Уеб реферат (SPA)

OWASP A06

Уязвими и остарели компоненти

Петър Христов
0MI0800202

OWASP A06: Уязвими и остарели компоненти

Vulnerable and Outdated Components • OWASP Top 10 (2021)

План

Какво ще покрием

- Какво е OWASP A06 и защо е високорисков фактор
- Какво се счита за „компонент“ и защо транзитивните зависимости са проблем
- Кога възниква рискът (типични сценарии в проектите)
- Ефектът на доминото: как уязвим компонент компрометира цялото приложение
- Реални инциденти: Log4Shell и Apache Struts (Equifax)
- Откриване и превенция: SBOM, SCA, процеси и автоматизация в CI/CD

Какво е OWASP A06?

Дефиниция

Рискът възниква, когато приложението използва външни зависимости (библиотеки, плъгини, услуги), които са:

- уязвими (известни CVE)
- остарели (не са обновявани)
- неподдържани / end-of-life

Проблемът е критичен, защото модерните системи съдържат голям процент „назаем“ код.

Защо е толкова опасно?

- Уязвимите компоненти често дават директно RCE (Remote Code Execution).
- Може да се стигне до пробив в автентикация/авторизация.
- Изтичане на данни и компрометиране на ключове/конфигурации.
- „Дребен“ ъпдейт може да предотврати масивен инцидент.

Какво е „компонент“?

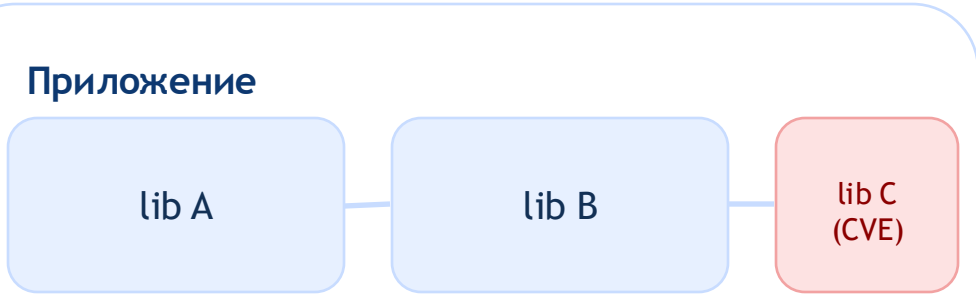
Типове компоненти

- Библиотеки (напр. Log4j, JSON/crypto parsers)
- CMS плъгини (WordPress/Joomla/Drupal)
- Контейнери и базови образи (Docker base images)
- Middleware / сървъри / runtime среди

Транзитивни зависимости

Дори да не добавиш библиотека директно, тя може да влезе през друга зависимост.

Затова сканирането трябва да обхваща цялото dependency дърво (direct + transitive).



Кога възниква рискът?

Типични сценарии в проектите

• Няма инвентар на зависимости (SBOM) и не се знае кои версии се ползват.
• „Работи – не пипай“: ъпдейтите се отлагат с месеци.
• Има пач, но не се прилага навреме (липса на процес/тестове/отговорност).
• Компонентът е end-of-life (няма бъдещи поправки).
• Зависимости от съмнителни източници
• Наследени admin панели, default пароли.

Ключов капан

Транзитивните зависимости често са „невидими“ за екипа, но носят същия риск.

Сканирането трябва да е върху целия dependency graph.

• Знаем ли всички зависимости?
• Следим ли CVE за тях?
• Имаме ли SLA за пачване?
• Имаме ли автоматизация в CI/CD?

Защо е опасно? Ефектът на доминото

Какво се случва на практика

- Уязвимостите са публични (CVE/NVD) и PoC експлойти се появяват бързо.
- Прозорецът за реакция е кратък – „patch or be owned“.
- Компонентът има правата на приложението (DB достъп, файлове, мрежа).
- Възможни последствия: RCE, изтичане на данни, lateral movement.

Log4Shell (Log4j) — пример за масов impact

Ключова идея: „нашият“ код може да е перфектен, но една уязвима зависимост е достатъчна за компрометиране.

Реални инциденти

Log4Shell (CVE-2021-44228)

- RCE чрез специално подготвен низ, който се логва.
- Широко разпространена библиотека → огромен брой засегнати системи.
- Показа, че една зависимост може да създаде глобален риск.

Apache Struts (CVE-2017-5638) — Equifax

- Публичен пач, но забавено прилагане.
- Компрометирани данни на ~140+ милиона потребители (2017).
- Извод: липса на процес за обновяване = бизнес риск.

Откриване: SBOM и Software Composition Analysis (SCA)

SBOM (Software Bill of Materials)

- Инвентар на всички зависимости и версии (вкл. транзитивни).
- Помага за бърза оценка „засегнати ли сме?“ при ново CVE.
- Полезен за одит и compliance.

SCA в CI/CD

- Сканиране на dependency файлове (pom.xml, package.json, lockfiles).
- Автоматично засичане на CVE и лицензни рискове.
- Gate: проваляй билд при критични уязвимости.
- Мониторинг на runtime/контейнери (по възможност).

Инструменти (пример)

Превенция: практики и процеси

Основни мерки

- Поддържай SBOM / инвентар на зависимости и версии.
- Планирай регулярни ъпдейти (пач прозорци + тестове).
- SCA сканиране в CI/CD + ясни прагове за риск (severity gates).
- Политика за версии и отговорности: кой и кога обновява.
- Trusted sources + проверка на интегритет (официални репота, подписи).

Автоматизация: Dependabot / Renovate могат да създават PR-и за обновяване.
Комбинирай с задължителни проверки при merge, за да стане ъпдейтът рутина, а не кризисна реакция.

Ако незабавен ъпдейт е невъзможен → временно „виртуално пачване“ (напр. WAF правила), но целта остава реален ъпдейт.

Ключови изводи

Какво да запомним

• Сигурността не е само „нашия“ код — зависимостите са част от атакуемата повърхност.
• Инвентар (SBOM) + автоматизирани проверки (SCA) са базова хигиена.
• Регулярното пачване и ясните отговорности намаляват риска и времето за реакция.
• Следи целия dependency graph (direct + transitive) и контейнерите/runtime.
• Когато има ново CVE: оценка → приоритизация → patch → проверка.

Използвана литература

Основни източници

- OWASP Top 10 (2021): A06 — Vulnerable and Outdated Components
- OWASP Top 10 — Official Documentation
- NIST National Vulnerability Database (NVD)
- MITRE CVE (Common Vulnerabilities and Exposures)
- CVE-2017-5638 (Apache Struts) — MITRE CVE entry
- OWASP Cheat Sheet Series — Dependency Management
- The Cautionary Saga of Log4Shell (security analysis)

Въпроси?

OWASP A06 • Уязвими и остарели компоненти