컴퓨터보안 보고서

실습과제 #01

강좌 명: 컴퓨터보안

교수님: 정준호 교수님

학과: 컴퓨터공학과

학년: 3학년

학번: 2017112138

이름: 정여준

소스코드

```
#include<iostream>
#include<stdlib.h>
#include<time.h>
using namespace std;

void caseRun(int keyLen, char type[]);
float findKey(int keyLen, char type[]);
int sum(int[], int);

void caseRun(int keyLen, char type[])
{
	cout << "================================" << endl;
	cout << "키 길이 : " << keyLen << endl;
	cout << "가능한 키 : " << type << endl;
	float sum = 0; //평균을 구하기 위한 실수형 변수 sum 선언
	float temp = 0; //각 케이스 별로 걸린시간을 저장하기 위한 실수형 변수 temp 선언
	for (int i = 0; i < 100; i++)
	{
	temp = findKey(keyLen, type);
	cout << "<case " << i + 1 << "> : " << temp << endl;
	sum += temp;
	}
	cout << "평균 : " << sum / 100 << endl;
	cout << "================================" << endl;
}

float findKey(int keyLen, char type[])
{
	srand((unsigned)time(NULL)); //시드별로 다른 난수를 생성
	clock_t start, end; //시간 측정을 위해 선언
	int alen = 0; //type 배열의 길이를 구하기 위한 정수형 변수
	while (type[alen] != NULL)
	{
	alen++;
	}
	int keys[8] = { 0 };
	int correctKey[8] = { 0 };
	int flag = 0;
	for (int i = 0; i < keyLen; i++) //암호 생성
	{
	correctKey[i] = rand() % alen;
	cout << type[correctKey[i]];
	}
	cout << " / ";
	start = clock();
	do //암호 찾기
	{
	for (int k = 0; k < keyLen; k++)
	{
	keys[k]++;
	if (keys[k] == alen)
	keys[k] = 0;
	else
	break;
	}
	flag = 0;
	for (int i = 0; i < keyLen; i++)
	{
	if (keys[i] != correctKey[i])
	{
	flag = 1;
	}

	}
```

```
        } while (sum(keys, keyLen) != 0 && flag == 1);
        end = clock();
        return (float)(end - start) / CLOCKS_PER_SEC; //시간을 초단위로 계산
}
int sum(int keys[], int keyLen)
{
        int result = 0;
        for (int i = 0; i < keyLen; i++)
        {
        result += keys[i];
        }
        return result;
}
int main()
{
        char typeA[] = "0123456789"; //숫자만
        char typeB[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";  //알파벳
대소문자
        char typeC[] = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
//숫자와 알파벳 대소문자
        char typeD[] =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ~!@#$%^&*()_+W<>?/,.|[]{}'
;:"; //숫자와 알파벳 대소문자 그리고 특수문자

        caseRun(4, typeA);


}
```

결과분석

	1회	2회	3회
숫자 4	0.00031	0.00036	0.00027
숫자 5	0.0315	0.00332	0.00316
숫자 6	0.02807	0.02888	0.02793
숫자 7	0.25986	0.25947	0.25472
숫자 8	2.58361	2.87018	2.74879
알파벳 4	0.15633	0.14668	0.13739
알파벳 5	7.06255	9.68932	8.375935
알파벳 6	357.32122	341.79521	368.37618
알파벳 7			
알파벳 8			
숫자 알파벳 4	0.30607	0.41686	0.40568
숫자 알파벳 5	24.76781	25.95791	24.81234
숫자 알파벳 6	329.3772	363.21314	345.16682
숫자 알파벳 7			
숫자 알파벳 8			
숫자 알파벳 특수문자 4	1.32692	1.2955	1.13521
숫자 알파벳 특수문자 5	189.46351	194.18872	192.10766
숫자 알파벳 특수문자 6			
숫자 알파벳 특수문자 7			
숫자 알파벳 특수문자 8			

분석 및 소감문

무작위 공격이란 브루트 포스 공격이라도 불리며 무차별로 대입해서 억지로 비밀번호를 푸는 것 같은 단순한 공격 기법이다. 위에 표를 보면 알 수 있듯이 숫자만 가지고 비밀번호를 설정했을 때는 조합가능한 문자가 그렇게 많지 않기 때문에 많은 시간이 걸리지 않는다. 하지만 이 방법은 케이스가 조금만 더 많아져도 푸는데 걸리는 시간은 기하급수적으로 증가하게 된다. 위에 표에서 비어있는 부분들이 그런 부분들이다. 케이스가 증가해서 프로그램을 돌렸을 때 너무 오래 걸려 측정을 할 수 없는 부분인 것이다. 여기서 알 수 있는 것은 비밀번호는 길고 복잡할수록 무작위 공격 방식으로 알아내기가 힘들다. 여기서 표를 보면 역시 가장 복잡한 숫자와 알파벳 그리고 특수문자를 모두 사용한 것이 굉장히 비밀번호를 찾아내기 힘든 것을 볼 수 있다. 따라서 비밀번호는 복잡하고 길게 설정하는 것이 좋을 것 같다. 또 보안성을 높이기 위한 방법으로는 비밀번호를 입력하는 시도 수를 제한하는 방법이다. 무작위 공격 같은 경우에는 모든 조합을 다 해보는 것이기 때문에 시도 자체가 많을 수밖에 없다. 그렇기 때문에 패스워드를 입력할 수 있는 시도 수에 제한을 두고 제한을 초과할 시에는 잠가버린다면 보안성을 높일 수 있을 것 같다. 또한 유추해내기 힘든 비밀번호로 설정하는 것이 좋을 것 같다. 아무리 비밀번호를 길게 만들고 복잡하게 하고 시도 수를 제한하더라도 그 사람에게서 유추할 수 있는 비밀번호로 설정했다면 경우의 수가 굉장히 줄어들기 때문이다. 예를 들어 내가

축구 팀 첼시를 좋아한다고 첼시11과 같은 비밀번호를 사용한다 하면 내가 첼시를 좋아한다는 사실을 알아낸 해커가 비밀번호를 쉽게 유추할 수 있기 때문이다. 따라서 연관성이 없는 유추하기 힘든 비밀번호로 설정하는 것이 좋을 것 같다. 이번 실습 과제를 통해 무작위 공격에 취약한 비밀번호와 강한 비밀번호에 대해서 알 수 있게 되었다. 이를 통해서 더욱 컴퓨터 보안에 신경을 써야겠다고 다짐했다.