рівноцінним кодом, написаним на C++. Різниця може сягати десятків та сотень разів, хоча розробники Blueprint постійно працюють над зменшенням цього розриву.

До того ж функціонал Blueprint є обмеженим, порівняно з можливостями мов програмування, як-от C++. Тому розробники мають можливість використовувати додаткові плагіни, розроблені спільнотою, або писати певний функціонал мовою C++, оскільки Unreal Engine дає змогу використовувати гібридну розробку із застосуванням обох технологій.

Враховуючи розглянуті переваги та недоліки, можна зробити висновок, що Blueprint, хоч і менш функціональний та менш швидкий, порівняно з C++, є відмінним інструментом для початківців та дизайнерів без глибоких технічних знань у програмуванні чи технології Unreal Engine. Досвідчені розробники використовують обидві технології: Blueprint – для швидкого написання та тестування ігрових механік, C++ – для написання ресурсномістких систем та забезпечення оптимальної продуктивності кінцевого продукту.

Список використаних джерел

1. Share of video gamers worldwide in 2022: вебсайт. URL: https://www.statista.com/statistics/297874/number-mobile-gamers-region/ (дата звернення: 09.05.2024)

2. Blueprint documentation: вебсайт. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/introduction-to-blueprints-visual-scripting-in-unreal-engine?application_version=5.3 (дата звернення: 09.05.2024)

УДК 004.056.5

Поліщук В. С., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Фриз І. В., *канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій*

ВИПАДКОВІ ВЕЛИЧИНИ ТА РОЗПОДІЛИ ЙМОВІРНОСТЕЙ У КІБЕРБЕЗПЕЦІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Проблеми захисту інформації стають все більш актуальними на сучасному етапі розвитку суспільства, коли загрози для безпеки інформації та приватності постійно зростають. Застосування математичного апарату в інформаційній та кібернетичній безпеці є ключовими аспектами для створення надійних систем захисту даних та мереж. Розглянемо це на прикладі випадкових величин та розподілів ймовірностей.

Одним з основних застосувань випадкових величин у кібербезпеці є генерація криптографічних ключів. Випадковість ключів є важливою для забезпечення стійкості криптографічних алгоритмів шифрування та автентифікації. Застосування різних розподілів ймовірностей, як-от рівномірний, нормальний або експо-

ненціальний, може забезпечити відповідний рівень випадковості генерованих ключів [1].

Випадкові величини та розподіли ймовірностей використовуються для аналізу вразливостей та прогнозування ризиків у кібербезпеці [2]. Моделювання випадкових процесів може допомогти ідентифікувати потенційні загрози та виявляти слабкі місця в інформаційних системах і мережах. Використання випадкових величин дає змогу проводити статистичний аналіз даних щодо вразливості програмного забезпечення та мереж [3]. Це може допомогти в розумінні розподілу часу між виявленням вразливості та випуском виправлення, що в свою чергу допомагає в управлінні ризиками та розробці стратегій захисту.

Використання випадкових величин та розподілів ймовірностей у кібербезпеці є важливим для створення надійних систем захисту даних та мереж. Розглянемо код на C# (рис. 1), який використовує випадкові величини для створення мережі з 50 вузлів, де кожен вузол має певну ймовірність атаки. Симуляція атаки на мережу дає змогу обчислити кількість вразливих вузлів та відсоток вразливості, що допомагає аналізувати стійкість мережі до потенційних загроз.

```
using System;
using System.IO;
using System.Text;

class Program
{
   static void Main()
   {
      // Параметри симуляції
      int networkSize = 50; // Розмір мережі (кількість вузлів)
      double attackProbability = 0.1; // Ймовірність атаки на кожен вузол

      // Створення мережі
      bool[] network = new bool[networkSize]; // Масив для представлення стану кожного вузла (true - вразливий, false - невразливий)

      // Ініціалізація мережі: встановлення вразливості для певної частини вузлів
      Random rand = new Random();
      for (int i = 0; i < networkSize; i++)
      {
         network[i] = rand.NextDouble() < 0.5; // 50% вузлів вважається вразливими
      }

      // Симуляція атаки на мережу
      int vulnerableNodes = 0; // Лічильник вразливих вузлів
      for (int i = 0; i < networkSize; i++)
      {
         if (network[i] && rand.NextDouble() < attackProbability)
         {
            // Якщо вузол вразливий і атака відбувається
            Console.WriteLine($"Вузол {i} був атакований!");
            vulnerableNodes++;
         }
      }
```

```
        // Виведення результатів симуляції
        Console.WriteLine($"У мережі з {networkSize} вузлів {vulnerableNodes} вразливих.");
        double vulnerabilityRatio = (double)vulnerableNodes / networkSize;
        Console.WriteLine($"Відсоток вразливих вузлів: {vulnerabilityRatio:P}");

        // Збереження у файл з кодуванням UTF-8
        string filePath = "output.txt";
        using (StreamWriter writer = new StreamWriter(filePath, false, Encoding.UTF8))
        {
            writer.WriteLine($"У мережі з {networkSize} вузлів {vulnerableNodes} вразливих.");
            writer.WriteLine($"Відсоток вразливих вузлів: {vulnerabilityRatio:P}");
        }
    }
}
```

Рис. 1. Код дослідження вразливостей на C#

Після проведення симуляції програма обчислює кількість вразливих вузлів та відсоток вразливості мережі (рис. 2).

```
Вузол 32 був атакований!
Вузол 35 був атакований!
Вузол 40 був атакований!
У мережі з 50 вузлів 3 вразливих.
Відсоток вразливих вузлів: 6,00%
Press any key to continue . . .
```

Рис. 2. Результати роботи коду

Ці результати важливі для аналізу та управління ризиками, а також для вдосконалення системи захисту. Збереження результатів у файл дає змогу вести запис та дослідження вразливостей, що сприяє подальшому аналізу та поліпшенню системи.

Отже, використання випадкових величин та розподілів ймовірностей у кібербезпеці допомагає не лише оцінювати стійкість системи, але й ідентифікувати потенційні загрози та виявляти слабкі місця у мережі, що робить її більш надійною та захищеною.

Список використаних джерел

1. Гулак Г. М. Методологія захисту інформації. Аспекти кібербезпеки: навчальний посібник. Київ: Видавництво НА СБ України. 2020. 256 с.
2. Sánchez-García I. D., Mejía J., San Feliu T. Cybersecurity Risk Assessment: A Systematic Mapping Review, Proposal, and Validation. *Applied Sciences.* 2023. № 13(1). 29 p. DOI: 10.3390/app13010395 (дата звернення 18.05.2024).
3. Дудикевич В. Б. Основи інформаційної безпеки: навч. посібник. Вінниця: ВНТУ. 2018. 316 с.