УДК 004.056.55

Крупін Є.Г., магістрант, Скрипник І. А., доцент, науковий керівник

КРИПТОГРАФІЧНИЙ АНАЛІЗ ВРАЗЛИВОСТЕЙ ХЕШОВАНОЇ СИСТЕМИ

Запорізька державна інженерна академія, кафедра ПЗАС

Сьогодні наш світ переповнений інформацією. Технології проникли на кожен рівень нашого життя. Ми прокидаємося поруч зі смартфоном, що синхронізує час з сервером, пошту з вашою поштовою скринькою, версії встановленого ПЗ і додатків з найбільш актуальною. Приходимо на роботу і входимо в Інтернет, що надає нам необмежені можливості в плані комунікації, поглиблення знань, торгівлі, просування наших товарів і послуг на ринку. Ми можемо знайти потрібну нам інформацію в мережі принаймні в два - три кліки.

Доступність інформації і технологій зробила нас сильнішими, але вона ж робить нас вразливими . Нам потрібна можливість зберігати в незайманому вигляді нашу автентичність, конфіденційність, а також наші дані, що не підлягають оприлюдненню. Тому на допомогу нам приходить **криптографія**[1].

Існує сила-силенна досить просунутих алгоритмів, здатних прийти на виручку нам, простим користувачам, які не бажають піддавати свою інформацію ризику. Що ми поставимо між їй і передбачуваним зловмисником? Звичайно, славнозвісний дует з **логіна і пароля**. А в якому вигляді ми будемо зберігати пароль на сервері? Ну природно, в **хешованому**.

І, звичайно, хочеться бути впевненими в ступені захищеності нашого пароля. І тут на сцену виходимо ми.

Мета даної роботи - ознайомитися зі ступенем захищеності паролів на прикладі алгоритму хешування **MD5** - не найбільшого свіжого і надійного, але вельми доступного і поширеного. Ми досліджуємо паролі різної довжини використовуючи різні методи злому, починаючи з простого брутфорса, потім брутфорса із заданою наперед базою пар хеш - пароль, а потім застосуємо трохи магії у вигляді так званих "Райдужних таблиць". "Райдужні таблиці" – більш просунутий метод підбору паролів, що використовує функцію редукції та дозволяє значно прискорити процес пошуку необхідної строки за її хешем. Ми зберігаємо проміжні кроки в нашій таблиці та постійно порівнюємо їх з необхідним результатом, а якщо вони співпадають ми просто повертаємося на крок назад та отримуємо необхідну строку.

Кожен, хто хоч трохи знайомий з криптографією знає, що всі сучасні алгоритми спираються в першу чергу на те, що тривалість злому нівелює користь від отриманої інформації.

Природно, що сучасні настільні комп'ютери і ноутбуки не так швидкі, як нам хотілося б. Але знову ж таки завжди можна знайти вихід і ми його знайшли - ми будемо проводити необхідні обчислення **паралельно**. На жаль, на CPU ми маємо не дуже вражаючу кількість ядер, що в свою чергу впливає на кількість одночасно працюючих потоків, адже якщо число паралельних потоків більше числа ядер, то настає уявна паралельність, яка часто зводить нанівець, якщо не заганяє в мінус всю користь від паралельності, адже велика частина процесорного часу витрачається на перемикання між потоками, а не на корисну роботу. Тому ми підемо іншим шляхом і скористаємося графічним процесором (**GPU)**, що щедро надає нам велику кількість ядер для справжньої паралельності .

Існує дійсно корисний інструментарій, який ми можемо використовувати в своїх цілях: OpenCL , Cuda[2], що дозволяють писати не графічні програми для графічних процесорів і використовують в якості мови програмування всіма улюблену мову **C**[3]**.** У російськомовному сегменті тема поки залишається не дуже освітленою, я б навіть сказав, слабо порушеною, хоча і цікавою.

Таким чином ми розробляємо систему, яка допоможе випробувати на міцність паролі різної довжини і різної природи на предмет уразливості.

Архітектура нашої системи буде складатися з кількох найважливіших елементів:

По-перше, ми пишемо ядро , що дозволяє нам проводити паралельні обчислення на графічному процесорі. Воно представляє собою пакет функцій мовою С, що використовують велику кількість ядер графічного процесора одночасно, враховуючи номер-ідентифікатор процесу як індекс елементу масиву. Ми загружаємо в функцію з директивою **_kernel** вказівник на масив байтів, що включає в себе усі паролі, що ми одночасно обраховуємо, а також масив з інформацію про довжину кожного з цих паролів. Далі наша функція викликає усі необхідні локальні функції для кожного процесу паралельно, що дозволяє нам обчислювати велику кількість хеш-функцій одночасно. Ми реалізували MD5 мовою С, щоб навести приклад роботи.

По-друге, створюємо необхідні утиліти, які допоможуть нам генерувати словник, підбирати паролі, а так само складати графіки і т.п. За їх допомогою ми складемо величезні файли з серіалізованою інформацією, де будуть зберігатися пари пароль - хеш, що ми будемо використовувати підчас брутфорсу, а також візуалізувати різницю в ступені захищеності паролів різної довжини. Це потрібно нам щоб робити висновки про роботу нашої системи та залежність захищеності від природи паролю, наприклад, алфавіту, з якого він складається та його довжини.

По-третє, нам знадобиться обгортка - клієнт , за допомогою якої ми будемо працювати з нашим ядром. В неї загружається наша програма-ядро мовою С, використовуючи бібліотеку “Cloo”, що дозволяє нам ідентифікувати обладнання на нашій робочій станції, а також підготувати контекст, загрузити в нього нашу програму та необхідні вхідні дані, а потім вивантажити з неї вихідні. Ми користуємося цією обгорткою для демонстрації роботи продукту в реальних умовах.

Перший пункт виконаний на мові С, два останніх – об’єктно -орієнтованих [4] - на **C#**, нині вельми популярному і доброзичливому до розробника[5], [6].

Система може бути використана для реального тестування рівню захищеності паролів, або для наочної демонстрації їх уразливостей.

Висновки:

- вивчена ступінь захищеності паролів;
- виконано порівняння паролів різної довжини і природи , а так само методів злому;
- розглянуті технології для написання математичних програм для GPU;
- розглянуті питання і проблеми паралельних обчислень;
- розглянуті і використані сучасні бібліотеки для паралельних обчислень.

Література

1. Уельшенбах М. – Криптография и С. / Триумф 2004.
2. Боресков А. та Харламов А. – Основы работы с технологией CUDA. / ДМК Пресс 2010.
3. Керниган Б. та Ритчи Д. – Язык С./ Вильямс 2003.
4. Буч Г. – Объектно-орієнтованый анализ и проектирование / Вильямс 2008.
5. Рихтер Дж. – CLR via C#. Программирование на платформе .NET. / Питер, 3-е издание 2012.
6. Мартин Р. – Чистый код. Создание, анализ, рефакторинг. / ЗАО "Издательский дом "Питер" 2012.