Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Спеціальність 125 «Кібербезпека та захист інформації»

Освітня програма «Безпека інформаційних і комунікаційних систем»

«Допущено до захисту»

В.о. зав. кафедрою БІСТ

Марина ЄСІНА

_____

« » 2024 р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Аналіз питань із забезпечення функціональної безпеки сучасних інформаційних систем за рахунок впровадження принципу діверсності»

оцінка « »

Голова ЕК

Олександр ЛЕМЕШКО _____

Керівник к.т.н., доцент каф. КБІСМіТ

Малахов С. В.

Рецензент PhD, доцент кафедри інтелектуальних програмних систем і технологій

Родінко М. Ю.

Виконавець: студент групи КБ-61

Бойко К.В.

Харків – 2024

РЕФЕРАТ

Пояснювальна записка містить 72 сторінки, 13 рисунків, 3 таблиці, 2 додатка, 67 джерел.

Метою дипломної роботи є аналіз досвіду та методів імплементації принципу диверсності для вирішення питань інформаційної і функціональної безпеки при побудові, та експлуатації сучасних інформаційно-комунікаційних систем (ІКС).

Об’єкт дослідження: - підвищення рівня безпеки сучасних ІКС за рахунок широкого впровадження концепції диверсності програмного забезпечення (ПЗ).

Предмет дослідження: - напрями та способи забезпечення диверсності ПЗ, як інструменту з протидії сучасним загрозам безпеки.

Основними методами досліджень є аналіз та порівняння.

У роботі проведено аналіз динаміки виникнення інцидентів безпеки (в т.ч. аварій) через вплив різного роду кіберзагроз. Розглянуто існуючі механізмів захисту та стандартів безпеки, шляхи й способи реалізації диверсності на прикладах відомих мультиверсійних систем (в т.ч. програмних рішень). Виконано узагальнення результатів сучасних досліджень у сфері застосування принципу диверсності в програмних рішеннях та особливостей його впровадження в сучасних ІКС. Досліджено методи оцінювання рівня впровадження диверсності в сучасних ІКС. Надано рекомендації, щодо можливостей впровадження різних диверсних стратегій (напрямів) при реалізації комплексних систем захисту сучасних ІКС.

Сформульовані рекомендації можуть слугувати довідковим матеріалом для подальшої розробки нових програмних рішень, які забезпечують більшу стійкість до атак, котрі використовують спільні вразливості ІКС.

Результати роботи можуть бути використані в освітніх цілях з курсу дисципліни «Управління інформаційною безпекою» та, як довідковий матеріал

для розширення рівня професійної обізнаності персоналу підрозділів (відділів) з інформаційної безпеки (ІБ).

Ключові слова: ДИВЕРСІЙНІСТЬ, РІЗНОМАНІТНІСТЬ, ІНФОРМАЦІЙНІ СИСТЕМИ, ІНФОРМАЦІЙНА БЕЗПЕКА, ЗАГРОЗИ БЕЗПЕКИ, CCF.

ABSTRACT

Explanatory note consists of 72 pages, 13 figures, 3 tables, 2 appendices, and 67 sources.

The objective of the thesis is to analyze the experience and methods of implementing the principle of diversity to address issues of information and functional security during the development and operation of modern information and communication systems (ICS).

Research object: improving the security level of modern ICS through the broad application of the concept of software diversity. Research subject: directions and methods of ensuring software diversity as a tool to counter modern security threats.

The main research methods are analysis and comparison. The study examines the dynamics of security incident occurrences (including failures) caused by various types of cyber threats. Existing protection mechanisms and security standards are reviewed, as well as approaches and methods for implementing diversity using examples of well-known multiversion systems (including software solutions). The results of modern research on applying the principle of diversity in software solutions and its implementation features in modern ICS are summarized. Methods for evaluating the level of diversity implementation in modern ICS are investigated. Recommendations are provided regarding the feasibility of adopting various diversity strategies when implementing comprehensive security systems for modern ICS.

The formulated recommendations can serve as reference material for the further development of new software solutions that provide greater resilience to attacks exploiting common vulnerabilities in ICS. The results of the study can be utilized for educational purposes in the discipline Information Security Management and as reference material to enhance the professional awareness of personnel in information security (IS) departments.

Keywords: SABOTAGE, DIVERSITY, INFORMATION SYSTEMS, INFORMATION SECURITY, SECURITY THREATS, CSF.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ІКС	–	Інформаційно-комунікаційна система
ІС	–	Інформаційні системи
ІБ	–	Інформаційна безпека
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення
ШІ	–	Штучний інтелект (*Artificial intelligence, AI*)
ФБ	–	Функціональна безпека
CCF	–	Common Cause Failure
CROW	–	Code Replacement and Optimization Workflow
DAEDALUS	–	Defense Against firmware ROP Exploits using stochastic software Diversity
FPGA	–	Field-Programmable Gate Arrays
ISA	–	Instruction Set Architecture
ISO	–	International Organization for Standardization
IoT	–	Internet of Things
MTD	–	Moving Target Defense
MVEE	–	Multi-Variant Execution Environment
PFD	–	Probability of Failure on Demand
SARA	–	Self-Aware Reconfigurable Architecture
SIF	–	Safety Instrumented Function
SIL	–	Safety Integrity Level
SIS	–	Safety Instrumented System
STRIDE	–	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
TMR	–	Triple Modular Redundancy

ВТУП

Розвиток хмарних сервісів, Інтернету речей (IoT), мобільних платформ та інших технологій значно розширює можливості інформаційно-комунікаційних систем (ІКС). Однак цей прогрес також відкриває нові вразливості до зовнішніх і внутрішніх загроз. Забезпечення надійності та стійкості до кібератак у зв'язку з розвитком таких технологій, як *AI,* хмарні сервіси, *Mesh*- мережі, IoT та ін., вимагає впровадження нових підходів до управління ризиками. Концепція резервування є одним із шляхів запобігання несправностям у системах, але це не вирішує всі небезпечні вектори атак, особливо якщо елементи цільової системи є ідентичними, що характерно для сучасної «монокультури» ІКС. Традиційно відмовостійкість сучасних ІКС досягається завдяки певній надмірності в реалізованих програмно-апаратних рішеннях. Це підвищує захист від фізичних збоїв і спроб проведення атак, але він також має свої обмеження. Так, наприклад, все більше трапляється інцидентів, пов'язаних із загальними несправностями (причиною).

В загальному випадку, диверсійність можна описати, як реалізацію однієї функції різними способами або методами. Цей процес може включати різноманітність в апаратному та програмному забезпеченні (ПЗ) через використання різних дизайнів (*тобто, конструктивних рішень у тому числі топології*) чи платформ для вирішення проблеми з загальною причиною (*Common Cause Failure, CCF*). Зазвичай для більшої ефективності такі системи розробляються різними незалежними командами фахівців. Універсальність підходу диверсності обумовлена тим, що він може бути реалізований на будь-якому етапі проектування відповідної системи. Тому важливо дослідити та визначити ефективність методів різноманітності для підвищення надійності ІКС від збоїв із загальної причини та атак на їх спільні вразливості (як в цілому, так і їх окремих складових).

Актуальність дослідження: - впровадження принципу диверсності

програмних рішень відкриває широкі можливості для підвищення рівня функціональної безпеки сучасних ІКС та покращення захисту їх інформаційних ресурсів. Використання діверсності технологій і підходів зменшує ризики «вдалої» реалізації однотипних атак та підвищує загальну стійкість систем, наприклад проти вразливостей нульового дня *(т.з. zero-day, зеродеї)*, що є особливо важливим для захисту критичної інфраструктури та забезпечення її стабільної/штатної роботи.

В цьому контексті, головною метою роботи є дослідження сучасного досвіду імплементації принципу диверсності для вирішення завдань покращення рівня безпеки на етапах побудові і експлуатації сучасних ІКС. Для досягнення цієї мети на прикладах відомих мультиверсійних систем, виконано аналіз й узагальнення можливих шляхів і способів реалізації диверсності. Це створює необхідний базис для подальшого формулювання рекомендацій, щодо можливостей впровадження різних диверсних стратегій (напрямів) при реалізації комплексних систем захисту сучасних ІКС.

Результати роботи можуть бути застосовані фахівцями з питань ІБ для вирішення завдань підвищення функціональної безпеки сучасних ІКС шляхом широкої імплементації принципу диверсності, що дозволить зменшити їх вразливості до атак та звузити ризики відмов. В цілому отримані рекомендації можуть використовуватися, як довідковий матеріал при вирішенні завдань розробки нових програмних рішень, які забезпечують більшу стійкість до атак, що експлуатують спільні вразливості ІКС.

1 СУЧАСНІ ЗАГРОЗИ В СФЕРІ БЕЗПЕКИ СУЧАСНИХ ІКС, РОЛЬ ТА МІСЦЕ ФУНКЦІОНАЛЬНОЇ БЕЗПЕКИ

1.1 Стан проблематики сучасних загроз інформаційної безпеки та деякі особливості їх класифікації

Основою цифрової інфраструктури сучасного інформаційного суспільства є інформаційно-комунікаційні системи (ІКС) широко спектра призначення. Їх створення і використання є результатом активного розвитку й впровадження нових технологій, зокрема, хмарних сервісів, IoT, ШІ, мобільних платформ, котрі значно розширюють можливості діючих ІКС. Однак, широке впровадження нових технологій, нажаль, відкриває шлях для експлуатації вразливостей до зовнішніх і внутрішніх загроз [1]. Загрози безпеки в сучасних ІКС виникають через вразливості їх інфраструктури (в т.ч. топології) та окремих складових, зокрема недоліки фізичного доступу чи програмні помилки, що можуть призводити до атак, поки вони не будуть усунуті [1]. Вирішення питань ІБ в ІКС, передусім, передбачає використання таких комплексних заходів, як постійний моніторинг мережевих взаємодій для виявлення аномалій, періодичний аудит, програмно-апаратне резервування, розподіл повноважень і сегментація ресурсів ІКС, управління доступом та шифрування даних [2-4].

Надійний захист має гарантувати доступність, конфіденційність і цілісність інформації, а також відновлення стану після можливих атак чи збоїв. Підготовка персоналу та організація чіткого процесу реагування на загрози ІБ, що є важливою організаційною складовою захисту, включає розробку й удосконалення відповідних протоколів реагування, щодо можливих загроз [3].

Класифікація загроз допомагає визначити і зрозуміти характеристики і джерела потенційних загроз і визначає ризики безпеки, які загрожують цим системам. Від цього залежить вибір найбільш доцільних заходів безпеки. Серед фахівців з питань ІБ, відомі кілька моделей класифікації загроз [2-4]:

1) Трьохвимірна модель – класифікація за мотивацією, локалізацією та агентом загрози;

2) Гібридна модель (С3) – класифікація за частотою, областю впливу та джерелом загроз;

3) Пірамідальна модель – класифікація за рівнем знань нападника, критичністю області та розміром (наслідками) втрат;

4) STRIDE – класифікація загроз за цілями атаки, наприклад, підробка ідентичності чи розголошення інформації (в т.ч. інсайд);

5) ISO модель – класифікація за впливом, зокрема, руйнування, зміна (модифікація) чи втрата інформації.

В цих класифікаціях можна виділити два ключові підходи, які є широко визнаними в літературі, а саме класифікації на основі технік атак, котра фокусується на методах що використовують зловмисники та класифікації на основі впливу загроз, яка оцінює можливі наслідки для ІКС [1].

Згідно з парадигмою подібної класифікації, загрози конфіденційності виникають, коли сторонні особи отримують несанкціонований доступ до захищеної інформації [1, 5]. Загрози цілісності даних більшою мірою пов'язані із збереженням чи використанням інформації без дозволу, що може бути наслідком як навмисних дій, так і технічних несправностей. Загрози доступності виникають, коли доступ до інформації чи послуг стає неможливим або ускладненим, що перешкоджає виконанню завдань.

Загрози ІБ поділяються на зовнішні загрози, які включають найбільш поширені кібератаки, шкідливе ПЗ, фішинг і таргетовані атаки, які здебільшого спрямовані на доступ до чутливих даних або нанесення шкоди ІКС, та внутрішні загрози, котрі пов’язані з діями співробітників, які мають безпосередній доступ до систем, зокрема недбалість чи зловмисні дії інсайдерів [1,5-8]. На рисунку 1.1 показано класифікацію загроз, яка включає 7 основних категорій, котрі охоплюють різні критерії. Внутрішні загрози можна передбачити та знешкодити, але вони можуть проявлятися в різних формах (проявах).

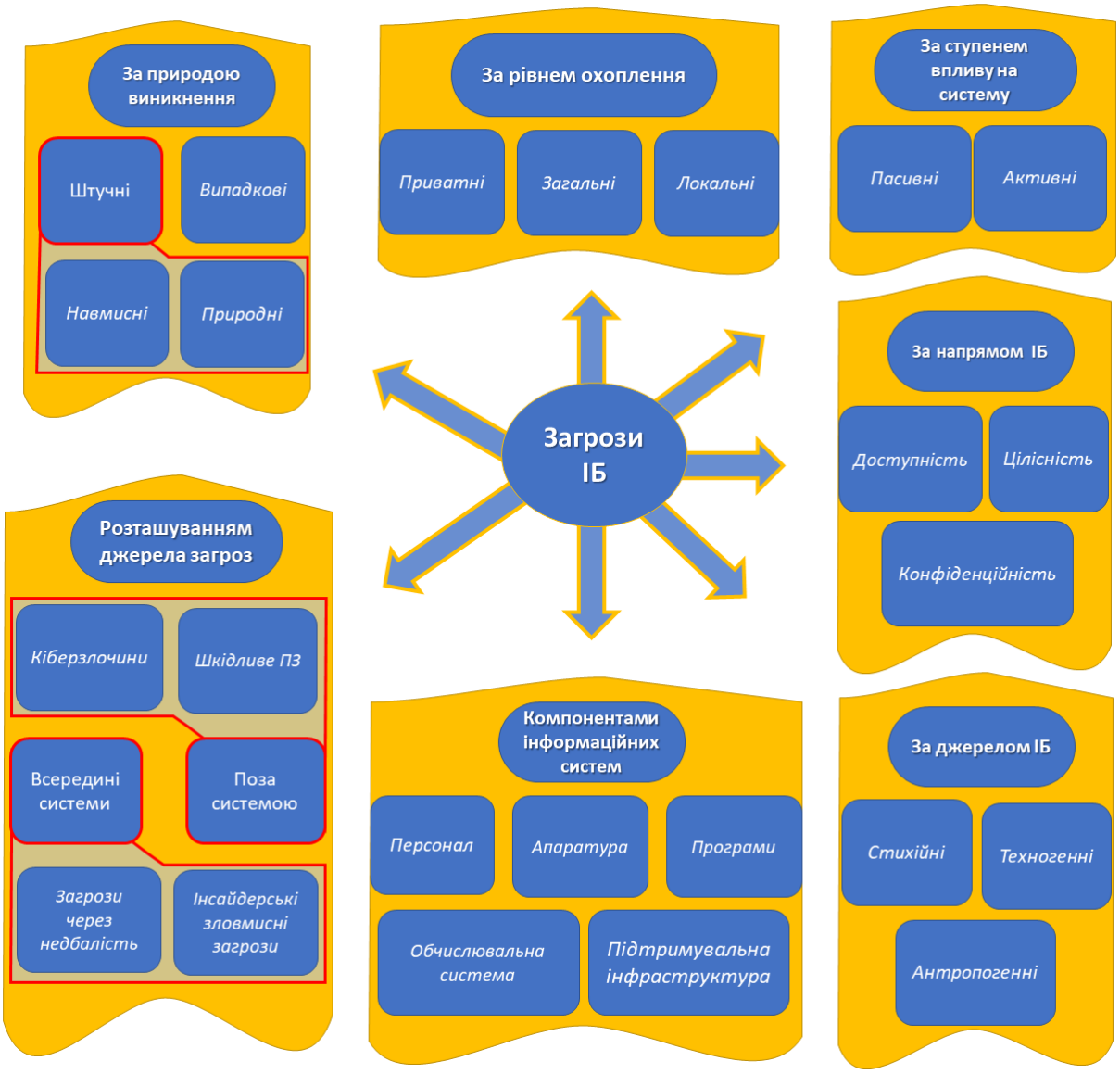


Рисунок 1.1 – Класифікація загроз ІБ (*за даними* [1,5-8]).

Серед таких загроз, це перш за все, помилки персоналу, ненавмисне видалення даних чи неправильно налаштовані системи безпеки. Таблиця 1.1 відображає деякі субстантивні особливості загроз ІБ за трьома показниками: - мета, рівень впливу/втручання та наслідки. Також трапляються навмисні загрози, як-от інсайдерські атаки, коли співробітники використовують свій доступ до інформації для особистої вигоди, та соціальна інженерія, що включає фішингові атаки й підкуп персоналу [6-8].

Таблиця 1.1 – Особливості активних і пасивних загроз (узагальнення даних [9])

Різновид атаки	Мета	Рівень втручання	Вплив
Пасивні атаки	Отримання несанкціонованого доступу (НСД) до інформації або даних без створення видимих змін у системі чи її роботі. Основна мета таких атак – збір інформації чи моніторинг конфіденційних даних.	Мають низький рівень втручання в систему. Зловмисники намагаються залишитися непоміченими під час перехоплення та аналізу даних.	Зазвичай не призводять до негайних змін у системі, однак зібрана інформація може бути використана для подальших більш складніших атак.
Активні атаки	Втручання в систему з метою її порушення, модифікації або маніпуляції даними чи комунікаціями. Зловмисники прагнуть завдати системі видимої шкоди, отримати несанкціонований доступ або порушити цілісність і доступність даних.	Включають більш агресивні методи втручання в систему. Нападники активно взаємодіють із ціллю, впроваджуючи шкідливі коди або втручаючись у процес передачу даних.	Можуть викликати негайні наслідки. Наприклад: - зупинка роботи ІКС, втрата даних, фінансові шахрайства, НСД до чутливих даних чи функцій управління.

Техногенні загрози виникають через несправності обладнання, наприклад, поломку жорстких дисків чи збої мережевого обладнання. Класифікація загроз допомагає краще розуміти ризики й обирати оптимальні заходи, що підвищує безпеку інформаційної системи.

1.1.1 Особливості стандартизації питань управління ризиками ІБ

В цілому існує кілька десятків різного роду методик і підходів до оцінки ризиків ІБ, таких як: - Austrian IT Security Handbook, AS / NZS4360, BSI 100-3, CRISAM, EBIOS, HB167: 200X, ISF IRAM, CRAMM, ISO 27005, MAGERIT, MARION, MEHARI, NIST SP800-30, OCTAVE, OSSTMMRAV, SOMAP та інші. Частина з них вже застаріла і не розвивається, частина не володіє актуальними перекладами на англійську мову з мови країни походження, що робить складним їх вивчення для широкої аудиторії [10].

ISO 27005 – це стандарт, що визначає процес управління ризиками інформаційної безпеки. Він декларує, як оцінювати, обробляти, приймати, контролювати та переглядати ризики. В даному контексті «ризик» – це відхилення від очікуваних результатів, яке може бути як позитивним, так і

негативним. Стандарт пропонує ідентифікувати активи, загрози, вразливості та можливі наслідки, оцінювати ризики за якісними чи кількісними показниками. Основними етапами є визначення обставин, аналіз ризиків та їхнє зіставлення з критеріями впливу та прийняття [11]. Є чотири варіанти дій щодо ризиків: змінити, прийняти, усунути або розподілити їх. Ці кроки супроводжуються постійним моніторингом та переглядом, обговоренням і консультаціями, що забезпечує безперервність процесу управління ризиками [11].

NIST SP800-30 — це стандарт, що описує процес управління ризиками ІБ на різних рівнях організації, від стратегічного до технічного рівня окремих інформаційних систем [12]. Стандарт базується на оцінці двох параметрів: потенційного збитку та ймовірності виникнення загрози. Ця оцінка ризиків відбувається за трирівневою шкалою і допомагає організаціям визначити рівень ризику, щоб вжити відповідних заходів для його зменшення. Процес управління ризиками зображений на рис. 1.2 й включає [12]: - визначення важливих параметрів ІТ- системи, виявлення потенційних загроз і слабких місць системи, оцінка поточних заходів безпеки, визначення ймовірності виникнення загроз і аналіз їх можливих наслідків, визначення рівня ризику на основі вищезгаданих параметрів, вибір заходів для зниження ризиків до прийнятного рівня та фіксація всіх отриманих висновків та дій.

COBIT 5 (*Control Objectives for Information and Related Technologies*) — це фреймворк для управління інформаційними технологіями, що надає інструменти для управління ризиками. Основні етапи оцінки ризиків включають визначення всіх критично важливих активів (ресурсів в широкому розумінні) і процесів, аналіз можливих загроз і вразливостей, аналіз існуючих заходів безпеки та їхньої ефективності, оцінка ризиків на основі ймовірності виникнення загрози та її потенційного впливу. Завершальними етапами є формування оцінки ризику для визначення критичності загроз та вибір варіантів: прийняття, усунення або зменшення ризиків.

Рисунок 1.2 – Загальна схема оцінки ризику за NIST SP800-30
(за узагальненням відомостей [12])

1.2 Функціональна безпека та її складові

Функціональна безпека (*Functional Safety*) спрямована на мінімізацію ризиків, що можуть виникнути внаслідок випадкових технічних збоїв і загроз за допомогою правильної роботи логічних контролерів, сенсорів і виконавчих механізмів [13-14]. Національні регулятори у сфері ядерної безпеки, такі як *Stralsakerhetsmyndigheten*, рекомендують інтегрувати дизайн, верифікацію та валідацію систем із врахуванням кібербезпеки протягом всього життєвого циклу [15]. Безпекові системи SIS *(Safety Integrity Level)* розроблені для автоматичного переведення процесу в безпечний режим у випадку загрозливої події. Кожна SIS включає функції безпеки SIF (*Safety Instrumented Function*),

кожна з яких має власний рівень SIL, що вказує на ймовірність відмови при активації PFD (*Probability of Failure on Demand*) [16]. Показник повноти безпеки (Safety Integrity Level, SIL) описує здатність системи виконувати безпечні функції, знижуючи ризики до необхідного рівня. Стандарт IEC 61508 визначає чотири рівні SIL (від SIL 1 до SIL 4), де SIL 4 є найвищим і відповідає найбільш жорстким вимогам, тоді як SIL 1 — найнижчим (див. рис. 1.3). Чим серйозніші ризики, пов'язані з процесами чи обладнанням, тим більш жорсткими є вимоги до необхідного рівня SIL, який гарантує безпеку для людей, майна та громадськості [13].

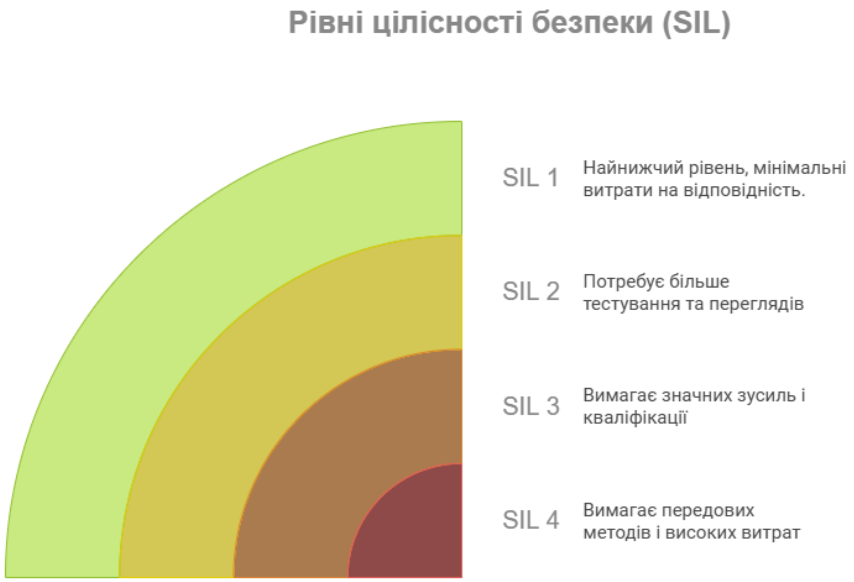


Рисунок 1.3 – Вкладеність рівнів SIL

Для визначення рівня повноти безпеки потрібно враховувати інтенсивність відмов та частку безпечних відмов. Безпечні відмови призводять до зупинки технологічного процесу без шкоди, а небезпечні — до втрати функціональної безпеки або безпечного стану системи [13]. Нижче, в таблиці 1.2 зазначені рівні повноти безпеки для різних режимів запитів. Відмови можуть бути випадковими, наприклад, через зношування компонентів, або систематичними, коли вони викликані дефектами на етапі проектування або непередбаченою взаємодією з іншими системами. На рис. 1.4 зображено візуальна інтерпретація варіанту класифікації відмов.

Таблиця 1.2 – Рівні повноти безпеки для різних режимів запитів

Рівень повноти безпеки (SIL)	Середня імовірність небезпечної відмови функції безпеки за вимогою	Середня частота небезпечної відмови функції безпеки за вимогою
1	$\geq 10^{-9}\ to\ < 10^{-8}$	$\geq 10^{-5}\ to\ < 10^{-4}$
2	$\geq 10^{-8}\ to\ < 10^{-7}$	$\geq 10^{-4}\ to\ < 10^{-3}$
3	$\geq 10^{-7}\ to\ < 10^{-6}$	$\geq 10^{-3}\ to\ < 10^{-2}$
4	$\geq 10^{-6}\ to\ < 10^{-5}$	$\geq 10^{-2}\ to\ < 10^{-1}$

Прим: - за даними [17]

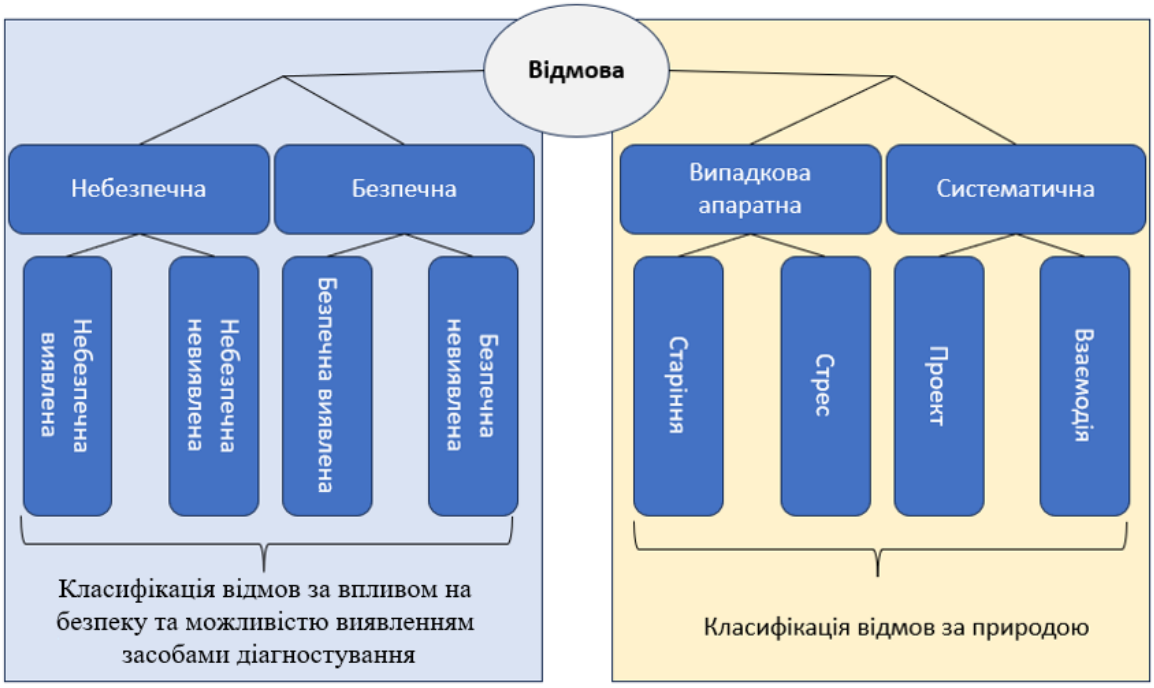


Рисунок 1.4 – Класифікація відмов (за даними [14])

На кожному етапі життєвого циклу у стандарті IEC 61508 визначені завдання й вимоги, яких слід дотримуватися а також необхідно здійснювати ретельний аналіз потенційних небезпек і ризиків. На цих етапах також мають бути створені детальні плани експлуатації, обслуговування, встановлення та перевірки безпеки продукту [14,18-19]. Спрощену версію життєвого циклу наведено на рисунку 1.5. Загальний життєвий цикл відповідно до стандарту IEC 61508 наведений в Додатку А.

Рисунок 1.5 – Етапи життєвого циклу безпеки (за даними [15])

Єтап реалізації продукту відокремлює впровадження апаратного та програмного забезпечення і розбиває його на кілька підетапів/шагів [18]:

- Розробка комплексної специфікації функцій безпеки продукту, присвоєння цим функціям рівня SIL та визначення заходів для зниження ризиків, пов'язаних з ними.
- Створення плану, що описує, як система буде перевірятися на відповідність визначеним вимогам безпеки.
- Розробка та впровадження необхідного критично важливого для безпеки програмного та апаратного забезпечення
- Об'єднання розроблених програмних і апаратних підсистем у повнофункціональний продукт
- Забезпечення правильної та безпечної роботи розробленого програмного та апаратного забезпечення, виключаючи будь-які модифікації системи.

1.2.1 Приклад засобів оцінки рівню функціональної безпеки та основні складові плану управління функціональної безпеки

AXMEA є спеціалізованим програмним продуктом, розробленим для підвищення рівня функціональної безпеки у складних технічних системах.

Створене фахівцями Національного аерокосмічного університету ім. М.Є. Жуковського «ХАІ» ПЗ, забезпечує оригінальний підхід до оцінки ризиків та управління безпекою, відповідаючи вимогам міжнародних стандартів [17]. Цей інструмент дозволяє проводити поглиблену оцінку можливих загроз чим знижуючи рівень небезпеки до мінімального допустимого рівня. AXMEA надає інструменти комплексної верифікації, валідації а також автоматично генерує деталізовану документацію, яка відображає усі кроки розробки та прийняті рішення, що спрощує управління і підвищує прозорість процесу. Його інтеграція з іншими системами, які використовуються в розробці, дозволяє легко взаємодіяти з іншим програмним забезпеченням, що сприяє побудові єдиної «екосистеми» для управління функціональною безпекою [17].

Типова структура плану управління функціональної безпеки (ФБ) включає різні елементи, з яких більш масштабні вимоги пов'язані із розробкою окремих планів, що включають такі процеси, як: - управління кадрами, управління конфігурацією, вибір і оцінка інструментальних засобів, верифікація (*підтвердження відповідності*), валідація (*підтвердження того, що вимоги задоволені*), трасування вимог (*тобто єдиний задум*), управління документацією та оцінка ФБ. Другу категорію складають «менш» значимі вимоги, які можуть бути реалізовані в рамках загального плану управління ФБ. Рисунок 1.6 уточнює структуру плану управління ФБ, поєднує основні компоненти та висвітлює їх взаємозв'язки у процесі забезпечення ФБ (*для умовного проекту*).

Важливим аспектом є також збереження окремого звіту про трасування вимог (тобто їх концептуальний взаємозв’язок), який завершується лише на фінальному етапі умовного проекту. План повинен містити розділ «Політика та стратегія проєкту» де визначені загальні положення проекту та основні положення плану управління. Це також включає життєвий цикл функціональної безпеки, основні рекомендації для користувача щодо безпечної експлуатації системи та розділ який присвячений захисту інформації (тобто питанням ІБ).

Самостійні управлінські процеси, що потребують розробки окремих планів	Імплементація вимог відбувається в рамках плану управління функціональною безпекою
• Управління персоналом (*Human Resource Management*); • Управління конфігурацією (*Configuration Management*); • Вибір і оцінювання інструментальних засобів розроблення (*Tools Selection and Evaluation*); • Верифікація та валідація (*Verification and Validation*); • Трасування вимог (*Requirements Tracing*); • Управління документацією (*Documentation Management*); • Оцінювання функційної безпеки (*Functional Safety Assessment*).	• Політика і стратегія проєкту (*Project Policy and Strategy*); • Управління проєктом (*Project Management*); • Система менеджменту якості (*Quality Management System*); • Життєвий цикл функціональної безпеки (*Functional Safety Life Cycle*); • Керівництво користувача з безпеки (*Product Safety Manual*); • Інформаційна безпека (*Information Security*)

Звіт з трасування вимог

• Політика та стратегія проєкту; • Життєвий цикл функціональної безпеки;	• Управління проєктом; • Система менеджменту якості;	• Інструкція з безпеки продукту; • Інформаційна безпека.

Рисунок 1.6 – Характерна структура плану управління ФБ

1.3 Узагальнення динаміки ризиків виникнення аварій та інцидентів з безпеки, внаслідок впливу кіберзагроз

Зростання масштабів впливу різних кіберзагроз у світі набуває все більшої актуальності, оскільки сучасні інформаційні технології (IT) не тільки дають нові можливості, але й створюють додаткові ризики [20]. Кіберінциденти, які включають атаки з використанням програм-вимагачів, витоки даних, а також збої в роботі IT-систем, вже неодноразово потрапляли до списку основних загроз, згідно з даними *Allianz Risk Barometer*. Вперше кіберзагрози стали головним ризиком для компаній різних розмірів та сфер діяльності [20]. У 2024 році в звіті Outlook, 81% керівників компаній відзначили, що вразливість їхніх IT систем значно зросла порівняно з попередніми роками, а 94% вважають, що їхні організації недостатньо захищені від кіберзлочинів. Офіційно, 29% організацій повідомили, що за останні 12 місяців вони зазнали серйозного впливу кіберінцидентів різного типу [21]. Так, наприклад, за 2022-2023 роки зафіксовано

значне зростання кількості атак (див. табл. 1.3) [22]. Наприклад, атака на Коста-Рику спричинила збитки в 1,3 млрд. доларів, тоді як інші атаки, такі як злом *«Axie Infinity»* та *«Binance»*, «коштували» понад 500 млн. доларів кожна [22]. При цьому, саме підвищена залежність від цифрових технологій у багатьох аспектах стала основною причиною того, що кібербезпека вийшла на перший план.

Таблиця 1.3 – Наслідки атак

Рік	**Жертва кібератаки**	**Наслідки**
2022	Коста-Рика	Збитки становлять 1,3 мільярда доларів США (30 млн USD на день за півтора місяця).
	Axie Infinity (Ronin Network)	Викрадено 600 млн доларів США.
	Binance	Викрадено 570 млн доларів США.
	Wormhole token bridge	Викрадено 321 млн доларів США.
	Common Spirit Health	Збитки становлять 160 мільйонів доларів США.
	Medibank	Збитки компанії за 2022-2023 рр. становлять 46,4 млн USD, прогнозовані збитки – 80 млн USD.
	Neopet	Викрадено особисту інформацію 69 млн людей.
	Twitter	Викрадено особисту інформацію про 5,4 мільйона користувачів.
	Advocate Aurora Health	Викрадено особисту інформацію 3 млн пацієнтів.
	Optus	Розкрито ідентифікаційні номери 2,1 млн клієнтів.
	Червоний Хрест	Викрадено особисту інформацію про 515 тис людей, включаючи їхню локацію.
	Міністерство національної оборони Португалії	Викрадено конфіденційні документи НАТО.
	Сайти аеропортів	Скомпрометовано сайти 23 аеропортів США, Японії, Естонії та Литви.
2023	MOVEit	Постраждало 2393 організації та орієнтовно від 69 до 73,8 млн людей.
	DarkBeam	Викрадено понад 3,8 мільярда електронних скриньок із паролями.
	23andMe	Викрадено потенційно мільйони наборів даних ДНК (точна кількість не визначена).
	Johnson Controls International	Викрадено понад 27 ТБ корпоративних даних і зашифровано віртуальні машини VMware ESXi.
	Barts Health NHS Data Breach	Витік 7 ТБ даних.
	Tampa General Hospital	Викрадено медичні дані 1,2 млн пацієнтів.

Як зазначають фахівці з ІБ компанії *Allianz Commercial*, ця тенденція є наслідком зростаючої залежності економіки та підприємств від цифрових послуг і інфраструктури. Оскільки майже кожен аспект бізнес-діяльності тісно пов'язаний із ІТ, то підключення до мереж надає широкій спектр можливостей для хакерів [20].

Також, найбільше занепокоєння викликає витік даних, за ним ідуть кібератаки на критично важливу інфраструктуру та фізичні активи, а також збільшення кількості атак з вимогами викупу [20]. Більш того, використання штучного інтелекту (ШІ) для автоматизації атак, таких як створення нових шкідливих програм і фішин, призводить до того, що навіть менш досвідчені зловмисники можуть здійснювати більш масштабні та складні атаки [20]. В цьому контексті, залучення можливостей ШІ на кшталт *ChatGPT*, здатне сприяти атакуючої стороні, значно швидше створювати нові варіації злочинного програмного забезпечення (ПЗ), що збільшує частоту та інтенсивність атак. Так, наприклад, у 2023 спостерігалася рекордна кількість атак програм-вимагачів та витоків даних. Витоки інформації, наприклад такі як із *Johnson Controls International* (27 ТБ корпоративних даних), не лише завдають фінансових збитків, а й порушують штатне функціонування їх ІКС. При цьому, зростання числа атак на 10% порівняно з 2022 роком свідчить про підвищену активність кіберзлочинців. Водночас багато атак залишаються нерозкритими, що ускладнює оцінку реального масштабу загроз [22]. Таким чином, питання протидії сучасним кіберзагрозам вимагає не лише технологічних інвестицій, а й стратегічного планування всіх заходів. 1

1.4 Узагальнення відмов програмних систем через загальні причини

Резервування, ключова стратегія досягнення високої надійності системи, передбачає інтеграцію надлишкових елементів у конструкцію системи. Забезпечення незалежності цих надлишкових компонентів є критично важливим, оскільки взаємозалежність під час проектування, виробництва або впливу зовнішнього середовища створює ризик одночасного виходу з ладу декількох

компонентів, незалежно від типу надмірності, що використовується [23]. Більше того, багато критично важливих для надійності систем за своєю суттю передбачають резервування, щоб обмежити вплив відмов окремих компонентів на катастрофічні події. Включення надлишкової потужності в ці системи має на меті пом'якшити потенційні наслідки відмови окремого компонента, що відповідає вимогам стандарту IEC 61508, згідно з яким кожна функція системи повинна витримувати щонайменше одну апаратну відмову [23]. Наявність CCF значно підвищує ймовірність спільної відмови, що суттєво впливає на загальну ненадійність системи [23-24]. Врахування впливу CCFs при моделюванні та оцінці надійності системи є обов'язковим. CCF можуть бути спровоковані зовнішніми факторами, такими як атаки, віруси, людські помилки або екстремальні умови навколишнього середовища. Крім того, CCF можуть виникати внаслідок поширених збоїв у системі, коли несправність одного компонента призводить до відмови інших [24]. Вони часто вносять найбільший внесок до рівня ризику, і тому завжди повинні бути ретельно розглянуті.

Відмови із загальних причин є важливою складовою, оскільки вони часто мають високу ймовірність виникнення, порівняно з комбінацією випадкових незалежних відмов окремих компонентів/підсистем/систем [25]. Визначення *Common Cause Failure* (CCF) пройшло кілька етапів уточнень протягом десятиліть і представлено різними стандартами й організаціями. Сюди можна віднести, Міжнародну електротехнічну комісію (IEC), Комісію з ядерного регулювання США (*NUREG*), Міжнародну організацію зі стандартизації (ISO), *Rausand* (який об'єднав два визначення з ядерної промисловості та IEC 61508) та інші [25-27]. Різні визначення CCF мають певні розходження, наприклад, вимога одночасності відмов за стандартом IEC 61508 [28], порівняно з можливістю короткого проміжку часу між відмовами за NUREG та ISO/TR 12489 [29]. PDS (Hauge et.al. 2013) [30] не чітко визначає одночасність, але використовує модифікаційні фактори для врахування часової залежності між відмовами, що може впливати на оцінки ймовірності CCF. Відзначається, що ключові критерії визначення CCF включають неможливість функціонування компонентів та

наявність множинних відмов у конфігураціях з резервуванням. Однак визначення CCF може різнитися в залежності від галузі, типу системи, мети аналізу та інших факторів. Так наприклад, згідно з думкою *Smith & Watson*, не існує універсального визначення CCF, яке б було прийнятим у всіх сферах діяльності сучасних ІКС. Відповідно, це підкреслює необхідність уточнення визначення CCF для конкретного застосування [27].

Однак, аналізуючи різні визначення CCF, можна виділити чотири основні аспекти, які є спільними для більшості з них:

- Наявність кількох відмов
- Вплив відмов на функціонування системи
- Обмеження часу між відмовами
- Наявність спільної причини відмов

Ці аспекти можуть бути використані як критерії для ідентифікації та класифікації CCF у різних контекстах. Так, відмови за загальною причиною є важливою частиною будь-якої моделі надійності або небезпеки, оскільки вони зводять нанівець поліпшення, пропоновані резервуванням. Вони часто вносять найбільший внесок у рівень ризику, і тому завжди повинні бути ретельно розглянуті. Існує багато методів системного аналізу, які пропонують способи врахування збоїв з загальних причин. Однак ці методи, як правило, прості, наприклад, беруть відсоток відмов компонентів і відносять їх до загальних причин. При цьому часто робляться певні припущення, наприклад, що всі компоненти мають однакові загальні причини або однакову частоту та розподіл відмов. Наприклад, модель бета-фактора є дуже поширеним методом, який можна знайти в таких стандартах, як IEC 61508. Щоб розрахувати частоту відмов через загальні причини, бета-фактор множиться на частоту відмов компонента. Таким чином, бета-фактор просто представляє відсоток відмов компонентів, які відбуваються через загальні причини [31].

Відмови можна розділити на дві категорії: - випадкові і систематичні (див. рис. 1.7).

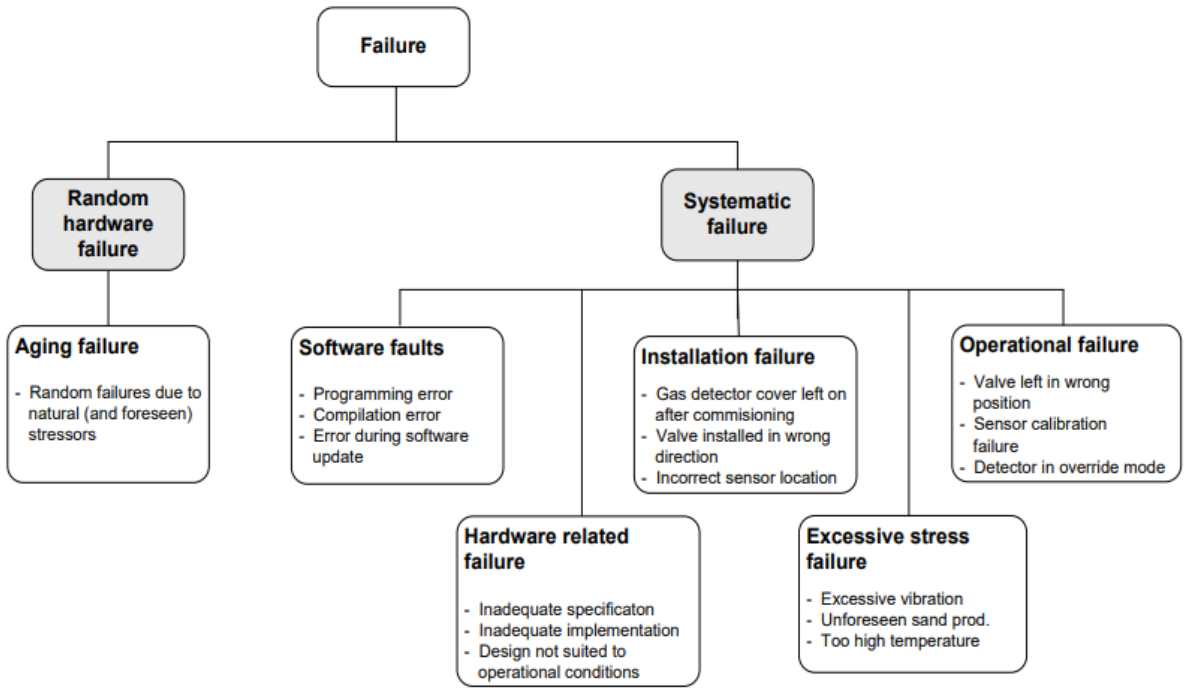


Рисунок 1.7 – Класифікація відмов за методом PDS (за даними [27])

Для забезпечення безпеки при проектуванні засобів контролю необхідно враховувати всі ці види відмов і мінімізувати їх. Випадкові відмови (*Random failure*) - це ті, що стаються непередбачувано в різні моменти часу і можуть бути спровоковані перевантаженням, дефектами виробництва або впливом середовища. Їх можна знизити, використовуючи компоненти з високою надійністю. Систематичні відмови (*Systematic failures*) - це ті, що пов'язані з помилками з загальних причин у процесах проектування, специфікації, експлуатації, обслуговування та монтажу [32].

Це не випадкові помилки, які складно статистично аналізувати, і вони можуть зачіпати резервні системи, виникаючи при певних умовах. До них належать помилки програмного та апаратного забезпечення, некоректна концепція проекту, проблеми з інтерфейсом, неправильне виконання завдань та обробка ситуацій безпеки.

Для розуміння причин та механізмів виникнення збоїв з загальною причиною, необхідно розрізняти два ключові атрибути: - «root causes» (*кореневі причини*) та «coupling factors» (*фактори зв'язку*).

Root causes - це базові причини відмов компонентів, які, якщо виправити або уникнути, можуть запобігти повторенню відмов. Збої з кореневою причиною можуть виникати внаслідок миттєвого впливу (наприклад, удар) або через збільшення тривалості небажаної ситуації (наприклад, корозія). [27].

Root causes включають в себе такі фактори, як [27]:

- Помилки проектування: - недостатнє або невірне визначення вимог, специфікацій, стандартів, методів аналізу.
- Помилки виробництва: - недотримання якості, дефекти матеріалів, неправильна збірка, неконтрольовані зміни.
- Помилки впровадження: - *хибна (нештатна)* інсталяція ПЗ, налаштування, тестування, документація.
- Помилки експлуатації та обслуговування: - неправильна робота, недостатнє або невірне обслуговування, невідповідність процедурам, недостатній контроль.
- Антропогенні помилки: - помилки ухвалення рішень персоналом ІКС, помилки виконання, помилки сприйняття, помилки пам’яті, помилки комунікації.
- Помилки ПЗ: - помилки кодування, помилки логіки, помилки сумісності, помилки взаємодії.

Coupling factors - це характеристики групи компонентів, які роблять їх вразливими до спільних механізмів відмов. Прикладами coupling factors можуть бути [26][33]:

- Використання однакового принципу проектування Same design (principles).
- Використання однакового обладнання або програмного забезпечення *Same hardwar*.
- Використання одного і того ж операційного або технічного персоналу.

- Використання однакових інтерфейсів (персонал-машина чи системно-системні) *Same interfaces*: або подібних засобів взаємодії, які мають однакові або подібні формати, символи, коди тощо
- Використання однакового середовища або місця розташування *Same environment / physical location.*

Базова причина, така як помилка ПЗ, у її поєднанні з фактором зв'язку, таким як, однакове ПЗ на кількох серверах, може призвести до відмови всіх серверів одночасно. Це може мати серйозні наслідки для безпеки, продуктивності та доступності ІКС зокрема, вплив "*Root causes*" та "*Coupling factors*" на відмови окремих компонентів ІКС та систем в цілому [34].

Common cmode failures. Загальнорежимний збій - це специфічний тип збою ІКС, коли кілька її підсистем виходять з ладу однаковим чином з однієї причини. При цьому, відмови можуть виникати в різний час, а загальною причиною може бути дефект конструкції чи повторювана подія. Це тип відмов притаманний до автономних запасних частин/складових. Подібні визначення є в декількох європейських джерелах (наприклад, Borcsok et al., R&M)[33].

Independent failures. Незалежна відмова виникає, коли компонент системи виходить з ладу без впливу інших частин, часто через вроджені дефекти або зовнішні фактори. Розуміння цих відмов має вирішальне значення для підвищення надійності та безпеки. Як розпізнавання незалежних відмов може вплинути на те, як ми проектуємо та взаємодіємо з технологіями навколо нас? Приєднуйтесь до дискусії та діліться своїми думками [35].

Dependent failures. Залежна відмова виникає, коли ймовірність відмови двох або більше систем більша, ніж добуток ймовірностей відмови кожної з них. Залежні відмови можуть мати одну причину, кілька непов'язаних причин, або не мати жодної причини.

Cascading failures. Каскадні відмови, які є поширюваними (прогресуючими) відмовами, тобто режим відмови одного або декількох компонентів, що призводить до інших умов експлуатації, навколишнього середовища і т.д., таким чином, що інші компоненти виходять з ладу [27].

Каскадні відмови є підвидами залежних відмов, які враховують взаємозв'язок між компонентами системи. Залежність може бути пов'язана з дизайном, середовищем або експлуатацією системи. Каскадні відмови важко виявити, крім того деякі каскадні відмови можуть бути помилково прийняті за CCF (*спільні причини відмов*) через недостатність та/чи неточність інформації про причини відмов [36]. Як вже було зазначено вище, CCF - це відмови, які виникають через спільну причину, яка впливає на декілька компонентів одночасно або протягом короткого проміжку часу. Таким чином, CCF мають кореневу причину, котра полягає у спільній вразливості декількох компонентів ІКС. Фактори взаємозв'язку між окремими компонентами можуть пояснити, чому декілька компонентів зазнають пошкоджень від однієї спільної небезпечної події, наприклад, низької температури, сильного снігопаду або відмови електропостачання [36].

Каскадні відмови відрізняються від CCF тим, що вони виникають через послідовну реакцію, яка починається з відмови одного компонента і поширюється на інші через залежності між ними. Каскадні відмови залежать від попередніх відмов, а не від спільної причини. Фактори взаємозв'язку між компонентами можуть пояснити, чому декілька компонентів постраждають від несправностей пов'язаних компонентів. Наприклад, якщо одна обробна одиниця вийшла з ладу, це може збільшити навантаження на іншу обробну одиницю і підвищити ймовірність її відмови [36].

Щоб відрізнити ці методи, можна звернути увагу на такі аспекти [36]:

- CCFs виникають через спільну причину, яка впливає на декілька компонентів одночасно або протягом короткого проміжку часу.
- CCFs є першими в ланцюжку відмов, які безпосередньо пов'язані з причиною відмови.
- CCFs мають скінченні наслідки, оскільки вони обмежені кількістю компонентів, які можуть бути піддані спільній причині.

Coupling effects - це термін, який використовується у [37], щоб охопити ключові концепції, які відрізняють каскадні відмови від інших видів відмов.

Coupling effects включають в себе дві підкатегорії: – «coupling factors» та «coupling paths». *Coupling paths* - це маршрути, по яких *coupling factors* передаються від одного компонента до іншого, викликаючи відмови в різних частинах системи. При моделюванні каскадних відмов розглядаються різні методи, наприклад такі як, аналіз топології, ймовірнісна оцінка ризиків, оптимізація обслуговування та аналіз надійності. При цьому аналіз надійності слід розглядати, як загальний метод для вимірювання продуктивності системи, що враховує вплив «*coupling effects»* на ефективність системи в умовах існування каскадних відмов [16].

Існує різноманіття методів аналізу збоїв, враховуючи мету і масштаб дослідження. Один з підходів - це аналіз першопричин, систематичний процес виявлення та запобігання основним причинам відмов, зокрема і факторам зв'язку загальних відмов [38]. Інший метод - використання блок-схем надійності для графічного відображення структури системи та режимів відмов компонентів, з розрахунком ймовірності відмови системи на вимогу. Це дозволяє включати відмови за загальними причинами, створюючи явну модель. На рисунку 1.8 наведено графічна інтерпретація залежних типів відмов.

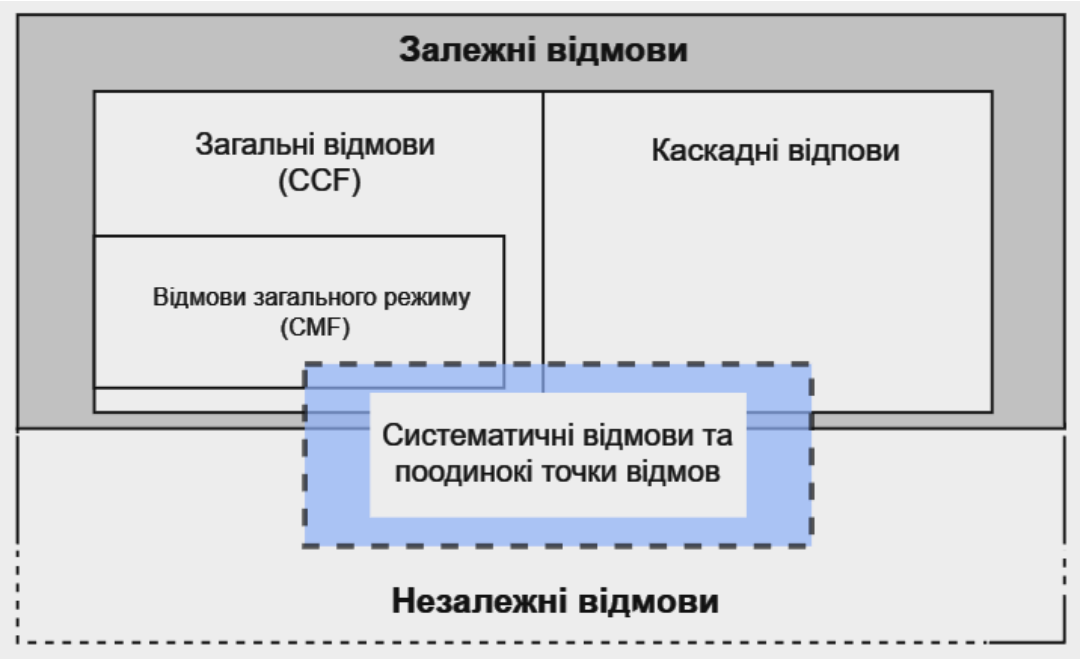


Рисунок 1.8 – Залежні типи відмов [34]

Третій підхід - аналіз рівня захисту, напівкількісний метод, що оцінює ризик небезпечних сценаріїв та ефективність незалежних рівнів захисту. Він враховує необхідний рівень цілісності безпеки для кожного рівня та ймовірність відмови з загальної причини, враховуючи часову послідовність подій чи залежність стану системи [34].

Параметричні моделі - математичні моделі, що описують частоту та серйозність відмов з загальних причин на основі статистичних даних і експертних оцінок. Такі моделі можуть бути «шоковими» або «не шоковими», в залежності від того, як вони враховують механізми відмов. Перші вимагають більше даних про час відмови, а другі більше даних про частоту відмов. Прикладами шокових моделей є модель біноміального розподілу та модель гіпергеометричного розподілу. Прикладами «не шокових» моделей є модель базових параметрів, модель бета-фактора, модель альфа-фактора, модель альфа-бета-фактора та інші. Ці моделі можуть бути використані, як явні або неявні, залежно від того, як вони включають загальні причини відмов в модель надійності [39].

Стисло розглянемо сутність двох основних підходів до моделювання відмов - явного та неявного. Неявне моделювання означає додавання подій, які охоплюють залишкові причини відмов, які не були змодельовані явно. Неявне моделювання може бути обрано, коли дані не доступні для підтримки явного моделювання. Прикладами неявних моделей є [15,40]:

- I C-factor model
- I Beta-factor model
- I Alpha-factor model
- I Multiple Greek Leer model
- I Multiple beta-factor model
- I Binomial failure rate model

З іншого боку, явне моделювання означає використання даних про загальні причини відмов для побудови детальної моделі, яка відображає механізми відмов. Явне моделювання може бути обрано, коли дані достатньо «надійні» та

повні для підтримки моделі. Прикладами явних моделей є моделі блок-діаграми надійності, дерева збоїв та ланцюги Маркова є прикладами явних моделей [20, 41]. Ці методи дозволяють враховувати часову послідовність подій або залежність стану системи.

Аналіз відмов за загальними причинами (тобто CCF) є критичним етапом оцінки надійності та безпеки IT систем. CCF, які можуть викликати відмову декількох компонентів одночасно, представляють серйозну загрозу, що породжується різними факторами: - проектні помилки, несправності обладнання та помилки персоналу. Їх наслідки включають втрату (виток) даних, порушення бізнес/технологічних процесів, фінансові та репутаційні втрати.

Запобігання та управління CCF вимагає детального проектування (врахування), заходів контролю якості та процедур управління операціями. Для аналізу відмов використовуються різні методи, такі як FMEA, FTA (расшифровать), аналіз першопричин та блок-схеми надійності. Розрізнення кореневих причин та факторів зв'язку є ключовим для розуміння та моделювання CCF. Таким чином, здійснення аналізу відмов за загальними причинами, є важливою складовою стратегії забезпечення надійності IT систем та мінімізації потенційних втрат і порушень основних технологічних процесів.

1.4.1 Особливості різниці між *CCF*, *SPOF* та каскадних відмов

Загальні причини відмов (CCF) слід чітко відрізняти від одноточкових збоїв (SPOF) і каскадних відмов. У випадку одноточкових збоїв відмова тільки одного компонента чи підсистеми призводить до відмови усієї системи чи функції. Це зазвичай наслідок недостатнього резервування або різноманітності в дизайні системи. При цьому для CCF характерно, що одна загальна причина одночасно впливає на кілька компонентів, що призводить до їх синхронного або майже одночасного виходу з ладу. У CCF ініціююча подія впливає на кілька компонентів одночасно або з невеликим часовим розривом. На відміну від них каскадні збої ініціює несправність одного компонента, яка через взаємодії та залежності з іншими елементами системи викликає ланцюгову реакцію збоїв в інших

частинах системи. У таких збоїв важлива саме динаміка поширення, де причини кожного наступного збою можуть відрізнятися [36].

Висновки за розділом: таким чином можна стверджувати, що ефективне забезпечення безпеки сучасних ІКС вимагає необхідність застосування комплексного підходу, який поєднує детальний аналіз системи для виявлення її характерних вразливостей, оцінку ризиків безпеки (через різні методи), розробку й впровадження плану підтримки функціональної безпеки з чітко визначеними вимогами (критеріями), та дотримання вимог міжнародних стандартів, наприклад таких як, IEC 61508, котрий сприяє запобіганню потенційних інцидентів, і забезпеченню стабільної роботи цільової системи.

2 ДИВЕРСІЙНІСТЬ ЯК ІНСТРУМЕНТ ЗАБЕЗПЕЧЕННЯ ФУНКЦІОНАЛЬНОЇ БЕЗПЕКИ СУЧАСНИХ ІКС

2.1 Особливості впровадження концепції диверсності в сучасні ІКС

Термін інформаційно-комунікаційна система, який об'єднує інформаційні та електронні комунікаційні системи в єдине ціле, виник на заміну попереднього «інформаційно-телекомунікаційні системи» відображає нові реалії у сфері інформаційної безпеки, які стали наслідком стрімкого розвитку технологій і зростання кіберзагроз. ІКС формувалися під впливом таких факторів, як технологічний прогрес, зростання мобільності та IoT, а також активне впровадження різних хмарних рішень та ШІ (*AI*). Ці зміни сприяли більшій інтеграції систем обробки даних і передачі інформації, що вимагає нових підходів до управління інформацією, її захисту та забезпечення стабільної роботи систем навіть у критичних умовах.

Функціональна безпека (Functional Safety) описує властивість системи, яка спрямована на мінімізацію ризиків несправностей або відмов. Концепція функціональної безпеки тісно пов'язана з інформаційною безпекою, однак акцентує увагу на безперебійному функціонуванні системи в критичних для безпеки умовах, тоді як інформаційна безпека націлена на захист даних і конфіденційності. У сучасних реаліях швидкого розвитку технологій і стрімкого зростання кіберзагроз стає критично важливим, щоб ІКС продовжували функціонувати навіть попри несприятливі обставини.

В цьому контексті, один з ключових факторів, що забезпечує покращення поточного рівня функціональної безпеки, є широке впровадження принципу диверсності, який використовує різноманітні технології, методи та стратегії для зменшення впливу різноманітних ризиків. Завдяки йому сучасні IT системи стають менш вразливими до однотипних загроз і набувають підвищеної стійкості перед різноманітними потенційними впливами. Різноманітність у структурі та функціонуванні, як це видно на прикладі природних екосистем, сприяє протидії

та зменшенню поширених збоїв. Інтеграція диверсифікації в стратегію кіберзахисту дозволить підвищити надійність системи та забезпечити гнучкість до атак, де компрометація однієї частини не призведе до повної втрати функціональності. Цей принцип виступає не тільки як додатковий рівень безпеки, а й як стратегічно важливий інструмент, який може забезпечити динамічний захист проти зростаючої хвилі атак, що постійно еволюціонують унаслідок стрімкого розвитку технологій.

2.2 Аналіз взаємозв'язку інцидентів з безпеки і масштабів впровадження диверсності в сучасних ІКС

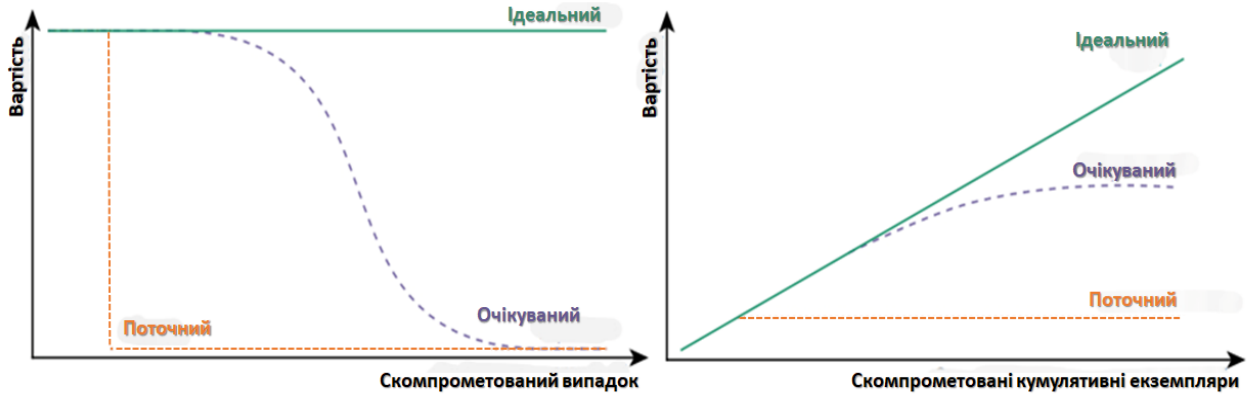
Ключовою властивістю будь-якої ІКС, є її здатність підтримувати коректне виконання основних функцій, незалежно від збоїв, що виникають внаслідок старіння, зношування чи пошкодження її апаратних компонентів, а також помилок у проектуванні, недоліків ПЗ чи наслідків кібератак [42, 43]. Як свідчить аналіз інцидентів безпеки, сучасні ІКС в багатьох випадках мають схожі вразливості, що надає потенційним зловмисникам значну свободу дій у виборі засобів та стратегії атаки [44-46]. В даному разі достатньо знайти одну вразливість, щоб створити універсальний вектор атаки, здатний виводити з ладу одразу декілька цільових ІКС [43]. Безумовно, що використання принципів однорідності та стандартизації ПЗ, сприяють економії часу і коштів на масштабах впровадження, його стабільності та полегшенню порядку розповсюдження та підтримки [47]. Однак, ця стратегія формує відповідні передумови для атак на такі ІКС, оскільки зловмисники можуть завантажити точну копію відповідного ПЗ для тестування, виявлення в ньому вразливостей й подальшого створення таргетованих експлойтів [47, 45], здатних загрожувати всім ІКС, що використовують відповідне ПЗ. Таким чином, «монокультура» ПЗ забезпечує економію на масштабах та в часі, не лише для користувачів, але й для зловмисників [43]. Вочевидь, що прагнення до стійкості ІКС [48] до відмов та/чи атак, вимагає впровадження концепції різноманітності ПЗ. Традиційно відмовостійкість сучасних ІС досягається завдяки певної надмірності у реалізованих програмно-апаратних рішеннях, що в свою чергу посилює захист

від фізичних збоїв та спроб проведення атак [42-43]. Як зазначено у звіті «Перспективи глобальної кібербезпеки до 2024 року», кібербезпека залишається одним головних пріоритетів на найближчі 10 років, при цьому прогнозується посилення атак на фінансові системи, енергетику і комунікаційну інфраструктуру [43,49]. У 2024 році в звіті Outlook, 81% керівників компаній відзначили, що вразливість їхніх ІКС значно зросла порівняно з попередніми роками, а 94% вважають, що їхні організації недостатньо захищені від кіберзлочинів. Офіційно, 29% організацій повідомили, що за останні 12 місяців вони зазнали серйозного впливу кіберінцидентів різного типу [43,50-51].

Використання принципу різноманітності в реальних комп'ютерних системах запозичує ідею з генетичної різноманітності, яка надає живим організмам стійкість до захворювань та несприятливих умов їх існування, оскільки різні генетичні варіації роблять ці організми більш стійкими й менш вразливими до одного й того ж патогену [52]. Дослідження цих процесів дозволило усвідомити, що перенесення принципу біологічної різноманітності у сферу створення й використання ПЗ, є ефективним рішенням для забезпечення функціональної безпеки сучасних ІКС та підвищення їх стійкості до різних типів кібератак, зокрема атак нульового дня [45,48]. Серед методів реалізації мережевої різноманітності, використовують такі техніки, як регулярна динамічна реконфігурація мережі, багатошляхова маршрутизація та застосування різних протоколів на кожному рівні [52]. Наприклад, динамічне оновлення адрес, протоколів та таймінгів, може помітно ускладнити атаки, які залежать від стабільних/типових внутрішніх та/чи зворотних реакцій цільової системи, зберігаючи при цьому доступність для легітимних користувачів [49,52]. Така реалізація мережевої різноманітності передбачає створення альтернативних версій ПЗ, компонентів і апаратного забезпечення, а також розробку різноманітних архітектур. Все це сприяє зниженню ймовірності одночасних відмов, забезпечуючи автономність компонентів [52]. Крім того, різноманітність значно ускладнює дії зловмисників, вимагаючи від нього більшого часу на мережеву розвідку та усвідомлення поточного стану справ. Все це посилює

«глибину захисту» (*defense in depth*) і зміцнює загальну стійкість ІТ систем до потенційних кіберзагроз [50,53]. Таким чином, щоб ускладнити експлуатацію типових вразливостей ІС, необхідно знизити передбачуваність цих систем.

Завдяки впровадження принципу диверсності, навіть у разі «успішної» атаки, вплив може бути обмежений лише одним екземпляром програмної системи чи окремим структурним елементом ІС. При цьому зрозуміло, що для «успішної» атаки зловмисник має володіти детальною інформацією про кожну реалізацію на всіх подібних системах [52]. В іншому випадку доведеться витрачати додаткові ресурси і час на адаптацію методики і засобів злому під кожну конкретну ІС. Вочевидь, що це суттєво підвищує умовний бар'єр для масових атак [51-52]. Так, діаграми на рисунку 2.1 [42] ілюструють, яким чином програмне різноманіття впливає на витрати атакуючих. Зліва відображається вартість компрометації окремих екземплярів ПЗ, а справа (*б*) - кумулятивні витрати умовного зловмисника на компрометацію кількох варіантів. Видно, що спочатку вартість залишається сталою, але з часом зменшується, оскільки зловмисник виявляє деякі спільні риси ПЗ, тобто врешті-решт, адаптація до програмного різноманіття знижує його ефективність [52].



а) *б)*

Рисунок 2.1 – Наслідки програмної диверсності для атакуючої сторони

В цілому, втілення концепції різноманітності не тільки ускладнює атакуючій стороні їх «роботу», але й зменшує масштаб впливу у разі «успішних»

атак. Цілеспрямовані атаки, також, потребують створення адресного варіанту експлойту для кожної цілі [4], що суттєво знижує ймовірність успіху таких атак. Наприклад, атакуючий може спробувати експортувати існуючі версії ПЗ для його аналізу чи тестувати експлойти «онлайн», але різноманітність програм ускладнює реалізацію. В результаті зусилля зловмисника зростають нелінійно зі збільшенням кількості інфікованих програмних пакетів [42,52].

Крім того, концепцію різноманітності можна використовувати і як інструмент для тестування ПЗ, що дозволяє виявляти потенційні вразливості в ньому. Так, у 2015-2016 роках агентство *DARPA* в межах реалізації програми *«Cyber Grand Challenge»*, створила відповідну збірку проблемних двійкових файлів з вже відомими вразливостями ПЗ та працюючими експлойтами [44, 45], що надало можливість фахівцям з питань ІБ, дослідити наслідки втілення в сучасних ІС концепції різноманітності та «лабораторно» оцінити ефективність різних методів диверсифікації [42,46,52,54]. В межах досліджень, щодо визначення ефективності диверсності для запобігання впливу експлойтам, була зафіксована проблема кількох викликових бінарних файлів (*challenge binaries - CBs*), коли їхні *CBs* постійно завершувалися помилками сегментації [42,52]. Крім того, було встановлено, що різноманітність не стала причиною цих збоїв, натомість сам процес диверсифікації виявив такі дефекти ПЗ. Це підтвердило можливість виявлення «слабких місць» у програмному коді за рахунок його тестінгу, а автоматизована різноманітність може бути використана, як дієвий інструмент для тестування ПЗ [52]. Однак, слід зазначити, що впровадження концепції диверсності ПЗ не завжди є доцільним, так як може ускладнитися застосування деяких засобів ІБ. Така ситуація може бути в разі використання систем фільтрації реєстрів/списків даних, що зберігають криптографічні підписи бінарних файлів, які вважаються «безпечними» для певного процесу чи організації, в цілому. В даному разі такі системи ІБ вимагають контролю підписів для кожної версії ПЗ [42], тому потрібна зважена імплементація всього концепту диверсності ПЗ [52].

2.3 Узагальнення питань класифікації видів виверсійності.

Забезпечення стійкості та надійності комунікаційних систем є важливим завданням, особливо в умовах загроз з боку зловмисників. Одним з підходів до вирішення цієї проблеми є використання різноманітних реалізацій або конфігурацій, які можуть ускладнити атаки. Наприклад, можна змінювати конкретні екземпляри реалізацій з плином часу, викликаючи часово-змінну невизначеність для зловмисника, або запускати багато паралельних екземплярів, що створює просторово-змінну невизначеність. Такі підходи можуть не лише ускладнити пошук невідомих вразливостей, але й сприяти їх усуненню. Розглянемо детальніше класифікацію видів диверсій, які можуть бути використані для підвищення стійкості комунікаційних систем.

2.3.1 Різноманітність на основі компілятора

Підходом до реалізації диверсійності компілятора є трансформація високорівневого коду у «змішаний» формат, що дозволяє компілятору автоматично генерувати різні варіанти ПЗ. Це є вирішенням проблеми підвищення стійкості до цілеспрямованих атак, які потребують створення унікальних експлойтів для кожної цілі, ускладнюючи роботу зловмисників і зменшуючи масштаб впливу у разі успішних атак [43,52].

Різноманітність, яку забезпечує компілятор, ґрунтується на кількох техніках, які використовуються в системах з відкритим доступом [43]. Наприклад, вставка «сміттєвого» коду (*тобто що зволікає*) передбачає ймовірнісне додавання в ПЗ однієї або відразу кількох «сміттєвих» інструкцій перед поточною діючою інструкцією. Такий код може варіюватися від простої інструкції *x86 NOP* до більш складних, які зберігають стан процесора (*наприклад, mov esp, esp*). Інша техніка, заміна інструкцій, що дозволяє виразити арифметичні операції кількома способами. Ще один спосіб – обфускація функцій, де порядок функцій у компіляторі ПЗ змінюється для кожного блоку, створюючи унікальні варіанти об’єктного коду [43,52]. Можливе рішення полягає у перетворенні високорівневого коду на «змішаний», що дозволяє компілятору автоматично генерувати різні версії ПЗ. Завдяки цьому кожен користувач отримує

«унікальну» версію програмного коду, а потенційні зловмисники не мають актуальних відомостей щодо внутрішньої структури цього релізу ПЗ та не можуть відразу спланувати атаку [46]. Так, репозиторії додатків *Apple App Store* чи *Android Marketplace*, використовують механізм диверсифікації (*мультикомпілятор*), який автоматично генерує унікальний, але функціонально подібний варіант ПЗ на кожен запит завантаження. У такому середовищі багатоваріантного виконання (*Multi-Variant Execution Environment, MVEE*), вирішується завдання підвищення безпеки додатків, адже спроба атаки може вплинути лише на обмежену кількість релізів ПЗ [43,46,52].

2.3.2 Диверсність даних

Різноманітність даних тісно пов'язана з механізмами обробки помилок і доповнює інші напрями диверсності. Ця техніка базується на ідеї, що два подібні вхідні значення повинні давати подібні вихідні результати. Навіть незначна зміна вхідного сигналу (даних), який контролює зловмисник, може значно знизити ефективність шкідливого впливу, зокрема шляхом запобігання аварійному завершенню процесу чи перехопленню контролю над системою, або її окремими підсистемами. Цей різновид диверсності сприяє покращенню відмовостійкості загальної структури ПЗ та має певний потенціал для попередження перспективних атак [43,49,52].

Для реалізації цього варіанту забезпечення диверсності потрібні зміни коду ПЗ, які враховують різницю в обробці даних для досягнення нормальної еквівалентності та дивергентної (іншої) поведінки. Нормальна еквівалентність передбачає, що всі варіанти обробки даних залишаються семантично еквівалентними. Це означає, що довірені дані для кожного варіанта трансформуються відповідними функціями, а інструкції для роботи з цими даними – модифікуються, щоб зберегти семантику [43,52]. Проте, у разі атаки на один з варіантів, всі інші повинні демонструвати відмінні від нього зворотні реакції, що дозволяє виявляти ознаки втручання. Це забезпечується застосуванням спеціальних функцій перетворення, які гарантують, що однакові вхідні дані, подані до різних варіантів, генеруватимуть різні результати, якщо

один з варіантів буде атакований. Така диверсність даних сприяє виявленню несанкціонованих ін'єкцій у конкретні типи даних через порівняння поведінки різних інтерпретаторів [43,52]. Методика особливо корисна для захисту типів даних, які часто стають ціллю атак (адреси пам'яті, інструкції процесора чи ідентифікатори користувачів тощо).

2.3.3 Різноманітність на основі дизайну

Різноманітність дизайну (побудови) являє собою наступний напрям, згідно з яким апаратні і програмні компоненти, призначені для повторюваних обчислень, не є точними копіями, а розробляються незалежно один від одного для задоволення вимог/функціоналу системи. Однією відомих технік впровадження цієї концепції є *n*-версійне програмування, яке забезпечує ізоляцію можливих збоїв один від одного. Цей підхід передбачає роздільну розробку апаратних і програмних компонентів при збереженні відповідності загальним базовим вимогам [43,52]. Цій підхід декларує «незалежність» виникнення помилок, що важливо, оскільки кількість випадкових збоїв часто перевищує очікування на основі припущень саме про незалежність помилок [42]. В межах цієї концепції кілька незалежних реалізацій ПЗ створюються на основі єдиної специфікації, що визнано дієвим методом підвищення надійності. Так автори роботи [53] описують власну реалізацію дизайн-диверсності для підвищення відмовостійкості систем потрійного модульного резервування (*TMR*) на базі польових програмованих вентильних матриць (*FPGA*) [43,52]. Стверджується, що різноманітність дизайну дозволяє системі *TMR* продовжувати функціонувати навіть за наявності несправності в одному з модулів, тоді як традиційна система *TMR* без застосування дизайн-диверсності могла б зазнати збою в аналогічній ситуації. При цьому, хоча дизайн-диверсність підвищує надійність ІС, цей підхід може бути дорогим і трудомістким, особливо для випадків реалізації некритичних ІС [43,52].

Для нівелювання цих особливостей впроваджуються автоматизовані способи забезпечення різноманітності, такі як *диверсний компілятор*, що є одним із засобів реалізації дизайн-диверсності. Диверсифікаційний компілятор працює

подібно до «звичайного» компілятора, але не гарантує ідентичності результатів при однакових вхідних даних [53]. Наприклад, він може вирішувати, чи видаляти баластовий (*недіючий, зашумляючий тощо*) програмний код, із певною ймовірністю, додаючи випадкові зміни до вихідного виконуваного файлу [43,52-53]. Хоча така варіативність може викликати певне занепокоєння, стосовно зниження швидкодії чи збільшення розміру коду, варто зазначити, що компілятори рідко гарантують оптимальність синтезованого ними ПЗ, покладаючись на відомі евристики і профілі виконання для зменшення ресурсів, необхідних для виконання обчислень. Оскільки такі трансформації ПЗ виконуються на рівні компілятора, то вони не ускладнюють супроводжуваність вихідного коду і зберігають його функціональну ідентичність, тобто не змінюють поведінку кінцевого ПЗ, стосовно вхідних/вихідних даних [42-43,52].

2.3.4 Диверсійність програмного забезпеченя

Програмна різноманітність (Software Diversity) – це підхід, що передбачає різні реалізації спільного функціоналу. Він водночас підходить для підвищення як функціональної надійності, так і безпеки ІС, оскільки застосування різних версій ПЗ зменшує ризик «успішної» атаки, так як зловмиснику необхідно долати різні механізми захисту. Різні підходи до впровадження програмної диверсності можуть бути реалізовані «вручну» програмістами, або ж автоматизовано, за допомогою вже згаданого вище компілятора чи бінарного переписувача [43,52].

Загалом, методи забезпечення стійкості ПЗ можна умовно поділити на техніки з однією чи кількома версіями. Наприклад, *n-версійне програмування*, яке базується на різноманітності дизайну, передбачає створення компонентів різними командами із застосуванням різних мов програмування та алгоритмів. Це знижує ймовірність того, що програмні компоненти міститимуть однакові помилки [54]. Оскільки забезпечення *різноманітності дизайну* потребує значних працевитрат, цей спосіб є досить дорогим і обмежується переважно спеціалізованими галузями [43,52]. На противагу ньому, техніки, що втілюють парадигму автоматичного створення коду через *диверсний компілятор* або бінарний переписувач, спрямовані на підвищення безпеки. Вони можуть бути повністю

автоматизованими та впроваджуються на пізніших етапах життєвого циклу розробки ПЗ.

2.3.5 Штучна диверсифікація

Штучна диверсність (Artificial Diversity) застосовується до бінарних програм і вихідного коду. Вона ускладнює експлуатацію вразливостей ПЗ, збільшуючи час і ресурси, котрі необхідні для проведення атаки. Вона додає випадковість та непередбачуваність у ПЗ без необхідності впровадження складних «секретів». Так наприклад, один із варіантів цієї техніки передбачає одночасне використання кількох ключів рандомізації для створення деякої сукупності (*N*) релізів вихідного коду [43,52]. У такому випадку всі *N* варіантів виконуються паралельно з однаковими вхідними даними, а резидентний монітор порівнює результати виконання [55]. *Штучна диверсність* характеризується динамічністю й адаптивністю, що дозволяє змінювати конфігурацію системи (її ПЗ) в реальному часі, мінімізуючи передбачуваність подій для зловмисників, які зазвичай покладаються на статичну природу цільових/атакованих ІС [43,52].

Ще одним підходом є *випадковий розподіл адресного простору* (*ASLR - Address-Space Layout Randomization*), який наділяє випадковістю процедуру розміщення в пам'яті ключових компонентів створюваного ПЗ, таких як стек, купа та бібліотеки коду, щоразу при виконанні цього ПЗ. Це ускладнює зловмисникам прогнозування розташування певного коду, тим самим затягуючи час й знижуючи ефективність атак, які залежать від знання конкретних місць розташування в пам'яті різних компонентів створюваного ПЗ [43,52,56].

2.3.6 Апаратна диверсійність

Апаратна диверсність (Hardware Diversity) впроваджує ідею посилення захисту за допомогою випадкового компонування чіпів, що значно ускладнює атаки, котрі спрямовані на фізичне розташування схем. Процесори, що підтримують рандомізацію набору інструкцій, змінюють поточні параметри кодування, тим самим роблячи результати потенційного ін'єкційного нападу/атаки на двійковий код, непередбачуваними [42-43,49,52]. При цьому, у системах з надмірністю, принципово важливо уникати однакових станів в

резервних компонентах, що використовуються під час відмов. Це критично, коли резервування базується на ідентичних ядрах, які виконують одну і ту ж програму. Тому, для зменшення ризику відмов, через загальні фактори, також, необхідно забезпечити відмінності у внутрішніх елементах всіх резервних компонентів.

Питання коректності вибору етапу введення різноманітності у життєвий цикл ПЗ є важливим фактором, оскільки це впливає на вартість і можливість обслуговування кінцевого продукту й цільової ІС в цілому. Складнощі з їх обслуговуванням можна нівелювати, застосовуючи автоматизовані техніки на етапах компіляції ПЗ та його виконання [43,49,52]. Так, в роботі [54] її автори пропонують деякі рекомендації з цього питання, наприклад: - різноманітність дизайну (*включно з n-версійним програмуванням*) передбачає відмінності на концептуальному рівні. Компіляторна диверсифікація, що застосована під час компіляції ПЗ, вносить низькорівневу випадковість, зберігаючи при цьому цільову функціональність коду, але змінюючи його структуру [43,52]. На етапі встановлення ПЗ, різні методи рандомізації можуть інтегруватися в інсталяційні скрипти, що може ефективно протидіяти атакам на розташування в пам'яті. Під час виконання ПЗ, система може використовувати динамічну рандомізацію для організації пам'яті або черговості окремих функцій, збільшуючи непередбачуваність коду.

2.3.7 Методи розмаїття мереж

Методи розмаїття мереж (Networks Diversity) - техніки для реалізації різноманітності та стійкості в комунікаціях включають часту динамічну реконфігурацію мережі, багатошляхову маршрутизацію та використання кількох різних реалізованих протоколів на кожному рівні. Динамічна перестановка адрес і протоколів могла б перешкодити атакам, які залежать від послідовних внутрішніх реакцій, зберігаючи при цьому функціональність, необхідну для законних користувачів [43,49,52]. Network Diversity був застосований у проекті Moving Target Defense (MTD), де реалізували динамічну зміну конфігурацій програмного забезпечення в мережах для підвищення безпеки. У цьому проекті різні компоненти, такі як веб-сервери, сервери додатків і бази даних, постійно

змінювалися, щоб ускладнити зловмисникам можливість експлуатації вразливостей [43,52].Основними факторами в мережевих налаштуваннях є топологія мережі та системні компоненти, встановлені в кожному вузлі мережі. Системні компоненти можуть включати як апаратні, так і програмні елементи. Документ [48] акцентує увагу на концепції мережевої діверсійності, як механізму безпеки та передбачає використання різноманітних ресурсів у межах мережі для підвищення її стійкості до атак, зокрема нульових днів.

2.4 Динамічна диверсифікація

Динамічна (або нападницька) диверсифікація – це підхід, який використовується шкідливим ПЗ для уникнення можливостей його виявлення (*тобто активної протидії*). Вона допомагає атакуючої стороні створювати програмний код, який постійно змінюється і стає менш помітним для антивірусів. Її головна мета – зробити аналіз і виявлення складнішими для захисних систем [57].

Є кілька основних реалізацій (способів) нападницької диверсифікації:

- Олігоморфізм – це найбільш простий спосіб, де шкідливий код приховується (шифрується) за допомогою відносно простих операцій, наприклад XOR, а потім розшифровується перед виконанням.
- Поліморфізм – більш вдосконалений варіант, де програмний код кожен раз змінюється по новому, а в ньому залишається лише невелика частина, яка розпаковує основну шкідливу програму.
- Обфускація через віртуалізацію – код перетворюється у байткод, який виконується віртуальною машиною. Щоразу при поширенні байткод і віртуальна машина змінюються.
- Метаморфізм – код модифікується так, щоб виглядати по-різному, але виконувати ті самі функції. Наприклад, додаються непотрібні команди або змінюється порядок інструкцій.
- Метаморфізм – найскладніший для виявлення, бо не має помітних шаблонів, таких як шифрування чи віртуальні машини. Зазвичай такий код спершу розбирається, змінюється, а потім збирається знову в зміненій формі.

Всі ці методи створюють високий рівень різноманітності шкідливих програм, через що статичний і навіть динамічний аналіз стають менш ефективними. На відміну від захисної диверсифікації, яка спрямована на усунення вразливостей, нападницька фокусується на маскуванні і уникненні виявлення [57].

На рисунку 2.2 представлена інфографіка процесу умовної еволюції атак за допомогою диверсифікації, починаючи від статичних методів до складних модифікованих форм. Центральна частина, «Зона диверсифікації», відображає ключові техніки, які використовують атакувальники: олігоморфізм, поліморфізм, обфускацію через віртуалізацію, метаморфізм і «Франкенштейн» методику [57]. Ці методи дозволяють створювати більш стійкі до виявлення шкідливі програми, які важко аналізувати. В результаті, покращені атаки мають мінімальні статистичні ознаки, що ускладнює їх відрізнення від легітимного коду, ефективно уникають статичного детектування і характеризуються високою складністю аналізу поведінки завдяки самомодифікації та обхідним технікам.

Нападницька диверсифікація використовує техніки, схожі на захисну диверсифікацію, але має протилежну мету — уникнути виявлення та протидіяти аналітичним системам. Методи обфускації, які застосовуються на рівні інструкцій, базових блоків і функцій, нагадують підходи, що використовуються для підвищення захищеності легітимного ПЗ.

Метаморфізм є найбільш складним для виявлення методом, оскільки він усуває очевидні шаблони, наприклад, шифрування або віртуальні машини. У процесі він змінює код шляхом розбору на проміжний рівень, модифікації та збирання у новій формі [57]. Ці підходи значно ускладнюють використання стандартних інструментів аналізу, як статичних, так і динамічних. Основна різниця між нападницькою та захисною диверсифікацією полягає в цілях та обмеженнях їх застосування.

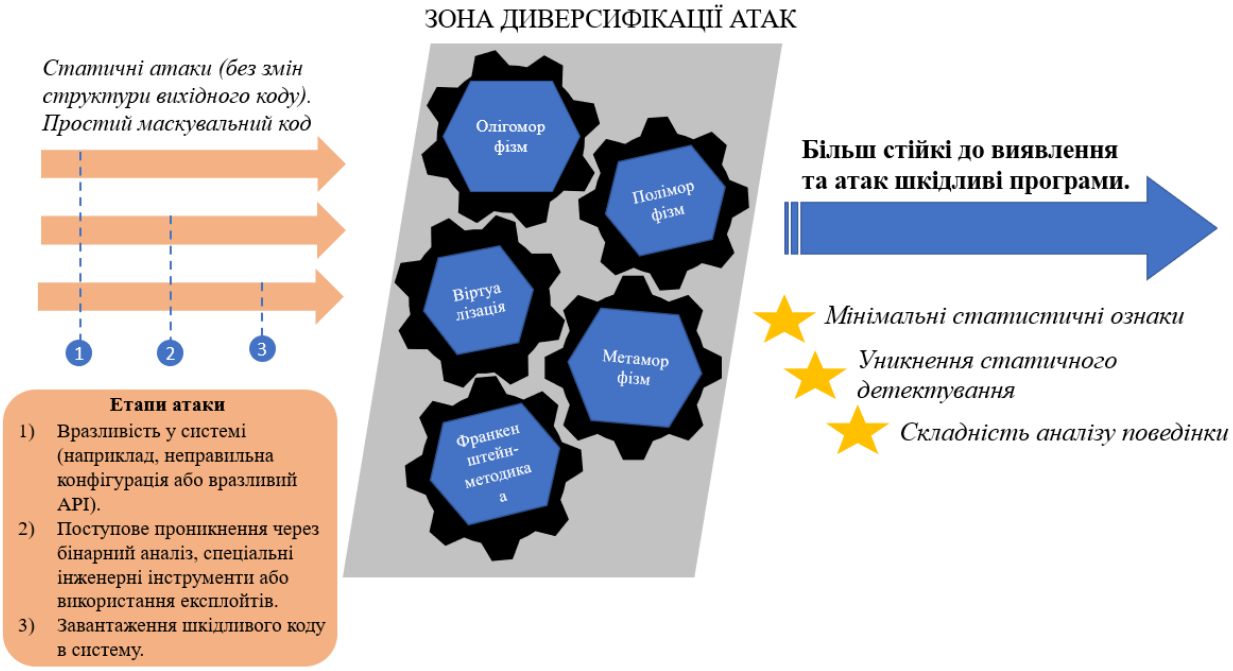


Рисунок 2.2 – Еволюція атак за рахунок диверсифікації

Динамічна диверсифікація активно використовується під час поширення шкідливого коду, щоб уникнути виявлення у стані спокою чи під час передачі. У свою чергу, захисна диверсифікація зосереджена на усуненні загальних вразливостей і шляхів експлуатації. Крім того, захисна диверсифікація обмежується вимогами до продуктивності та підтримки ІС, тоді як у шкідливому коді пріоритетом є прихованість і стійкість до виявлення, навіть якщо це підвищує обчислювальні витрати. Ці відмінності визначають, як саме реалізуються методи програмного синтезу, орієнтовані на обфускацію та створення різноманітності, в залежності від поставлених цілей. Нападницька диверсифікація, таким чином, є ключовим елементом у постійному розвитку сучасних атак [57].

Висновки за розділом: Сучасні інформаційно-комунікаційні системи стикаються з численними кіберзагрозами, які вимагають впровадження нових підходів до забезпечення їх стійкості до впливу нових різновидів атак та покращення поточного рівня їх функціональної безпеки. Використання концепції різноманітності передбачає створення альтернативних версій ПЗ, компонентів і апаратного забезпечення, а також розробку різноманітних, близьких за

функціональною парадигмою, архітектур. Все це сприяє зниженню ймовірності одночасних відмов, забезпечуючи широку автономність та незалежність складових компонентів. Впровадження принципу диверсності, що передбачає використання різноманітних технологій і методів, є ключовим для зменшення ризиків атак, оскільки він помітно (рис. 2.1) ускладнює діяльність потенційних зловмисників, позбавляючи їх універсальних інструментів та уніфікованих векторів атак, котрі здатні виводити з ладу одразу декілька цільових систем (чи їх окремих компонентів), що в свою чергу, ефективно обмежує масштаб деструктивного впливу на цільові системи.

3 ДОСЛІДЖЕННЯ ПОШИРЕНИХ ЗАСОБІВ ТА МЕТОДІВ ВПРОВАДЖЕННЯ РІЗНОМАНІТНОСТІ В ІКС

3.1 Фреймворки

CROW (*Code Replacement and Optimization Workflow*) – це система з відкритим вихідним кодом, яка призначена для автоматизації диверсифікації програм, написаних для середовища *WebAssembly* [58]. Основна мета – підвищення стійкості програмного забезпечення до атак шляхом генерації набору різних двійкових файлів WebAssembly з однієї вихідної програми, написаної на C/C++. Диверсифікація в CROW здійснюється через процес синтезу різноманітних кодових фрагментів. Це дозволяє генерувати різні варіанти програми, зменшуючи інформаційну асиметрію між атакуючою стороною та захисником, а також впроваджує стратегію рухомої цілі (moving target defense). Завдяки такому підходу кожен новий HTTP-запит може оброблятися унікальною версією програми, що значно ускладнює використання вразливостей [58].

Диверсифікація в CROW реалізована як синтез різних кодових фрагментів, що дозволяє генерувати варіанти програм, які відрізняються один від одного. Це створює різницю, зменшуючи інформаційну асиметрію між атакуючою стороною та захисником, а також впроваджує стратегію рухомої цілі (moving target defense). Такий підхід гарантує, що новий HTTP-запит обробляється унікальною версією програми, що створює нові перепони для нападника при спробі скористатися вразливостями системи [58]. Принцип роботи системи CROW складається з трьох ключових етапів: етап дослідження (Exploration Stage), етап генерації (Generation Stage) та етап компіляції (Compilation Stage). На першому етапі за допомогою компілятора вхідна програма перетворюється на потік бітового коду. Аналіз коду з метою виявлення "чистих блоків", які завжди демонструють однакові вихідні результати для заданих вхідних даних, є завданням цього інструменту. Для кожного такого блоку створюються альтернативні варіанти, які перевіряються на функціональну еквівалентність з

оригінальним блоком. Усі варіанти, що пройшли успішну перевірку, зберігаються для подальшого використання [58]. Наступним етапом є об’єднання створених альтернативних рішень і формування різноманітних варіантів бітового коду. Цей процес необхідний для налаштування таких параметрів, як максимальна кількість інструкцій у блоці або набір інструкцій для синтезу, що забезпечує оптимальний баланс між кількістю варіантів і витраченим часом на їх створення. Завершальним етапом є трансформація всіх згенерованих варіантів бітового коду в двійкові файли WebAssembly, які вже будуть повністю готовими для експлуатації в різних середовищах. Ця стадія реалізує концепцію рухомої цілі, що підвищує рівень захисту програми шляхом постійного створення нових варіантів коду [58]. Варто зазначити, що CROW генерує безліч варіантів системи замість одного оптимального рішення. Це називається супердиверсифікацією і значно підвищує рівень безпеки програмного забезпечення.

DAEDALUS (*Defense Against firmware ROP Exploits using stochastic software Diversity*) – це фреймворк, що використовує різноманітність ПЗ для захисту пристроїв IoT на базі Linux від *ROP*-атак.

Основна концепція DAEDALUS полягає в розробці унікальних двійкових перезаписів прошивки IoT, які є семантично еквівалентними, але синтаксично відмінними. Це значно ускладнює виконання атак зловмисниками [59]. ROP-атаки потребують точних знань про структуру двійкового коду, включаючи адреси пам’яті та використовувані фрагменти для побудови корисного навантаження. DAEDALUS порушує цю модель, створюючи синтаксично різноманітні перезаписи, які унеможливлюють застосування однакового навантаження на різних пристроях . Це особливо актуально в контексті DDoS-атак, оскільки зловмисникові доводиться генерувати унікальні атаки для кожного окремого пристрою. Унікальною рисою DAEDALUS є його вибірковість. Він зосереджений на диверсифікації критично важливих базових блоків, які можуть бути вразливими до помилок пам'яті або використовуватися для здійснення *ROP*-атак. Такий підхід дозволяє зменшити накладні витрати та зберегти продуктивність IoT-пристроїв, які зазвичай мають обмежені ресурси [59].

Фреймворк діє на рівні двійкових файлів, що забезпечує його сумісність навіть без доступу до вихідного коду. Він автоматично аналізує локалізовані потоки даних в базових блоках і генерує семантично еквівалентні варіанти коду, які відображають оригінальний керуючий потік. Крім того, DAEDALUS контролює розмір створених блоків, що є вкрай важливим для ресурсозалежних IoT-пристроїв.

Переваги DAEDALUS включають наступне [59]:

- Захист від масштабних атак, адже різноманітність прошивки запобігає реплікації одного корисного навантаження.
- Автономність і низькі накладні витрати, що робить його придатним для використання в IoT-домені.
- Відсутність необхідності у спеціальному апаратному забезпеченні або додаткових бібліотеках.

DAEDALUS пройшов тестування за допомогою реалістичних симуляцій, які продемонстрували, що навіть два унікальних перезаписи базових блоків ефективно блокують зловмисників, не даючи їм можливості отримати несанкціонований доступ до пристроїв. Унікальність цього підходу дозволяє інтегрувати DAEDALUS у масштабні середовища розгортання, такі як побутові та критично важливі IoT-пристрої [59]. Рисунок 3.1 демонструє двоетапний робочий процес фреймворку DAEDALUS, призначений для захисту IoT-пристроїв від ROP-атак. На першому етапі виконується аналіз двійкового файлу прошивки з метою виявлення базових блоків, які є критично важливими для безпеки та можуть містити уразливості до помилок пам’яті (*Type M*) або інструкції, що можуть бути використані для *ROP*-атак (Type R). На другому етапі ці блоки підлягають локалізованому аналізу потоку даних, що дозволяє синтезувати їх семантично еквівалентні, але синтаксично різні версії за допомогою інструменту STOKE, після чого генеруються перезаписи прошивки [59]. Ці модифіковані прошивки піддаються тестуванню в середовищі DDoSim для оцінки здатності DAEDALUS запобігати атакам, забезпечуючи надійний рівень захисту без значного споживання ресурсів.

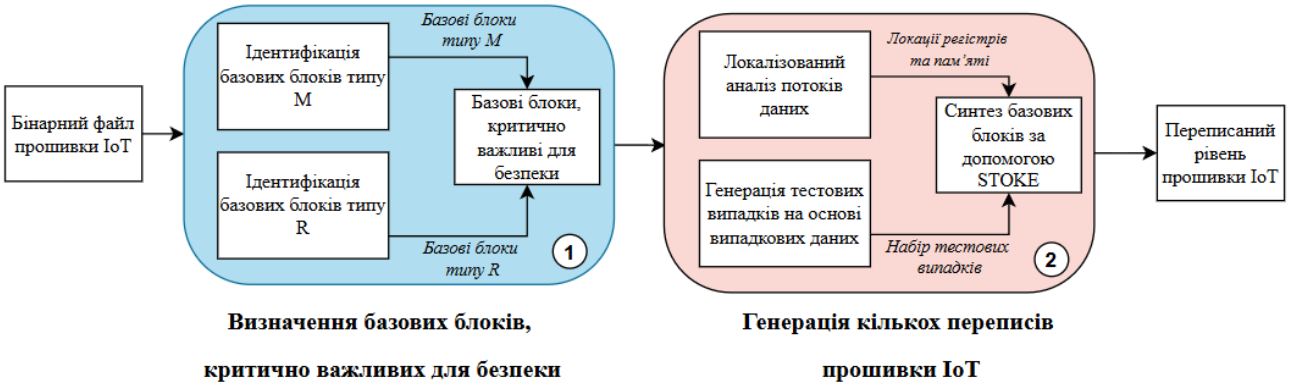


Рисунок 3.1 – Двоетапний робочий процес фреймворку DAEDALUS [59]

Таким чином, DAEDALUS є ефективним рішенням для покращення безпеки прошивок IoT, пропонуючи практичний підхід до боротьби з сучасними загрозами, пов'язаними з *ROP*-атаками.

Rave це фреймворк створений для динамічної диверсифікації ПЗ в розподілених середовищах з механізмом захисту рухомих цілей (MTD). Він забезпечує високий рівень захисту від атак, що ґрунтуються на пошкодженні пам’яті. Основна мета *Rave* полягає в рандомізації внутрішніх станів програм, таких як, компонування стека та послідовність інструкцій, а також у перенесенні процесів між вузлами з різними апаратними або програмними налаштуваннями [60]. Фреймворк складається з двох ключових компонентів: *librave* та *CRIU-Rave*. *Librave* виконує статичний аналіз бінарного коду та забезпечує необхідні інструменти для реалізації політик рандомізації, тоді як CRIU-Rave є вдосконаленою версією перевіреного інструменту CRIU для міграції процесів у Linux. CRIU-Rave відповідає за динамічне оновлення стану програм під час міграції, включаючи зміни в структурі даних та внутрішньому коді. Обидва компоненти функціонують у просторі користувача, що виключає потребу у втручанні в цільову програму та забезпечує повну ізоляцію від її адресного простору, що запобігає компрометації захисної логіки Rave зловмисником [60]. Rave інтегрує механізм обробки помилок сторінок у просторі користувача для динамічного переписування випадкових сторінок коду та даних. Це дозволяє системі постійно оновлювати та перемішувати внутрішні стани програм,

мінімізуючи витрати на продуктивність. Важливою перевагою є можливість прозорого перенесення процесів у розподілене середовище, що дозволяє динамічно коригувати апаратні та програмні стеки та уникати ризиків, пов’язаних із статичною прив’язкою до конкретного середовища виконання.

Rave розширює традиційні методи захисту, впроваджуючи перевірені механізми міграції процесів і власну логіку рандомізації, що довели свою ефективність у реальних умовах. Це дозволяє ефективно протистояти атакам, які спираються на передбачуваність середовища виконання або структури даних. Навіть якщо зловмисник отримає доступ до програмного середовища, динамічне оновлення станів і безперервна зміна компонування коду та стека ускладнюють виконання попередньо підготовлених шкідливих дій. Таким чином, Rave забезпечує високу програмну ентропію при мінімальних витратах, встановлюючи новий стандарт у захисті програмного забезпечення в умовах сучасних загроз [60].

Sphinx – це фреймворк, який використовує обфускацію як основний механізм для захисту програм від атак, особливо в контексті монолітних систем. Основна ідея Sphinx – ускладнити життя зловмисникам, маскуючи структуру та функціональність програми, що значно ускладнює реверс-інжиніринг. Цей фреймворк реалізує динамічну обфускацію: у процесі виконання програми змінюються назви змінних, функцій і навіть алгоритмів [61]. Це робить аналіз коду для атакуючих практично неможливим. Важливо, що Sphinx легко інтегрується в існуючі системи, що робить його універсальним інструментом у сфері захисту. Sphinx складається з двох компонентів: програмного обфускатора та апаратного блоку SARA (Self-Aware Reconfigurable Architecture). Обфускатор генерує випадкові інструкції, додаючи бінарну маску, яка відокремлює справжні інструкції від фальшивих. Це забезпечує [61]:

1) Неможливість ефективного аналізу бінарного коду;

2) Можливість динамічної адаптації програми до апаратного середовища.

Щодо апаратної частини, SARA надає різні способи виконання однієї інструкції. Це дозволяє варіювати час виконання, енергоспоживання, а також доступ до пам'яті. На рисунку 3.2 (*Sphinx obfuscation-based secure system*), представлена архітектура системи *Sphinx*, яка поєднує програмну обфускацію та апаратну платформу SARA [61]. В даному випадку, верхній рівень (програмний) включає етапи компіляції, обфускації коду та створення обфускованої програми. Нижній рівень (апаратний) представлений модулем SARA, що забезпечує динамічну обробку обфускованого коду за допомогою блоків обфускації, дешифрування, маскування виконання та пам'яті. Така архітектура дозволяє захищати програму від атак реверс-інжинірингу та на основі бічних каналів.

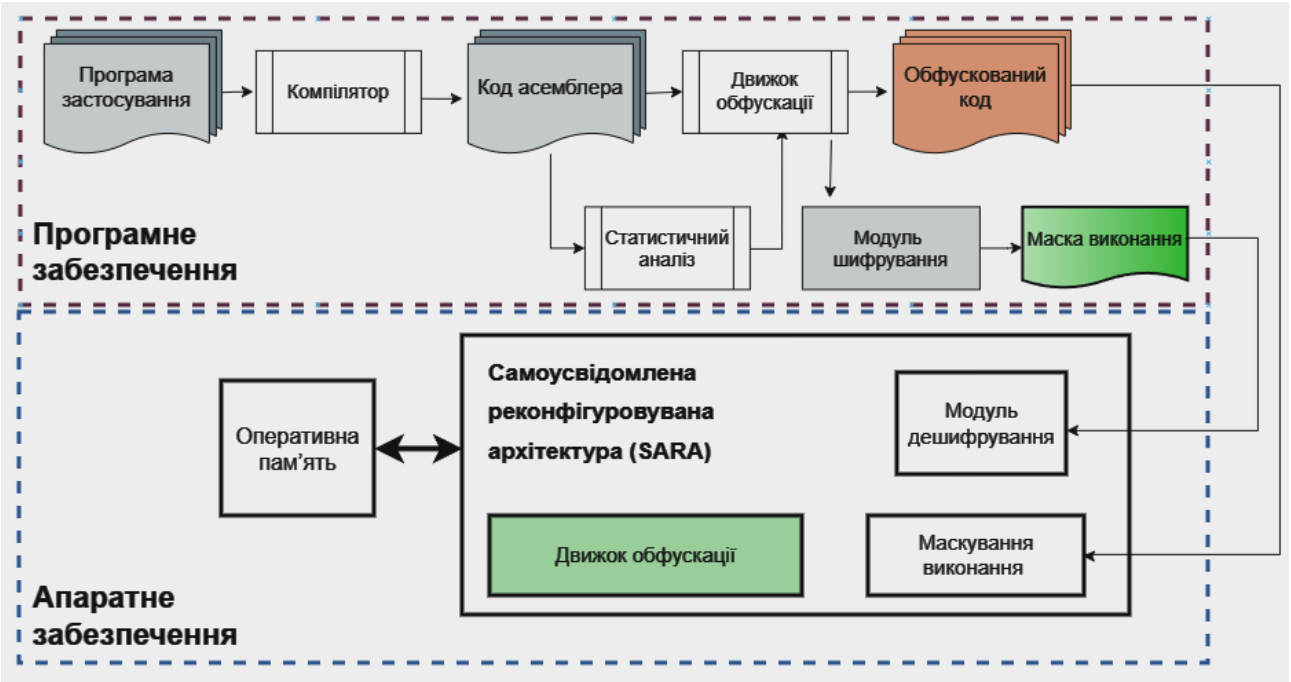


Рисунок 3.2 – Архітектура системи *Sphinx* (за даними [61])

На відміну від Sphinx, який фокусується на обфускації, *HeterSec* використовує гетерогенні архітектури інструкційного набору (ISA) як засіб для підвищення захисту програмного забезпечення. Основна концепція полягає в імітації мульти-ISA середовища, що дозволяє програмам працювати в умовах динамічно змінюваних ISA [62].

Цей підхід відкриває нові можливості для захисту, такі як:

- Рухомий захист (Moving Target Defense), який ускладнює аналіз програми через постійну зміну середовища виконання.
- Багатоваріантне виконання (Multi-Variant Execution), де кілька варіантів програми запускаються одночасно, а результати порівнюються для виявлення аномалій.

Головною перевагою *HeterSec* є те, що він працює на реальних машинах із різними ISA, на відміну від попередніх підходів, що обмежувалися симуляціями. Це дозволяє HeterSec забезпечувати реальний захист програм, навіть у найскладніших середовищах. Основні сценарії використання включають динамічну зміну середовища ISA для створення додаткового захисту та виявлення атак через аналіз аномальних поведінок під час виконання [62].

Обидва фреймворки спрямовані на покращення безпеки через диверсифікацію. Sphinx акцентує увагу на обфускації та захисті програмного коду від реверс-інжинірингу, тоді як HeterSec досягає захисту через варіативність архітектур ISA. Обидва підходи ускладнюють роботу зловмисників, підвищуючи загальний рівень безпеки систем [62].

3.2 Особливості оцінки наслідків від впровадження диверсності

Важливою темою є кількісна оцінка ефективності диверсифікації ПЗ, яка слугує для підвищення його стійкості до атак і забезпечення функціональної безпеки, щоб зберегти його ефективність. На жаль, немає можливості прямо виміряти результати впровадження різноманітності, тому дослідники використовують непрямі методи, такі як аналіз ентропії. Цей метод оцінює непередбачуваність бінарного коду після внесення змін: чим вищий рівень ентропії, тим більший поріг складності атак для зловмисників. Наступним методом є аналіз атак на основі конкретного коду, зокрема засоби для генерації ROP-ланцюгів. Вони здатні виявляти слабкі місця в коді, які можуть залишитися після диверсифікації системи. Час від часу використовується логічна аргументація, що ґрунтується на припущенні: якщо атака залежить від певної властивості системи, яка в свою чергу варіюється в процесі диверсифікації, то

захист спрацює. Недоліком цього методу є неможливість кількісно оцінити рівень зусиль, необхідних для зловмисника, щоб обійти захист [62]. При тестуванні конкретних атак вибирається загроза, що була успішною до впровадження диверсифікації, та перевіряється, чи залишиться вона такою після застосування різноманітності. Однак такий підхід не гарантує високого рівня ентропії [62]. Для якісної оцінки ефективності диверсифікації та її впливу на швидкість і витрати ресурсів застосовуються стандартні бенчмарки, наприклад, SPEC CPU. Також використовуються різноманітні метрики, такі як кількість вцілілих гаджетів під час *ROP*-атак або відсоток незмінного коду. Проте бракує матеріалів та досліджень, які б показували розподіл усіх варіантів диверсифікованих систем у просторі рішень, оскільки не всі перетворення здатні забезпечити прийнятний рівень різноманіття [62]. Методи оцінки ефективності можна класифікувати на теоретико-множинні підходи (наприклад, діаграми Ейлера), метрики різноманіття, імовірнісні методи (такі як блок-схеми надійності і ланцюги Маркова) та статичні підходи, які включають аналіз тестових даних, розрахунок релевантних метрик і вибір моделей зростання надійності [54]. Також доцільно додати методи ін’єкції несправностей для оцінки впливу змін у коді системи на безпеку і різноманітність. Серед цих методів варто відзначити експертно-орієнтовані підходи, які поєднують процеси розробки, оцінки безпеки та метрики складності продукту. Ці методи можуть інтегруватися з алгоритмами м’яких обчислень, такими як генетичні алгоритми чи нечітка логіка, що створює більш гнучку систему аналізу [54].

3.3 Дослідження практичних прикладів реалізації диверсності

3.3.1 Застосування диверсності дизайну

DMON – це система N-Variant Execution (NVX), яка функціонує як розподілена та використовує природну різноманітність платформ для підвищення стійкості до експлойтів пам’яті. Основна ідея полягає в моніторингу поведінки кількох варіантів однієї програми, запущених паралельно. Ці варіанти можуть працювати на різноманітних фізичних машинах з різними архітектурами та інтерфейсами додатків. Процес моніторингу і контролю поведінки програм

відбувається через мережевий зв’язок, що дозволяє перехресно перевіряти стан програм під час системних викликів [63]. DMON використовує модель лідера, який виконує операції вводу-виводу, а також послідовників, що мають завдання емулювати поведінку лідера, отримуючи результати його операцій. Різноманітність платформ зменшує кількість гаджетів, доступних для зловмисників. Серія експериментів, що базувалися на реалістичних серверних додатках, таких як Nginx, Lighttpd, Redis і ProFTPD, ще раз підтвердила ефективність цього рішення з прийнятними накладними витратами. Завдяки різнорідним архітектурам (ISA) та інтерфейсам (ABI) DMON забезпечує високий рівень стійкості до експлойтів. Для зменшення витрат процес було оптимізовано за рахунок асинхронної перевірки, кешування станів та оптимізації мережевих протоколів. Таким чином, DMON пропонує надійний захист від широкого спектра атак, зберігаючи прийнятну продуктивність [63].

Метод *N-Version Design* (диверсійність дизайну) для блокчейн-вузлів базується на одночасному використанні кількох незалежних реалізацій одного і того ж протоколу [64]. Такий підхід дозволяє значно підвищити надійність, доступність та безпеку систем навіть за умов нестабільного виконання чи цілеспрямованих атак. Його основна ідея полягає в тому, що різні реалізації одного протоколу (*N*-версії), розроблені різними командами або з використанням різних підходів, компенсують вразливості одна одної, знижуючи ймовірність спільних відмов. У статті представлено концепцію *N-Version Blockchain Node*, яка інтегрує кілька реалізацій клієнтів блокчейну в єдину архітектуру. Основні елементи цієї архітектури включають маршрутизацію запитів, обробку помилок і порівняння відповідей різних клієнтів [64]. Центральним компонентом є проксі-сервер, що виступає інтерфейсом для зовнішніх клієнтів, об'єднуючи підвузли, які працюють на основі різних реалізацій. У разі збою або атаки на одну з реалізацій система продовжує роботу за рахунок інших клієнтів, що забезпечує безперервність сервісу навіть у критичних ситуаціях. Було розроблено прототип N-ETH, який використовує різноманітні реалізації клієнтів Ethereum (GETH, BESU, ERIGON, NETHERMIND). У ході експериментів застосовувалися

стратегії ін’єкції помилок на рівні системних викликів для моделювання нестабільного виконання [64]. Результати показали, що різні реалізації клієнтів поводяться асиметрично за однакових умов нестабільності, тобто сильні сторони одних компенсують слабкі місця інших. Завдяки цьому доступність N-ETH досягла 98,5%, тоді як найкраща окрема реалізація забезпечувала лише 84,7%. Архітектура *N-Version Blockchain Node* має не лише високу доступність, але й забезпечує захист від цілеспрямованих атак, спрямованих на експлуатацію вразливостей окремих клієнтів. Цей метод є особливо важливим для бізнесів, які залежать від безперервної роботи блокчейн-вузлів, і відкриває нові можливості для створення стабільних і стійких систем у сучасному цифровому середовищі [64].

3.3.2 Дослідження застосувань диверсності, стосовно стратегії захисту рухомих цілей

Ций підхід використовує так звані «вузли-приманки», як метод обману та впровадження різноманітності операційних систем у рамках стратегії *Moving Target Defense* (MTD) для зменшення «поверхні» атаки в IoT-мережах. Це дозволяє знизити витрати на захист. Ефективність підходу оцінювалася за допомогою графічної моделі безпеки, котра базується на програмно-визначених мережах (SDN), що підтвердило її здатність мінімізувати вплив атак без втрати продуктивності IoT-системи/мережи [65].

Відповідна структура, представлена на рис. 3.3, складається з п'яти етапів впровадження та оцінки ефективності методики різноманітності ОС [65]:

Стадія 1. Користувач надає відповідну системну інформацію, таку як початкова топологія мережі та дані про вразливість вузла, генератору IoT для створення мережевої архітектури IoT.

Стадія 2. На цьому етапі розробляється початкове розгортання мережі IoT на основі приманки, пов'язаної з реальною мережею IoT.

Стадія 3. Модуль розгортання різноманітності генерує мережу IoT. Мережа IoT змінює ОС для вузлів відповідно до запропонованої методики, а

також оновлює інформацію про вразливості випадковим чином і на основі ІМ різноманітності ОС (операційних систем).

Стадія 4. Генератор моделей безпеки автоматично генерує значення HARM на основі як початкової мережі IoT, так і оновленої ОС, яка відображає всі шляхи атаки. Модель HARM складається з двох шарів. Верхній рівень представляє інформацію про досяжність, а нижній – інформацію про вразливості кожного вузла.

Стадія 5. Оцінювач використовує вихідні дані моделі HARM, яка є графічною моделлю безпеки для розрахунку результатів за заданими показниками оцінки.

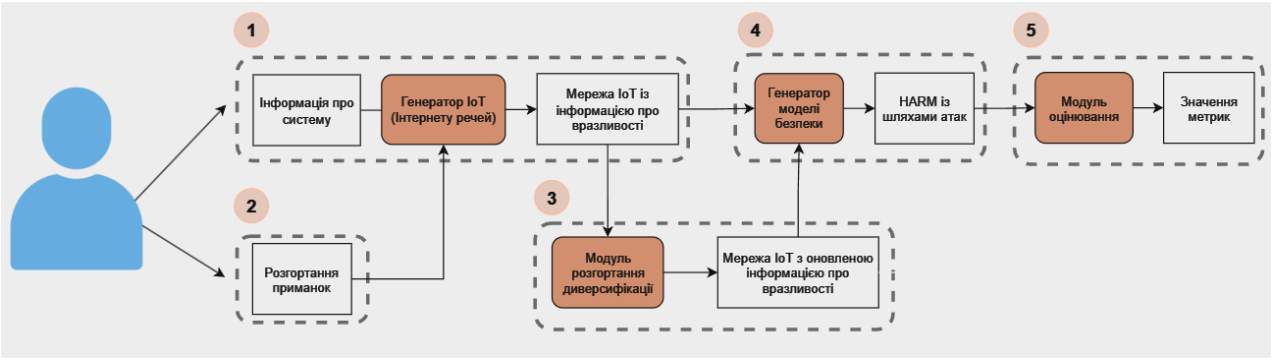


Рисунок 3.3 – Методика поєднує MTD (за узагальненням [65])

Постійна передача незашифрованих даних у хмару з IoT-пристроїв створює значні ризики, зокрема маніпуляції даними та запуск DDoS-атак через зараження пристроїв. Методи кіберзахисту, такі як MTD, створюють невизначеність для зловмисників через динамічну зміну компонентів ІКС, а процес імітації (тобто впровадження пасток), ефективно вводить нападника в оману, використовуючи приманки (накшталт Honeypot) та фальшиві дані [65].

Попередні дослідження переважно зосереджувалися або на MTD, або на кіберобмані. У роботі *«Evaluation for Combination of Shuffle and Diversity on Moving Target Defense Strategy for Cloud Computing»* досліджували інтеграцію двох MTD-технік – перемішування (Shuffle) та різноманітності (Diversity). Використовуючи модель Hierarchical Attack Representation Model (HARM),

автори оцінювали три ключові показники: системний ризик (Risk), вартість атаки (AC) та вигоду від атаки (RoA). Результати показали, що комбінація Shuffle+Diversity знижує ризик системи, підвищує вартість атак і зменшує їхню вигоду, роблячи атаки менш імовірними. Запропонована методика поєднує MTD і кіберобман, інтегровані в IoT-мережі на основі SDN, що забезпечує програмованість, масштабованість і оптимізацію витрат.

Модель включає 3 стратегії: – нульову; – випадкове різноманіття ОС; – різноманіття для критичних вузлів. Початкове розгортання відповідної IoT-мережі передбачає інтеграцію вузлів-приманок, що маскують реальні вузли. Модуль HARM виявляє критичні вузли та застосовує до них різноманітність ОС. Цей підхід адаптивно змінює середовище, знижуючи вразливість і підвищуючи захист. Для оцінки ефективності використовують вісім ключових метрик, серед яких: - MTTC (час до компрометації), PDR (доставка пакетів), System Risk, Attack Cost (AC), Return on Attack (RoA), кількість шляхів атак, EOC (додаткові витрати) та BCR (баланс вигод і витрат). Узагальнення результатів моделювання свідчить, що взаємодія з вузлами-приманками змушує зловмисників витрачати ресурси на хибні цілі, знижуючи вплив на реальні вузли [65]. Комбінація різноманітності ОС та технології пасток (т.з. кіберобману), ефективно мінімізує ризики атак, зберігаючи продуктивність IoT-мереж при оптимальних для них витратах. Однак часті мутації/зміни компонентів можуть впливати на продуктивність, особливо в обмежених IoT-середовищах. При цьому, «легка» структура на основі SDN допомагає оптимізувати безпеку при мінімальних супутних витратах [65].

3.3.3 Мережеве різноманіття, як інструмент оцінки надійності мереж

Модель, розглянута в роботі [66] будується на аналізі ресурсів у мережі, їхньої кількості, розподілу та схожості між ними. Її основним елементом є граф ресурсів, який відображає доступні в мережі сервіси, їхні взаємозв’язки та потенційні вразливості.

На основі цього графа визначаються шляхи атаки, які зловмисник може використати для компрометації критичних активів. Автори визначають три основні метрики [66]:

1) *Ефективна різноманітність ресурсів*, яка враховує кількість різних ресурсів у мережі, їхній розподіл і схожість та дозволяє оцінити масштаб потенційного поширення шкідливого ПЗ.

2) *Метрика мінімальних зусиль атак*и яка оцінює кількість і тип ресурсів, які зловмисник повинен використати для успішної атаки. Цей підхід дозволяє аналізувати причинно-наслідкові зв’язки між ресурсами та визначати найкоротші шляхи для атак.

3) *Ймовірнісна модель на основі баєсових мереж* враховує середні зусилля, які зловмисник повинен витратити, і забезпечує доповнення до інших підходів.

В ході циклу моделювань використовувалися реальні конфігурації мереж, що були доповнені випадково згенерованими компонентами. За результатами відповідних експериментів [66] зазначено, що збільшення диверсності ресурсів значно ускладнює «успіх» атак. Так, метрика мінімальних зусиль атаки показала, які частини мережі є найбільш вразливими, тоді як ймовірнісна модель дозволила оцінити загальний рівень безпеки мережі. Моделювання підтвердило, що диверсійність знижує ймовірність успішної атаки та підвищує стійкість мережі до шкідливих впливів. При цьому є і певні обмеження, що пов'язані з труднощами у визначенні точних характеристик ресурсів та їх взаємозв'язків, а також відсутністю «великих» масивів даних, які могли б посилити грунтовність перевірці моделей.

В роботі [67], реалізовано мережеву різноманітність на основі моделі розширеного графу ресурсів, як спосібу підвищення стійкості мереж до атак «нульового дня» (*т.з. Zero-day, «зеродеї»*), використовуючи три метрики мережевої різноманітності: - d1 *(ефективна кількість унікальних ресурсів)*; - d2 *(мінімальна кількість унікальних ресурсів, необхідних для компрометації критичного активу)* та d3 *(ймовірність компрометації активу)*.

В цьому випадку, вузли графу представляють мережеві сервіси, тоді як ребра вказують на причинно-наслідкові зв’язки між ними. Граф також враховує початкові умови та потенційні кінцеві цілі атак.

Оптимізація здійснюється шляхом використання генетичного алгоритму, який модифікує конфігурації мережевих сервісів для максимізації значень d1 та d2 або для мінімізації d3, при цьому враховуються бюджетні обмеження. Витрати на диверсифікацію також беруться до уваги, включаючи витрати на установку, обслуговування та можливі простії. Нижче наведено вираз вартісті диверсифікації в межах мережі. В даному випадку ідея полягає у визначенні: - скільки коштує заміна поточних сервісів на інші варіанти для всіх вузлів мережі. Ций вираз підсумовує витрати усіх таких замін для кожного вузла, щоб визначити загальну вартість диверсифікації для всієї системи [67].

$$Q_d = \sum_{i=1}^{|E|} DCM_{u_s(e_i)} \left(\vec{V_0}(i), \vec{V}(i)\right) \tag{3.1}$$

де $Q_d -$ загальна вартість диверсифікації

$|E| -$ кількість вузлів або елементів мережіб які підлягають дивенрсифікації

$DCM_{us(e_i)} -$ елемент матриці вартості диверсифікації для конкретного сервісу

$us(e_i) -$ сервісб пов^′ язанийіз змінною оптимізації $v(e\_i$)

$\vec{V}_0(i) -$ початковий стан сервісу на вузлі і

$\vec{V}(i) -$ новий стан сервісу після диверсифікації

Дослідження продемонструвало значне поліпшення стійкості мережі до невідомих атак внаслідок зменшення кількості спільних вразливостей (залежностей), що призвело до зниження кількості критичних атак на 40–60% [67]. Це стало можливим через необхідність використання більшої кількості унікальних експлойтів для їх успішного здійснення. Евристичний алгоритм забезпечує масштабованість, демонструючи прийнятний час обробки та точність навіть для великих мереж з численними вузлами. Навіть за умов суворих бюджетних обмежень можна досягти значних покращень у безпеці.

3.4. Узагальнення перспектив впровадження принципу диверсності

Важливим етапом є застосування широкого діапазону технологій та стратегій для формування багаторівневої системи захисту. Така інтеграція допомагає запобігти негативному впливу можливо скомпрометованих компонентів на всю структуру, гарантуючи її стабільність та ефективність навіть при локальних атаках. Використання штучного інтелекту (ШІ) для автоматизації процесів виявлення і реагування на загрози може значно зменшити час, необхідний для відповіді на нові атаки. Активне впровадження AI здатне не лише швидко виявляти аномалії, а й вживати відповідні контрзаходи без участі людини. Для збільшення стійкості мереж сучасних ІТ систем до атак, доцільно застосовувати інтегрований підхід, що одночасно впроваджує різні методи, наприклад, такі як: - динамічна реконфігурація мережі, багатоканальна маршрутизація та використання різних протоколів на кожному рівні. Таке комплексування помітно ускладнює сценарії атак, які прагнуть скористатися слабкими місцями в конфігураціях мережі. Запровадження дизайн-диверсності, тобто використання різних архітектурних (конструктивних) підходів при розробці інформаційних систем, значно підвищує їх надійність. Хоча цей метод може бути витратним і трудомістким, особливо для не критичних систем, він є суттєвим для зменшення ризиків. Для точного оцінювання ефективності впровадження принципу диверсності можна використовувати ймовірнісні методи, статистичні підходи та моделі теорії множин. Такі інструменти сприяють адекватній оцінці впливу диверсності на стабільність і захищеність інформаційних систем на різних етапах їхнього функціонування.

Впровадження принципу диверсності в кіберзахисті сприяє формуванню адаптивних систем, здатних ефективно реагувати на змінювані умови. Це забезпечує, що збій одного з компонентів не веде до повного колапсу всієї системи. Завдяки цим підходам підвищується стійкість до непередбачуваних загроз і гарантована стабільна діяльність навіть під час атак. Диверсність виступає також як стратегія, що забезпечує довготривалу стійкість сучасних інформаційних систем. Використання різноманітних елементів дозволяє

зменшити ймовірність повного збоїв, особливо коли атаки націлені на вразливі місця однорідних структур. Це забезпечує безперервність функціонування навіть у критичних ситуаціях. Крім того, застосування принципів, схожих на механізми природних екосистем, істотно знижує ризик масових відмов. Якщо більшість елементів системи виявляться скомпрометованими, інші залишаться працездатними. Таким чином, диверсність підвищує загальну стійкість і безпеку, забезпечуючи функціонування систем навіть під час часткових порушень.

Важливо також враховувати економічні чинники впровадження дизайн-диверсності. Для критично важливих інфраструктур такі інвестиції стають необхідними, хоча й потребують помітних фінансових та трудових витрат. Головним викликом є пошук оптимальних рішень, які сприятимуть зменшенню витрат без зниження рівня захисту.

Проте появлення нових технологій, таких як генеративний AI, ставить перед кібербезпекою нові виклики. Зловмисники все активніше використовують ці інновації для розробки більш складних атак. Це потребує постійного вдосконалення заходів безпеки та адаптації систем до сучасних загроз, що вимагає гнучкого підходу до управління ризиками.

Використання різноманітних технологій і методів є невід'ємною складовою для зміцнення стійкості систем перед зовнішніми атаками. Наприклад, комбінування різних протоколів та динамічної реконфігурації мереж може суттєво ускладнити зловмисникам доступ до критично важливих систем, навіть якщо деякі їх компоненти піддаються компрометації. ШІ може бути застосований не лише для автоматизації виявлення атак, але й для адаптації захисних заходів у реальному часі. Він має здатність ідентифікувати нові загрози та знижувати кількість успішних атак, оскільки може вчитися на основі попередніх спроб злому та миттєво реагувати на нові виклики.

Вибір різних мов програмування та архітектур при розробці програмного забезпечення дозволяє уникнути "монокультур", які створюють єдині точки вразливості для масових атак. Застосування різноманітних технологічних стратегій сприяє зниженню ймовірності одночасного компрометування великої

кількості систем. Принцип диверсності у мережах IoT дозволяє формувати більш стійкі структури до атак, оскільки використання різних протоколів і архітектур на рівні мережі підвищує їхню захищеність від масованих атак, спрямованих на однотипні елементи системи. Таким чином, інтеграція цих підходів є ключем до забезпечення надійності та безпеки сучасних інформаційних інфраструктур.

Висновки за розділом:

1. Якість впровадження диверсифікації визначається не лише рівнем безпеки, але й стабільністю продуктивності програмного забезпечення. Метою диверсифікації є підвищення стійкості систем до атак, при цьому вона має мінімально впливати на ключові параметри продуктивності, зокрема на швидкість і ресурсозатратність.

2. Оцінка ефективності впровадження диверсифікації програмного забезпечення залишається важливим завданням, що потребує уваги та досліджень. Дослідники використовують різноманітні непрямі методи оцінки, такі як аналіз ентропії, дослідження вразливостей через конкретний код, логічна аргументація та тестування атак, а також фреймворки, метрики різноманітності, теоретико-множинні методи, статистичні підходи й експертні оцінки для визначення ефективності диверсифікації.

3. Сучасні виклики в кібербезпеці включають зростання складності атак і неможливість прогнозування всіх потенційних вразливостей [66]. Для вирішення цих проблем розроблено багато моделей та метрик, які базуються на принципах різноманітності, що ускладнює повторне використання експлойтів та інструментів, а також знижує ймовірність поширення атак через мережу.

4. Проактивні методики кіберзахисту з використанням різноманітності операційних систем або вузлів-приманок також набувають актуальності. Диверсифікація є складним завданням, особливо якщо її виконувати вручну, оскільки навіть у невеликих мережах існує складна взаємодія між різними сервісами. Ці взаємозв'язки створюють велике число залежностей, які важко аналізувати і контролювати без спеціалізованих інструментів, а також вимагають прогнозування всіх можливих шляхів атак і забезпечення різноманітності в

компонентах мережі. Можливим рішенням цих туднощей є автоматизовані фреймворки [67] для диверсифікації та систематизовані підходи, такі як оптимізація конфігурації мережевих послуг на основі алгоритмів.

5. Запровадження принципу диверсності в сучасних ІКС є важливим етапом у забезпеченні безпеки та надійності сучасних інформаційних систем.

6. Різноманіття технологій і методів дозволяє створювати більш стійкі і адаптивні рішення, які здатні ефективно протистояти постійно еволюціонуючим загрозам. Хоча під час впровадження таких підходів можуть виникнути економічні та технічні труднощі, принцип диверсності залишається необхідною умовою для формування ефективної та захищеної інфраструктури в умовах сучасних кіберзагроз.

7. Адаптація до змінюваного середовища безпеки стає ключовою для довготривалої стійкості інформаційних систем, що підкреслює важливість диверсифікації в стратегіях кіберзахисту.

ВИСНОВКИ

Ефективне забезпечення безпеки сучасних ІКС потребує комплексного підходу. Це включає детальне дослідження слабких місць системи, оцінку ризиків за допомогою різних методів, створення планів для підтримки надійності та дотримання міжнародних стандартів, таких як IEC 61508. Цей підхід допомагає не тільки зменшити ймовірність інцидентів, але й забезпечує стабільну роботу навіть у разі технічних збоїв або кібератак.

Сучасні ІКС часто стикаються з постійним зростанням кількості і складності загроз. Використання принципу різноманітності дозволяє створювати альтернативні варіанти ПЗ, компонентів і архітектур. Це зменшує ризик одночасного виходу з ладу багатьох елементів і підвищує їх незалежність. Завдяки такому підходу, зловмисникам стає значно складніше здійснювати однотипні атаки, оскільки вони втрачають можливість використовувати універсальні інструменти для порушення роботи багатьох систем. Наприклад, методи статичної диверсифікації, такі як різноманіття в дизайні або апаратних рішеннях, добре працюють у ситуаціях, коли важливо уникнути однакових вразливостей між компонентами. Динамічна диверсифікація, наприклад, зміна програмного середовища або шифрування даних у реальному часі, дозволяє активно реагувати на можливі загрози, створюючи додаткові перешкоди для атакуючих. Мережева диверсифікація, зі свого боку, забезпечує високу гнучкість шляхом розподілу навантажень та змін конфігурацій у мережі, що особливо ефективно для великих інформаційних систем. Кожен із цих методів має свої переваги: – статичні підходи забезпечують стабільність, динамічні – гнучкість, а мережеві– адаптивність і захист від найбільш поширених вразливостей у зв’язках між системами (чи їх елементами).

Оцінка ефективності впровадження принципу різноманітності залишається важливим завданням, яке вимагає врахування кількох ключових критеріїв. Це включає аналіз впливу різних стратегій на продуктивність і

ресурсозатратність, вивчення вразливостей у контексті конкретних сценаріїв атак, а також моделювання можливих ситуацій використання. Особливу увагу слід приділяти розробці метрик, які дозволяють кількісно оцінити рівень впровадження різноманітності та її ефективність. Результати таких оцінок забезпечують обґрунтованість вибору стратегій, оптимізуючи баланс між безпекою та продуктивністю систем.

У нинішніх умовах постійних атак різноманіття рішень і технологій надає можливість створювати більш захищені й стійкі до загроз системи. Хоча впровадження таких підходів може супроводжуватися додатковими витратами, їх ефективність у протидії потенційним загрозам робить ці інвестиції виправданими. Важливо підкреслити, що саме поєднання різних методів диверсифікації, а не використання одного підходу, створює найбільшу стійкість систем до потенційних атак.

Таким чином, різноманіття, як стратегія кіберзахисту – це не лише спосіб покращення поточного рівня безпеки, а й можливість зробити системи більш адаптивними до змін середовища й умов їх функціонування. У підсумку, реалізація таких рішень сприяє створенню інфраструктури та елементів ІКС, які ефективно функціонують навіть у складних умовах деструктивного впливу на них, тим самим підвищуючи її спроможність парирувати нові виклики безпеки.

СПИСОК ДЖЕРЕЛ ПОСИЛАННЯ

1. Класифікації-загроз-інформаційної-безпеки. Режим доступу: https://bit.ly/3AYM98B
2. Що таке інформаційна безпека (InfoSec)? Режим доступу: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-information-security-infosec
3. Дудикевич В.Б. Основи інформаційної безпеки: навч. посібник/ В.Б. Дудикевич, В.О. Хорошко, Ю.Є. Яремчук.- Вінниця: ВНТУ, 2018.- 316 с.
4. Jouini M. Classification of Security Threats in Information Systems [Electronic resource] / Mouna Jouini, Latifa Ben Arfa Rabai, Anis Ben Aissa // Procedia Computer Science. – 2014. – Vol. 32. – P. 489–496. Режим доступу: https://doi.org/10.1016/j.procs.2014.05.452
5. Гребенюк А.М. Основи управління інформаційною безпекою: навч. посіб. / А.М. Гребенюк, Л.В. Рибальченко. – Дніпро: ДДУВС, 2020. – 144 с.
6. Гончаренко, Є. О. Вибір підходу до оцінки ризиків інформаційної безпеки для підприємств роздрібної торгівлі : магістерська дис. : 125 Кібербезпека / Гончаренко Євгенія Олександрівна. - Київ. - 2019. - 92 с.
7. С. В. Кавун, В. В. Носов, О. В. Манжай. «Інформаційна безпека». - 2008.
8. ZOLOTAR O. Classification of threats to information security [Електронний ресурс] / O. ZOLOTAR, I. TRUBIN // INFORMATION AND LAW. – 2013. – № 3(9). – С. 105–112. Режим доступу: https://doi.org/10.37750/2616-6798.2013.3(9).272386
9. Understanding Cybersecurity Attacks: Passive and Active Threats (Part 3) Режим доступу: https://medium.com/@MakeComputerScienceGreatAgain/understanding-cybersecurity-attacks-passive-and-active-threats-part-3-d2c5df16bdfa

10. What is threat management? Режим доступу: https://www.cisco.com/site/us/en/learn/topics/security/what-is-threat-management.html
11. ISO/IEC 27005:200.335. Інформаційні технології. Методи захисту. Управління ризиками інформаційної безпеки . – Київ: ДП "УкрНДНЦ", 200.335. – 60 с
12. Guide for conducting risk assessments [Electronic resource]. – Gaithersburg, MD : National Institute of Standards and Technology, 2012. Режим доступу: https://doi.org/10.6028/nist.sp.800-30r1
13. Функційна безпека індустріальних систем. Режим доступу: https://tk185.appau.org.ua/whitepapers/aCampus-whitepaper-IEC-61508+++.pdf
14. Overview of IEC 61508 & Functional Safety. Режим доступу: https://assets.iec.ch/public/acos/IEC%2061508%20&%20Functional%20Safety-2022.pdf?2023040501
15. Clarification of the Cybersecurity and Functional Safety Interrelationship in Industrial Control Systems: Barrier Concepts and Essential Functions [Electronic resource] / Bálint Z. Téglásy [et al.] // Proceedings of the 29th European Safety and Reliability Conference (ESREL), 1–5 November 2020. – Singapore, 2020. Режим доступу: https://doi.org/10.3850/978-981-14-8593-0_3786-cd
16. What Safety Integrity Level (SIL) Means and How to Calculate It. Режим доступу:https://blog.msasafety.com/what-safety-integrity-level-means/#:~:text=SIL%20stands%20for%20Safety%20Integrity,failure%20on%20demand%20(PFD)
17. Функційна безпека у відповідності з ДСТУ EN 61508. Режим доступу: https://tk185.appau.org.ua/guide/aCampus-users-guides-IEC61508+++.pdf
18. Preschern, C., Kajtazovic, N., & Kreiner, C. (2013). Catalog of safety tactics in the light of the IEC 61508 safety lifecycle. In Proceedings of VikingPLoP 2013 Conference (p. 79). Режим доступу:

https://graz.elsevierpure.com/en/publications/catalog-of-safety-tactics-in-the-light-of-the-iec-61508-safety-li

19. Bell R. Introduction and Revision of IEC 61508 [Electronic resource] / Ron Bell // Advances in Systems Safety. – London, 2010. – P. 273–291. Режим доступу: https://doi.org/10.1007/978-0-85729-133-2_16
20. Allianz Risk Barometer 2024. Режим доступу: https://commercial.allianz.com/news-and-insights/reports/allianz-risk-barometer.html
21. Лєсная, Ю., & Малахов, С. (2023). Аналіз розвитку, типові цілі та механізми здійснення фішингових атак. Комп'ютерні науки та кібербезпека, (1), 6-27. Режим доступу: https://doi.org/10.26565/2519-2310-2023-1-01
22. Кібератаки 2022-2023: огляд найбільших інцидентів, та що нас чекає у 2024 році. Режим доступу: https://www.h-x.technology/ua/blog-ua/cyber-threats-forecast-2024-ua
23. International Electrotechnical Commission, IEC 61508-2: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, Brussels: IEC, 2010.
24. Wang C. Explicit and implicit methods for probabilistic common-cause failure analysis [Electronic resource] / Chaonan Wang, Liudong Xing, Gregory Levitin // Reliability Engineering & System Safety. – 2014. – Vol. 131. – P. 175–184. Режим доступу: https://doi.org/10.1016/j.ress.2014.06.024
25. Quantitative Evaluation of Common Cause Failures in High Safety-significant Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants [Electronic resource] / Han Bao [et al.] // Reliability Engineering & System Safety. – 2022. – P. 108973. Режим доступу: https://doi.org/10.1016/j.ress.2022.108973
26. Common cause failures (CCF). What are they and how are they mitigated. Режим доступу: https://bit.ly/3B0G1Ntz

27. Common Cause Failures in Safety Instrumented Systems Beta-factors and equipment specific checklists based on operational experience / Stein Hauge [et al.]. – Norway : SINTEF Technology and Society, 2015. – 72 p. Режим доступу: https://www.sintef.no/globalassets/project/pds/reports/sintef-a26922-common-cause-failures-in-safety-instrumented-systems-beta....pdf
28. IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems. Geneva: International Electrotechnical Commission. 2010
29. ISO/TR 12489. Petroleum, petrochemical and natural gas industries – Reliability modelling and calculation of safety systems. Ver. 01, 2013
30. Hauge S. Reliability Prediction Method for Safety Instrumented Systems – PDS Example collection, 2013 Edition / Stein Hauge, Solfrid Håbrekke, Mary Ann Lundteigen. – Norway : SINTEF Technology and Society Safety Research, 2010. – 50 p. Режим доступу: https://www.sintef.no/globalassets/project/pds/reports/pds-example-collection-24-01-11_open.pdf
31. Belland J. R. Modeling common cause failures in diverse components with fault tree applications [Electronic resource] / Joseph R. Belland // 2017 Annual Reliability and Maintainability Symposium (RAMS), Orlando, FL, 23–26 January 2017. – [S. l.], 2017. Режим доступу: https://doi.org/10.1109/ram.2017.7889659
32. What is a Common Cause Failure? [Електронний ресурс] / Doug Nix. (2019). ISO 13849-1 Analysis — Part 6: CCF — Common Cause Failures. Режим доступу: https://machinerysafety101.com/2017/03/20/iso-13849-1-analysis-part-6-ccf/
33. Jones H. Common Cause Failures and Ultra Reliability [Electronic resource] / Harry Jones // 42nd International Conference on Environmental Systems, San Diego, California. – Reston, Virigina, 2012. Режим доступу: https://doi.org/10.2514/6.2012-3602

34. Kunt, T. Common Cause Failure: What Are They and How to Mitigate Them? [Electronic resource] / Tekin Kunt // PSRG Inc. – 2021. Режим доступу: https://bit.ly/3B0ZUnl
35. Terry Masters, 2024. What Is an Independent Failure? Режим доступу: https://www.smartcapitalmind.com/what-is-an-independent-failure.htm
36. Xie L. Common cause failures and cascading failures in technical systems: Similarities, differences and barriers [Electronic resource] / L. Xie, M. A. Lundteigen, Y. L. Liu // Safety and Reliability – Safe Societies in a Changing World. – [S. l.], 2018. – P. 2401–2407. Режим доступу: https://doi.org/10.1201/9781351174664-302
37. Xie L. Safety Barriers Against Common Cause Failure and Cascading Failure: Literature Reviews and Modeling Strategies [Electronic resource] / L. Xie, M. A. Lundteigen, Y. L. Liu // 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, 16–19 December 2018. – [S. l.], 2018. Режим доступу: https://doi.org/10.1109/ieem.2018.8607769
38. Čepin M. Assessment of Power System Reliability [Electronic resource] / Marko Čepin. – London : Springer London, 2011. Режим доступу: https://doi.org/10.1007/978-0-85729-688-7
39. Reliability modeling and analysis of communication networks [Electronic resource] / Waqar Ahmad [et al.] // Journal of Network and Computer Applications. – 2017. – Vol. 78. – P. 191–215. Режим доступу: https://doi.org/10.1016/j.jnca.2016.11.008
40. Morikawa A. Safety design concepts for statistical machine learning components toward accordance with functional safety standards. / Yutaka Matsubara, Nagoya University, 2020. – 13 p. - (Preprint). Режим доступу: https://doi.org/10.48550/arXiv.2008.01263
41. Marsh McLennan, Zurich Insurance Group. The global risks report 2023 - 18th edition - today's crisis, tomorrow's catastrophes. Режим доступу:

https://www.zurich.com/knowledge/topics/global-risks/the-global-risks-report-2023

42. Coffman, J. Diversity as an Enabler for Cyber Resilience [Electronic resource] / Joel Coffman and Andrew S. Gearhart – 2019. Режим доступу: https://www.sintef.no/globalassets/project/pds/reports/sintef-a26922-common-cause-failures-in-safety-instrumented-systems-beta....pdf
43. Бойко, К., Чудновський, В. & Малахов, С. (2024). Узагальнення напрямів програмної диверсності для покращення безпеки інформаційних систем. Proceedings of the XIII International Scientific and Practical Conference. Valencia, Spain. International Science Group. 2024. Pp. 299-308. Режим доступу: https://isg-konf.com/cultural-and-artistic-processes-in-the-context-of-the-european-scientific-space
44. Яремчук, К., Воскобойников, Д., & Мелкозьорова, О. (2022). Сучасні загрози та способи забезпечення безпеки веб-застосунків. Комп'ютерні науки та кібербезпека, (2), 28-34. Режим доступу: https://doi.org/10.26565/2519-2310-2022-2-03
45. Богданова, Е., & Малахов, С. (2022). Узагальнення специфіки застосування експлойтів. Збірник наукових праць SCIENTIA, (Vol.2), 28-32. Режим доступу: https://previous.scientia.report/index.php/archive/issue/view/24.06.2022
46. Compiler-Generated Software Diversity [Electronic resource] / Todd Jackson [et al.] // Advances in Information Security. – New York, NY, 2011. – P. 77–98. Режим доступу: https://doi.org/10.1007/978-1-4614-0977-9_4
47. Global cybersecurity outlook 2024. Режим доступу: https://www3.weforum.org/docs/WEF_Global_Cybersecurity_Outlook_2024.pdf
48. Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks [Electronic resource] / Mengyuan Zhang [et al.] // IEEE Transactions on Information Forensics and Security. – 2016. –

Vol. 11, no. 5. – P. 1071–1086. Режим доступу: https://doi.org/10.1109/tifs.2016.2516916

49. Mayo, J. R., Torgerson, M. D., Walker, A. M., Armstrong, R. C., Allan, B. A., & Pierson, L. G. (2010). The theory of diversity and redundancy in information system security: LDRD final report (No. SAND2010-7055). Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA (United States). Режим доступу: https://doi.org/10.2172/992781
50. Gashi I. Diversity, Safety and Security in Embedded Systems: Modelling Adversary Effort and Supply Chain Risks [Electronic resource] / Ilir Gashi, Andrey Povyakalo, Lorenzo Strigini // 2016 12th European Dependable Computing Conference (EDCC), Gothenburg, Sweden, 5–9 September 2016. – [S. l.], 2016. Режим доступу: https://doi.org/10.1109/edcc.2016.27
51. World Economic Forum. (2023). Global risks report 2023 (18th ed.). Режим доступу: www3.weforum.org/docs/WEF_Global_Risks_Report_2023.pdf
52. Бойко, К., Гальцева, І., & Малахов, С. (2024). Програмна диверсність як інструмент забезпечення функціональної безпеки сучасних інформаційних систем. Збірник наукових праць ΛΌГОΣ, 170-178. Режим доступу: https://go-vropejska.esclick.me/1dKnhzlofriG65Ls8k
53. Design-for-Diversity for Improved Fault-Tolerance of TMR Systems on FPGAs [Electronic resource] / Rizwan A. Ashraf [et al.] // 2011 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2011), Cancun, Mexico, 30 November – 2 December 2011. – [S. l.], 2011. Режим доступу: https://doi.org/10.1109/reconfig.2011.26
54. SoK: Automated Software Diversity [Electronic resource] / Per Larsen [et al.] // 2014 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 18–21 May 2014. – [S. l.], 2014. Режим доступу: https://doi.org/10.1109/sp.2014.25
55. Яновский М. Э. Оценка диверсности программного обеспечения с использованием косвенных метрик / М. Э. Яновский // Радіоелектронні і комп’ютерні системи, 2009, № 7 (41) . - С. 255-260.

56. Diversity in Cybersecurity [Electronic resource] / John Knight [et al.] // Computer. – 2016. – Vol. 49, no. 4. – P. 94–98. Режим доступу: https://doi.org/10.1109/mc.2016.102
57. Lundquist G. R. Searching for software diversity [Electronic resource] / Gilmore R. Lundquist, Vishwath Mohan, Kevin W. Hamlen // NSPW '16: New Security Paradigms Workshop 2016, Granby Colorado USA. – New York, NY, USA, 2016. Режим доступу: https://doi.org/10.1145/3011883.3011891
58. CROW: Code Diversification for WebAssembly [Electronic resource] / Javier Cabrera Arteaga [et al.] // Workshop on Measurements, Attacks, and Defenses for the Web, Virtual. – Reston, VA, 2021. Режим доступу: https://doi.org/10.14722/madweb.2021.23004
59. Obaidat I. DAEDALUS: Defense Against Firmware ROP Exploits Using Stochastic Software Diversity [Electronic resource] / Islam Obaidat, Meera Sridhar, Tavakoli Fatemeh. – 2024. – 20 p. – (Preprint). Режим доступу: https://doi.org/10.48550/arXiv.2401.16234
60. Blackburn C. Rave: A Modular and Extensible Framework for Program State Re-Randomization [Electronic resource] / Christopher Blackburn, Xiaoguang Wang, Binoy Ravindran // CCS '22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles CA USA. – New York, NY, USA, 2022. Режим доступу: https://doi.org/10.1145/3560828.3564008
61. Michel Kinsy. Sphinx: A Secure Architecture Based on Binary Code Diversification and Execution Obfuscation [Electronic resource] / Michel Kinsy. Boston Area Architecture Workshop (BARC18), 2018. – 2 p. – (Preprint). Режим доступу: https://doi.org/10.48550/arXiv.1802.04259
62. Wang X. et al. A Framework to Secure Applications with ISA Heterogeneity //The 9th Workshop on Systems for Multi-core and Heterogeneous Architectures (SFMA). – 2019. Режим доступу: https://www.researchgate.net/publication/331559202_A_Framework_to_Secure_Applications_with_ISA_Heterogeneity

63. Voulimeneas, A., Song, D., Parzefall, F., Na, Y., Larsen, P., Franz, M., & Volckaert, S. (2019). DMON: A distributed heterogeneous n-variant system. arXiv preprint arXiv:1903.03643. Режим доступу: https://doi.org/10.48550/arXiv.1903.03643
64. Ron, J., Soto-Valero, C., Zhang, L., Baudry, B., & Monperrus, M. (2023). Highly Available Blockchain Nodes With N-Version Design. IEEE Transactions on Dependable and Secure Computing. Режим доступу: https://doi.org/10.1109/TDSC.2023.3346195
65. Rehman, Z., Gondal, I., Ge, M., Dong, H., Gregory, M., & Tari, Z. (2024). Proactive defense mechanism: Enhancing IoT security through diversity-based moving target defense and cyber deception. Computers & Security, 139, 103685. Режим доступу: https://doi.org/10.1016/j.cose.2023.103685
66. Wang, L., Zhang, M., Jajodia, S., Singhal, A., & Albanese, M. (2014). Modeling network diversity for evaluating the robustness of networks against zero-day attacks. In Computer Security-ESORICS 2014: 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II 19 (pp. 494-511). Springer International Publishing. Режим доступу: https://link.springer.com/chapter/10.1007/978-3-319-11212-1_28
67. Borbor, D., Wang, L., Jajodia, S., & Singhal, A. (2019). Optimizing the network diversity to improve the resilience of networks against unknown attacks. Computer Communications, 145, 96-112. Режим доступу: https://doi.org/10.1016/j.comcom.2019.06.004

ДОДАТОК А

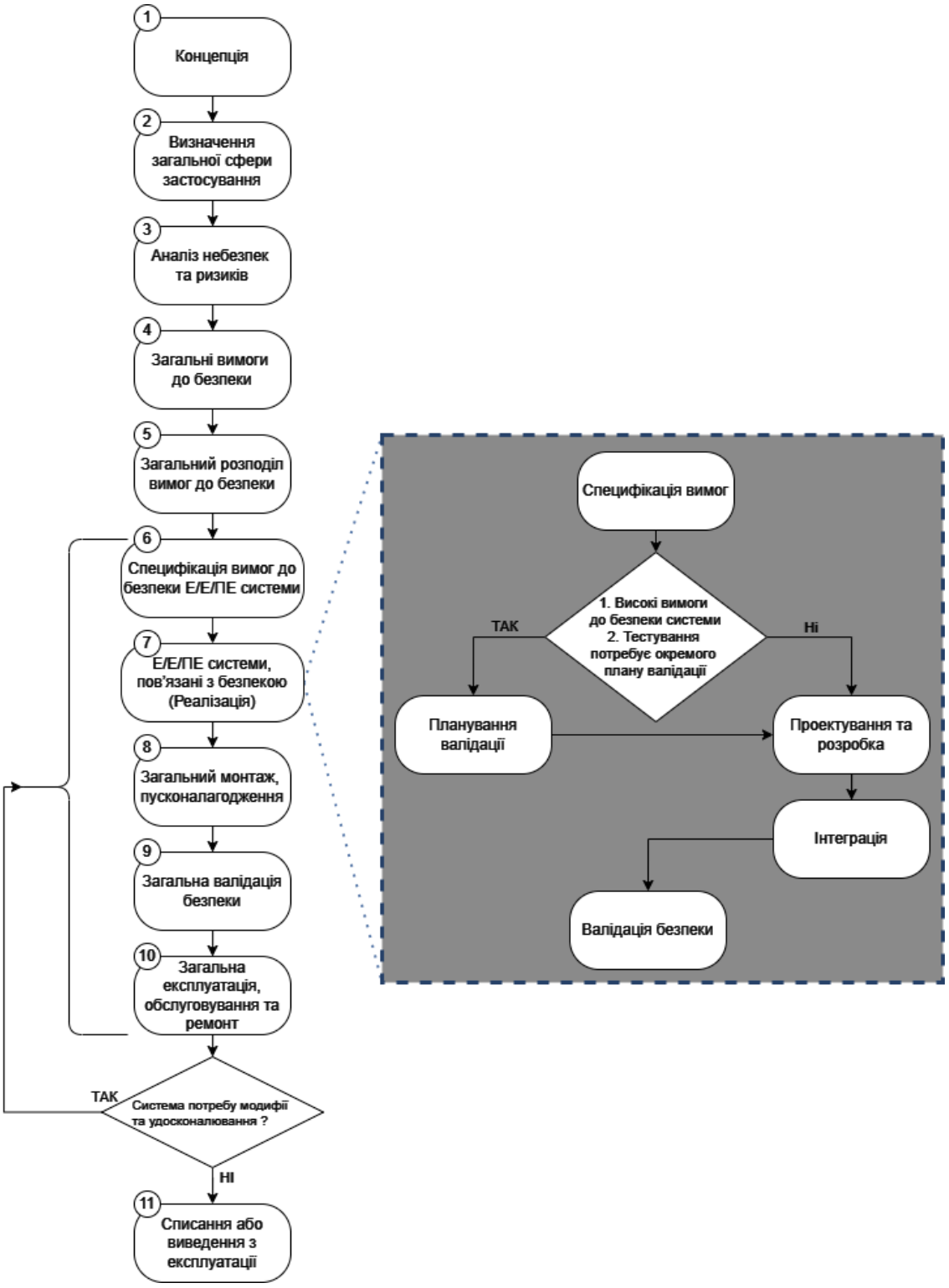


Рис. А.1 - Загальний життєвий цикл відповідно до стандарту IEC 61508

ДОДАТОК Б

Таблиця Б.1 – Зміст та особливості порівняння різних методів оцінки ризиків

Методи оцінки ризиків	Наявність перекладу	Орієнтація на розмір підприємства	Наявність програмного інструментарію	Фази підходу *(етапи)*	Тип оцінки ризику	Обробка ризиків	Необхідність в ресурсах
COBIT 5	Доступний	Можливе застосування для організацій різного розміру і галузей	Програмні інструменти доступні	Ідентифікація процесів та активів Оцінка вразливостей Визначення ймовірностей Оцінка ризиків на основі бізнес-цілей	Змішана оцінка (кількісна і якісна	•Прийняття •Запобігання •Планування	Необхідне залучення співробітників як з боку IT, так і з боку бізнесу. Можливе залучення третіх сторін
ISO 27005	Доступний	Для організацій будь-якого розміру та галузі	Програмні інструменти доступні	Оцінка обставин Визначення ризиків Аналіз ризиків Оцінка ризиків Планування ризиків Прийняття рішень	Основні рекомендації для якісної або кількісної оцінки	•Модифікація •Прийняття •Усунення •Розподіл	Потрібні ресурси з IT та бізнесу. Можливе залучення зовнішніх організацій для впровадження
NIST SP800-30	Недоступний	Для організацій будь-якого розміру, основний акцент на федеральних установах США	Програмні інструменти доступні	Аналіз системи Визначення загроз Виявлення вразливостей Оцінка контрольних заходів Оцінка ймовірності та впливу Ризик-менеджмент Рекомендації щодо контролю Документування результатів	Комбінована якісна та кількісна оцінка	•Прийняття •Запобігання •Обмеження •Планування •Дослідження та повідомлення •Передача	Потрібні ресурси з IT та бізнесу. Можливе залучення зовнішніх організацій для впровадження