективним напрямом підвищення безпеки також є використання ізольованих середовищ виконання для перевірки сторонніх компонентів перед їх інтеграцією до ПЗ при його збірці та компіляції. Застосування віртуальних машин, технологій контейнеризації (Docker, containerd, LXC/LXD), системних обмежень засобами AppArmor/SELinux та виконання коду у середовищах на основі WebAssembly (Wasm) дозволяє аналізувати поведінку бібліотек та пакетів-залежностей у контрольованих умовах.

Таким чином, ефективна протидія атакам на ланцюжки постачання ПЗ потребує комплексного підходу, що поєднує автоматизований аналіз складу програмних компонентів, контроль цілісності програмних артефактів, а також поведінковий аналіз сторонніх бібліотек.

Список використаних джерел

1. Securing the Software Supply Chain: Recommended Practices for Developers. URL: https://www.cisa.gov/resources-tools/resources/securing-software-supply-chain-recommended-practices-developers (дата звернення: 18.03.2026).

2. CVE-2024-3094 Detail. URL: https://nvd.nist.gov/vuln/detail/CVE-2024-3094 (дата звернення: 18.03.2026).

3. SUNSPOT: An Implant in the Build Process (by CrowdStrike Intelligence Team). URL: https://www.crowdstrike.com/en-us/blog/sunspot-malware-technical-analysis/ (дата звернення: 18.03.2026).