КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (28), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.28.855
УДК 004.056.53

Добришин Юрій Євгенович
кандидат технічних наук, доцент
Національна академія Служби безпеки України, Київ, Україна
ORCID ID: 0000-0003-2473-9507
ydobryshyn@gmail.com

ФОРМАЛІЗАЦІЯ ВИБОРУ СПОСОБУ ВІДНОВЛЕННЯ ПОШКОДЖЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВНАСЛІДОК ДІЇ КІБЕРАТАК

Анотація. У статті запропоновані методика формалізованого вибору способу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак. Розробка приведеної методики здійснювалася на підставі аналізу сучасних формальних методів та інструментів, які включають логіко-алгебраїчні підходи щодо аналізу та відновлення програмного забезпечення інформаційно-комунікаційних систем. Для розробки методичних матеріалів, приведених у даній статті, використовувались наукові розробки вітчизняних та закордонних фахівців щодо моделювання та перевірки роботи різних моделей, які на теперішній час застосовуються у сфері кібербезпеки. У якості інструменту щодо формалізації вибору способу відновлення пошкодженого програмного забезпечення були вибрані графові моделі, які дозволяють візуалізувати можливі рішення щодо вибору способу відновлення пошкодженого програмного забезпечення, а також надають можливість спроектувати технологічний процес та призначити оптимальний маршрут відновлення дефектів програмного забезпечення. На підставі аналізу компонентів технологічного процесу відновлення пошкодженого програмного забезпечення, визначені основні елементи, що потребують дослідження та побудований граф відношень, за допомогою якого визначені відношення між елементами одного технологічного об'єкта та відношення між елементами різних об'єктів, що належать до структури технологічного процесу відновлення пошкодженого програмного забезпечення. Сформульовані умови, які підтверджують істинність відношень, визначених за результатами задачі вибору способу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак. Визначені основні властивості елементів, що реалізують структуру технологічного процесу відновлення пошкодженого програмного забезпечення. На підставі відношень та основних властивостей елементів, сформульовані технологічні умови, які дозволяють здійснювати вибір раціонального способу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак. Формалізовані правила, що були розроблені, дозволяють здійснювати побудову алгоритмів вибору раціонального способу відновлення програмного забезпечення а також розробку та впровадження програмних засобів з метою автоматизації складних процесів щодо аналізу та відновлення систем після кібератак.

Ключові слова: формалізація; спосіб відновлення; пошкоджене програмне забезпечення технологічні умови; відношення; властивості; технологічний процес відновлення; модель; графові моделі.

ВСТУП

Постановка проблеми. Розглядаючи та аналізуючи експлуатацію сучасних інформаційно-комунікаційних систем, технологію їх працездатності, заходи щодо супроводження та технічного обслуговування програмних компонентів, що належать до їх складу, необхідно зазначити, що на теперішній час спостерігається зростання кількості та складності кібератак, які руйнують працездатність інфраструктури та функціонування програмного забезпечення вказаних систем, порушуючи конфіденційність та доступність даних.

Детальний аналіз сучасних кібератак свідчить про те, що атаки стають більш розумними та витонченими, внаслідок чого значно ускладняються процедури виявлення дефектів пошкодженого програмного забезпечення, його діагностики, а особливо оптимального відновлення та приведення до працездатного стану його компонентів.

Дефекти пошкодженого програмного забезпечення, які виявлені після впливу кібератак, можуть призвести до серйозних наслідків у роботі інформаційно-телекомунікаційних систем, а саме, зупинки технологічних процесів, втрат інформації, порушення політики безпеки.

Через широкий спектр кібератак важко кількісно оцінити їхній вплив на працездатність автоматизованих систем та програмних комплексів, а особливо швидко відновити працездатність пошкодженого програмного забезпечення шляхом вибору оптимального способу відновлення, який залежить від процедури виявлення та аналізу дефектів програмного забезпечення, які з'являються унаслідок наявних вразливостей у складі програмних модулів та компонентів автоматизованих систем. Для цього необхідно здійснювати класифікацію дефектів програмного забезпечення з подальшим їх групуванням в технологічно — подібні групи з призначенням оптимальної технології відновлення.

Окрім того складно забезпечити автоматизацію процесу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак, тому що на теперішній час відсутні формалізовані методики опису внутрішнього змісту та взаємозв'язків окремих технологічних операцій з відновлення дефектів пошкодженого програмного забезпечення, тобто основною проблемою є відсутність систематизованого, формалізованого підходу до вибору способу відновлення програмного забезпечення.

Проблема вибору оптимального способу відновлення пошкодженого програмного забезпечення полягає ще в тому, що у багатьох випадках рішення про вибір одного або іншого способу відновлення приймається інтуїтивно без урахування специфіки атаки, характеру дефекту, взаємозв'язків між дефектами, способами їх відновлення, наявними технічними ресурсами.

Такий підхід базуються виходячи з досвіду фахівців, які виконують операції діагностування на місці події. Зазначені підходи як правило не мають чітких процедурних дій та алгоритму виконання певних етапів щодо кінцевого результату та контролю за якістю відновлення.

У наслідок відсутності методичних матеріалів щодо формалізації вибору способу відновлення програмного забезпечення внаслідок впливу кібератак, підходи до автоматизації технологічних процесів з відновлення становяться малопридатними та неефективними. Виникає проблема щодо розробки або вибору формалізованих методів з використанням різних моделей, які спроможні математично оцінити весь процес відновлення пошкодженого програмного забезпечення внаслідок впливу кібератак.

Аналіз останніх досліджень і публікацій. Питання щодо формалізації технологічних процесів з відновлення пошкодженого програмного забезпечення внаслідок впливу кібератак розглядаються та досліджуються у багатьох наукових працях вітчизняних та закордонних фахівців.

Однією з основоположних праць, є робота наукових фахівців [1] щодо опису формалізованих процесів відновлення програмного та апаратного забезпечення внаслідок впливу кібератак. Фахівцями документу представлено план, який визначає поняття критичних ресурсів, визначення персоналу, що здійснює експлуатацію програмного забезпечення а також ролей та прав доступу до ресурсів автоматизованих систем та програмних модулів.

Питання дослідження моделей та методів захисту інформації в комп'ютерних системах, найшли своє відображення у наукових працях [2] – [4]

Матеріали вказаних робіт охоплюють різні аспекти кібербезпеки, зокрема моделі, що можуть бути корисними для розробки та формалізації процесів відновлення пошкодженого програмного забезпечення.

Огляд сучасних формальних методів, застосовуваних у сфері безпеки та алгоритми відновлення пошкодженого програмного забезпечення досліджуються у роботах наукових фахівців [5] – [7]. Автори у своїх роботах пропонують формалізований підхід до відновлення цільових фізичних станів у інформаційно-комунікаційних системах після атак, розглядають алгоритми відновлення програмного забезпечення в умовах обмежених ресурсів та постійних атак.

Необхідність інтеграції управління інформаційною безпекою та реагування на інциденти, з акцентом на формалізацію процесів відновлення щодо забезпечення цілісного підходу, досліджуються у наукових роботах [8], [9].

Окремі автори [10], [11] розглядають використання формальних методів для забезпечення ефективності у підвищенні стійкості програмного забезпечення до кібератак та відновлення після них, а також використовують формальні методи для оцінки пошкоджень та вибору оптимальної стратегії відновлення під час застосування інтелектуальної системи підтримки прийняття рішень.

Інтерес представляють наукові роботи [12], [13], де обговорюються питання використання формальних методів для забезпечення безпеки програмного забезпечення протягом усього життєвого циклу, включаючи етапи відновлення після атак, а також розробки автоматизованих систем, які використовують ієрархічні операційні моделі щодо моніторингу, діагностики та швидкого реагування на атаки.

Таким чином, формалізація вибору способу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак є актуальною та необхідною умовою для розробки ефективного технологічного процесу діагностування та відновлення програмного забезпечення. Дослідження у цієї сфері повинні зосереджуватися на використанні сучасних моделей, що дозволяють формалізувати процес ефективного відновлення дефектів програмного забезпечення.

В таких умовах актуальність набувають моделі на основі графових структур. Графові моделі дозволяють не лише візуалізувати можливі рішення щодо вибору способу відновлення пошкодженого програмного забезпечення, такі моделі надають можливість спроектувати певний технологічний процес та призначити оптимальний маршрут відновлення дефектів програмного забезпечення.

Використанню графових моделей, як інструменту формалізації вибору способу відновлення пошкодженого програмного забезпечення присвячена значна кількість наукових праць.

Так у дослідженні [14] представлено близько 70 графових моделей, що застосовуються для аналізу кібератак. У статті [15] та наукової роботі [16] запропоновано модель виявлення програм, що використовує графові нейронні мережі для аналізу поведінки систем, виявлення аномалій та навчання. Моделювання кібератак для візуальної аналітики представлено у роботі [17]. Підхід запропонований у роботі допомагає краще виявити та розпізнати структуру атак та визначитися з ефективним способом відновлення.

Захист віртуального хмарного середовища на основі графа структури атаки розглядається у науковій праці [18]. Авторами статті, описується захист віртуального хмарного середовища на підставі графа, побудованого відповідно до структури атаки.

Такий підхід дозволяє виявляти вразливості та здійснювати заходи з відновлення пошкодженого програмного забезпечення внаслідок дії кібератак.

Таким чином формалізація процесу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак є головним фактором ефективного виявлення дефектів, призначення способів їх відновлення та проектування оптимального технологічного процесу щодо забезпечення працездатності інформаційно-комунікаційних систем.

Під час формалізації процесів відновлення, особливе місце приділяється графовим моделям. Вказані моделі виступають потужним інструментом щодо аналізу, виявлення та відновлення після кібератак. За допомогою графових моделей можна здійснити формалізацію складних зв'язків між структурними компонентами інформаційно-комунікаційних систем, автоматичних комплексів, різних програмних модулів.

Сучасні графові моделі дозволяють виявляти вразливості, які виникають внаслідок дії кібератак та створюють умови щодо планування ефективної стратегії відновлення пошкодженого програмного забезпечення. Вказані моделі допомагають побудувати структуру **процесу** відновлення пошкодженого програмного забезпечення, а також проаналізувати варіанти відновлення дефектів програмного забезпечення та здійснювати аналітичну оцінку кожного з них.

Дослідження, приведені у наукових роботах вітчизняних та закордонних фахівців свідчать про актуальність та перспективність застосування вказаних моделей для вирішення різних задач у кібербезпеці.

Мета дослідження. Метою дослідження є розробка формалізованої методики щодо вибору способу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак, яка передбачає визначення основних структурних елементів технологічного процесу відновлення та їх властивостей, а також розробку формалізованих правил з використанням графової моделі та логіко-алгебраїчних підходів, в основу яких покладені відношення між елементами, що належать до структури технологічного процесу відновлення пошкодженого програмного забезпечення.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Математичне модулювання базується на системному підході, який є концептуальним по відношенню щодо дослідження структури технологічного процесу відновлення пошкодженого програмного забезпечення у наслідок дії кібератак.

Розгляд структури технологічного процесу відновлення показує, що головним елементом дослідження є саме пошкоджене програмне забезпечення, програмні модулі, які належать до її складу, дефекти, що виникають у наслідок дії кібератак, а також способи відновлення, так як вони забезпечують побудову рішень та технологію оптимального відновлення.

Тому предметом дослідження будуть виступати відношення між множинами $\{PR\}$, $\{PM\}, \{GP\}, \{VP\}$, які породжують аналітичні взаємозв'язки, що доступні для математичної інтерпретації, які в свою чергу дозволяють перейти до формалізованого опису технологічного процесу відновлення пошкодженого програмного забезпечення.

Множені, що потребують дослідження, є технологічними об'єктами та мають певні властивості (функціональність). Між предметом та властивістю існують відношення, які можуть визначатися одномісним предикатом P(x), тобто функцією змінної $\{x\}$, визначеної на деякій множині, яка приймає значення ∧ та ∨.

Виходячи із зазначеного, побудуємо граф відношень між вказаними множинами (рис. 1). За допомогою графа визначим відношення між елементами одного технологічного об'єкта та відношення між елементами різних об'єктів, що належать до структури технологічного процесу відновлення пошкодженого програмного забезпечення.

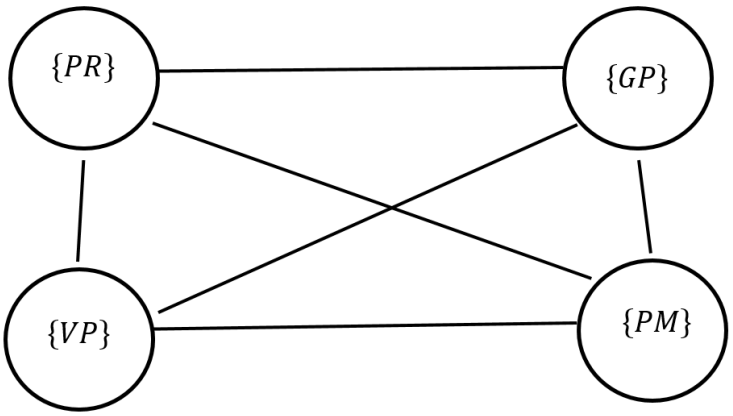

Рис. 1. Граф множин технологічної середи відновлення пошкодженого програмного забезпечення внаслідок дії кібератак

Програмне забезпечення P_z, де $z \in \{PR\}$, яке пошкоджено у наслідок дії кібератак, можна представити у вигляді множини дефектних програмних модулів P_i, де $i \in \{PM\}$

$$P_z = \{P_1, P_2 \dots \dots \dots \dots, P_i \dots \dots \dots P_M\} \quad (1)$$

У свою чергу працездатність кожного дефектного програмного модулю P_i може бути забезпечена з використанням альтернативних способів відновлення C_j, де $j \in \{VP\}$ Враховуючи вищезазначене, можна записати:

$$P_i = \left[\vee C_j^i\right] \quad (2)$$

Тоді для всього програмного забезпечення P_z, де $z \in \{PR\}$, яке пошкоджено у наслідок дії кібератак, отримуємо наступний математичний вираз:

$$P_z = \wedge P_i \left[\vee C_j^i\right] \quad (3)$$

Зазначимо технологічні можливості способів відновлення дефектних програмних модулів P_i, де $i \in \{PM\}$ у вигляді предикатної матриці, а саме, відношенням між елементами множини P_i, та способів відновлення C_j.

$$M = \begin{array}{cccccccc} & C_1 & C_2 & \dots\dots & C_j & \dots\dots & C_v \\ P_1 & y_{11} & y_{12} & \dots\dots & y_{1j} & \dots\dots & y_{1v} \\ P_2 & y_{21} & y_{22} & \dots\dots & y_{2j} & \dots\dots & y_{2v} \\ \dots\dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ P_i & y_{i1} & y_{i2} & \dots\dots & y_{ij} & \dots\dots & y_{iv} \\ P_{pm} & y_{pm1} & y_{pm2} & \dots\dots & y_{pmj} & \dots\dots & y_{pmv} \end{array} \quad (4)$$

Якщо елемент y_{ij} =1, це означає що програмний модуль може бути відновлений, у разі y_{ij} =0 відновлення здійснити неможливо.

Побудуємо вектор $y_j = (y_{1j}, y_{2j}, \dots\dots y_{nj})$, не нулеві елементи якого визначають способи відновлення, якими може бути усунутий дефект програмного модуля та підставимо вираз до формули (2)

$$P_1 = (y_{11}, y_{21}, \dots \dots \dots y_{pm1})$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad (5)$$
$$P_m = (y_{1v}, y_{2v}, \dots \dots \dots y_{pmv})$$

На підставі цього отримаємо наступне:

$$P_i = C_1^i\,(y_{11}\,, y_{21}, \dots y_{i1} \dots \dots . y_{pm1}) \wedge C_2^i\,(y_{12}\,, y_{22}, \dots y_{i2} \dots \dots . y_{pm2}) \wedge \dots..\wedge\ C_j^i\ (y_{1j}\,, y_{2j}, \dots y_{ij} \dots \dots . y_{pmj}) \wedge C_v^i\,(y_{1v}\,, y_{2v}, \dots y_{iv} \dots \dots . y_{pmv}) \quad (6)$$

Для пошкодженого програмного модуля після операції перетворення, отримаємо наступне:

$$P_i = \wedge\ C_j^i\ (\vee\ y_{ij}) \quad (7)$$

За результатами цього для пошкодженого програмного забезпечення P_z, де $z \in \{PR\}$, отримуємо:

$$P_z = \wedge\, P_i[\wedge\, C_j^i\ (\vee\ y_{ij})] \quad (8)$$

Зазначимо, що для кожного пошкодженого програмного модуля P_i, де $i \in \{PM\}$ існує хоча б один з способів відновлення, у разі існування декількох способів відновлення, необхідно вибирати «min» спосіб по собівартості та «max» по продуктивності.

Відбір та скорочення альтернативних варіантів способів відновлення пошкодженого програмного забезпечення P_z де $z \in \{PR\}$, забезпечується шляхом визначення та аналізу властивостей елементів, які приймають участь у формуванні технологічного процесу відновлення та сформованих відношень між ними.

Для формалізованого опису вибору способу відновлення пошкодженого програмного забезпечення розглянемо основні властивості елементів технологічного процесу відновлення пошкодженого програмного забезпечення, які приведені у табл. 1.

Таблиця 1

Приклад властивостей основних елементів технологічного процесу відновлення пошкодженого програмного забезпечення (фрагмент)

Назва елементу	Властивість (функціональність)	Позначення властивості (функціональності)
Пошкоджене програмне забезпечення P_z де $z \in \{PR\}$	Системне програмне забезпечення	$[S_{pz}^1]$
	Прикладне програмне забезпечення	$[S_{pz}^2]$
	Інструментальне програмне забезпечення	$[S_{pz}^3]$
Програмний модуль P_i де $i \in \{PM\}$	Модулі ядра	$[S_p^1]$
	Системні модулі	$[S_p^2]$
	Модулі програмних бібліотек	$[S_p^3]$
	Модулі драйверів пристроїв	$[S_p^4]$
	Інтерфейсні модулі	$[S_p^5]$
	Модулі бізнес-логіки	$[S_p^6]$
	Модулі бази даних	$[S_p^7]$
	Модулі доступу до даних	$[S_p^8]$
	Комунікаційні модулі	$[S_p^9]$
Дефект g_f, де $f \in \{GP\}$	Втрата контролю над системою	$[S_g^1]$
	Вилучення критичних файлів	$[S_g^{21}]$
	Зміна критичних файлів	$[S_g^3]$
	Відсутність відповіді системи на запити	$[S_g^4]$
	Втрата доступу до функцій системи	$[S_g^5]$
	Системні збої у роботі програмного забезпечення	$[S_g^6]$
	Системні аварії у роботі програмного забезпечення	$[S_g^7]$
	Невідповідність логіки роботи програми	$[S_g^8]$

ISSN 2663 - 4023

Назва елементу	**Властивість (функціональність)**	**Позначення властивості (функціональності)**
	Витік оперативної пам'яті системи	$[S_g^9]$
	Невірне використання процесорного часу	$[S_g^{10}]$
	Порушення роботи з базами даних	$[S_g^{11}]$
	Відсутність патчів безпеки	$[S_g^{12}]$
	Некоректне оновлення програмного забезпечення	$[S_g^{13}]$
	Зміна інформації у системних журналах	$[S_g^{14}]$
	Видалення системних журналів	$[S_g^{15}]$
	Порушення роботи мережевих протоколів	$[S_g^{16}]$
	Порушення конфіденційності	$[S_g^{17}]$
	Пошкоджена цільність даних	$[S_g^{18}]$
	Порушення цільності програмного коду	$[S_g^{19}]$
	Небезпека витоку даних	$[S_g^{20}]$
	Виток даних	$[S_g^{21}]$
	Пошкодження даних	$[S_g^{22}]$
	Шифрування файлів для вимагання	$[S_g^{23}]$
Спосіб відновлення C_j де $j \in \{VP\}$	Повне переустановлення операційної системи	$[S_v^1]$
	Повне переустановлення прикладного програмного забезпечення	$[S_v^2]$
	Повне переустановлення інструментального програмного забезпечення	$[S_v^3]$
	Заміна всіх облікових даних	$[S_v^4]$
	Проведення аудиту системи	$[S_v^5]$
	Встановлення оновлень безпеки	$[S_v^6]$
	Відновлення програмного забезпечення з резервної копії	$[S_v^7]$
	Усунення пошкоджених програмних модулів	$[S_v^8]$
	Оновлення програмного забезпечення	$[S_v^9]$
	Сканування антивірусним програмним забезпеченням	$[S_v^{10}]$
	Видалення шкідливого програмного забезпечення	$[S_v^{11}]$
	Аудит мережевої активності	$[S_v^{12}]$
	Використання спеціальних утиліт для дешифрування	$[S_v^{13}]$
	Повне очищення системи	$[S_v^{14}]$
	Аудит конфігурацій програмних модулів	$[S_v^{15}]$
	Виявлення змін у коді або налаштуваннях програмних модулів	$[S_v^{16}]$
	Повернення роботи програмного забезпечення до контрольної точки	$[S_v^{17}]$
	Зміна параметрів безпеки	$[S_v^{18}]$
	Проведення тестування програмних модулів на проникнення	$[S_v^{19}]$
	Перевірка на наявність шкідливих компонентів	$[S_v^{20}]$
	Верифікація цілісності даних	$[S_v^{21}]$

На підставі відношень та властивостей елементів, зазначених у таблиці, сформулюємо технологічні умови, які дозволяють здійснювати вибір способів відновлення пошкодженого програмного забезпечення. Формалізований опис таких технологічних умов можна отримати за допомогою графа приведеного на рис. 1.

Зазначимо умови, які дозволяють визначати відношення сумісності (↔) між елементами технологічного процесу. Так, наприклад відношення сумісності між

дефектним програмним модулем P_i та способом відновлення формально можна описати наступним чином:

$$\forall P_i\ \forall C_j\ \exists S_p\ \exists S_v\ \ni P\left\{(P_i\left[S_p^7\right] = C_j[S_v^7]) \wedge (P_i\left[S_p^7\right] = C_j[S_v^{17}] \wedge (P_i\left[S_p^7\right] = C_j[S_v^{21}])\right\} \rightarrow \left[P_i\ \leftrightarrow\ C_j\right] \quad (9)$$

де: P — предикат, що характеризує істинність відношення.

Для пошкодженого програмного забезпечення серед багатьох способів відновлення необхідно визначити підмножину способів, які за своїми характеристиками спроможні відновити програмний модуль. З урахуванням графа відношень сформулюємо умови сумісності між дефектом g_f, де $f \in \{GP\}$ та способом його відновлення. У формальному вигляді це можливо записати таким чином:

$$\forall g_f\ \forall C_j\ \exists S_g\ \exists S_v\ \ni P\left\{(g_f\left[S_g^{11}\right] = C_j[S_v^7])\right\} \rightarrow \left[g_f\ \leftrightarrow\ C_j\right] \quad (10)$$

Виходячи з цього не складно визначити, що на підставі приведених умов можливо побудувати властивість транзитивності:

$$P\left(\left[P_i\ \leftrightarrow\ C_j\right] \wedge \left[g_f\ \leftrightarrow\ C_j\right]\right) \rightarrow \left[P_i\ \leftrightarrow\ g_f\right] \quad (11)$$

Дефект $g_{f_i}^1$ пошкодженого програмного модуля домінує на дефектом g_f^2 тільки тоді, коли виконується відношення домінування ($\gg$):

$$\forall g_{fx} \forall g_{fy}\ \exists\ S_{gx}\ \exists\ S_{gy}\ \ni P\left\{\left[(P(g_{fx}\left[S_{gx}\right] = 1) \wedge \left[(P\left(g_{fy}\left[S_{gy}\right] = 0\right)\right)\right]\right\} \rightarrow \left[g_{fx} \gg g_{fy}\right] \quad (12)$$

Наприклад, під час проведення діагностики пошкодженого програмного забезпечення, виявляється дефект, усунення якого потребує повного переустановлення та нового налаштування програмного забезпечення. Такий дефект буде домінувати над іншими та в подальшому впливає на послідовність проектування всього технологічного процесу відновлення.

Враховуючи властивості основних елементів технологічного процесу відновлення пошкодженого програмного забезпечення, які приведені у табл. 1, відношення домінування ($\gg$) може бути представлено таким чином:

$$\forall\ g_f^1\ \forall\ g_f^2\ \exists S_g^6\ \exists S_g^7\ \ni P\left\{\left[(P(\ g_f^1\left[S_g^7\right] = 1) \wedge \left[(P\left(g_f^2\left[S_g^6\right] = 0\right)\right)\right]\right\} \rightarrow \left[g_f^1 \gg g_f^2\right] \quad (13)$$

Два дефекти вважаються еквівалентними ($\approx$) один до одного, якщо виконується наступне відношення:

$$\forall g_{fx} \forall g_{fy} \exists S_{gx} \exists S_{gy}\ \ni P\left\{g_{fx}\left[S_{gx}\right] = g_{fy}\left[S_{gy}\right] \wedge (g_{fx} \neq g_{fy}\right\} \rightarrow g_{fx} \approx g_{fy} \quad (14)$$

Поява такого відношення вважається при наявності однакових властивостей дефектів пошкодженого програмного забезпечення.

На прикладі властивостей основних елементів технологічного процесу відновлення програмного забезпечення (таблиця 1), відношення еквівалентності ($\approx$) для дефекту буде записане таким чином:

$$\forall g_f^1 \forall g_f^2 \exists S_g^{18}\ \exists S_g^{22}\ \ni P\left\{g_f^1\left[S_g^{18}\right] = g_f^2\left[S_g^{22}\right] \wedge (g_f^1 \neq g_f^2\right\} \rightarrow g_f^1 \approx g_f^2 \quad (15)$$

Особливе місце серед технологічних умов займає відношення слідування ($\sim$), тому що воно визначає послідовність усунення дефектів, а також визначення послідовності способів відновлення. Тобто першим під час відновлення буде відновлюватися більш важливий дефект, без усунення якого працездатність програмного забезпечення неможлива, а потім інший, при якому програмне забезпечення ще спроможне працювати.

На підставі приведених властивостей (таблиця 1) технологічна послідовність усунення дефектів визначається із істинності наступних умов:

$$\forall g_f^1\ \forall g_f^2\ \exists S_g^1\ \exists\ S_g^2\ \ni P\left\{[P\ (g_f^1\left[S_g^1\right] >\ g_f^2\left[S_g^4\right])\right\} \rightarrow \left[g_f^1 \sim g_f^2\right] \quad (16)$$

Як в наслідок виникає властивості транзитивності дефектів:

$$([g_f^1 \leftrightarrow g_f^2] \wedge [g_f^2 \leftrightarrow g_f^3]) \rightarrow [g_{f_i}^1 \leftrightarrow g_f^3] \quad (17)$$

Проведений аналіз показує, що технологічні умови щодо вибору способу відновлення пошкодженого програмного забезпечення можна отримати на підставі властивості сумісності між елементами технологічного процесу, а так як умови сумісності в неявному вигляді містять умови еквівалентності, то перевірка істинності для різних елементів технологічного процесу передбачає встановлення певної направленості між ними.

Це дозволяє із урахуванням раніше сформованих умов зробити визначення, що два і болі елемента технологічного процесу відновлення пошкодженого програмного забезпечення будуть взаємодіяти (≅) в процесі вибору способу відновлення, якщо для них виконується умови сумісності технологічних параметрів.

З урахуванням цього, можна записати наступні загальні умови:

$$\forall P_z \, \forall P_i \, \forall g_f \forall C_j \, \exists S_z \, \exists S_p \, \exists S_g \exists S_v \ni P\{(P_z[S_z] \leftrightarrow C_j[S_v]) \wedge (P_i[S_p] \leftrightarrow g_f[S_v]) \wedge (g_f[S_g] \leftrightarrow C_j[S_v]) \wedge (P_i[S_p] \leftrightarrow g_f[S_g])\} \rightarrow [(P_z \cong P_i \cong g_f \cong C_j)] \quad (18)$$

Необхідно зазначити, що якщо пошкоджене програмне забезпечення відновлюється різними способами, які за своїми технологічними можливостями еквівалентні між собою, то спосіб відновлення програмного забезпечення буде вибиратися той, який домінує на іншим способом.

Символічно зазначене правило можна записати наступним чином:

$$\forall P_z \, \exists C_j^1 \, \exists C_j^2 \, \exists S_v \ni P[(C_j^1[S_v] \gg C_j^2[S_v]) \wedge (C_j^1 \approx C_j^2)] \rightarrow P_z \leftrightarrow C_j^1 \quad (19)$$

Якщо пошкоджене програмне забезпечення відновлюється різними способами, які за своїми технологічними можливостями не еквівалентні між собою, то спосіб відновлення програмного забезпечення буде включати в себе всі способи, що призначені для відновлення програмних модулів, що належать до відповідного програмного забезпечення. Виходячи з цього, дане правило можна записати наступним чином:

$$\forall P_z \, \exists C_j^1 \, \exists C_j^2 \, \exists S_v \ni P[(C_j^1[S_v] \nleftrightarrow C_j^2[S_v])] \rightarrow (P_z \leftrightarrow C_j^1) \wedge (P_z \leftrightarrow C_j^2) \quad (20)$$

Таким чином на підставі приведеного графу (рисунок 1) та визначених технологічних умов, розроблених з урахуванням логіко-алгебраїчного підходу, можна навести матрицю (табл. 2) відношень елементів технологічного процесу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак.

Таблиця 2

Матриця відношень елементів технологічного процесу відновлення пошкодженого програмного забезпечення внаслідок дії кібератак

Елемент технологічного процесу відновлення	$\{P_z\}$	$\{P_i\}$	$\{g_f\}$	$\{C_j\}$
$\{P_z\}$	≅	≅ ↔	≅ ↔	≅ ↔
$\{P_i\}$		≈	↔	↔
$\{g_f\}$		↔	≫ ≈ ~	↔
$\{C_j\}$		↔	↔	≈ ≫

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Таким чином необхідно зазначити, що формалізовані підходи щодо вибору способу відновлення пошкодженого програмного забезпечення у наслідок дії кібератак дозволяють впровадити обґрунтовані, структуровані та прозорі рішення щодо автоматизації технологічних процесів відновлення програмного забезпечення, а також забезпечити відповідність технологічного процесу відновлення обраній стратегії реагування на кіберризики в сучасному інформаційному середовищі.

Застосування практичних формальних методів, що використовуються для забезпечення відновлення пошкодженого програмного забезпечення внаслідок дії кібератак, з використанням логіко-алгебраїчних підходів, дозволив визначити основні властивості елементів, що реалізують структуру технологічного процесу відновлення пошкодженого програмного забезпечення та їх відношення.

На підставі відношень та основних властивостей елементів технологічного процесу відновлення пошкодженого програмного забезпечення сформульовані технологічні умови та формалізовані правила, які дозволяють здійснювати вибір раціонального способу відновлення дефектів пошкодженого програмного забезпечення.

Наведені формалізовані правила забезпечують побудову алгоритмів вибору раціонального способу відновлення пошкодженого програмного забезпечення у наслідок дії кібератак, а також дозволяють здійснювати розробку та впровадження програмних засобів з метою автоматизації складних процесів щодо аналізу та відновлення систем після кібератак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bartock, M., Cichonski, J., & Souppaya, M. (2018). Guide for Cybersecurity Event Recovery. *NIST Special Publication 800-184.* https://doi.org/10.6028/NIST.SP.800-184
2. Pogashiy, S.S. (2022). Models and methods of information protection in cyber-physical systems. *Ukrainian Scientific Journal of Information Security, 28(2),* 67–79.
3. Kovalenko, O.V. (2020). Models and methods for developing secure software for computer systems. (Dissertation of Doctor of Technical Sciences). Central Ukrainian National Technical University, Cherkasy.
4. Obodyak, V.K., Shelekhov, I.V., & Dovbysh, A.S. (2021). *Modern information technologies in cybersecurity: monograph*. Sumy: Sumy State University.
5. Pengyuan, L., Mengyu L., Zhang, L., Sokolsky, O., Sridhar, K., & Kong, F. (2024). Recovery from Adversarial Attacks in Cyber-physical Systems: Shallow, Deep and Exploratory Works. *ACM Comput. Surv, 1,* 1–38. https://doi.org/10.1145/3653974
6. Raffaele, R., Griffioen, P., & Sinopoli, B. (2020). Software Rejuvenation Under Persistent Attacks in Constrained Environments. *IFAC PapersOnLine, 53-2,* 4088–4094. https://www.sciencedirect.com
7. Kulik, T., Dongol, B., Gorm, P., & Schneider, S. (2022). A Survey of Practical Formal Methods for Security. Form. *Asp. Comput. 34,* 1–39. https://dl.acm.org/doi/full/10.1145/3522582
8. Ojo, C., Osoko, E., Okolo, J., & Jaji, M. (2024). Incident response: A structured model from detection to containment and recovery. *World Journal of Advanced Research and Reviews 24(1),* 1401–1407. https://doi:10.30574/wjarr.2024.24.1.3148
9. Drinkwater, K., & Sultan, S. (2022). *Integrating Cybersecurity and Disaster Recovery: A Unified Approach to Business Continuity*. https://www.researchgate.net/publication/383268245
10. Kulik, T., & Gorm, P. (2018). Towards Formal Verification of Cyber Security Standards. *Proceedings of the Institute for System Programming of RAS, 30(4),* 79–94. https:// DOI:10.15514/ISPRAS-2018-30(4)-5
11. Chow, K., Deshpande, U., Seshadri, S., & Liu, L. (2021). *SRA*: *Smart Recovery Advisor for Cyber Attacks, SIGMOD/PODS '21: International Conference on Management of Data.* Shaanxi, China: PODS.
12. Mishra, A., & Khurram, M. (2023). Security requirements specification by formal methods: a research metadata analysis. *Multimedia Tools and Applications, 83,* 41847–41866. https://link.springer.com/article/10.1007/s11042-023-17218-4

13. Sunandita Patra, S., Velazquez, A., Kang, M., & Nau, D. (2021). *Using Online Planning and Acting to Recover from Cyberattacks on Software-defined Networks, AAAI Conference on Artificial Intelligence.* Vancouver, Canada: AAAI-21.
14. Wachter, J. (2023). *Graph models for cybersecurity - a survey*. Klagenfurt, Austria: University of Klagenfurt. https://doi.org/10.48550/arXiv.2311.10050
15. Li, J., Yang, G., & Shao, Y. (2023). *Ransomware Detection Metho-Based on TextGCN, 6th International Conference on Artificial Intelligence and Big Data (ICAIBD),* Chengdu, China: MDPI Journal.
16. Wei, R., Cai, L., Aimin Yu, A., & Meng, D. (2021). DeepHunter: A Graph Neural Network Based Approach for Robust Cyber Threat Hunting. *Security and Privacy in Communication Networks,* 3–24. https://arxiv.org/abs/2104.09806
17. Rabzelj, M., Bohak, C., Južnič, L., Kos, A., & Sedlar, U. (2023). Cyberattack Graph Modeling for Visual Analytics. *IEEE Access, 1,* 1–34. https://doi.org/10.1109/access.2023.3304640
18. Vyshnivskyi, V., & Danilov I. (2023). A method for protecting a virtual cloud environment based on an attack structure graph. *Modern Information Security, 1(53),* 24–33. https://doi: 10.31673/2409-7292.2023.010003

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (28), 2025
ISSN 2663 - 4023

Yurii Dobryshyn
PhD, Associate Professor,
National Academy of the Security Service of Ukraine, Kyiv, Ukraine
ORCID ID: 0000-0003-2473-9507
ydobryshyn@gmail.com

FORMALIZATION OF THE CHOICE OF A METHOD FOR RESTORING DAMAGED SOFTWARE AS A RESULT OF CYBERATTACKS

Abstract. The article proposes a methodology for the formalized choice of a method for restoring damaged software as a result of cyberattacks. The development of the presented methodology was carried out based on an analysis of modern formal methods and tools, which include logical-algebraic approaches to the analysis and restoration of software of information and communication systems. To develop the methodological materials given in this article, scientific developments of domestic and foreign specialists on modeling and verification of the operation of various models currently used in the field of cybersecurity were used. As a tool for formalizing the choice of a method for restoring damaged software, graph models were selected, which allow visualization of possible solutions for choosing a method for restoring damaged software and also provide an opportunity to design a technological process and assign an optimal route for restoring software defects. Based on the analysis of the components of the technological process of restoring damaged software, the main elements that require research are identified and a graph of relationships is constructed, with the help of which the relationships between the elements of one technological object and the relationships between the elements of different objects belonging to the structure of the technological process of restoring damaged software are determined. Conditions are formulated that confirm the truth of the relationships determined based on the results of the task of choosing a method for restoring damaged software as a result of cyberattacks. The main properties of the elements that implement the structure of the technological process of restoring damaged software are determined. Based on the relationships and main properties of the elements, technological conditions are formulated that allow choosing a rational method for restoring damaged software as a result of cyberattacks. The formalized rules that have been developed allow building algorithms for choosing a rational method for restoring software, as well as developing and implementing software tools to automate complex processes for analyzing and restoring systems after cyberattacks.

Keywords: formalization; method of restoration; damaged software; technological conditions; relationships; properties; technological process of restoration; model; graph models.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Bartock, M., Cichonski, J., & Souppaya, M. (2018). Guide for Cybersecurity Event Recovery. *NIST Special Publication 800-184.* https://doi.org/10.6028/NIST.SP.800-184
2. Pogashiy, S.S. (2022). Models and methods of information protection in cyber-physical systems. *Ukrainian Scientific Journal of Information Security, 28(2),* 67–79.
3. Kovalenko, O.V. (2020). Models and methods for developing secure software for computer systems. (Dissertation of Doctor of Technical Sciences). Central Ukrainian National Technical University, Cherkasy.
4. Obodyak, V.K., Shelekhov, I.V., & Dovbysh, A.S. (2021). *Modern information technologies in cybersecurity: monograph*. Sumy: Sumy State University.
5. Pengyuan, L., Mengyu L., Zhang, L., Sokolsky, O., Sridhar, K., & Kong, F. (2024). Recovery from Adversarial Attacks in Cyber-physical Systems: Shallow, Deep and Exploratory Works. *ACM Comput. Surv, 1,* 1–38. https://doi.org/10.1145/3653974
6. Raffaele, R., Griffioen, P., & Sinopoli, B. (2020). Software Rejuvenation Under Persistent Attacks in Constrained Environments. *IFAC PapersOnLine, 53-2,* 4088–4094. https://www.sciencedirect.com
7. Kulik, T., Dongol, B., Gorm, P., & Schneider, S. (2022). A Survey of Practical Formal Methods for Security. Form. *Asp. Comput. 34,* 1–39. https://dl.acm.org/doi/full/10.1145/3522582

8. Ojo, C., Osoko, E., Okolo, J., & Jaji, M. (2024). Incident response: A structured model from detection to containment and recovery. *World Journal of Advanced Research and Reviews 24(1),* 1401–1407. https://doi:10.30574/wjarr.2024.24.1.3148
9. Drinkwater, K., & Sultan, S. (2022). *Integrating Cybersecurity and Disaster Recovery: A Unified Approach to Business Continuity*. https://www.researchgate.net/publication/383268245
10. Kulik, T., & Gorm, P. (2018). Towards Formal Verification of Cyber Security Standards. *Proceedings of the Institute for System Programming of RAS, 30(4),* 79–94. https:// DOI:10.15514/ISPRAS-2018-30(4)-5
11. Chow, K., Deshpande, U., Seshadri, S., & Liu, L. (2021). *SRA*: *Smart Recovery Advisor for Cyber Attacks, SIGMOD/PODS '21: International Conference on Management of Data.* Shaanxi, China: PODS.
12. Mishra, A., & Khurram, M. (2023). Security requirements specification by formal methods: a research metadata analysis. *Multimedia Tools and Applications, 83,* 41847–41866. https://link.springer.com/article/10.1007/s11042-023-17218-4
13. Sunandita Patra, S., Velazquez, A., Kang, M., & Nau, D. (2021). *Using Online Planning and Acting to Recover from Cyberattacks on Software-defined Networks, AAAI Conference on Artificial Intelligence.* Vancouver, Canada: AAAI-21.
14. Wachter, J. (2023). *Graph models for cybersecurity - a survey*. Klagenfurt, Austria: University of Klagenfurt. https://doi.org/10.48550/arXiv.2311.10050
15. Li, J., Yang, G., & Shao, Y. (2023). *Ransomware Detection Metho-Based on TextGCN, 6th International Conference on Artificial Intelligence and Big Data (ICAIBD),* Chengdu, China: MDPI Journal.
16. Wei, R., Cai, L., Aimin Yu, A., & Meng, D. (2021). DeepHunter: A Graph Neural Network Based Approach for Robust Cyber Threat Hunting. *Security and Privacy in Communication Networks,* 3–24. https://arxiv.org/abs/2104.09806
17. Rabzelj, M., Bohak, C., Južnič, L., Kos, A., & Sedlar, U. (2023). Cyberattack Graph Modeling for Visual Analytics. *IEEE Access, 1,* 1–34. https://doi.org/10.1109/access.2023.3304640
18. Vyshnivskyi, V., & Danilov I. (2023). A method for protecting a virtual cloud environment based on an attack structure graph. *Modern Information Security, 1(53),* 24–33. https://doi: 10.31673/2409-7292.2023.010003