REPORT

무작위 공격 테스트

과목명 :	컴퓨터보안
담당교수 :	정준호 교수님
제출일 :	2021 년 03 월 22 일
공과 대학	컴퓨터공학 과
학번 : 2016112154	이름: 정동구

1. 소스코드

```
std::string CreatePasswd(int key_length, int passwd_type)
{
    //char* passwd = new char[key_length];
    std::string passwd;
    char alphabet[] = "abcdefghijklmnopqrstuvwxyz";
    char number[] = "0123456789";
    char alpha_num[] = "abcdefghijklmnopqrstuvwxyz0123456789";
    char whole[] = "abcdefghijklmnopqrstuvwxyz0123456789~!@#$%^&*()_+`,./<>?;':[]{}₩"|";


    if (passwd_type == 1)
    {
        for (int i = 0; i < key_length; i++)
        {
            passwd += number[rand() % 10];
        }
    }
    else if (passwd_type == 2)
    {   for (int i = 0; i < key_length; i++)
        {
            passwd += alphabet[rand() % 26];
        }
    }
    else if (passwd_type == 3)
    {
        for (int i = 0; i < key_length; i++)
            passwd += alpha_num[rand() % strlen(alpha_num)];
    }
    else if (passwd_type == 4)
    {
        for (int i = 0; i < key_length; i++)
            passwd += whole[rand() % strlen(whole)];
    }
    return passwd;
}
```

랜덤한 비밀번호를 생성하는 함수.

```
void ComparePasswd(int key_length,int case_num)
{

    srand(GetTickCount64());
    std::string case_passwd;
    std::string compare_passwd;
    clock_t start, end;

    for (int i = 0; i < 25; i++)
    {
        case_passwd = CreatePasswd(key_length, case_num);
        for (int i = 0; i < key_length; i++)
        {
            std::cout << case_passwd[i];
        }std::cout << " : ";
        start = clock();
        while (case_passwd.compare(compare_passwd) != 0)
        {
            compare_passwd = CreatePasswd(key_length, 4);

        }
        end = clock();
        double elapse_time = (end - start) / (double)1000;
        vec.push_back(std::make_pair(case_passwd, elapse_time));
        std::cout.precision(8);
        std::cout << (end - start) / (double)1000 << "₩n";
    }
}
```

무작위로 문자열을 생성하여 비밀번호와 일치 할 때 까지 생성, 비교를 하는 함수.

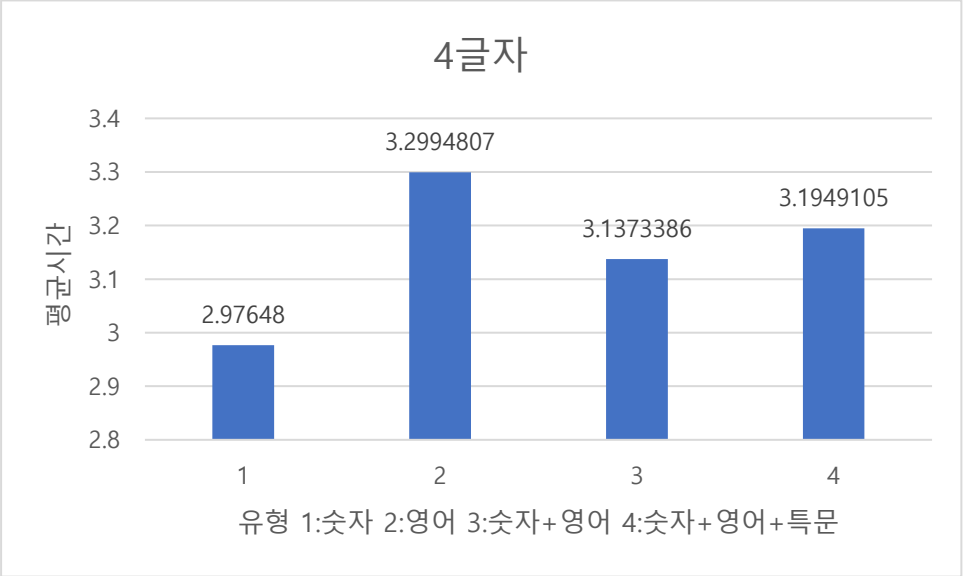
```
void run()
{

    int case_num;
    int key_length;
    std::cout << "input key length : ";
    std::cin >> key_length;
    /*char* case_passwd = new char[key_length];
    char* compare_passwd = new char[key_length];*/
    std::string case_passwd;
    std::string compare_passwd;
    std::cout << "choose case number(case 1 : number , case 2 : alphabet , case 3 : alphabet+number)₩n  :";
    std::cin >> case_num;
    clock_t start, end;

    std::thread t1(ComparePasswd,key_length,case_num);
    Sleep(1000);
    std::thread t2(ComparePasswd, key_length, case_num);
    Sleep(1000);
    std::thread t3(ComparePasswd, key_length, case_num);
    Sleep(1000);
    std::thread t4(ComparePasswd, key_length, case_num);

    t1.join();
    t2.join();
    t3.join();
    t4.join();


    float sum = 0;
    for (int i = 0; i < 100; i++)
    {
        sum += vec[i].second;
    }
    sum = sum / (float)100;
    std::string out_filename = "result_";
    out_filename += std::to_string(key_length);
    out_filename += "_";
    out_filename += std::to_string(case_num) +".txt";
    std::ofstream fout(out_filename);
```

CreatePasswd 와 ComparePasswd 를 이용하여 전체 과정을 수행하는 함수.
하단에 잘린부분은 텍스트 파일로 출력하는 부분입니다.

소스코드는 별도로 첨부하겠습니다.

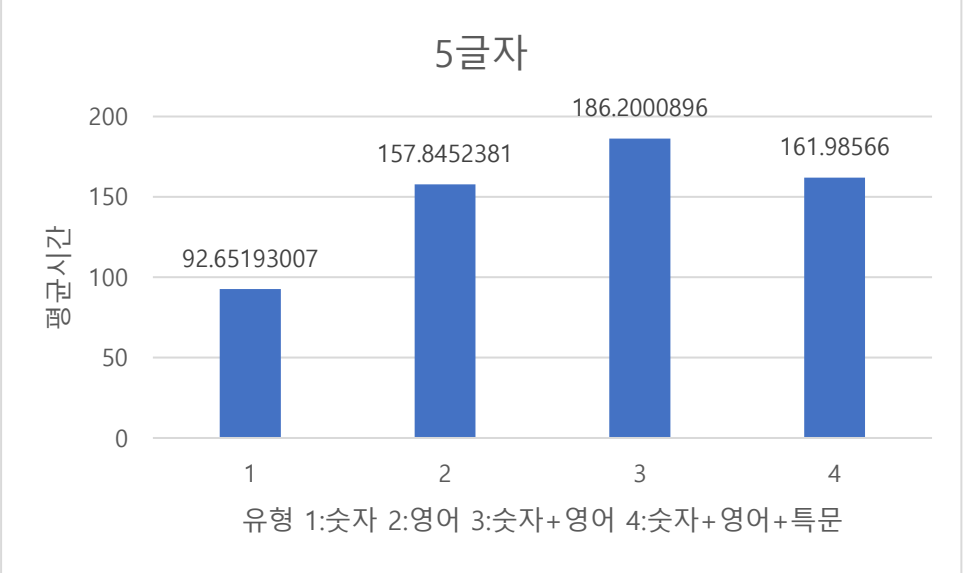
2. 결과

-4 글자



세로 축 범위가 촘촘해 차이가 많이 나 보이지만 0.2 초 이내의 차이를 보인다.

-5 글자



비밀번호가 숫자로만 이루어진 경우를 제외하면 평균 소요시간이 비슷하다.

3. 분석

4 글자 비밀번호는 비교적 비밀번호 구성 유형에 관계 없이 결과가 일정하게 나왔다. 소요되는 시간도 굉장히 짧았다. 반면에 5 글자 비밀번호는 소요되는 시간도 굉장히 길어지고, 결과도 4 글자에 비해 비교적 차이가 났다. 아마 값이 나오는 범위를 생각하면 4 글자 비밀번호도 차이가 꽤 난 것일수도 있다. 실제 시행을 해보기 전 논리적으로 생각해 보았을 때 비밀번호 유형 관계 없이, 공격시 생성되는 문자열이 영어, 숫자, 특수문자를 모두 포함하기 때문에 확률이 어차피 같지 않을까 생각을 했다. 실제로도 그러한 부분들이 어느정도는 반영이 된 결과가 도출되었다. 어느정도 차이는 있지만 영어,숫자,특수문자가 합쳐진 결과만 압도적으로 오래걸린다거나 하는 문제는 안생겼다.

유형 별 결과와 관계 없이 분명히 알 수 있는 것은 자리수가 하나 늘어 날 때 마다 시간이 정말 기하급수적으로 증가한다는 것 이다. 4 자리 비밀번호를 분석하는데 걸린 평균시간은 3 초정도인 것에 반해 5 글자는 160 초 이상이 평균적으로 걸렸다. 영어, 숫자 , 특수문자를 합친 개수가 60 여가지 되므로 자리가 하나 늘어 날 때마다 경우의 수가 60 배가 늘어나기 때문에 시간도 그와 비슷하게 증가했다고 예측 할 수 있다. 이를 통해 만약 6 글자의 비밀번호를 테스트를 한다면 약 9600 초 정도 걸릴 것으로 예측 할 수 있다.

5 글자 비밀번호에서 숫자로만 이루어진 비밀번호가 유독 작게 나온 이유는 테스트 하는 과정에 생긴 문제일 것으로 추측한다. 프로그램을 실행하는 과정중에 중간에 멈추는 경우가 있어 10-20 개 내외로 나온 결과들만을 취합하고 프로그램을 재실행하는 방식을 4 회 정도 반복하였는데 그러는 과정중에 유독 짧게 걸린 값들이 많이 섞여 평균이 많이 낮아진것으로 보인다. 이것도 시행 회수를 충분히 한다면 여타 다른 유형들과 비슷한 값이 나올 것으로 생각된다.

4. 소감

프로그램 구현보다 구현한 프로그램을 실행하고 기다리는 시간이 참 힘들었다. 처음에는 그냥 단순하게 프로그램을 구현하고 실행하였었다. 그렇게 해서 5 글자 비밀번호 4 유형으로 돌리는데 약 6 시간정도 소요되었던 것 같다. 이렇게 해선 너무 오래걸리겠다는 생각에 멀티스레드로 프로그램을 구현 하여보았는데, 아마 이 과정에 문제가 조금 있어서 프로그램이 돌아가다가 멈추는 현상이 생긴 것 같다. 중간에 멈추는 문제 때문에 20 여회정도 계속 시행을 해보았는데, 도중에 딱 한번 끝까지 실행이 되었다. 시간은 약 1 시간 30 분정도 소요되었는데, 4 스레드로 구현하였으니 기존 6 시간과 비교했을 때 납득할 만한 시간이 소요 된 것 같다. 급하게 만드느라 어디에 문제가 있는지 완전히 파악을 못하여 완벽하게 하지 못한 것이 못내 아쉽다. 시간이 오래걸리는지라 컴퓨터 여러대로 테스트를 해보았는데, 컴퓨터 사양에 따른 연산속도 차이가 확연히 났다. 위 보고서에는 포함하지 않았지만, 별도로 첨부할 엑셀에는 5 글자 4 유형 테스트 결과가 두가지 있다. 하나는 노트북, 하나는 데스크탑 PC 에서 실행한 것인데, 두 시행의 평균시간이 161 과 241 로 상당히 차이가 많이 났다. 프로그램을 잘 짜는 것도 중요하지만 그만큼 좋은 컴퓨터로 연산을 하는게 또 중요하구나 라는 큰 깨달음을 얻었다.

내가 생각하기에 프로그램 자체가 무겁지 않을 것이라 판단하였는데 전혀 아니었다. 6 년이 넘은 내 노트북은 여태껏 낸적 없는 팬소리가 나며 살려달라고 비명을 질렀고, 나의 데스크탑도 CPU 가용율이 계속 90 프로 이상을 보였다. 아 내가 생각하는 것 보다 난수를 생성하고 비교하는 것이 횟수가 많아지면 컴퓨터에 엄청난 무리를 줄 수 있겠구나 싶었다. 덕분에 본의 아니게 나와 내 친구들의 컴퓨터를 간접적으로 벤치마크 하는 재미있는 경험을 할 수 있었다.

영화에 보면 pda 단말기로 6-8 자리 숫자 비밀번호를 몇 초만에 뚝딱 풀어내는 장면들이 종종 나온다.심심해서 해보았는데 정말 1 초도 안되는 시간에 다 맞춘다. 숫자로만 이루어진 비밀번호는 정말 강도가 약한 보안이구나 직접적으로 체감이 되었다. 그렇다고 영어+숫자+특수문자로 이루어진 비밀번호는 안전하냐 한다면 이전에 손윤식 교수님께 프로그래밍 언어 개론을 수강할 때 들었던 말씀이 생각난다. 인간은 비슷해서 영어 대소문자 숫자 특수문자로 비밀번호를 구성해도 보통 첫글자 대문자와 영문자 몇글자, !나 @와 숫자로 비밀번호를 구성한다고. 나만 그런줄 알았는데 다들 그랬던 모양이다. 역시 보안에서 제일 중요한 것은 인간 자신이 아닐까 싶다.