КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (32), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.32.1195
УДК 004.421.5

Бугай Данило Артемович
студент кафедри «Захист інформації»
Національний Університет "Львівська Політехніка", Львів, Україна
ORCID: 0009-0001-9858-0482
danylo.buhai.kb.2022@lpnu.ua

Власюк Ігор Дмитрович
аспірант кафедри «Захист інформації»
Національний Університет "Львівська Політехніка", Львів, Україна
ORCID: 0009-0008-3600-750X
ihor.d.vlasiuk@lpnu.ua

Сусукайло Віталій Андрійович
доктор філософії, старший викладач кафедри «Захист інформації»
Національний Університет "Львівська Політехніка", Львів, Україна
ORCID: 0000-0003-4431-9964
vitalii.a.susukailo@lpnu.ua

Курій Євгеній Олегович
доктор філософії, старший викладач кафедри «Захист інформації»
Національний Університет "Львівська Політехніка", Львів, Україна
ORCID: 0000-0002-3423-5655
Yevhenii.O.Kurii@lpnu.ua

МОДЕЛЬ ПЛАНУ РЕАГУВАННЯ НА ІНЦИДЕНТИ ПОВ'ЯЗАНІ З ВИКОРИСТАННЯМ СЛАБКИХ ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

Анотація. У статті проведено аналіз генераторів псевдовипадкових чисел (ГПВЧ) у криптографії, розглянуто їхні теоретичні основи, класифікацію та основні вимоги до безпеки. Зокрема, розглянуто, як детерміновані алгоритми на основі початкового seed формують довгі послідовності бітів, практично невідрізнені від істинно випадкових, тому було проаналізовано різні типи генераторів: від простих лінійних конгруентних генераторів (LCG) і Mersenne Twister, що добре підходять для моделювання та чисельних обчислень, до криптографічно стійких алгоритмів ChaCha20 і Yarrow, які забезпечують непередбачуваність та стійкість до прогнозування вихідної послідовності. Запропоновано універсальну модель плану реагування на інциденти, пов'язані з вразливостями генераторів псевдовипадкових чисел та алгоритм автоматизації реагування на зазначені інциденти. Результати дослідження можуть бути застосовані розробниками криптографічних алгоритмів та фахівцями з інформаційної безпеки для покращення оцінки щодо вибору генераторів псевдовипадкових чисел у практичних та навчальних цілях.

Ключові слова: генератор псевдовипадкових чисел, Mersenne Twister, ChaCha20, Yarrow, псевдовипадковість, seed, статистичні вимоги, безпека даних, CI/CD, інциденти інформаційної безпеки.

ВСТУП

Стрімкий розвиток цифрових технологій і збільшення обсягів оброблюваних даних висувають підвищені вимоги до захисту інформації та надійності її обробки. Криптографічні методи забезпечують конфіденційність, цілісність та контроль доступу, а їх ефективність значною мірою визначається якістю допоміжних алгоритмічних

компонентів. Генератори псевдовипадкових чисел (ГПВЧ) займають важливе місце серед таких компонентів, оскільки вони формують послідовності, необхідні для безпечного функціонування криптографічних систем.

Аналіз генераторів псевдовипадкових чисел охоплює дослідження математичних моделей, принципів побудови та методів оцінювання алгоритмічної надійності. У наукових роботах здійснюється класифікація генераторів за алгоритмічними підходами, визначаються їх основні властивості та критерії, що забезпечують придатність для застосування в криптографії. Дослідження таких аспектів дозволяє оцінити сильні та слабкі сторони існуючих методів генерації псевдовипадкових послідовностей.

Постановка проблеми. У сучасних інформаційних системах генератори псевдовипадкових чисел відіграють ключову роль у забезпеченні автентифікації, шифрування та управління сесіями, однак використання нестійких алгоритмів, таких як Mersenne Twister чи Linear Congruential Generator, недостатня ентропія в механізмах на кшталт Yarrow або помилки реалізації режимів автентифікованого шифрування, зокрема при застосуванні ChaCha20-Poly1305 із повторним nonce, створюють ризик передбачуваності вихідних значень і компрометації даних. Відсутність структурованої моделі реагування, яка б враховувала тип генератора та характер криптографічної помилки, ускладнює мінімізацію наслідків інцидентів і відновлення безпечного стану системи, що стимулює необхідність розроблення узагальненого підходу до реагування на такі порушення.

Аналіз останніх досліджень і публікацій. У фундаментальній праці з криптографії Handbook of Applied Cryptography авторів Menezes A., van Oorschot P., Vanstone S. ґрунтовно розглянуто теоретичні засади побудови генераторів псевдовипадкових чисел та сформульовано основні вимоги до їх криптографічної стійкості [1], а сучасні дослідження акцентують увагу на тестуванні криптографічно захищених генераторів із використанням статистичних методик, зокрема NIST SP 800-22 [6], Almaraz Luengo E., Román Villaizán J. Криптографічно захищені генератори псевдовипадкових чисел: аналіз та тестування із застосуванням статистичного набору тестів NIST [2]. Проблематика вразливості некриптографічних генераторів, таких як Mersenne Twister, а також питання їх відмінності від CSPRNG детально проаналізовані у дослідженні Stoen H. The Mersenne Twister and Cryptographically-Secure PRNGs , тоді як практичні аспекти реалізації ChaCha20-Poly1305 регламентуються в RFC 8439 і супроводжуються прикладами вразливостей, зафіксованих у NVD CVE-2019-1543 . Разом із тим у наявних публікаціях основна увага зосереджена на математичному або прикладному аналізі алгоритмів, тоді як недостатньо досліджено питання інтеграції результатів криптографічного аналізу в структурований процес реагування на інциденти відповідно до підходів NIST SP 800-61 [7], що й становить невирішену частину проблеми, якій присвячена дана стаття.

Мета дослідження. Метою статті є дослідження вразливостей та інцидентів пов’язаних зі слабкими генераторами псевдовипадкових чисел та формування моделі реагування на інциденти пов’язані з використанням слабких генераторів псевдовипадкових чисел.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Генератори псевдовипадкових чисел у криптографії визначаються як алгоритми, що дозволяють на основі невеликої кількості істинно випадкових бітів формувати

більш довгі послідовності, які виглядають випадковими. Псевдовипадковість- це здатність вихідної послідовності формуватися детерміновано, залишаючись при цьому практично невідрізненою від істинно випадкової для стороннього спостерігача [1]. Іншими словами, хоча алгоритм виконує обчислення за заданим правилом, результат виглядає випадковим і не дозволяє ефективно передбачити наступні біти або відновити внутрішній стан без знання вихідних параметрів.

Головним елементом, який визначає детермінованість алгоритму, є seed - початкова послідовність бітів, що задає стартову точку генерації псевдовипадкових бітів [1]. Seed може бути фіксованим для відтворюваності послідовності, але у практичних криптографічних застосуваннях його зазвичай беруть випадковим, щоб забезпечити непередбачуваність вихідних даних. Саме через seed алгоритм здатний відтворювати послідовність за потреби, водночас зберігаючи властивості псевдовипадковості.

Для ілюстрації принципу роботи генераторів псевдовипадкових чисел можна розглянути лінійний конгруентний генератор (LCG), описаний у науковій літературі як один із простих прикладів детермінованого алгоритму формування псевдовипадкових послідовностей. У цьому алгоритмі наступний елемент послідовності обчислюється за рекурентною формулою:

$$x_n = ax_{n-1} + b \; mod \; m, \tag{1}$$

де X_{n-1} – початкове значення (seed), а a, c, m – константи, що визначають властивості

генератора. Змінюючи seed, можна отримати різні псевдовипадкові послідовності, водночас використання одного й того ж початкового значення дозволяє відтворювати послідовність, що підкреслює детермінованість алгоритму. Такі генератори добре ілюструють, як невелика істинно випадкова послідовність може бути розширена до довшої послідовності з властивостями, близькими до випадкових [1]. Також, орім дослідження псевдовипадкових генераторів, огляд включає аналіз істинно випадкових послідовностей, що є еталонними для оцінки властивостей генераторів.

Опираючись на це визначення, істинно випадкові послідовності характеризуються статистичною незалежністю та рівномірністю розподілу бітів, що забезпечує максимальну непередбачуваність кожного наступного елемента послідовності. На відміну ж від псевдовипадкових генераторів, що детерміновано формують вихідну послідовність на основі seed, як зазначалося раніше, істинні випадкові послідовності не залежать від попередніх значень та не можуть бути відтворені. Тому, аналіз властивостей випадкових послідовностей дозволяє встановити критерії, за якими оцінюється наближеність псевдовипадкових бітів до істинної випадковості, а також визначити межі застосування ГПВЧ у криптографічних системах, де непередбачуваність та стійкість є критично важливими. У науковій літературі генератори псевдовипадкових чисел зазвичай класифікують залежно від їхніх характеристик стійкості та призначення. Зокрема, деякі алгоритми підходять лише для статистичних задач і моделювань, тоді як інші відповідають підвищеним вимогам криптографічної безпеки. Умовно виділяють дві групи генераторів:

- Загальні або нестійкі ГПВЧ – алгоритми, які формують послідовності з рівномірним розподілом бітів, проте без гарантій непередбачуваності чи стійкості до криптоаналізу. Вони зазвичай визначаються простотою реалізації та високою

швидкістю генерації, що робить їх придатними для задач моделювання, чисельних обчислень, статистичних симуляцій та тестування алгоритмів.

- Криптографічно стійкі ГПВЧ (КСГПВЧ) – генератори, що забезпечують непередбачуваність вихідної послідовності, стійкість до відновлення внутрішнього стану та захист від прогнозування наступних значень. Вони реалізуються на основі криптографічних примітивів, таких як хеш-функції, блокові шифри або струмові шифри. Використовуються у захищених комунікаційних протоколах, цифрових підписах, шифруванні та інших критичних криптографічних системах. Відмітною рисою КСГПВЧ є те, що навіть при частковому відомому виході генератора, відновлення початкового стану або прогнозування наступних бітів практично неможливе [2].

Серед нестійких алгоритмів у даній роботі розглянуто лінійний конгруентний генератор (LCG) та Mersenne Twister, оскільки вони широко застосовуються у задачах моделювання, чисельних обчислень та статистичних симуляцій завдяки простоті реалізації та високій швидкості генерації. Серед криптографічно стійких генераторів обрано ChaCha20 та Yarrow, які забезпечують непередбачуваність вихідної послідовності та стійкість до відновлення внутрішнього стану, завдяки використанню криптографічних примітивів, таких як блочні шифри, струмові шифри та хеш-функції. Слід зазначити, що наведений набір генераторів не є вичерпним для кожної з груп, проте він дозволяє продемонструвати ключові відмінності між нестійкими та криптографічно стійкими генераторами у контексті цієї статті, а також проаналізувати їхню поведінку та застосовність у практичних сценаріях.

Проаналізуємо теоретичні основи Mersenne Twister. Mersenne Twister - генератор псевдовипадкових чисел із довгим періодом 2^19937-1, який забезпечує рівномірний розподіл та проходить стандартні статистичні тести. Його внутрішній стан складається з 624 32-бітних слів, і вихідна послідовність обчислюється за допомогою лінійної трансформації та бітових операцій (twist і tempering) [3].

$$seed = x_i = \left(f \cdot \left(x_{i-1} \oplus \left(x_{i-1} \gg (w-2)\right)\right) + i\right) \; mod \; 2^w \;, \tag{2}$$

де x_(i-1) – початкове значення (seed), яке визначає весь внутрішній стан генератора, а константа f=0x6c078965та розмір слова w=32 біти визначають властивості генератора та змішування бітів при обчисленні наступних елементів стану. Змінюючи seed, можна отримати різні псевдовипадкові послідовності, водночас використання одного й того ж початкового значення дозволяє відтворювати ту ж послідовність, що підкреслює детермінованість алгоритму [3]. Mersenne Twister добре ілюструє, як невелика початкова послідовність може бути розширена до довгої послідовності з властивостями, близькими до випадкових, проте, як і LCG, не є криптографічно стійким.

ChaCha20 – це криптографічний потоковий шифр, який генерує ключовий потік для шифрування або аутентифікації даних. Він оперує 512-бітним внутрішнім станом, поділеним на шістнадцять 32-бітних слів, і використовує серію чверть-раундів (Quarter Round) з операціями додавання за модулем 2^32, XOR та циклічного зсуву для змішування цього стану [4]. Схема алгоритму, що зображена на Рис.3, ілюструє базову операцію алгоритму ChaCha20 - функцію чверть-раунду, яка описує процес нелінійного змішування чотирьох 32-бітних слів даних a,b,c,d з використанням трьох простих операцій: додавання за модулем 2^32(жовті квадрати), виключного “АБО” ((XOR) сині кола) та циклічних зсувів бітів (зелені прямокутники). Чверть-раунд складається з

чотирьох послідовних кроків, у межах яких слова багаторазово комбінуються між собою шляхом додавання, XOR та циклічних зсувів на фіксовану кількість бітів. У результаті всі чотири слова суттєво змінюються, що забезпечує ефективне перемішування даних у внутрішньому стані алгоритму. Криптографічна стійкість ChaCha20 зумовлена тим, що його внутрішній стан еволюціонує за допомогою нелінійних операцій додавання за модулем 2^32, XOR та циклічних зсувів, які не утворюють лінійних залежностей між вхідними та вихідними даними.

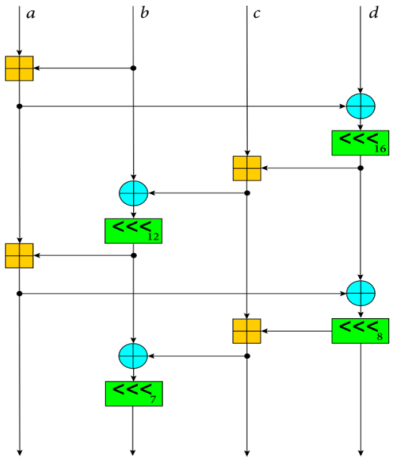

Рис. 1. Схема алгоритму ChaCha20 (Quarter Round)

Багаторазове застосування чверть-раундів призводить до повної дифузії: зміна одного біта ключа, лічильника або nonce впливає на всі слова внутрішнього стану. На відміну від лінійних конгруентних генераторів і Mersenne Twister, де внутрішній стан можна відновити за скінченною кількістю вихідних значень, у ChaCha20 відсутні ефективні методи відновлення стану або прогнозування ключового потоку без знання секретного ключа, що робить алгоритм придатним для криптографічного використання.

Yarrow – це криптографічно стійкий генератор псевдовипадкових чисел, призначений для забезпечення високої непередбачуваності вихідного потоку навіть при потужних атаках. Його робота побудована на зборі ентропії (Entropy Collection) з частково непередбачуваних джерел, яка надходить у два пулу – fast pool для частих оновлень і slow pool для довгострокового накопичення. Після накопичення достатньої кількості ентропії виконується пересів (Reseed), при якому внутрішній стан генератора оновлюється за допомогою хеш-функцій і криптографічних операцій, унеможливлюючи відновлення попереднього стану. Оновлений стан використовується для генерації псевдовипадкових чисел (Output Generation) через криптографічно стійкі функції, а механізм контролю пересіву (Reseed Control) забезпечує часті оновлення fast pool для швидкого реагування на нову ентропію і рідкіші reseed slow pool для довгострокової безпеки. В результаті формується ключовий потік (Output Delivery), криптографічно стійкий та непередбачуваний для атакуючого, придатний для генерації секретних ключів, сольових значень (salt) і ініціалізаційних векторів [5]. Саме через таку складну структуру та багатоетапне оновлення стану Yarrow важко передбачити, що забезпечує його надійність, на відміну від нестійких генератррів, які описувалися вище.

Вимоги до генераторів псевдовипадкових чисел. Вибір генератора псевдовипадкових чисел (ГПВЧ) визначається його здатністю формувати послідовності, які практично неможливо відрізнити від істинно випадкових. Одним із

основних підходів оцінки придатності генератора є вимог, визначених у NIST SP 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Дані формують набір критеріїв, за якими оцінюється якість вихідної послідовності. Перелік основних статистичних вимог до ГПВЧ, що випливають із NIST SP 800 22 наведено наступним списком:

- рівномірний розподіл бітів;
- відсутність закономірностей у послідовності;
- лінійна складність;
- висока ентропія коротких підпослідовностей;
- відсутність періодичних або систематичних відхилень.

Відповідність цим вимогам не гарантує криптографічної стійкості генератора, а надає ймовірнісну оцінку його надійності. Проте, якщо будь-яка з вимог не виконується або порушується кілька вимог одночасно, можна з високою ймовірністю стверджувати, що ГПВЧ не є криптографічно стійким, оскільки ймовірність його стійкості різко знижується [6]. Наведені вимоги слугують необхідним критерієм оцінки генераторів псевдовипадкових чисел для криптографічних застосувань для вибору алгоритмів із високою непередбачуваністю.

Аналіз інцидентів пов’язаних з використанням слабких ГПВЧ. Використання некоректно реалізованих або слабких генераторів псевдовипадкових чисел створює суттєві ризики для стійкості інформаційних систем. Основними наслідками використання вразливих PRNG є передбачуваність криптографічних ключів, компрометація TLS-сесій, можливість відновлення приватних ключів, підробка цифрових підписів і захоплення користувацьких сесій.

Надійність сучасних кіберсистем сильно залежить від якості генерації випадкових чисел, проте захист часто виявляється слабким через використання детермінованих алгоритмів, таких як Mersenne Twister та LCG. Інциденти з ними пов’язані з передбачуваністю токенів та потребують моделі реагування, що забезпечує усунення загроз і відновлення стану системи. Навіть при використанні криптографічно стійких генераторів, як ChaCha20 та Yarrow, ризики не зникають повністю.

З огляду на це доцільним є впровадження формалізованої моделі реагування на інциденти, яка передбачає своєчасне виявлення ознак криптографічної слабкості, локалізацію уражених компонентів, заміну генератора на криптографічно стійкий аналог та обов’язкову заміну ключового матеріалу.

Нижче наведено таблицю зв’язку між типом генератора, його вразливістю та типом інциденту, що може її використати:

Таблиця 1

Зв’язок між типом генератора, його вразливістю та типом інциденту

Тип генератора	Вразливість	Тип інциденту
Yarrow	Дефіцит ентропії (Entropy Starvation)	Недостатній рівень ентропії при ініціалізації
Mersenne Twister	Оборотність стану (State Recoverability)	Відновлення внутрішнього стану
LCG	Математична передбачуваність (Mathematical Predictability)	Передбачуваність сесійних токенів
ChaCha20	Чутливість до імплементації (Nonce Reuse)	Повторне використання nonce

Детальніше розглянемо про вразливості кожного з типів генераторів та змоделюємо сценарій інциденту.

Як зазначає L. Dorrendorf: “Відомим алгебраїчним генератором псевдовипадкових чисел є Mersenne Twister. Він є дуже ефективним і забезпечує високоякісний вихідний потік, придатний для будь-яких цілей, окрім криптографії: зловмисник може обчислити його внутрішній стан після спостереження певної кількості вихідних значень. Наприклад, для обчислення внутрішнього стану MT19937 - стандартної реалізації Mersenne Twister – достатньо 624 вихідних значень”[9].

Проаналізувавши дану наукову роботу можна визначити, що Mersenne Twister має вразливість “Оборотність стану”, адже генератор використовує фіксований внутрішній стан розміром 624 послідовні значення, який повністю визначає всі наступні виходи, тому при наявності достатньої кількості спостережуваних вихідних значень, внутрішній стан можна відновити, а отже й передбачити всі майбутні значення генератора.

А щоб відновити внутрішній стан генератора, необхідно виконати кроки, зворотні до операцій tempering (untempering) для кожного з отриманих вихідних 32-бітових значень, що дає можливість відновити відповідні значення внутрішнього стану [3].

$$\begin{gathered} 1.\, y \coloneqq x \oplus (x \gg u) \\ 2.\, y \coloneqq y \oplus \big((y \ll s)\&b\big) \\ 3.\, y \coloneqq y \oplus \big((y \ll t)\&c\big) \\ 4.\, z \coloneqq y \oplus (y \gg l) \end{gathered} \tag{3}$$

де u, s, t, l – сталі зсувів (для MT19937 відповідно 11, 7, 15 і 18), де b та c - бітові маски tempering, де символ << означає побітовий зсув вліво, а & – операцію побітового логічного “І” (AND).

Визначимо приклад інциденту, що може бути спровокований даною вразливістю.

У веб-сервісі виплати премій організації зловмисник виявляє, що для генерації результатів використовується ГПВЧ Mersenne Twister. Через відсутність обмежень на кількість звернень до API він автоматизовано збирає 624 послідовні 32-бітові вихідні значення, після чого, застосувавши зворотні до tempering перетворення (untempering), відновлює внутрішній стан генератора та починає передбачати всі наступні “випадкові” результати, маніпулюючи отриманням премій.

Генератор псевдовипадкових чисел LCG вважається поліноміально передбачуваним, якщо існує алгоритм поліноміального часу, який, отримавши достатньо довгу послідовність бітів, згенерованих цим генератором, здатний передбачити його наступні вихідні значення. Наприклад, лінійні конгруентні генератори та генератори типу $1/P$ є поліноміально передбачуваними [10].

Даний генератор є поліноміально передбачуваним, оскільки його робота визначається чіткою математичною формулою і знаючи частину параметрів, можна обчислити решту, що фактично дає можливість скомпрометувати генератора. Тому очевидно, що існує алгоритм, який за поліноміальний час, маючи достатньо довгу послідовність вихідних значень, може відновити внутрішній стан або параметри генератора, використовуючи систему конгруентних рівнянь. Після цього стає можливим точно передбачати всі наступні вихідні числа. Відновлення внутрішнього стану LCG за допомогою системи конгруентних рівнянь [1].

$$\{X_1 = (aX_0 + c)\ mod\ m\ X_2 = (aX_1 + c)\ mod\ m\ \vdots, \tag{4}$$

У системі онлайн-лотереї зловмисник виявляє, що для генерації “випадкових” номерів білетів у системі онлайн-лотереї використовується лінійний конгруентний генератор, тому спостерігаючи кілька послідовних номерів, він застосовує алгоритм LCG-break, вирішуючи систему конгруентних рівнянь для відновлення внутрішнього стану та параметрів генератора. В результаті зловмисник може передбачати всі наступні номери, що дозволяє йому купувати білети з гарантовано виграшними комбінаціями.

ChaCha20-Poly1305 є шифром типу AEAD (Authenticated Encryption with Associated Data) і вимагає унікального значення nonce для кожної операції шифрування. Відповідно до RFC 7539, значення nonce (IV) має становити 96 бітів (12 байтів). Бібліотека OpenSSL дозволяє використання nonce змінної довжини та доповнює його спереду нульовими байтами, якщо його довжина менша за 12 байтів. Однак вона також некоректно дозволяє встановлювати nonce довжиною до 16 байтів. У такому випадку враховуються лише останні 12 байтів, а додаткові початкові байти ігноруються. Використання цього шифру передбачає обов’язкову унікальність значень nonce. Повторне використання одного й того самого nonce призводить до серйозних атак на конфіденційність і цілісність зашифрованих повідомлень. [11].

Основна вразливість ChaCha20-Poly1305 полягає у Nonce Reuse, а саме у повторному використанні одноразового вектора ініціалізації для шифрування кількох повідомлень. Якщо один і той самий nonce застосувати більше ніж один раз, це вже серйозно скомпрометує конфіденційність та цілісність зашифрованих даних, незважаючи навіть не те, що алгоритм ChaCha20-Poly1305 сам по собі криптографічно стійкий. Як приклад, ця вразливість була виявлена у реалізації OpenSSL, де неправильне управління довжиною nonce дозволяло повторно використовувати однаковий nonce.

Прикладом інциденту може бути помилкова реалізація OpenSS, коли один і той самий nonce використали для шифрування кількох файлів за допомогою ChaCha20-Poly1305. Зловмисник скористався цією нагодою, бо він помітив, що деякі повідомлення були зашифровані з однаковим nonce, і порівняв відповідні зашифровані тексти, обчисливши між ними XOR, що дозволило йому виділити частину відкритого тексту та дізнатися, як виглядають інші повідомлення. Далі ж зловмисник підставив власні дані, сформувавши нові зашифровані повідомлення з тим самим nonce, щоб контролювати вміст переданих файлів і саме внаслідок даного інциденту були порушені конфіденційність і цілісність переданих даних. Наслідком став витік інформації, що завдав шкоди як фінансовому стану компанії, так і її безпековій репутації.

Як зазначає J. Viega у своїй роботі Practical Random Number Generation in Software: “Неможливо легко оцінити, чи є генератор псевдовипадкових чисел достатньо безпечним, лише на підставі кількості повторних ініціалізацій (reseed). Якщо скомпрометоване або “забруднене” джерело ентропії спричиняє часті повторні ініціалізації, це може призвести до того, що перші кілька пулів ентропії щоразу міститимуть незначну кількість фактичної ентропії під час reseed. У такому випадку зловмисник здатний тривалий час перешкоджати досягненню генератором безпечного стану” [12]. Автор зазначає, що “неможливо легко оцінити, чи PRNG буде безпечним на основі кількості пересіянь (reseeds)” і що “забруднене джерело може викликати багато пересіянь, гарантуючи, що перші кілька пулів завжди містять невелику кількість фактичної ентропії”, тому виходить, що Yarrow опирається на механізм накопичення ентропії у кількох “пулах”, які періодично пересіюють внутрішній стан генератора, і

проблема полягає в тому, що оцінити, скільки справжньої ентропії накопичено у кожному пулі, практично неможливо. Якщо ж джерела ентропії містять недостатньо випадкових подій), то пересіяння відбувається, але фактична кількість випадковості, яка впливає на стан генератора, дуже мала. Внаслідок даної вразливості внутрішній стан PRNG може залишатися передбачуваним протягом тривалого часу, навіть якщо формально відбуваються пересіяння і зловмисник, який спостерігає вихідні значення генератора, може скористатися цим дефіцитом ентропії для зворотного відновлення стану або прогнозування майбутніх значень. Прикладом інциденту може бути генератор Yarrow , що працює з обмеженим джерелом ентропії, і кілька пулів часто містять недостатньо справжньої випадковості через передбачувані джерела подій. Під час аудиту аналіз журналів подій показав, що перші пересіяння завжди відбувалися з низьким рівнем ентропії, а вихідні значення демонстрували повторювані шаблони. Зловмисник, отримавши доступ до вихідних значень через мережеві журнали, скористався дефіцитом ентропії, частково відновив внутрішній стан Yarrow і зміг прогнозувати наступні значення, отримавши можливість передбачати майбутні ключі сеансів користувачів, що може призвести до компрометації даних і несанкціонованого доступу до облікових записів. Після аналізу вразливостей та інцидентів було сформовано план реагування на дані інциденти.

Модель плану реагування на інциденти пов’язані з використанням слабких ГПВЧ. Запропонована модель реагування сформована на зв’язку між типом генератора та характером його вразливості та реалізована на загальному життєвому циклі управління інцидентами інформаційної безпеки, що відповідає підходу, визначеному у NIST SP 800-61 Revision 2: Computer Security Incident Handling Guide, де зазначено, що процес реагування включає фази підготовки, виявлення та аналізу, локалізації, усунення і відновлення, а також діяльності після інциденту [7]. Також нижче наведено візуальне відображення процесу плану реагування на інциденти (блок-схема):

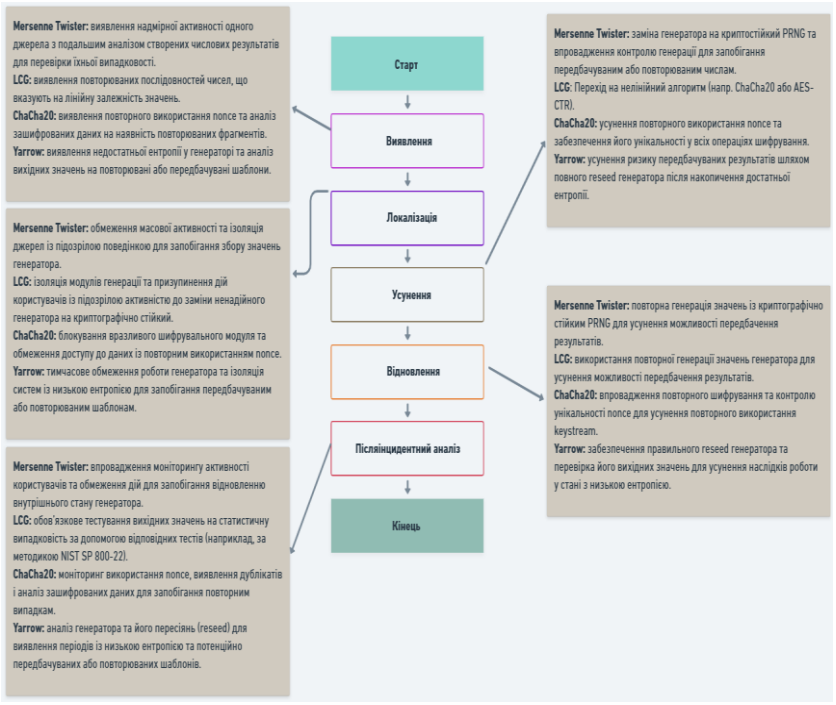

Рис. 2 Модель плану реагування на інциденти пов’язані з використанням слабких ГПВЧ

Узагальнено етапи плану реагування на інциденти подано наступним списком:

1. На етапі виявлення потрібно виявити аномалії та ознаки компрометації генераторів випадкових чисел.
2. Етап локалізації спрямований на мінімізацію масштабу потенційної компрометації та ізоляцію уражених систем.
3. Етап усунення спрямований на ліквідацію першопричини інциденту.
4. Етап відновлення повертає систему до безпечного, працездатного стану після інциденту.
5. Етап післяінцидентного аналізу детально оцінити наслідки інциденту, визначити масштаби компрометації, зрозуміти першопричини події та внести зміни у процеси і політики безпеки, щоб знизити ризик повторення подібних випадків у майбутньому.

Для побудови активної протидії даним загрозам та зменшення ймовірності їх виникнення заходи безпеки можуть бути додані до постійної інтеграції та постійного впровадження (CI/CD). Кожного разу, коли розробник створює код, він запускає інструмент CI/CD, який виконує весь необхідний процес, тобто передає код у спільне сховище та надсилає сповіщення іншим членам команди [8]. У такому випадку виявлення некоректно реалізованих або слабких генераторів псевдовипадкових чисел може бути автоматизованим.

Автоматизація реагування на криптографічні ризики ГПВЧ у CI/CD середовищі. З метою підвищення рівня контролю криптографічної безпеки запропонована модель реагування на інциденти, пов’язані з використанням слабких генераторів псевдовипадкових чисел, інтегрується до процесу безперервної інтеграції та доставки (CI/CD).

Запропонована реалізація механізму в середовищі Jenkins дозволяє виявляти ризики ще на етапі розробки та збірки програмного забезпечення, до його розгортання у продукційному середовищі. На відміну від традиційного реагування після виникнення інциденту, інтеграція до CI/CD систем модель управління криптографічними ризиками у превентивну систему контролю. У межах pipeline реалізуються автоматизовані механізми виявлення, класифікації та блокування потенційно небезпечних змін. Механізм активується під час виконання збірки та включає:

- статичний аналіз коду (SAST) на наявність некриптографічних PRNG у безпековому контексті;
- перевірку коректності використання AEAD-алгоритмів (зокрема унікальності nonce для ChaCha20-Poly1305);
- аналіз залежностей (SCA) для виявлення вразливих криптографічних бібліотек.

Результати сканування передаються до алгоритму класифікації інцидентів, який визначає тип потенційного інциденту та рівень його критичності. Залежно від результату Jenkins приймає одне з трьох рішень: блокування збірки (FAIL) у разі критичного ризику, позначення збірки як нестабільної (UNSTABLE) з необхідністю ручного погодження, або продовження процесу (PASS) у разі відсутності загроз. Таким чином, CI/CD середовище виконує функцію автоматизованого бар’єру, що запобігає потраплянню в продуктивне середовище коду з криптографічними дефектами, такими як використання LCG або Mersenne Twister для генерації токенів, повторне використання nonce, або застосування бібліотек із відомими вразливостями.

Нижче наведено приклад реалізації Jenkins-автоматизації, який демонструє логіку виявлення та блокування ризиків виникнення інциденту:

```
pipeline {
  agent any
  options { timestamps() }
  environment {
       REPORT_DIR = "reports"
  }
  stages {
       stage("Checkout") {
       steps { checkout scm }
       }
       stage("Prepare") {
       steps { sh "mkdir -p ${REPORT_DIR}" }
       }
       stage("Static Analysis (PRNG & Crypto)") {
       steps {
       sh """
       grep -R "Math.random()". > ${REPORT_DIR}/prng_findings.txt || true
       grep -R "java.util.Random". >> ${REPORT_DIR}/prng_findings.txt || true
       grep -R "random.random(". >> ${REPORT_DIR}/prng_findings.txt || true
       """
       }
       }
       stage("Dependency Scan") {
       steps {
       sh """
       # Приклад інтеграції зі сканером залежностей
       echo '{}' > ${REPORT_DIR}/sca_report.json
       """
       }
       }
       stage("Crypto Misuse Check") {
       steps {
       sh """
       grep -R "ChaCha20". > ${REPORT_DIR}/crypto_misuse.txt || true
       """
       }
       }
       stage("Decision") {
       steps {
       script {
       def findings = readFile("${REPORT_DIR}/prng_findings.txt")
       if (findings.trim()) {
              error("Build blocked: insecure PRNG usage detected.")
       } else {
              echo "No critical crypto risks detected."
       }
       }
       }
       }

       stage("Deploy") {
       when { expression { currentBuild.currentResult == "SUCCESS" } }
       steps { echo "Deploying..." }
       }
  }
  post {
       always {
       archiveArtifacts artifacts: 'reports/**', fingerprint: true
       }
  }
}
```

Запропонований підхід забезпечує відповідність етапам класичного циклу реагування на інциденти: виявлення здійснюється через SAST/SCA-аналіз, локалізація через блокування збірки, усунення через виправлення коду або оновлення

залежностей, відновлення через повторну перевірку після внесення змін, а профілактика реалізується шляхом обов'язкового контролю в кожному релізному циклі.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Генератори псевдовипадкових чисел є критично важливими компонентами криптографічних систем, оскільки дозволяють детерміновано формувати довгі послідовності бітів, що практично не відрізняються від істинно випадкових. Аналіз теоретичних основ, класифікації генераторів та конкретних прикладів, таких як LCG, Mersenne Twister, ChaCha20 і Yarrow, дозволяє виділити ключові відмінності між нестійкими та криптографічно стійкими алгоритмами, а також зрозуміти роль початкового стану (seed) у забезпеченні відтворюваності послідовностей.

У статті розроблено узагальнену модель плану реагування на інциденти, пов'язану з використанням слабких або некоректно реалізованих генераторів псевдовипадкових чисел, яка інтегрує типологію генераторів, характер криптографічної вразливості та етапи життєвого циклу реагування. Запропонований підхід дозволяє системно поєднати криптографічний аналіз, організаційні заходи та технічні механізми відновлення, що забезпечує мінімізацію наслідків інцидентів. Практична цінність моделі полягає в можливості її адаптації до різних класів інформаційних систем і використання як методичної основи для вдосконалення процедур безпеки.

Подальші наукові дослідження доцільно спрямувати на тестування автоматизованого підходу до реагування критеріїв оцінювання рівня критичності інцидентів залежно від типу генератора та сфери його застосування, а також на розроблення автоматизованих механізмів виявлення аномалій у роботі ГПВЧ у реальному часі. Перспективним є також створення математичної моделі кількісної оцінки ризику компрометації та інтеграція запропонованого підходу з системами безперервного моніторингу для підвищення адаптивності реагування на криптографічні інциденти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography.* https://theswissbay.ch/pdf/Gentoomen%20Library/Cryptography/Handbook%20of%20Applied%20Cryptography%20-%20Alfred%20J.%20Menezes.pdf
2. Almaraz Luengo, E., & Román Villaizán, J. (2023). Cryptographically secured pseudo-random number generators: Analysis and testing with NIST statistical test suite. *Mathematics, 11*(23), 4812. https://www.mdpi.com/2227-7390/11/23/4812
3. Stoen, H. (2024). *The Mersenne Twister and cryptographically secure PRNGs.* https://simonrs.com/eulercircle/crypto2024/henry-mersenne.pdf
4. Internet Engineering Task Force. (2018). *RFC 8439: ChaCha20 and Poly1305 for IETF protocols.* https://datatracker.ietf.org/doc/rfc8439/
5. Ferguson, N., & Schneier, B. (2000). *Notes on the design and analysis of the Yarrow cryptographic PRNG.* https://scispace.com/pdf/yarrow-160-notes-on-the-design-and-analysis-of-the-yarrow-47wqbflx00.pdf
6. National Institute of Standards and Technology. (2010). *A statistical test suite for random and pseudorandom number generators for cryptographic applications* (NIST Special Publication 800-22 Rev. 1a). https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf

7. National Institute of Standards and Technology. (2012). *Computer security incident handling guide* (NIST Special Publication 800-61 Rev. 2). https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf
8. Susukailo, V. (2021). Use of DevSecOps approach for analysis of modern information security threats. *Cybersecurity: Education, Science, Technique*, *2*(14), 26–35.
9. Dorrendorf, L., Gutterman, Z., & Pinkas, B. (2007). Cryptanalysis of the random number generator of the Windows operating system. https://www.cs.huji.ac.il/~dolev/pubs/thesis/msc-thesis-leo.pdf
10. Bikos, A., Nastou, P. E., Petroudis, G., & Stamatiou, Y. (2023). Random number generators: Principles and applications. *Cryptography*, *7*(4), 54. https://www.mdpi.com/2410-387X/7/4/54
11. National Vulnerability Database. (2019). *CVE-2019-1543 detail.* https://nvd.nist.gov/vuln/detail/cve-2019-1543
12. Viega, J. (2003). *Practical random number generation in software.* https://www.acsac.org/2003/papers/79.pdf

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (32), 2026
ISSN 2663 – 4023

Danylo Buhai
Student, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0009-0001-9858-0482
danylo.buhai.kb.2022@lpnu.ua

Ihor Vlasiuk
Postgraduate student, Assistant Professor, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0009-0008-3600-750X
ihor.d.vlasiuk@lpnu.ua

Vitalii Susukailo
PhD, Senior Lecturer, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0000-0003-4431-9964
vitalii.a.susukailo@lpnu.ua

Yevhenii Kurii
PhD, Senior Lecturer, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0000-0002-3423-5655
Yevhenii.O.Kurii@lpnu.ua

MODEL OF AN INCIDENT RESPONSE PLAN FOR INCIDENTS RELATED TO THE USE OF WEAK PSEUDORANDOM NUMBER GENERATORS

Abstract. The article analyzes pseudorandom number generators (PRNGs) in cryptography, considers their theoretical foundations, classification, and basic security requirements. In particular, it considers how deterministic algorithms based on the initial seed generate long sequences of bits that are practically indistinguishable from truly random ones, therefore, different types of generators were analyzed: from simple linear congruent generators (LCGs) and Mersenne Twister, which are well suited for modeling and numerical calculations, to cryptographically stable algorithms ChaCha20 and Yarrow, which provide unpredictability and resistance to predicting the initial sequence. A universal response plan for incidents involving vulnerabilities in pseudorandom number generators and an algorithm for automating responses to such incidents are proposed. The results of the study can be used by developers of cryptographic algorithms and information security specialists to improve the assessment of pseudorandom number generator choices for practical and educational purposes.

Keywords: pseudorandom number generator, Mersenne Twister, ChaCha20, Yarrow, pseudorandomness, seed, statistical requirements, data security, CI/CD, information security incidents.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. https://theswissbay.ch/pdf/Gentoomen%20Library/Cryptography/Handbook%20of%20Applied%20Cryptography%20-%20Alfred%20J.%20Menezes.pdf
2. Almaraz Luengo, E., & Román Villaizán, J. (2023). Cryptographically secured pseudo-random number generators: Analysis and testing with NIST statistical test suite. *Mathematics, 11*(23), 4812. https://www.mdpi.com/2227-7390/11/23/4812
3. Stoen, H. (2024). *The Mersenne Twister and cryptographically secure PRNGs.* https://simonrs.com/eulercircle/crypto2024/henry-mersenne.pdf
4. Internet Engineering Task Force. (2018). *RFC 8439: ChaCha20 and Poly1305 for IETF protocols.* https://datatracker.ietf.org/doc/rfc8439/

5. Ferguson, N., & Schneier, B. (2000). *Notes on the design and analysis of the Yarrow cryptographic PRNG.* https://scispace.com/pdf/yarrow-160-notes-on-the-design-and-analysis-of-the-yarrow-47wqbflx00.pdf
6. National Institute of Standards and Technology. (2010). *A statistical test suite for random and pseudorandom number generators for cryptographic applications* (NIST Special Publication 800-22 Rev. 1a). https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf
7. National Institute of Standards and Technology. (2012). *Computer security incident handling guide* (NIST Special Publication 800-61 Rev. 2). https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf
8. Susukailo, V. (2021). Use of DevSecOps approach for analysis of modern information security threats. *Cybersecurity: Education, Science, Technique*, *2*(14), 26–35.
9. Dorrendorf, L., Gutterman, Z., & Pinkas, B. (2007). Cryptanalysis of the random number generator of the Windows operating system. https://www.cs.huji.ac.il/~dolev/pubs/thesis/msc-thesis-leo.pdf
10. Bikos, A., Nastou, P. E., Petroudis, G., & Stamatiou, Y. (2023). Random number generators: Principles and applications. *Cryptography*, *7*(4), 54. https://www.mdpi.com/2410-387X/7/4/54
11. National Vulnerability Database. (2019). *CVE-2019-1543 detail.* https://nvd.nist.gov/vuln/detail/cve-2019-1543
12. Viega, J. (2003). *Practical random number generation in software.* https://www.acsac.org/2003/papers/79.pdf

Отримано редакцією журналу / Received: 16.01.26
Прорецензовано / Revised: 02.02.26
Схвалено до друку / Accepted: 26.03.26