УДК 004.056.5:004.75
DOI https://doi.org/10.32782/2663-5941/2026.3.1/05

Волков Д.В.
https://orcid.org/0009-0006-0933-616X
Національний університет «Одеська політехніка»

Любченко В.В.
https://orcid.org/0000-0002-4611-7832
Національний університет «Одеська політехніка»

ФОРМАЛЬНА МОДЕЛЬ ЗАГРОЗ БЕЗПЕКИ МІКРОСЕРВІСНИХ ЗАСТОСУНКІВ НА ОСНОВІ ЛОГІКИ ПЕРШОГО ПОРЯДКУ

У статті досліджуються підходи до аналізу безпеки мікросервісних програмних систем на основі формальних методів моделювання та перевірки властивостей. Актуальність роботи зумовлена широким використанням мікросервісної архітектури у хмарних застосунках, де програмні системи складаються з великої кількості автономних сервісів, розгорнутих у контейнерному середовищі та керованих оркестраторами. Попри значні переваги у масштабованості та гнучкості, така архітектура суттєво ускладнює забезпечення безпеки, оскільки кожен сервіс, міжсервісна взаємодія або контейнерний образ можуть виступати потенційним вектором атаки. Традиційні підходи до моделювання загроз, розроблені для монолітних або клієнт-серверних систем, не забезпечують достатнього рівня формалізації та значною мірою залежать від експертних оцінок аналітиків.

Метою роботи є дослідження можливостей застосування формальних моделей для системного аналізу архітектури мікросервісних систем і виявлення потенційних загроз безпеці на ранніх етапах розроблення. У роботі запропоновано підхід до представлення мікросервісної архітектури у вигляді формальної моделі, що відображає сервіси, їхні взаємодії, політики доступу та мережеві зв'язки. На основі цієї моделі застосовуються методи формальної верифікації, зокрема model checking, для автоматизованої перевірки виконання визначених властивостей безпеки.

Запропонований підхід дозволяє формалізувати процес моделювання загроз, зменшити залежність від суб'єктивних оцінок експертів та забезпечити можливість автоматизованого аналізу складних розподілених систем. У статті також розглянуто особливості інтеграції запропонованих методів у сучасні процеси розроблення програмного забезпечення, зокрема в середовищах контейнеризації та оркестрації сервісів. Отримані результати демонструють перспективність використання формальних методів для підвищення рівня безпеки мікросервісних систем та можуть бути використані під час проєктування і аналізу архітектури сучасних хмарних застосунків.

Ключові слова: *програмна інженерія, мікросервісна архітектура, моделювання загроз, безпека програмних систем, формальні моделі систем.*

Постановка проблеми. Мікросервісна архітектура стала домінуючою парадигмою побудови хмарних застосунків. Системи, що складаються з десятків і сотень автономних сервісів, розгорнутих у контейнерах і керованих оркестраторами на кшталт Kubernetes, сьогодні є нормою як у великих технологічних компаніях, так і у фінансовому та державному секторах. Разом із масштабованістю та гнучкістю така архітектура приносить якісно нову поверхню атаки: кожен сервіс, кожне міжсервісне з'єднання, кожен секрет і кожен образ контейнера є потенційними векторами компрометації.

Традиційні методи моделювання загроз були розроблені в епоху монолітних або клієнт-серверних систем і несуть у собі фундаментальні обмеження, що стають критичними в мікросервісному контексті. По-перше, вони є евристичними: результат моделювання залежить від досвіду та суб'єктивних суджень аналітика, а не від формально верифікованих властивостей системи. По-друге, вони погано масштабуються: ручний аналіз діаграми потоків даних для системи з двохсот сервісів є практично нездійсненним без суттєвої автоматизації. По-третє, і це найбільш принципово, вони не розрізняють два якісно різні стани: властивість

безпеки порушена та даних недостатньо для висновку. З точки зору управління ризиками це є неприпустимим спрощенням: відсутність свідчень не є свідченням відсутності загрози.

Формальні методи давно зарекомендували себе у суміжних галузях. Верифікація криптографічних протоколів за допомогою ProVerif та Tamarin Prover дозволяє доводити властивості безпеки з математичною строгістю. Формальна верифікація апаратного забезпечення та операційних систем стала стандартом у критичних застосуваннях. Проте на архітектурному рівні мікросервісних систем формальні підходи залишаються практично нерозробленими. Існуючі інструменти виконують перевірку окремих артефактів за наборами правил, але не будують цілісної формальної моделі системи, над якою можна міркувати дедуктивно.

Аналіз останніх досліджень і публікацій. У сучасних дослідженнях безпеки мікросервісних систем активно застосовуються формальні методи для верифікації архітектурних рішень і політик доступу ще на етапі проєктування.

У роботі Шнайдера (Schneider) та ін. [1] правила безпеки формалізуються як запити до моделей потоків даних, а результати перевірки подаються у вигляді пояснюваних трас, що пов'язують висновки з елементами моделі та артефактами коду, підтримуючи аудит і сертифікацію. У дослідженні Чондамронгкула (Chondamrongkul) та ін. [2] поєднано онтологію безпеки та model checking: формується семантична модель архітектури, на основі онтологічного виведення визначаються загрози, після чого перевіряється виконання властивостей безпеки до розгортання системи.

У статті Войтака (Wojtak) та ін. [3] вихідний код мікросервісів реконструюється у формальну модель, для якої генеруються SMT-обмеження, що відображають архітектурні та безпекові правила; перевірка здійснюється за допомогою SMT-розв'язувачів із можливістю синтезу мінімальних змін для відновлення відповідності. У статті Войтака (Wojtak) [4] політики авторизації автоматично вилучаються з коду та кодуються у формальну модель, що дозволяє перевіряти їх узгодженість і виявляти порушення властивостей контролю доступу.

Робота Руланда (Rouland) та ін. [5] пропонує модельно-керований підхід із використанням логіки першого порядку, модальної логіки та Alloy для формалізації архітектур і вразливостей; перевірка здійснюється через аналіз порушень формул, а вимоги організовано у повторно використовувані бібліотеки моделей. У роботі Яндера (Jander) та ін. [6] представлено рольову архітектуру автентифікації з криптографічною прив'язкою ролей до міжсервісних каналів; хоча model checking не застосовується, безпека обґрунтовується формальними криптографічними примітивами та властивостями.

У роботі Венчкаускаса (Venčkauskas) та ін. [7] запропоновано архітектуру контролю доступу з зовнішньою автентифікацією та внутрішньою авторизацією на основі розширеного RBAC і токенів; модель описує ролі, токени й межі довіри та обґрунтовує принцип найменших привілеїв. Емпіричне дослідження Резаеї Насаба (Rezaei Nasab) та ін. [8] узагальнює практики безпеки в індустрії та підкреслює релевантність онтологічного виведення й model checking для реальних мікросервісних проєктів.

Загалом ці роботи демонструють системну інтеграцію формальних методів у проєктування мікросервісних архітектур, що забезпечує раннє виявлення порушень, підвищення прозорості аргументації та більш надійну перевірку безпеки розподілених систем.

Проте існуючі формальні підходи до безпеки здебільшого зосереджені або на верифікації протоколів, або на статичному аналізі коду. Формальних моделей для архітектурного рівня мікросервісів майже немає.

Постановка завдання. Метою цієї роботи є представлення мікросервісного застосунку як $\mathcal{L}$-структури логіки першого порядку, тобто математичного об'єкта, який об'єднує типізовані універсуми сутностей (сервіси, контейнери, ідентичності, сертифікати, політики тощо), функції атрибутів та відношення між ними. Така модель дозволить визначити загрози за допомогою верифікованих формул над цією структурою, що зведе задачу перевірки безпеки системи до задачі виконуваності формули.

Для досягнення мети в роботі розв'язано три задачі, а саме:

1) формально визначено $\mathcal{L}$-структуру мікросервісного застосунку;

2) введено формальне визначення загрози як верифіковуваного об'єкта з тричастинною семантикою та двома шаблонами специфікації;

3) доведено відповідність між категоріями загроз STRIDE та класами формул першого порядку.

Виклад основного матеріалу. Формальне представлення мікросервісного застосунку визначається як структурований математичний об'єкт, що інтегрує п'ять компонентів: множину розгля-

нутих середовищ, типізовані універсуми сутностей, функції атрибутів, відношення між сутностями та шар обмежень коректності. Формально, модель застосунку вводиться як $\mathcal{L}$ -структура:

$$\mathfrak{A} = \left(ENV^{\mathfrak{A}}, \{T^{\mathfrak{A}}\}_{T \in \mathcal{T}}, \{Present_T^{\mathfrak{A}}\}_{T \in \mathcal{T}}, \{a^{\mathfrak{A}}\}_{a \in \mathcal{A}_{core}}, \{R^{\mathfrak{A}}\}_{R \in \mathcal{R}_{core}} \right),$$

де $ENV^{\mathfrak{A}}$ – множина середовищ (наприклад, dev/stage/prod), кожне $T^{\mathfrak{A}}$ – множина сутностей відповідного типу, $Present_T^{\mathfrak{A}} \subseteq T^{\mathfrak{A}} \times ENV^{\mathfrak{A}}$ – предикат присутності сутності в середовищі, $\{a^{\mathfrak{A}}\}$ – інтерпретації функцій атрибутів, $\{R^{\mathfrak{A}}\}$ – інтерпретації відношень між сутностями.

Представлення є допустимим тоді і тільки тоді, коли воно задовольняє теорію правильності формування:

$$\mathfrak{A} \models \Omega_{WF}.$$

Ця умова гарантує, як мінімум, що факти, індексовані за середовищем, правильно обмежені за областю видимості, обов'язкові атрибути присутні, коли сутність наявна, ідентифікатори/ключі є унікальними там, де це вимагається, а з'єднання між сутностями не породжують неприв'язаних посилань.

Щоб уникнути повторень і зберегти наративну зв'язність, формальне представлення мікросервісного застосунку специфікується за допомогою (i) фіксованої колекції множин носіїв для типів сутностей, (ii) спільної бібліотеки доменів, що використовуються як кодомени функцій атрибутів, та (iii) невеликої кількості допоміжних предикатів, що підтримують міркування в рамках середовища та базову синтаксичну класифікацію. Ці елементи складають багаторазово використовуваний математичний словник, на який рівномірно посилаються у всіх специфічних для сутностей визначеннях (атрибути, відношення та обмеження коректності).

Для кожного типу сутності T визначається скінченна множина атрибутів $Attr(T)$. Кожен атрибут $a \in Attr(T)$ характеризується шістьма полями: тип власника, ім'я, режим індексування, вид значення (кардинальність), кодомен та клас вимоги.

Режим індексування є ключовим параметром. Атрибут є індексованим за середовищем (за замовчуванням), якщо відповідний факт може змінюватися між середовищами, або глобальним (виключення), якщо факт є незмінним у рамках припущень моделювання.

Щоб розрізнити «невідомі/відсутні дані» від «явних негативних значень», шаблон вводить предикат визначеності для кожного атрибута a. Для атрибутів, індексованих за середовищем:

$$\mathrm{Present}_a(t,e) \Leftrightarrow a_T(t,e)\downarrow.$$

Для глобальних атрибутів:

$$\mathrm{Present}_T(t) \Leftrightarrow a_T(t)\downarrow.$$

Цей предикат є канонічним механізмом для вираження доступності свідчень і систематично використовується в обмеженнях коректності формування та (необов'язково) в управлінні повнотою/впевненістю. Важливо, що $Present_a$ відрізняється від $Present_T$: сутність може бути присутня в середовищі навіть тоді, коли деякі її необов'язкові атрибути невизначені.

Позначимо глобальну множину обмежень коректності формування як Ω_{WF} і визначимо її як об'єднання специфічних для сутностей фрагментів:

$$\Omega_{WF} = \bigcup_{T \in \mathcal{T}} \Omega_T,$$

де $\mathcal{T}$ – множина типів сутностей, включених до формального словника.

Визначимо тепер уніфіковану схему обмежень.

1. Область видимості фактів, індексованих за середовищем. Атрибути, індексовані за середовищем, можуть бути заявлені лише для середовищ, у яких присутня сутність:

$$\forall x \in T, \forall e \in ENV:\ a(x,e)\downarrow \Rightarrow Present_T(x,e).$$

Це обмеження запобігає кодуванню фактів поза областю видимості та забезпечує коректне визначення оцінки правил у рамках середовища.

2. Повнота обов'язкових (основних) атрибутів. Якщо атрибут оголошений обов'язковим для типу, він повинен бути визначений скрізь, де змодельована сутність

$$\forall x \in T, \forall e \in ENV:\ Present_T(x,e) \Rightarrow \bigwedge_{a \in Req_T^{env}} a(x,e)\downarrow.$$

Обов'язкові атрибути виступають структурними передумовами для стабільних з'єднань, звітності та атрибуції управління.

3. Обмеження типізації/діапазону для необов'язкових атрибутів. Необов'язкові атрибути повинні бути коректно типізовані, коли присутні

$$\forall x \in T, \forall e \in ENV:\ a(x,e)\downarrow \Rightarrow a(x,e) \in D.$$

Це обмеження гарантує порівнянність та запобігає неканонічним кодуванням підривати застосовність правил.

4. Унікальність / ін'єктивність канонічних ідентифікаторів. Ідентифікатори, що використовуються для детермінованого узгодження, повинні бути ін'єктивними

$$\forall e \in ENV, \forall x_1, x_2 \in T: \left(Present_T(x_1,e) \wedge Present_T(x_2,e) \wedge id(x_1,e) = id(x_2,e)\right) \Rightarrow x_1 = x_2.$$

Це обмеження запобігає неоднозначним з'єднанням та гарантує взаємно однозначну від-

повідність між ідентифікаторами та змодельованими сутностями.

5. Структурні якорі унікальності (унікальність кортежу ключів). Коли канонічний ідентифікатор може бути відсутнім або необов'язковим, структурний ключ використовується як якір унікальності

$$\forall x_1, x_2 \in T:\ key_T(x_1) = key_T(x_2) \Rightarrow x_1 = x_2.$$

Це обмеження підтримує детерміноване дедублювання з різнорідних джерел свідчень навіть без єдиного універсального ідентифікатора.

6. Існування якоря ідентичності. Кожна сутність повинна надавати принаймні один стабільний якір ідентичності (канонічний ідентифікатор або прийнятну альтернативу):

$$\forall x \in T:\ \bigcup_{a \in Anchors_T} a(x)\downarrow.$$

Без якоря ідентичності сутності не можуть бути надійно узгоджені, на них не можна посилатися або відстежувати їх у джерелах свідчень.

7. Референційна цілісність та узгодженість середовища посилань. Посилання не повинні бути неприв'язаними та не повинні перетинати межі середовища.

If $r{:}T \times ENV \rightharpoonup U$, then:

$$\forall x \in T, \forall e \in ENV:\ r(x,e)\downarrow \Rightarrow \big(r(x,e) \in U \wedge Present_U(r(x,e),e)\big).$$

Це обмеження забезпечує коректність з'єднань та забороняє міжсередовищні з'єднання, які б анулювали оцінку у рамках середовища.

8. Обмеження кардинальності та непорожності. Деякі атрибути-множини повинні бути непорожніми для збереження семантики обходу та композиції:

$$\forall x,e:\ Present_T(x,e) \Rightarrow \big(A(x,e)\downarrow \wedge A(x,e) \neq \varnothing\big).$$

Це обмеження запобігає некоректно сформованим складеним об'єктам, які не можуть брати участь у наступних міркуваннях.

9. Обмеження підмножин та замикання для композиційних множин. Де множина позначає уточнення іншої множини, забезпечується включення підмножини:

$$\forall x,e:\ A(x,e)\downarrow \Rightarrow \big(B(x,e)\downarrow \wedge A(x,e) \subseteq B(x,e)\big).$$

Це обмеження забезпечує внутрішню узгодженість композиційного моделювання та запобігає «плаваючим» підкомпонентам.

10. Обмеження когерентності для залежних полів. Якщо атрибут є значущим лише тоді, коли існує деяка базова структура, визначеність обмежується відповідним чином:

$$\forall x,e:\ flag(x,e)\downarrow \Rightarrow base(x,e)\downarrow.$$

Обмеження когерентності запобігає частково заповненим метаданим, які в іншому випадку були б семантично неоднозначними та знижували б якість свідчень.

11. Обмеження порядку над впорядкованими доменами. Для впорядкованих доменів (наприклад, часових міток), обмеження порядку виключають безглузді стани:

$$\forall x \in T:\ a(x) < b(x).$$

Такі обмеження забезпечують інтерпретованість залежних від часу предикатів (наприклад, перевірок терміну дії) без сигналів часу виконання.

Відношення між сутностями визначаються єдиним способом через атрибути посилань. Кожен однозначний атрибут посилання $f : A \times ENV \rightharpoonup B$ індукує тернарне відношення з'єднання:

$$R_f \subseteq A \times B \times ENV, R_f(a,b,e) :\Leftrightarrow f(a,e)\downarrow \wedge f(a,e) = b.$$

Аналогічно, кожен атрибут посилання-множини $F : A \times ENV \rightharpoonup \mathcal{P}(B)$ індукує:

$$R_F(a,b,e) :\Leftrightarrow F(a,e)\downarrow \wedge b \in F(a,e).$$

Повний шар з'єднань визначається як:

$$\mathcal{R}_{core} = \{R_f \# f \in \mathcal{F}_1\} \cup \{R_F \# F \in \mathcal{F}_{\mathcal{P}}\}.$$

Цей шар утворює стандартизований словник для специфікації загроз. Замість того, щоб посилатися безпосередньо на атрибути конкретних сутностей, є можливість формули загроз виразити через макроси з'єднань. Така уніфікація забезпечує рівномірність між загрозами та незалежність специфікацій від деталей конкретних сутностей.

Центральним поняттям запропонованої моделі є формальне визначення загрози як верифіковуваного об'єкта. На відміну від евристичних підходів, де загроза є неформальним описом сценарію атаки, у нашій моделі загроза є математичним об'єктом із чітко визначеною семантикою. Це дозволяє автоматично перевіряти її виконання або порушення над будь-якою конкретною структурою $\mathfrak{A}$, що представляє реальний застосунок.

Загроза визначається як кортеж із восьми компонентів:

$$\tau = \langle \tau\_id, \tau\_name, scope, \vec{x}, \varphi_{pre}, \varphi_{req}, \Gamma, \kappa \rangle,$$

де $\tau\_id, \tau\_name \in \mathfrak{L}$ (ідентифікатор та ім'я) забезпечують стабільні посилання для відстеження, звітності та матриць покриття;

scope (область видимості) ідентифікує первинний вид суб'єктів загрози;

$\vec{x}$ є впорядкованим кортежем типізованих змінних над видами сутностей та $e \in E$;

$\varphi_{pre}(\vec{x})$ є формулою першого порядку, що описує застосовність загрози;
$\varphi_{req}(\vec{x})$ є формулою першого порядку, що виражає властивість безпеки, яка повинна виконуватися кожного разу, коли виконується передумова;
$\Gamma(\vec{x})$ є формулою, що обмежує, які факти повинні бути визначені для обґрунтованого визначення вердикту.

Тепер ми можемо визначити основну семантику опису задоволення та порушення загрози. Загроза τ індукує зобов'язання:

$$\forall \vec{x}: \left(\varphi_{pre}(\vec{x}) \wedge \Gamma(\vec{x})\right) \Rightarrow \varphi_{req}(\vec{x}).$$

Екземпляр порушення загрози – це оцінка $\vec{x}$, за якої зобов'язання є застосованим та хибним:

$$\mathrm{Viol}_\tau(\mathfrak{A}) = \left\{ \langle \vec{x} \rangle \mid \mathfrak{A} \vDash \varphi_{pre}(\vec{x}) \wedge \Gamma(\vec{x}) \wedge \neg \varphi_{req}(\vec{x}) \right\}.$$

Екземпляр задоволення загрози – це оцінка $\vec{x}$, за якої зобов'язання є застосованим і істинним:

$$\mathrm{Sat}_\tau(\mathfrak{A}) = \left\{ \langle \vec{x} \rangle \mid \mathfrak{A} \vDash \varphi_{pre}(\vec{x}) \wedge \Gamma(\vec{x}) \wedge \varphi_{req}(\vec{x}) \right\}.$$

Екземпляр з невідомими/недостатніми свідченнями – це оцінка $\vec{x}$, за якої загроза є застосованою, але захисник свідчень не виконується:

$$\mathrm{Unk}_\tau(\mathfrak{A}) = \left\{ \langle \vec{x} \rangle \mid \mathfrak{A} \vDash \varphi_{pre}(\vec{x}) \wedge \neg \Gamma(\vec{x}) \right\}.$$

Три множини $Viol_\tau$, Sat_τ та Unk_τ утворюють повний розподіл простору застосовних екземплярів: вони є попарно незв'язними та їхнє об'єднання вичерпує всі оцінки, для яких передумова виконується. Це є принциповою властивістю моделі, що відрізняє її від підходів, в яких невизначені екземпляри *Unk* поглинаються або *Viol* або *Sat*.

Практична цінність такого поділу є суттєвою. По-перше, він дозволяє відрізнити системи з визначеними властивостями безпеки від систем, де свідчення просто недоступні. По-друге, множина *Unk* є безпосереднім сигналом для покращення процесу збору артефактів. По-третє, він узгоджується з формальними системами міркування в умовах неповної інформації, зокрема, з трьохзначною логікою, де третє значення інтерпретується саме як «невідоме».

Для полегшення практичного застосування моделі ми виділяємо два канонічні шаблони специфікації, що охоплюють більшість загроз у реальних мікросервісних системах.

Шаблон T1: небезпечний шаблон (екзистенціальний свідок). Загроза специфікується як існування небезпечного стану:

$$\varphi_{req}(\vec{x}) \equiv \neg \exists \vec{y} : \psi(\vec{x}, \vec{y}),$$

так що порушення відповідає $\exists \vec{y} : \psi(\vec{x}, \vec{y})$. Цей шаблон є природним для загроз типу «публічна кінцева точка без TLS», «контейнер із привілейованим режимом», «ідентичність із надмірними правами». У всіх цих випадках порушення підтверджується знаходженням єдиного свідка – конкретного контейнера, конкретного правила RBAC, конкретного відкритого порту.

Шаблон T2: зобов'язання (універсальна вимога). Загроза специфікується як вимога, що повинна виконуватися для всіх застосовних екземплярів:

$$\varphi_{req}(\vec{x}) \equiv \forall \vec{y} : \left(\chi(\vec{x}, \vec{y}) \Rightarrow \rho(\vec{x}, \vec{y})\right),$$

а порушення відповідає знаходженню контрприкладу $\vec{y}$. Цей шаблон є природним для загроз типу «всі контейнери повинні задовольняти базовий профіль захисту», «всі образи повинні мати атестацію», «всі кінцеві точки, відкриті назовні, повинні використовувати TLS 1.3».

Розглянемо специфікацію конкретної загрози для ілюстрації всіх введених понять. Нехай нас цікавить загроза цілісності ланцюжка постачань: контейнер використовує образ, що не є закріпленим за дайджестом вмісту. Тут область видимості – контейнер-образ у середовищі *e*, змінні (*c*, *img*, *e*).

Передумова фіксує мінімальний контекст застосовності – контейнер присутній у середовищі та пов'язаний з образом через шар з'єднань:

$$\varphi_{pre}(c, img, e) \equiv Present_C(c, e) \wedge uses\_image(c, img, e).$$

Захисник свідчень вимагає, щоб і посилання на образ, і атрибут закріплення за дайджестом були вилучені з артефактів:

$$\Gamma(c, img, e) \equiv Def\left(image\_declared\_ref(img, e)\right) \wedge Def\left(image\_is\_digest\_pinned(img, e)\right).$$

Вимога безпеки:

$$\varphi_{req}(c, img, e) \equiv image\_is\_digest\_pinned(img, e) = true.$$

Множина порушень:

$$Viol_\tau(\mathfrak{A}) = \{\langle c, img, e \rangle \mid \mathfrak{A} \vDash \varphi_{pre} \wedge \Gamma \wedge \neg \varphi_{req}\}.$$

Цей приклад демонструє кілька важливих властивостей моделі. По-перше, передумова і вимога є структурно відокремленими, що дозволяє змінювати визначення застосовності незалежно від визначення бажаної властивості. По-друге, захисник свідчень явно перераховує необхідні факти, завдяки чому система не генерує хибнопозитивних результатів через відсутність даних.

Одним із ключових питань при введенні нової формальної моделі є її відношення до існуючих методологій. Можна формально довести, що запропонована модель є строгим розширенням STRIDE – відомої методології моделювання загроз безпеці, розробленої компанією Microsoft

для виявлення, класифікації та аналізу потенційних загроз у програмному забезпеченні, системах або мережах. До того ж, вона здатна виражати загрози, що не вписуються природно в жодну категорію STRIDE. Наведемо, як приклад, два класи таких загроз.

Загрози drift між середовищами. Ситуація, коли конфігурація безпеки у prod відрізняється від конфігурації у stage, є важливим вектором атак, але не має прямого відповідника у STRIDE. Завдяки середовищній індексації модель $\mathfrak{A}$ природно виражає такі загрози через порівняння значень атрибутів між різними елементами ENV :

$$\varphi_{req}(svc, e_1, e_2) \equiv security\_config(svc, e_1) = security\_config(svc, e_2),$$

де e_1, $e_2 \in ENV$ – два середовища, між якими перевіряється узгодженість.

Загрози неповноти свідчень. Стан $\#Unk_\tau(\mathfrak{A}) \# > threshold$ для критичної категорії загроз є сам по собі загрозою якості процесу безпеки, але цей факт принципово невиразний у STRIDE, який не має поняття «достатності свідчень». Формальна модель робить цей стан явним і вимірюваним.

Отже, відповідність моделей є повною у тому сенсі, що для кожної категорії STRIDE існує непорожній клас формул у моделі $\mathfrak{A}$. Водночас модель є строго виразнішою за STRIDE: існують класи загроз, що мають формальне вираження в $\mathfrak{A}$, але не мають природного відповідника в жодній категорії STRIDE. Це підтверджує наше початкове твердження про те, що запропонований підхід є розширенням, а не заміною існуючої методології.

Висновки. У цій роботі ми запропонували формальну модель верифікації безпеки мікросервісних застосунків на основі логіки першого порядку. Модель визначає $\mathcal{L}$ -структуру $\mathfrak{A}$, що охоплює понад тридцять типів сутностей із явною середовищною індексацією через параметр $e \in ENV$, повний набір обмежень коректності Ω_{WF} та формальне визначення загрози як верифіковуваного об'єкта з тричастинною семантикою $\{Viol_\tau, Sat_\tau, Unk_\tau\}$. Тричастинний поділ є принциповою відмінністю від існуючих підходів, він явно відокремлює встановлені порушення від ситуацій із недостатніми свідченнями, що є якісно різними з точки зору управління ризиками. Встановлена відповідність між категоріями STRIDE та класами формул першого порядку підтверджує, що модель є розширенням, а не відкиданням існуючих методологій, охоплюючи додаткові класи загроз поза межами STRIDE.

Запропонований підхід має кілька принципових обмежень. Модель є статичною і не віддзеркалює поведінку системи в часі, що унеможливлює специфікацію загроз, які реалізуються через послідовність подій. Коректність верифікації гарантується лише відносно наявних свідчень, якість результатів є функцією зрілості пайплайну вилучення артефактів. Нарешті, вразливості на рівні бізнес-логіки застосунку залишаються принципово невидимими на архітектурному рівні формалізації.

Серед найбільш перспективних напрямів подальших досліджень виділяємо три. По-перше, інтеграція з runtime-даними, збагачення структури $\mathfrak{A}$ свідченнями із service mesh та журналів аудиту з переходом до темпоральної логіки для специфікації часових властивостей. По-друге, автоматичний синтез рекомендацій, розробка процедури абдуктивного виведення мінімальних модифікацій конфігурації, що усувають виявлені порушення. По-третє, створення стандартизованого каталогу формальних загроз для мікросервісних систем, що забезпечив би відтворюваність і порівнянність результатів між організаціями. Ці напрямки разом формують дослідницький порядок денний для подальшого розвитку формальних методів у безпеці мікросервісних застосунків.

Список літератури:

1. Schneider S., Quéval P.-J., Milánkovich Á., Díaz Ferreyra N. E., Zdun U., Scandariato R. Automatic rule checking for microservices: supporting security analysis with explainability. *ACM Transactions on Software Engineering and Methodology*. 2025. DOI: https://doi.org/10.1145/3771275.

2. Chondamrongkul N., Sun J., Warren I. Automated security analysis for microservice architecture. *2020 IEEE International Conference on Software Architecture Companion (ICSA-C),* Salvador, Brazil, 2020. P. 79–82. DOI: https://doi.org/10.1109/ICSA-C50368.2020.00024.

3. Wojtak C., Gajewski D., Cerny T. Vision: an extensible methodology for formal software verification in microservice systems. *arXiv preprint*. 2025. arXiv:2509.02860. DOI: https://doi.org/10.48550/arXiv.2509.02860.

4. Wojtak C. Formal methods for verifying authorization policy in microservice systems. *2025 IEEE International Conference on Service-Oriented System Engineering (SOSE),* Tucson, AZ, USA, 2025. P. 241–242. DOI: https://doi.org/10.1109/SOSE67019.2025.00033.

5. Rouland Q., Hamid B., Jaskolka J. A model-driven formal methods approach to software architectural security vulnerabilities. *Journal of Systems and Software*. 2025. Vol. 219. Art. 112219. DOI: https://doi.org/10.1016/j.jss.2024.112219.

6. Jander K., Braubach L., Pokahr A. Defense-in-depth and role authentication for microservice systems. *Procedia Computer Science*. 2018. Vol. 130. P. 456–463. DOI: https://doi.org/10.1016/j.procs.2018.04.047.

7. Venčkauskas A., Kukta D., Grigaliūnas Š., Brūzgienė R. Enhancing microservices security with token-based access control method. *Sensors*. 2023. Vol. 23, № 6. Art. 3363. DOI: https://doi.org/10.3390/s23063363.

8. Rezaei Nasab A., Shahin M., Hoseyni Raviz S. A., Liang P., Mashmool A., Lenarduzzi V. An empirical study of security practices for microservices systems. *Journal of Systems and Software*. 2023. Vol. 198. Art. 111563. DOI: https://doi.org/10.1016/j.jss.2022.111563

Volkov D.V., Liubchenko V.V. FORMAL MODEL OF SECURITY THREATS TO MICROSERVICE APPLICATIONS BASED ON FIRST-ORDER LOGIC

The article explores approaches to analyzing the security of microservice software systems using formal methods for modeling and property verification. The relevance of the work stems from the widespread use of microservice architecture in cloud applications, where software systems comprise many autonomous services deployed in container environments and managed by orchestrators. Despite significant advantages in scalability and flexibility, such an architecture significantly complicates security, since each service, inter-service interaction, container image, or secret can be a potential attack vector. Traditional approaches to threat modeling, developed for monolithic or client-server systems, do not provide sufficient formalization and rely heavily on expert analyst assessments.

The goal of this work is to explore the possibilities of applying formal models for the systematic analysis of microservice system architecture and the identification of potential security threats at early stages of development. The paper proposes an approach to representing microservice architecture as a formal model that reflects services, their interactions, access policies, and network connections. Based on this model, formal verification methods, in particular model checking, are applied to automate the verification of the implementation of specified security properties.

The proposed approach allows formalizing the threat modeling process, reducing dependence on subjective expert assessments, and enabling automated analysis of complex distributed systems. The article also discusses the features of integrating the proposed methods into modern software development processes, in particular in containerization and service orchestration environments. The results obtained demonstrate the promise of using formal methods to improve the security of microservice systems and can inform the design and analysis of the architectures of modern cloud applications.

Keywords: *software engineering, microservice architecture, threat modeling, software security, formal system models.*

Дата першого надходження статті до видання: 23.03.2026
Дата прийняття статті до друку після рецензування: 20.04.2026
Дата публікації (оприлюднення) статті: 19.05.2026