Computer Security
Term Project

- Programming a secure online bank -

학 과 컴퓨터공학과
학 번 201211704
이 름 김기홍

제 출 일 2016.09.25

부경대학교

<목 차>

<그림 목차>

1 서론

16년도 2학기 컴퓨터 보안 과목의 Term Project는 '온라인 뱅킹 시스템'을 기본 모델로 하여 인증, 인허, 암호, 보안 프로토콜 등의 보안 시스템을 이해할 수 있도록 프로그래밍 하는 것이다.

본 프로젝트의 범위는 '정보보안 이론과 실제' 교재의 내용을 기반으로 한다. 하지만 이는 추후 교수님의 추가적인 요구 사항에 따라 확장되어질 수 있음을 명시한다. 개인적으로 추후에 완성이 되어갈 즈음에는 16년도 1학기 컴퓨터 네트워크 과목에서 진행했던 Term Project#2의 소규모 네트워크에 접목시켜 프로젝트의 규모를 조금 더 확장시킬 생각이다.

이 프로젝트는 현대에 널리 사용되고 있는 '온라인 뱅킹 시스템'의 전체적인 보안이 실제로 이루어지는 과정을 이해하고 전체 구조를 그려 봄으로써 여러 파트의 보안 이슈들을 쉽게 이해하며 학습하는 데 목적을 둔다. 추가로 작성자 본인에게는 이 프로젝트를 진행하면서 과거에 학습했었던 소켓프로그래밍부터 새로 학습해나가는 보안 분야까지 모두 한 번에 학습할 수 있어 얻는 이점이 많은 프로젝트임을 밝힌다.

현재(2016. 09. 23) 프로젝트의 진행상황은 tcp 소켓 프로그래밍을 통해 서버와 클라이언트 간에 사용자 등록이 가능하고 ID와 PW를 통해 사용자 인증이 가능하다. 그리고 간단한 입출금 및 계좌 잔액 확인이 가능한 프로그램이며 모든 기능 트랜잭션은 텍스트 파일로 저장된다. 그리고 통신 간 암호 및 복호화 알고리즘 라이브러리를 제작하여 암호화 통신이 가능하다.

다음 진행 단계는 아직까지 공개되지 않았으며, 해당 보고서는 현재 프로젝트 진행에 맞추어 작성되어졌다.

해당 보고서는 서론을 마친 후 구축 환경과 사용된 프로그램을 소개하고 간단한 프로그래밍 기능 구현 설명과 프로그램 실행 결과를 함께 보이고, 그 다음으로 분석 도구를 통한 결과 분석, 끝으로 현재 시스템의 보안상 문제점들, 그리고 현재 구현까지 어려웠던 점, 아쉬웠던 점 그리고 배운 점 등을 개인적인 시각으로 소개하면서 마친다.

앞서 말했듯이 본 프로젝트는 강의 진행에 따라 보안에 초점을 맞춘 요구 기능이 추가된다. 따라서 현재(2016. 09. 23) 작성되어지는 해당 보고서는 최종 프로젝트 결과를 문서화한 것은 아님을 명시한다.

2 배경

우선 이 프로젝트를 진행한 컴퓨터의 시스템 정보는 다음과 같다.

운영체제	Microsoft Windows 10 Pro (64-bit)
프로세서	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz
메모리	8.00GB

그림 2.1 프로젝트 진행 컴퓨터 시스템 정보

그리고 다음은 사용된 툴들의 정보이다.

개발 툴	Microsoft Visual Studio Community 2015
분석 툴	Wireshark 2.2.0 (64-bit), RawCap

그림 2.2 사용된 툴들

해당 툴들을 선택한 이유는 작성자가 기본적으로 사용하는 툴들이고, 그만큼 조작에 있어서 다른 툴들에 비해 비교적 능숙하기 때문이다.

'온라인 뱅킹' 프로그램의 구현은 TCP 연결 기반의 서버/클라이언트로 이루어지고, 서버는 멀티스레딩 환경을 지원하기 때문에 여러 클라이언트가 동시에 접속할 수 있다. 현재는 루프백 인터페이스를 통해 로컬에서만 통신하므로 TCP 통신에 있어서 한 이슈인 메시지 경계 문제는 따로 처리하지 않았다.

3 기능

기능 파트에서는 각 기능에 대한 설명과 프로그램의 실행 예도 함께 병행적으로 보이려한다. 먼저 아래는 서버와 클라이언트가 연결되었을 때 프로그램의 최초 화면을 캡처한 것이다.

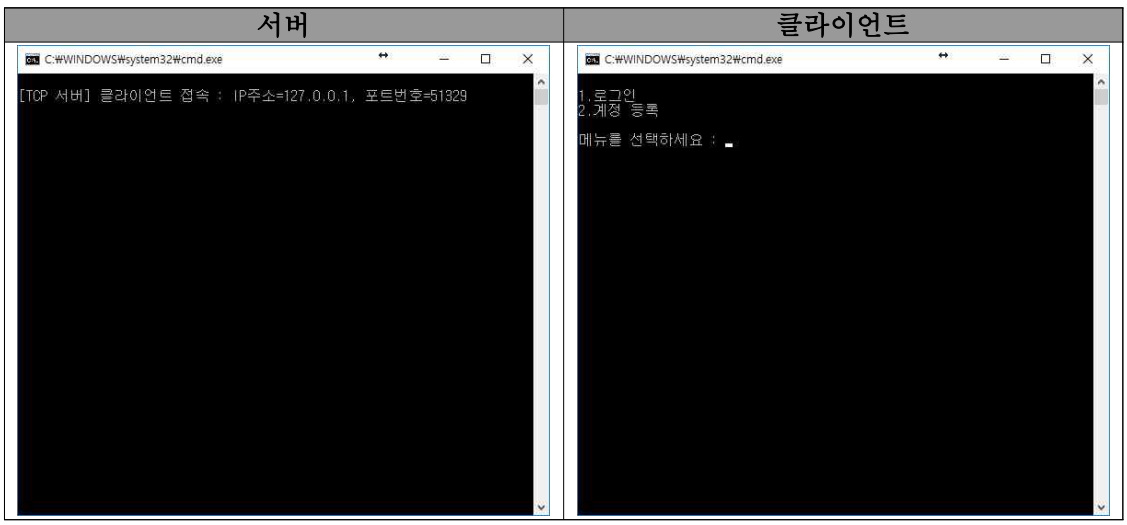


그림 3.1 프로그램 최초 실행 화면

3.1 사용자 등록

사용자 등록 기능은 우선 사용자에게 ID, PW, 이름, 계좌 번호를 입력 받는다. 그리고 그 정보들을 서버로 전송하게 되면, 서버에서는 내부의 테이블에 그 정보들을 저장한다. 아래는 정보가 저장되는 구조체의 도식이다.

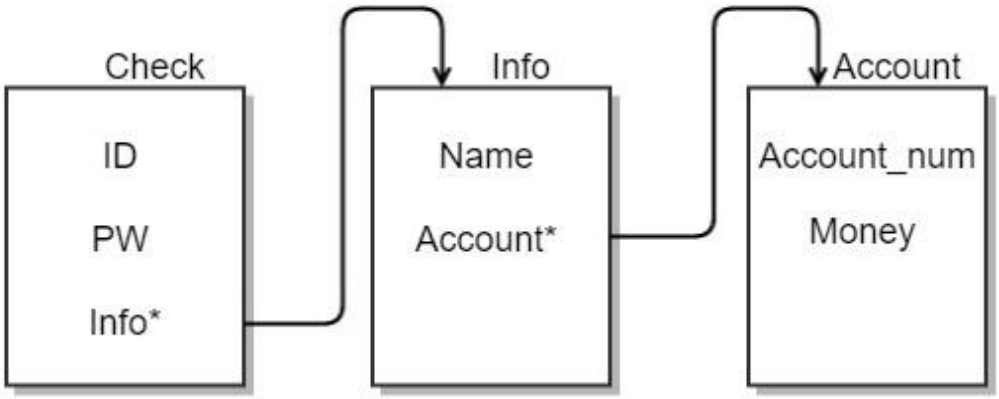


그림 3.1.1 서버 내부 데이터 저장 구조도

이러한 구조를 설계한 이유는 추후에 프로그램이 확장될 경우를 생각해서 (예를 들어 한 고객이 여러 계좌 소유 가능) 만들었기 때문이다. 다음은 클라이언트의 등록 화면을 캡처한 것이다.

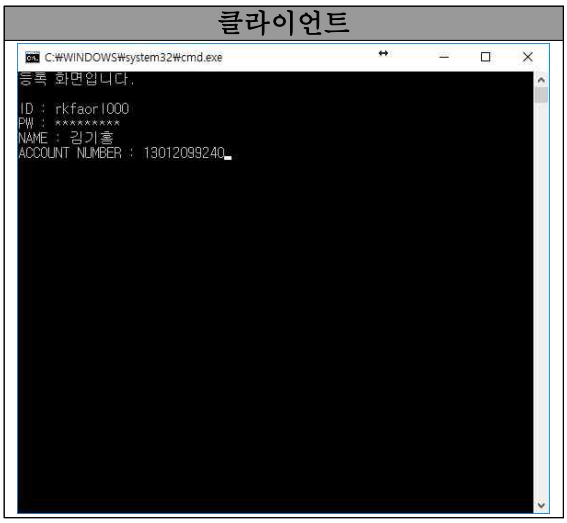


그림 3.1.2 클라이언트 등록 화면

등록 화면에서 고객의 정보를 입력받는 모습이다. PW는 입력이 노출되지 않도록 특별히 문자 '*'로 출력하였다.

3.2 사용자 인증

사용자 인증 기능은 이 프로그램에서 로그인 기능과 같다. 로그인 화면에서는 사용자의 ID와 PW를 입력받아 서버 측에서는 Check 구조체를 참조해 인증 절차를 거친다. 다음은 로그인 화면을 캡처한 것이다.

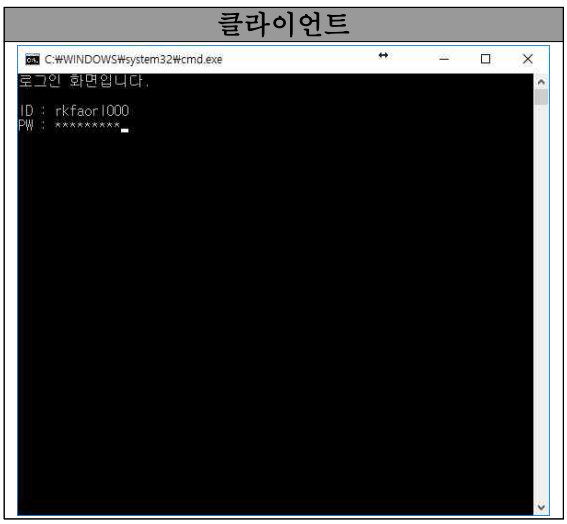


그림 3.2.1 클라이언트 로그인 화면

아이디와 패스워드가 잘 못 입력되었을 경우 각각에 다른 메시지가 띄워진다.

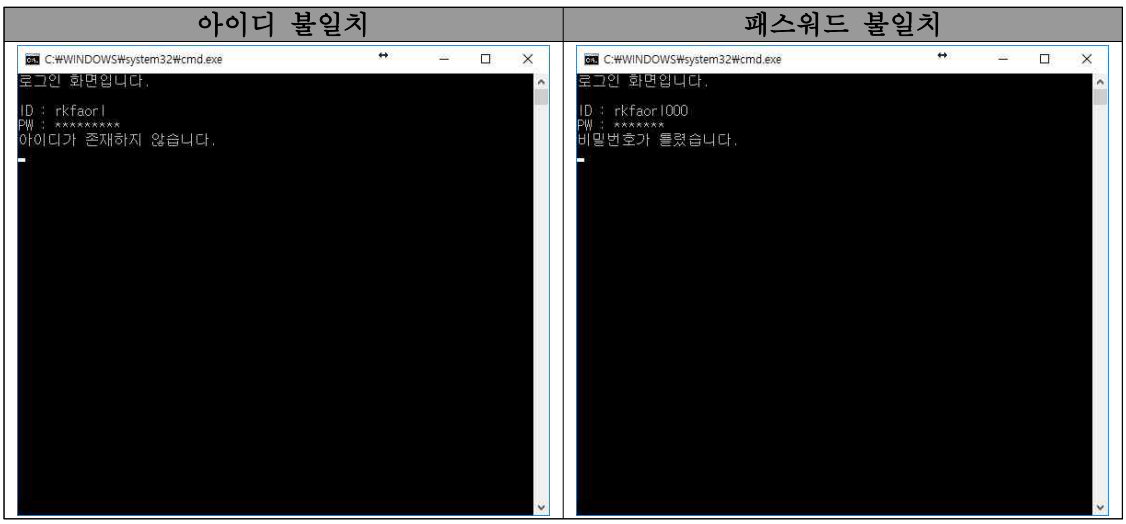


그림 3.2.2 로그인 입력 정보가 조회되지 않을 경우

물론 조회가 성공적으로 이루어지면 로그인이 성공했다는 메시지와 함께 다음 화면으로 넘어간다.

3.3 계좌 조회 및 입·출금

로그인이 성공한 후 부터는 해당 Check 구조체가 참조하고 있는 Info 구조체, Account 구조체로 접근이 가능하다. 이러한 방법으로 계좌 조회, 입금, 출금 모두 이루어진다. 다음은 로그인이 성공했을 때의 화면을 캡처한 것이다.

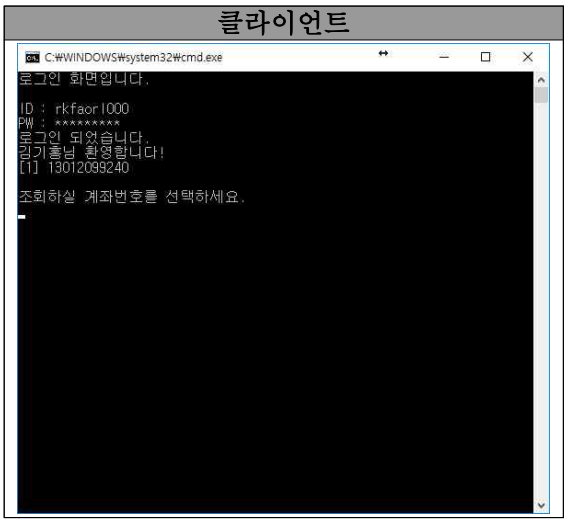


그림 3.3.1 클라이언트 로그인 성공 화면

그림 3.3.1을 보면 조회할 계좌번호 선택을 요구하는데, 이는 앞서 말했듯이 기능이 확장되었을 때 한 사람이 여러 계좌를 가질 수 있도록 미리 구현해둔 기능이다. 물론 지금은 한 사람당 한 계좌이기 때문에 번거롭게 생각할 수도 있다. 다음은 계좌 조회 화면을 캡처한 것이다.

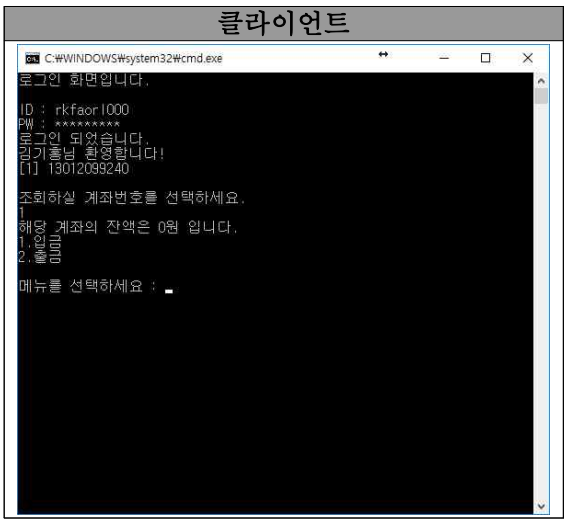
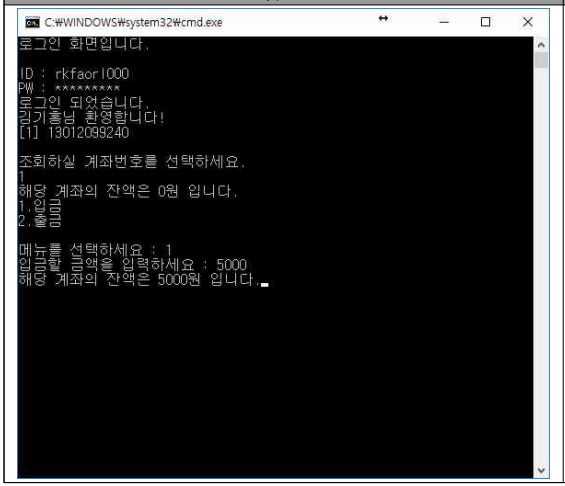


그림 3.3.2 계좌 조회 화면

계좌 조회는 현재 계좌의 잔액을 보여준다. 다음은 입금과 출금 기능을 사용한 후의 잔액 조회 결과를 캡처한 것이다.

입금



출금

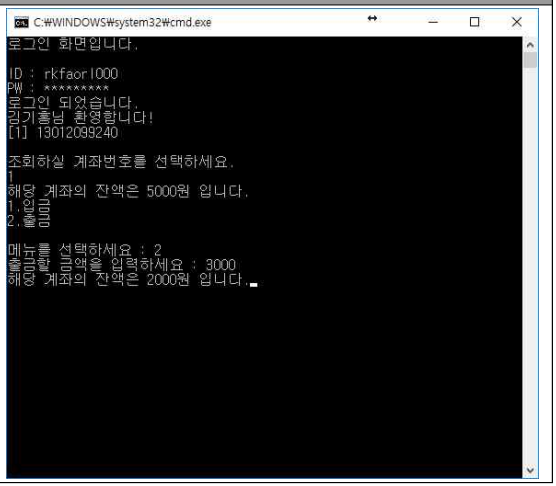


그림 3.3.3 입금 및 출금 화면

3.4 로그 남기기

해당 프로그램의 모든 트랜잭션은 서버 측에 Log.txt 파일로 저장된다. Log.txt에는 당시 트랜잭션이 수행된 시간과, 어떤 트랜잭션인지, 그리고 어떤 데이터를 주고받았는지 확인 가능하다. 다음은 Log.txt 내용의 일부분을 캡처한 화면이다.

Log.txt

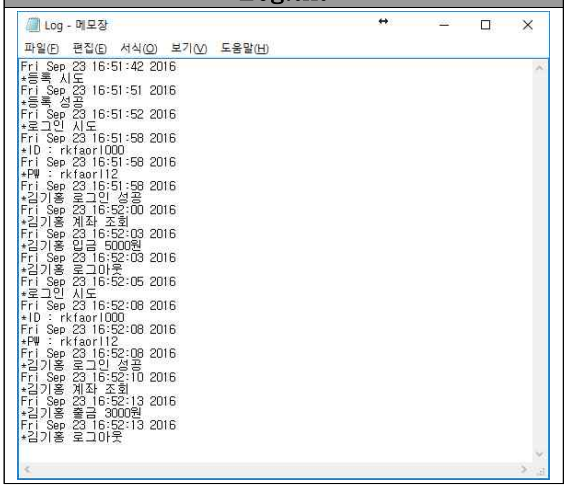


그림 3.4.1 계좌 조회 화면

3.5 암호화

암호화는 공개키 방식 중 RSA 암호화 알고리즘을 사용하였다. RSA 암호화 알고리즘을 선택한 이유는 이산수학 강의에서 배운 적이 있어서 한번쯤 구현해보고 싶었기 때문이다. RSA 암호화 라이브러리를 사용할 때는 프로그램 실행 중에 항상 윗부분에 RSA 알고리즘의 euler_pi, public_key, private_key의 값을 출력해준다.

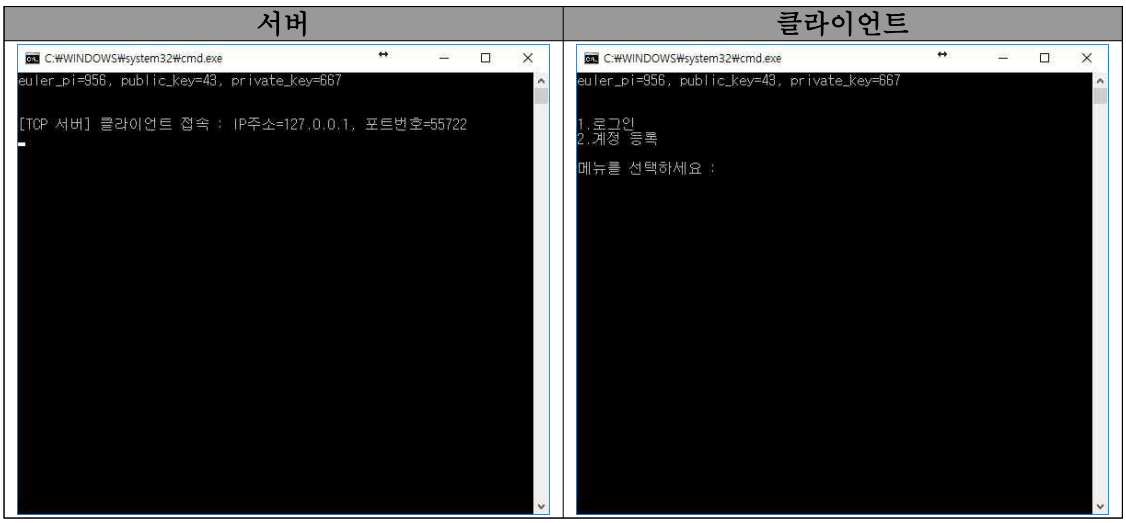


그림 3.5.1 RSA 암호화 라이브러리를 사용할 때 프로그램 실행 화면

다음은 직접 구현한 RSA 암호화 라이브러리 내부 Function Diagram이다.

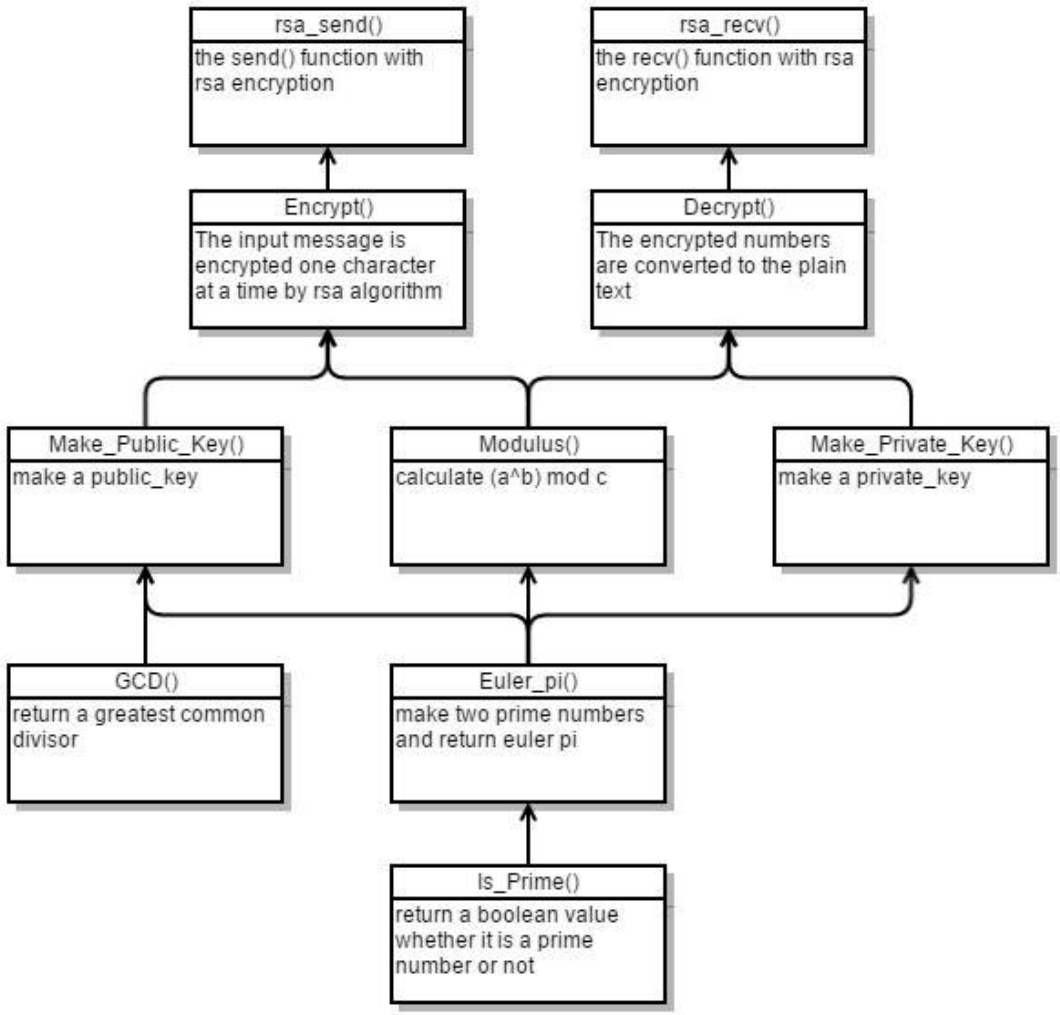


그림 3.5.2 RSA 암호화 라이브러리 내부 Function Diagram

RSA 암호화를 tcp 통신에 쓰이는 send() 함수와 recv() 함수와 결합하여 rsa_send() 함수와 rsa_recv() 함수를 작성하였다. 서버와 클라이언트는 각자 자신의 공개키를 주고받은 다음부터는 앞서 소개된 함수를 통해 쉽게 암호화된 메시지를 주고받을 수 있다. 하지만 아쉽게도 라이브러리 내부에서 암호화와 복호화가 이루어지므로 Log.txt 파일에는 암호화된 내용을 확인할 수 없다. 대신 다음 파트에서 RawCap과 Wireshark 스니핑 툴을 사용하여 암호화 전 후의 메시지를 살펴보자.

4 분석

분석을 위한 툴로는 Wireshark가 Windows OS에서 스니핑할 수 없는 루프백 인터페이스의 패킷을 스니핑해 줄 RawCap과 그 결과물을 분석해 줄 WireShark 툴을 사용한다. 아래는 RawCap 툴을 통해 스니핑 하고 있는 화면을 캡처한 것이다.

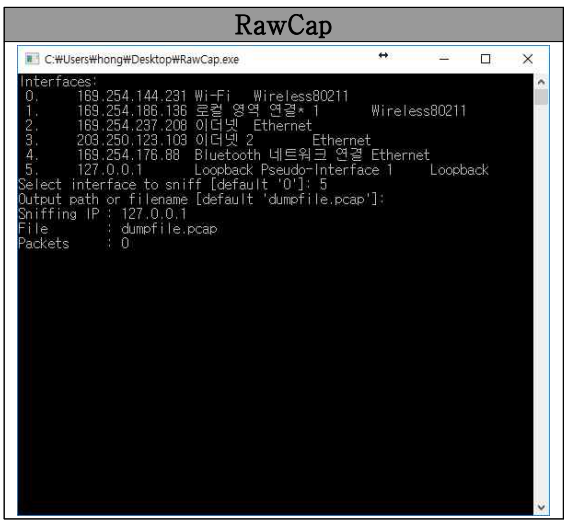


그림 4.1 RawCap 툴을 사용해 패킷을 스니핑 중인 화면

이제 루프백 인터페이스를 통하는 패킷을 스니핑 할 준비가 되었다. 이제 암호화가 되지 않은 상태에서 고객이 로그인할 경우 패킷이 어떻게 보여 지는지 살펴보자.

그림 4.1 WireShark를 통해 분석한 패킷

보다시피 서버와 클라이언트 사이에 통신 내용이 그대로 노출되는 것을 볼 수 있다. 이번에는 같은 부분을 암호화한 후에 살펴보도록 하자.

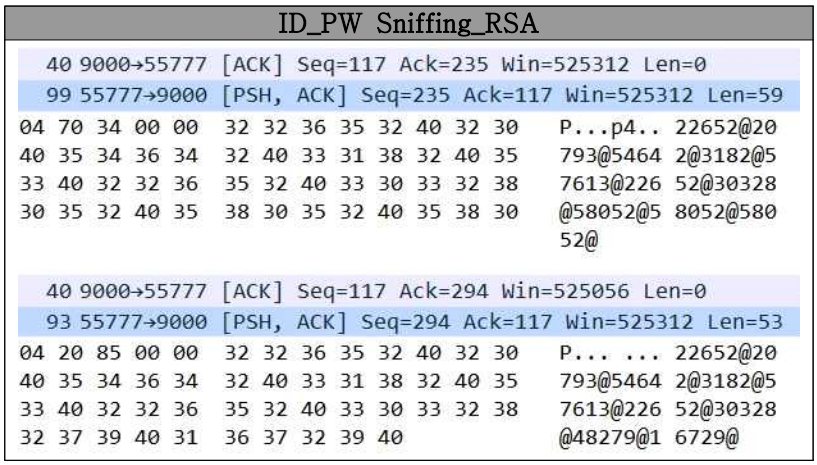


그림 4.2 WireShark를 통해 분석한 패킷(RSA)

이번에는 통신상에서 ID와 PW가 알 수 없는 숫자로 이어져 있는 것을 볼 수 있다. 이 숫자들은 각각 한 문자 당 RSA 암호화 알고리즘을 통해 암호화한 내용이다. 중간 중간 보이는 '@' 문자는 각 문자 단위로 구별할 수 있게 해주는 구별자 역할을 한다.(ID와 PW는 @문자가 들어가면 안 되도록 하거나 메시지의 길이를 첨부하는 등의 제약을 따로 둘 필요가 있다.)

5 회고

이번 파트에서는 현재 시스템에서 어떤 보안상 문제점들이 예상되어지는지, 그리고 현재 구현까지 어려웠던 점, 아쉬웠던 점 그리고 배운 점을 개인적인 시각에서 서술해보려 한다.

5.1 현재(2016. 09. 23) 시스템의 보안상 문제점

현재(2016. 09. 23) 구축된 시스템의 보안상 문제점에는 우선 Cut and Paste 공격에 매우 취약하다는 점이다. 즉, 무결성이 전혀 제공되지 않는다. 또한 서버로의 접근에 아무런 제약이 없다는 점과 서버가 분산되어 있지 않아 모든 트래픽을 서버 혼자 처리해야 한다는 점이다. 이것은 가용성과 관계가 있다. 우선 첫 번째 상황에 대해 그림으로 쉽게 살펴보자.

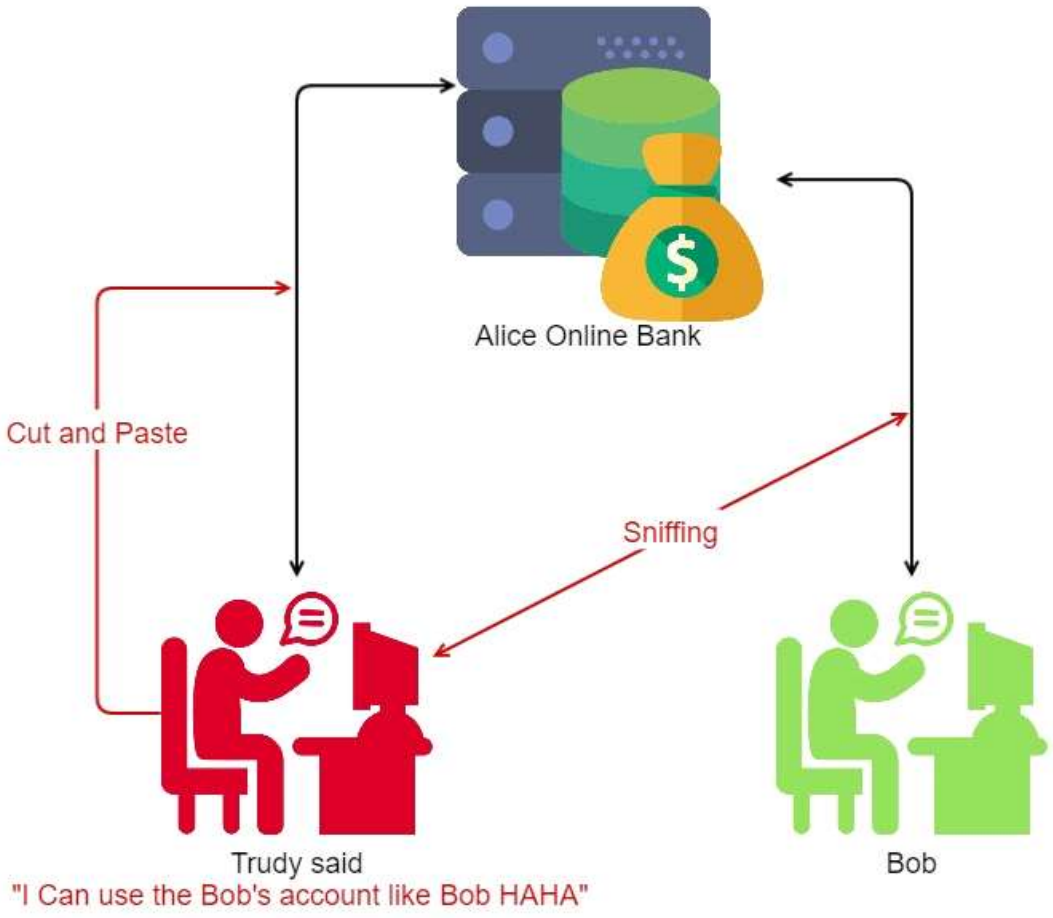


그림 5.1.1 Trudy가 Bob의 계정을 도용하는 상황

그림과 같이 Trudy는 Bob과 서버가 통신하는 내용을 스니핑 하고 있다. 하지만 통신 내용은 RSA로 암호화 되어있기 때문에 봐도 무슨 내용인지 알 수 없다. 그러나 Trudy는 그 암호문 통째로 사용할 수 있다. 서버는 수신한 데이터에 대한 무결성을 의심하지 않기 때문에 Trudy가 Cut and Paste 공격을 했을 때 Trudy와 Bob을 구별할 수 없다. 이는 정말 위험한 상황이다. 이번에는 다음 상황을 살펴보자.

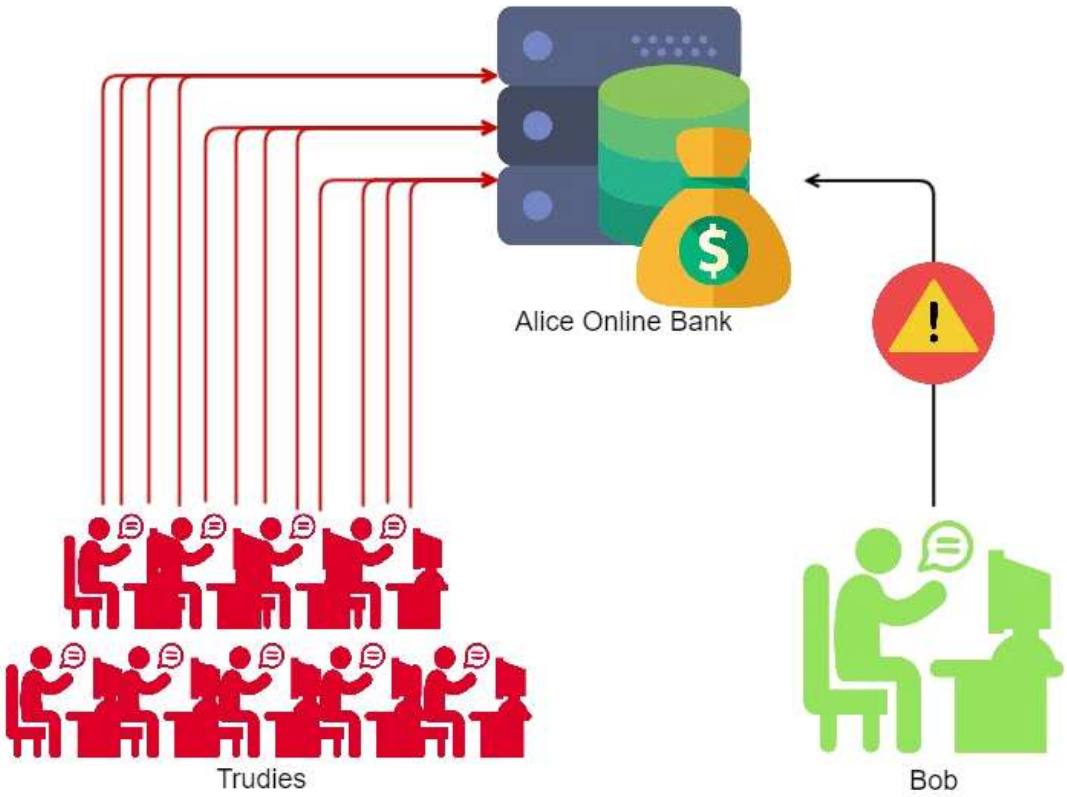


그림 5.1.2 Trudy가 Alice Online Bank Server로 Dos 공격을 하는 상황

이것은 Trudy가 Botnet을 사용해 서버를 향해 Dos 공격을 하고 있는 상황이다. 이렇게 되면 Bob은 물론 다른 모든 사용자들이 Online Banking 서비스를 사용하는데 큰 지장이 있을 것이다. 이렇게 가용성을 보장할 수 없다면 서버와 클라이언트 양 측 모두 큰 피해를 입을 수 있다.

5.2 어려웠던 점

구현하는 부분에서 어려웠던 점은 단연 RSA 암호화 알고리즘 구현이다. RSA 암호화 알고리즘은 수학적인 메카니즘으로 동작하기 때문에 연산을 실제로 구현하기 위해서 생각을 많이 필요로 했던 것 같다.

5.3 아쉬웠던 점

아쉬웠던 점은 몇 가지 있다. 첫째로 암호화 알고리즘을 너무 강력한 걸 사용해서 Trudy 입장이 되어서 중간에 암호를 Breaking하는 테스트를 하지 못했다는 점이다. 이 점은 추후에 현재는 Breaking 되었지만 과거에 널리 사용되었던 암호 알고리즘을 조금 취약하게 구현하여 Breaking 테스트도 진행해볼 예정이다. 둘째로 TCP 경계를 생각하지 않고 프로그래밍 했다는 점이다. 때문에 루프백 인터페이스가 아니면 송수신이 제대로 이루어지지 않는다. 이는 나중에 반드시 수정해야 할 점이다.

5.4 배운 점

Term Project의 초기 부분을 진행하면서 우선 보안이라는 것이 얼마나 넓은 범위를 포함하는 것인지 조금은 알 수 있었다. 그리고 특히 암호 부분을 전부 학습하지는 않았지만 암호 알고리즘의 기발함과 암호 Breaking 기법의 다양함에 머리가 아플 정도로 놀라웠다. 이런 것들을 하나씩 이해해나가고 학습 한다면 사고력 향상에 큰 도움이 될 것이다. 앞으로 학습할 내용도 기대가 된다.