신원기반 네트워크 패킷 접근제어 시스템 개발

팀 명 : 개발자들
지도 교수 : 양환석 교수님
팀 장 : 조재현
팀 원 : 김현진
허송이

2021. 10.

중부대학교 정보보호학과

목 차

1. 서론

1.1 연구 배경 4

1.2 연구 필요성 4

1.3 연구 목적 및 주제 선정 4

2. 관련 연구

2.1 지문인증기술 4

2.1.1 WinBIO 4

2.1.2 WinBioDatabase 5

2.2 Authentication 5

2.3 Inherence Factor 5

2.4 2-Factor Authentication 5

2.5 Challenge-Response Authentication 5

2.6 Socket 5

3. 본론

3.1 시스템 구성 6

3.2 프로그램 구성 6

3.2.1 클라이언트 6

3.2.2 보안 라우터(서버) 7

3.2.3 DB 서버 8

3.2.4 관리/관제 웹 사이트 8

3.3 FINET 프로토콜 구성 12

4. 결론

4.1 결론 13
4.2 기대 효과 13

5. 별첨

5.1 소스 코드 14
5.2 발표 자료 34

1. 서론

1.1 연구 배경

최근 코로나로 인해 많은 기업이 재택근무를 실시하고 있다. 직원들은 외부에서 내부 서버에 접속하기 위해 VPN 등 다양한 방법을 이용한다. 이러한 환경은 사실 보안에 많은 허점이 발생할 수 있다. 외부에 있는 사용자가 내부 직원이 맞는지 정확한 검증이 필요할 것이며, 로그인한 사용자 역시 내부 직원일 것이란 확증이 필요하다. 그래서 최근 OTP나 모바일 인증을 함께 적용하는 등 2-Factor 인증을 통해 기존 ID/PW 기반 인증의 보안을 강화하고 있다. 그러나 현재 2-Factor 인증방식엔 여러 불편함이 따른다. 서버에 접근할 때마다 ID/PW 입력을 수행하고 OTP나 문자 인증을 위해 핸드폰을 보아야 하는 등의 번거로움이 존재한다. 해당 연구에선 'Inherence Factor'에 대한 깊은 고찰을 통해 보안도를 유지하며 사용자 편의성을 높이는 방법을 찾고자 해당 연구를 기획하게 되었다.

1.2 연구 필요성

현재 다양한 인증 솔루션은 존재하나, PC에서 직접 지문 인증을 수행해 2-Factor 인증에 이용하는 솔루션은 존재하지 않는다. 최근 노트북에 지문인식 센서가 들어간 장비가 많아지고 있으며, USB 방식 지문인식 동글의 가격 역시 높지 않다. 이러한 환경을 이용해 2-Factor 인증을 접목시킨 서비스를 만든다면 빠른 속도의 인증과 보안을 동시에 강화된 서비스가 탄생할 것이다. 따라서 본 연구에서 지문 인증을 이용하여 간편한 인증을 제공하고 Challenge-Response 인증 프로토콜을 직접 개발하고, 내/외부망에 접근하는 패킷을 제어하는 시스템 구축을 통해 관리/관제 사이트에서 관리자가 사용자의 접근, 포트 관리와 로그들을 모니터링하여 부적절한 접근을 확인할 수 있는 시스템을 설계 및 개발하고자 한다.

1.3 연구 목적 및 주제 선정

이번 연구는 사용자의 편의성을 증가시키며 2-Factor 인증 실현에 도움을 주는 것을 목적에 둔다. 본 프로젝트는 보안 라우터에 연결된 클라이언트 장비에서의 지문 인증을 통한 패킷 제어를 실현한다. 기존의 핸드폰 인증 등의 불편함을 개선하고 신원인증을 통해 더 빠르고 편리한 본인인증을 수행할 수 있다.

2. 관련 연구

2.1 지문인증 기술

2.1.1 WinBIO

Windows 10과 11에서 지원하는 기능으로 지문 일치를 통해 잠재적인 스푸핑을 방지하는데 도움이 되는 생체 인증 기능이다. 이 기능을 사용하여 지문 센서를 통해 장치 잠금 해제를 하여 인증자는 네트워크에 인증하고 허용하도록 작동할 수 있다.

2.1.2 WinBioDatabase

지문인식 센서가 있는 Windows 시스템에서 발견할 수 있는 파일로 해당 파일은 Windows 생체인식 하드웨어 또는 샘플에 직접 액세스하지 않고 생체 데이터를 저장하는 데이터베이스로 로그온한 사용자의 GUID는 지문/얼굴 인식 정보/캐시가 포함되어 있다. 파일 경로는 C:₩Windows₩System32₩WinBioDatabase 이다.

2.2 Authentication

사용자가 누구인지 확인하는 절차가 곧 인증이라고 할 수 있다. 참이라는 근거가 있는 무언가를 확인하거나 확증하는 행위이다. 사람을 인증한다는 것은 사람들의 신분을 구성하는 것을 말한다.

2.3 Inherence Factor

사용자만의 고유한 속성으로 속성 기반 인증 곧 'What you are'로 표현을 한다. 지문, 홍채, 정맥, 얼굴과 같은 사용자가 고유로 소유한 특징을 통해 인증을 수행하는 것을 의미한다. 이외에도 Knowledge(지식기반) Factor, Possession(소유기반) Factor 등 다양한 Factor가 존재한다.

2.4 Multi-Factor Authentication

2개 이상의 Factor를 이용하여 인증을 구성한 기술이다. 지식, 소유, 속성 등 다양한 Factor 중 2개 이상을 사용하면 Multi-Factor Authentication에 해당한다. 2017년 Black-Hat Conference에서 250명을 대상으로 설문조사를 진행해보니, 38%의 사람들이 Multi-Factor Authentication을 해킹하기 가장 힘들었다고 뽑았다. 그러나 수단이 늘어나는 만큼 편리성이 감소하므로 일반적으로 2-Factor Authentication 방식을 채택하고 있다.

2.5 Challenge-Response Authentication

Hash를 이용한 OTP 인증방식이다. 클라이언트가 서버에 로그인 요청을 보내면 서버는 클라이언트에게 난수를 전달한다. 이후 클라이언트는 받은 난수에 패스워드를 결합하여 Hash를 생성해 서버에 전달한다. 서버는 클라이언트에게 받은 값과 DB에 저장된 패스워드와 전송했던 난수를 결합한 Hash 값을 비교한다. 해당 방식으로 인증을 진행하게 되면 중간자가 패킷을 들여다보아도 실제 패스워드값은 알 수 없다.

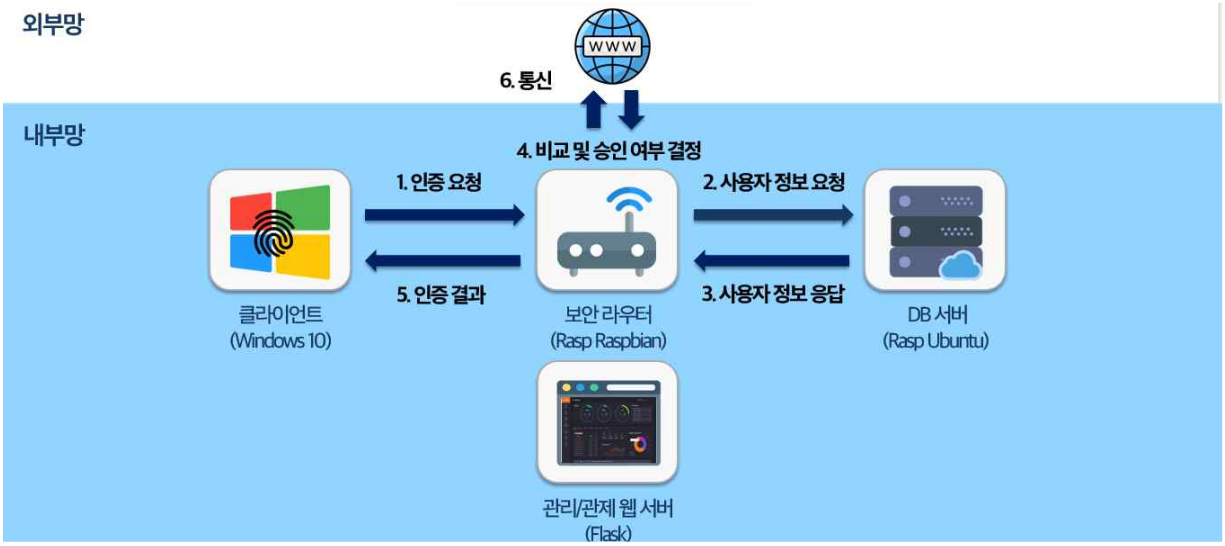
2.6 Socket

소켓이란 네트워크상에서 돌아가는 두 개의 프로그램 간 양방향 통신의 하나의 엔트포인트로 포트 번호에 바인딩되어 TCP 레이어에서 데이터가 전달되어야하는 애플리케이션을 식별할 수 있게 한다. 따라서 소켓을 이용하여 클라이언트와 서버간의 데이터를 주고 받을 수 있는 통신이 가능해진다.

3. 본론

3.1 시스템 구성

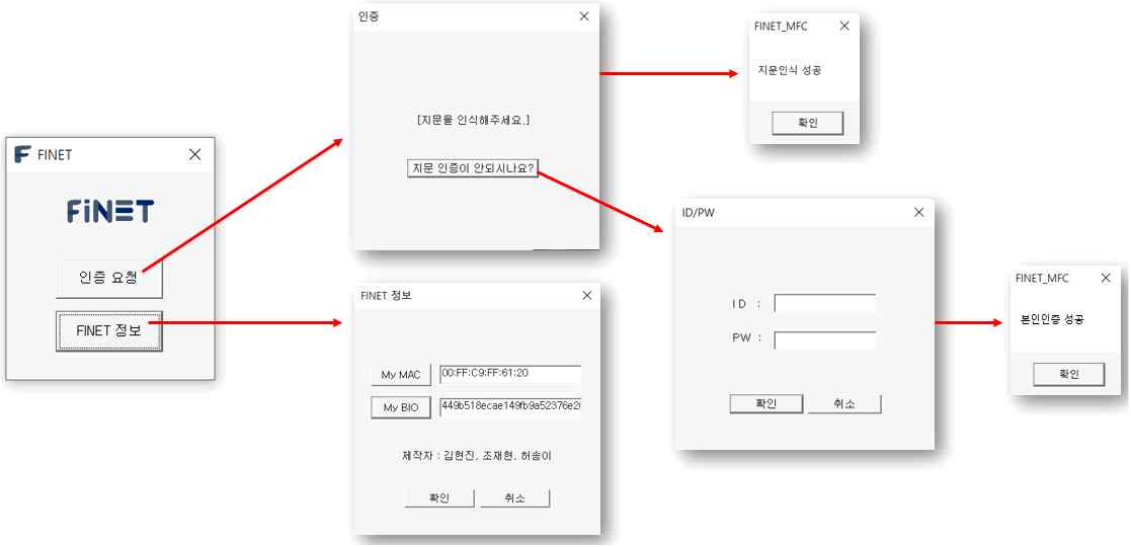
논문에서 제안하는 시스템 동작 과정은 다음과 같다. 해당 구성도에서의 라우터는 스위치 등으로 교체될 수 있으며 내부망에서 외부망으로 나가는 패킷을 통제하는 시나리오로 구성하였다. 우선 외부망에 접속하고자 하는 PC가 지문인식 혹은 ID/PW 방식으로 보안 라우터에 인증을 요청한다. 그럼 보안 라우터는 인증에 필요한 데이터를 DB 서버로부터 받아온 후 PC에게 받은 인증 정보와 비교를 수행한다. 이때 인증 정보가 일치한다면 보안 라우터는 일정 시간 동안 포트를 열어 PC가 외부망의 특정 포트로 접근할 수 있도록 한다. 모든 인증 과정은 로깅 되며 관리/관제 사이트에서 해당 로그를 시각화하여 제공하고 사용자나 포트를 등록하거나 수정, 삭제할 수 있다. 관리/관제 웹서버를 제외한 소프트웨어는 모두 C++ 언어를 통해 개발하였으며 관리/관제 웹서버는 Python 및 Apache, MYSQL 등을 통해 구현하였다.



3.2 프로그램 구성

3.2.1 클라이언트

클라이언트에선 소켓과 MFC를 이용해 GUI로 소켓 클라이언트를 구성하였다. [FINET 정보]를 통해 장비 등록 요청에 필요한 본인의 MAC 주소와 BIO 정보를 확인 가능하다. [인증 요청]을 누르면 지문인식 모듈이 활성화되며 지문인식을 수행한다. 올바른 지문인 경우 '지문인식 성공'이 출력되며 연결된 라우터에 설정된 특정 포트에 10초간 접근 가능하게 된다. 올바른 지문이 아닐 경우 '지문인식 실패'가 뜨며, 서버에 실패 기록이 전송되게 된다. 5회 이상 초과해서 연속적으로 틀리면 계정이 잠김 상태에 들어가게 된다. 추가로 지문인식을 수행 못하는 경우 ID/PW 인증을 수행할 수 있도록 대체 방안 역시 도입해두었다. 최초 장비 등록 시 함께 등록한 ID/PW 정보를 입력하여 지문인식과 같은 결과를 얻을 수 있다.



3.2.2 보안 라우터(서버)

보안 라우터에서는 소켓을 이용해 소켓 서버를 구성하였다. 서버에선 네 가지 기능을 제공한다. 첫째, 클라이언트로부터 받은 인증 정보와 DB에 저장된 인증 정보를 비교하여 Challenge-Response 인증을 수행한다. 둘째, 인가된 사용자에게 일정 시간 동안 포트를 열어준다. 셋째, 통신 과정에서 발생하는 모든 에러 로그들을 파일에 저장한다. 넷째, 인증에 5번 이상 실패 시 계정을 잠금 가능해야 한다. 해당 기능을 구현하기 위해 소켓 클라이언트에서 전송된 패킷에 맞게 다음 패킷을 조립해 전송하며 인증을 수행한다. 관제/관리 웹 서버도 해당 보안 라우터에서 실행되며 보안을 위해 localhost에 한해 접속가능하도록 하였다.

0 (ERR_NOERR) : 성공 (에러 없음)

1 (ERR_FLOW) : 플로우 에러

2 (ERR_CHKSUM) : 체크섬 에러

3 (ERR_OPTION) : 옵션 에러

4 (ERR_SENID) : SenID 에러

5 (ERR_DATA) : Data 에러

6 (ERR_FSCHK) : 초기인증 에러

7 (ERR_CRCHK) : C/R 인증 에러

```
⚡ root@kali  /var/log  cd /var/log/
⚡ root@kali  /var/log  ls -l finet.log
-rw-r--r-- 1 root root 1296 May 18 11:48 finet.log
⚡ root@kali  /var/log  cat finet.log
172.31.13.132-2021/04/14-15:23:27-0
172.31.13.132-2021/04/14-15:23:42-0
172.31.13.132-2021/04/14-15:30:39-0
172.31.13.132-2021/04/14-15:30:50-7
172.31.13.132-2021/04/14-15:33:46-7
172.31.13.132-2021/04/14-15:33:58-0
172.31.13.132-2021/04/14-15:34:07-7
172.31.13.132-2021/04/14-15:34:58-3
172.31.13.132-2021/04/14-15:35:40-1
172.31.13.132-2021/04/14-15:36:17-2
172.31.13.132-2021/04/14-15:37:08-6
172.31.13.132-2021/04/26-16:29:47-2
172.31.13.132-2021/05/10-16:29:47-1
172.31.13.132-2021/05/11-16:29:47-3
172.31.13.132-2021/05/12-16:29:47-0
```

3.2.3 DB 서버

실질적인 사용자 정보가 담긴 Database가 존재하는 서버이다. 보안 라우터에서 동작하는 소켓 서버에서만 접근 가능하며 설계는 다음과 같다. Client의 기기 및 지문정보인 MAC 주소와 WinBioDatabase ID 그리고 대체 수단 제공을 위한 User ID, User Password가 DB에 저장된다. 모든 정보는 원본을 알 수 없도록 SHA-256 암호화 알고리즘을 통해 일 방향 암호화된다.

Finetdb - Member_tb

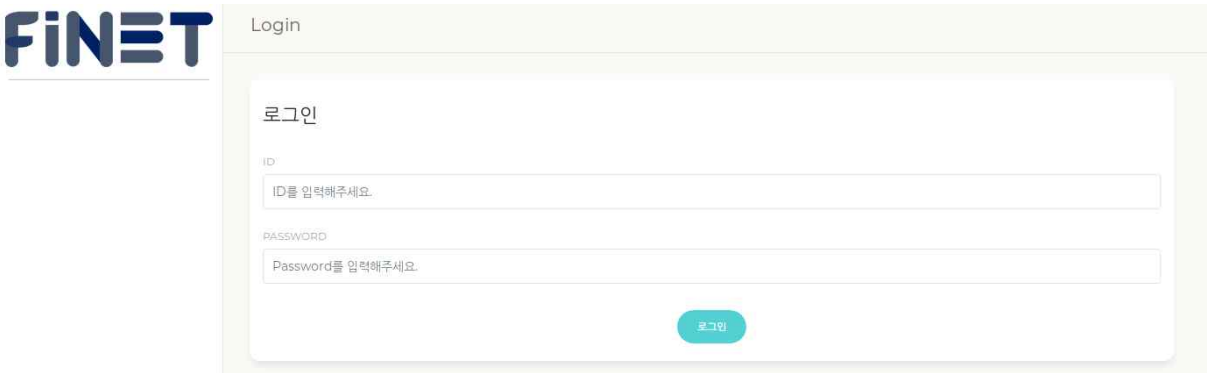
컬럼 명	타입	옵션	정보
member_idx	int(10)	PRI, NOT NULL, AUTO_INCREMENT	기본키, 인덱스
member_id	varchar(64)	UNIQ, NOT NULL	Client MAC Addr (SHA-256)
member_guid	varchar(64)	UNIQ, NOT NULL	Client GUID (SHA-256)
member_subid	varchar(20)	UNIQ, NOT NULL	Client ID
member_subpw	varchar(64)	UNIQ, NOT NULL	Client PW (SHA-256)

```
MariaDB [finetdb]> select * from member_tb;
+-----------+------------------------------------------------------------------+------------------------------------------------------------------+
| member_idx | member_id                                                        | member_guid                                                      |
+-----------+------------------------------------------------------------------+------------------------------------------------------------------+
|          1 | 62492fe30f8558566b6a2288fbae2f82cfde8d4a9d814125c3081f44f691724f | 1b89ef6d0da66e0d222547c223a2023efe58f76a1b7686acf426585684982da3 |
|          2 | c526a6ab42912dbdfc8edf54a399b1c92767141da8802ee9237e7e8c7a2c2fb1 | c6a39891197a579aef09a0fe72f669d2bab9807203a7fdeda7c16423182b9a8e |
+-----------+------------------------------------------------------------------+------------------------------------------------------------------+
```

```
+--------------+------------------------------------------------------------------+
| member_subid | member_subpw                                                     |
+--------------+------------------------------------------------------------------+
| finet        | 404872c562b48cc94b4db5f9c7bdaf4260ac98316672284d8d4d7c04befade82 |
| jaehyeon     | 63403912b98b029748d39be2c1c2ba87f3942dd4f0b5ae6377215fd7f592874e |
+--------------+------------------------------------------------------------------+
```

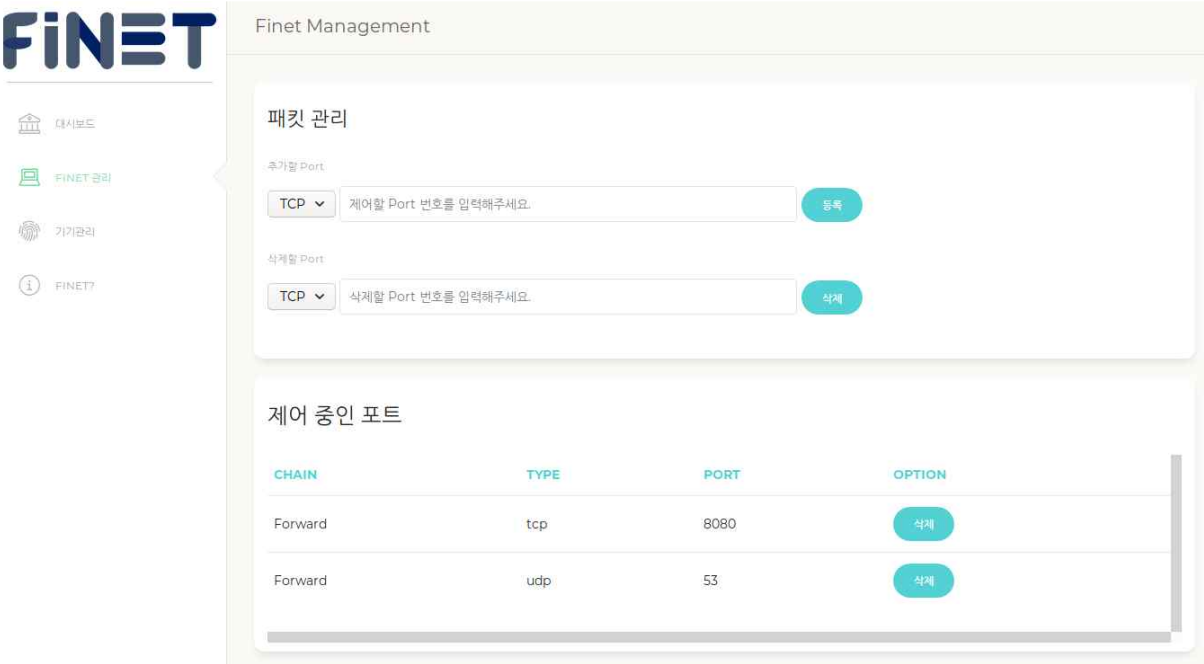
3.2.4 관리/관제 웹 사이트

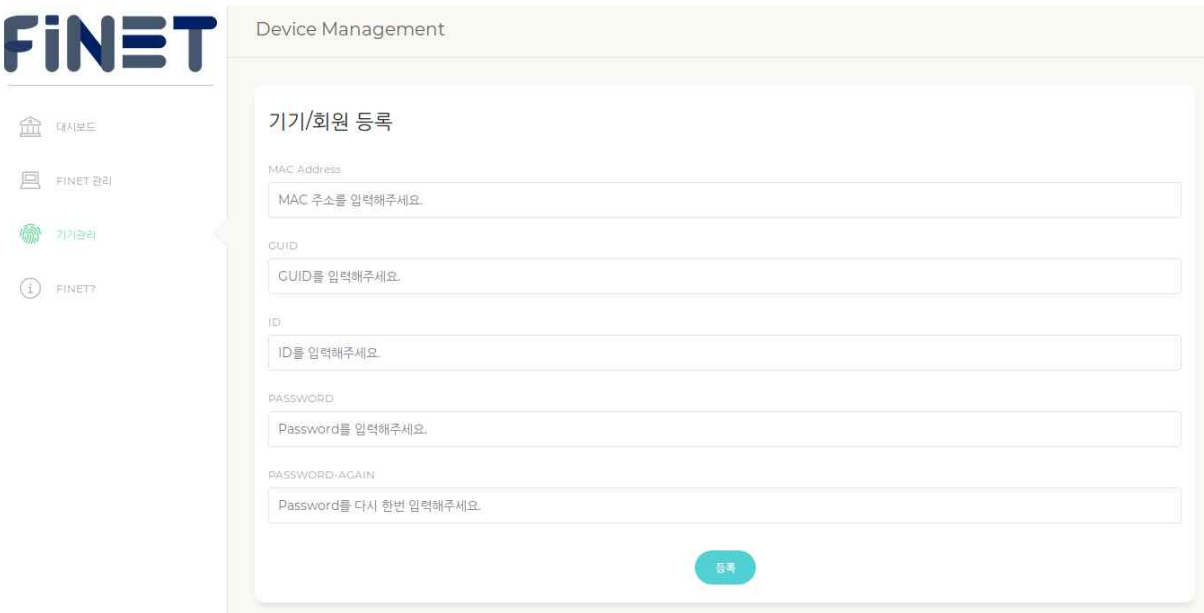
관리/관제 사이트에서는 사용자와 포트의 등록, 수정, 삭제를 편리하게 해주며 에러 로그들을 확인할 수 있어 부적절한 접근을 식별할 수 있다. 대시보드 형태로 요청량과 실패 로그 유형 등을 확인할 수 있으며 모니터링 수행 편리성을 제공한다. 세부 메뉴로 접속해 접근 가능한 포트 관리, 장비 관리, 정보 습득과 같은 활동이 가능하다.

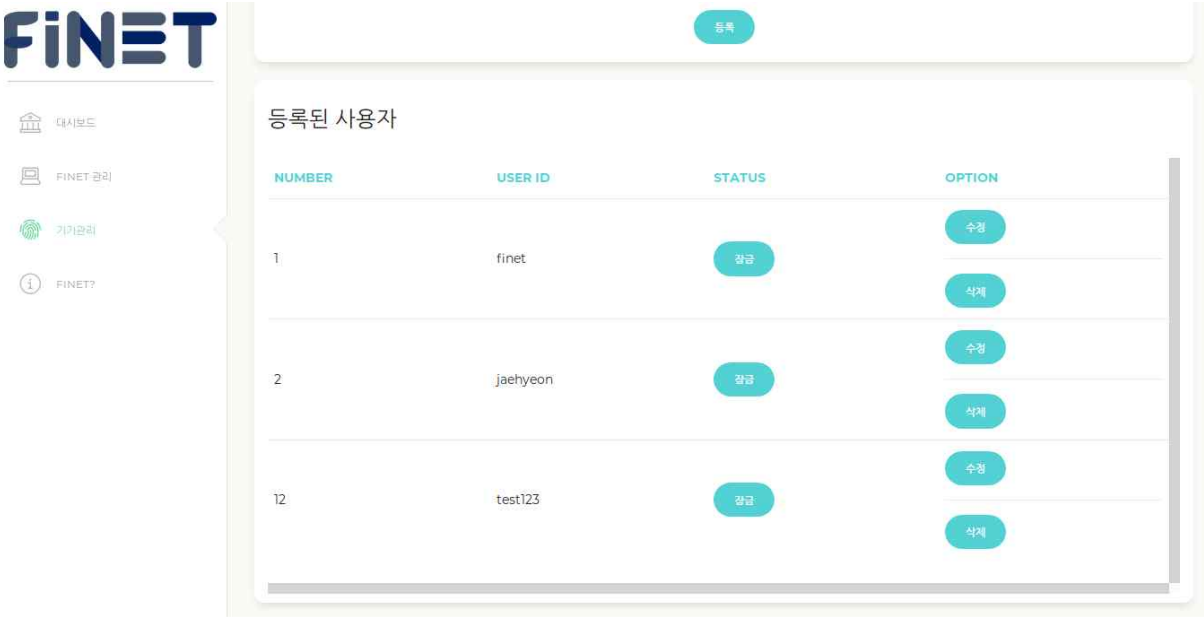

FiNET
Login
로그인
ID
ID를 입력해주세요.
PASSWORD
Password를 입력해주세요.
로그인

FiNET
Finet Dashboard
대시보드
FINET 관리
기기관리
FINET?
누적 요청량
36
Real-time Update
누적 성공
14
Real-time Update
누적 실패
22
Real-time Update
오늘의 요청량
1
Real-time Update
요청량 흐름
최근 요청량 흐름에 대한 정보를 보여줍니다.
6
4
2
0
2021/10/07
2021/10/08
2021/10/09
2021/10/10
2021/10/11
2021/10/12
2021/10/13
2021/10/14
2021/10/15
2021/10/16

FINET?
실패한 요청 분석
실패한 요청들에 대한 정보를 보여줍니다.
6
5
4
3
2
1
0
Flow Err
Checksum Err
Option Err
SerID Err
Data Err
FIDO Fail
C/R Fail
요청 성공/실패량 추이
요청이 성공하고 실패한 추이에 대한 정보를 보여줍니다.
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
2021/10/07
2021/10/08
2021/10/09
2021/10/10
2021/10/11
2021/10/12
2021/10/13
2021/10/14
2021/10/15
2021/10/16
성공
실패


FiNET
Finet Management
대시보드
FINET 관리
기기관리
FINET?
패킷 관리
추가할 Port
TCP
제어할 Port 번호를 입력해주세요.
등록
삭제할 Port
TCP
삭제할 Port 번호를 입력해주세요.
삭제
제어 중인 포트
CHAIN
TYPE
PORT
OPTION
Forward
tcp
8080
삭제
Forward
udp
53
삭제

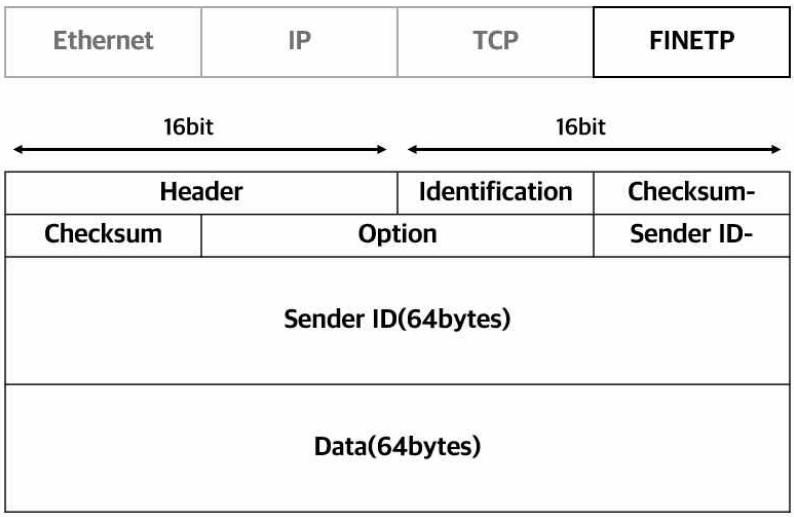

FiNET
Device Management
대시보드
FINET 관리
기기관리
FINET?
기기/회원 등록
MAC Address
MAC 주소를 입력해주세요.
GUID
GUID를 입력해주세요.
ID
ID를 입력해주세요.
PASSWORD
Password를 입력해주세요.
PASSWORD-AGAIN
Password를 다시 한번 입력해주세요.
등록


FINET
대시보드
FINET 관리
기기관리
FINET?
등록
등록된 사용자
NUMBER
USER ID
STATUS
OPTION
1
finet
잠금
수정
삭제
2
jaehyeon
잠금
수정
삭제
12
test123
잠금
수정
삭제

FINET
대시보드
FINET 관리
기기관리
FINET?
FINET?
FINET
FINET은 FIDO와 Network의 합성어로
'신원기반 네트워크 패킷 접근제어'를 실현하는 솔루션입니다.
지문인식을 통해 편리하게 특정 포트에 접근해보세요

3.3 FINET 프로토콜 구성

인증 과정에서의 속도와 보안성을 증가시키기 위해 직접 프로토콜을 설계하였다. 안전한 통신을 위해 OTP 방식 중 하나인 Challenge-Response 방식을 채택하였다.



Header (2bytes)

- 0xF1E7

Identification (1bytes)

- 0x01 (Authentication Request)
- 0x02 (Challenge-Response Challenge)
- 0x03 (Challenge-Response Response)
- 0x04 (Authentication Response)

Checksum (2bytes)

- ~(FINETP[0]+...+FINETP[134](Except [3],[4]))&0x0000FFFF

Option (2bytes)

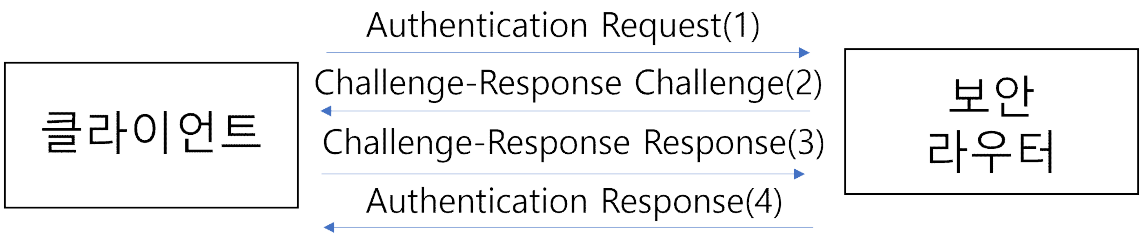
- 0x0001 (Success, When ID(0x03) FIDO Auth)
- 0x0002 (Fail, When ID(0x03) ID//PW Auth)
- 0xXXXX (When ID(0x02) Random Number(Challenge))

Sender ID (64bytes)

- 0x0000...0000 (Padding)
- 0xXXXX...XXXX (When ID(0x03)/Option(0x0001) Client MAC(Hashed))
- 0xXXXX...XXXX (When ID(0x03)/Option(0x0002) User ID)

Data (64bytes)

- 0x0000...0000 (Padding)
- 0xXXXX...XXXX (When ID(0x03)/Option(0x0001) GUID + Challenge(Hashed))
- 0xXXXX...XXXX (When ID(0x03)/Option(0x0002) User PW + Challenge(Hashed))



클라이언트에서 서버로 Authentication Request 패킷을 보내 인증을 요청한다. 그럼 서버는 난수 값을 담은 Challenge 패킷을 클라이언트에게 보낸다. 클라이언트는 받은 Challenge 값으로 Data를 암호화시키고 그 정보가 담긴 Response 패킷을 보낸다. 서버는 암호화된 인증 정보를 DB에 저장된 인증 정보와 비교를 수행하고, 최종적으로 인증 성공, 실패 여부를 포함한 Authentication Response 패킷을 클라이언트에게 보낸다.

4. 결론

4.1 결론

프로젝트를 통해 신원 기반 인증을 통해 인증된 사용자가 특정 IP 및 Port에 접근 가능한 시스템을 만드는데 성공했다. 철저한 설계를 통해 단지 사용자 인증을 수행하는 기능뿐만 아니라 인증 과정 자체에 대한 보안 역시 강화할 수 있었다. 직접 Challenge-Response 인증 패킷을 만들어 패킷에 대한 무결성을 보장하고 및 인증 실패에 관한 내용을 기록한다. 악의적인 사용자로부터 패킷 재전송, 변조를 막을 수 있으며, 인증 결과 로깅을 통해 송신지를 확인할 수 있어 DoS 공격과 같은 네트워크 공격에 대한 추적성을 확보할 수 있다. 사용자를 대상으로 하므로 GUI에 대한 신경도 많이 기울였다. 클라이언트에서는 Windows 10 64비트 운영체제에서 동작하는 GUI 클라이언트 프로그램에서 손쉽게 서버로 인증을 요청할 수 있다. 서버 관리자는 관제/관리 웹 서버를 통해 디바이스를 손쉽게 관리 가능하며, 열어줄 포트에 대한 관리 등이 가능하다. 인증 요청량이나 오류 로그도 시각화하여 쉽게 확인 가능해 모니터링에 대한 가시성을 높일 수 있었다. DB 서버는 라우터와 분리하여 보안도 역시 증가시켰다.

4.2 기대효과

기업에서 Multi-Factor 인증을 필요로 하는 환경에 해당 시스템을 적용한다면 ID/PW 입력을 제외한 하나의 Factor 인증을 매우 손쉽게 이룰 수 있다. 이를 통해 업무 수행에 대한 연속성을 유지 시키고 보안성 및 편리성을 높일 수 있어 업무의 생산성을 향상시킬 수 있는 기술로 기대한다.

5. 별첨

5.1 소스코드

(Client) fido.cpp

```
#include "pch.h"
#include "data.h"
#pragma warning(disable:4996)


// 현재 사용자 ID 검색 __inout : 표기만을 위한 매크로
// PWINBIO_IDENTITY : WINBIO_PIPELINE(단일 생체 센서 유닛, 엔진, 저장 어댑터에 의해 사용되는 공유 컨텍스트 정보를 포함하는 구조)
HRESULT GetCurrentUserIdentity(__inout PWINBIO_IDENTITY Identity) {

    HRESULT hr = S_OK;

    // 액세스 토큰에 열린 프로세스에 대한 핸들
    HANDLE tokenHandle = NULL;

    // 액세스 토큰에 대해 요청된 액세스 유형을 지정
    DWORD  byteReturned = 0;
    struct _tokenInfoBuffer {
        TOKEN_USER tokenUser;
        BYTE buffer[SECURITY_MAX_SID_SIZE];
    } tokenInfoBuffer;

    // 입력 ID 0으로 설정
    ZeroMemory(Identity, sizeof(WINBIO_IDENTITY));  // 구조체 0으로 초기화
    Identity->Type = WINBIO_ID_TYPE_NULL;

    // 현제 프로세스와 연결된 토큰 열기
    // 실패 > 반환값 = 0, GetCurrentProcess() : 현재 프로세스에 대한 핸들을 검색.
    if (!OpenProcessToken(GetCurrentProcess(), TOKEN_READ, &tokenHandle)) {

        // GetLastError() : 호출 스레드의 마지막 오류 코드 값을 검색.
        DWORD win32Status = GetLastError();
        wprintf_s(L"Can not Open Token Handle : %d₩n", win32Status);

        // HRESULT_FROM_WIN32 : 매핑 시스템 오류 코드에 대한 HRESULT 값
        hr = HRESULT_FROM_WIN32(win32Status);
        goto E_Exit;
```

```
    }

    // 토큰버퍼 구조를 0으로 설정
    ZeroMemory(&tokenInfoBuffer, sizeof(tokenInfoBuffer));

    // 액세스 토큰에 대한 정보를 검색한다. (SID검색)
    if (!GetTokenInformation(
        tokenHandle,
        TokenUser,
        &tokenInfoBuffer.tokenUser,
        sizeof(tokenInfoBuffer),
        &byteReturned))
    {
        DWORD win32Satus = GetLastError();
        wprintf_s(L"Can not Query Token Information : %d₩n", win32Satus);
        hr = HRESULT_FROM_WIN32(win32Satus);
        goto E_Exit;
    }

    //tokenInfoBuffer 에서 SID를 WINBIO_IDENTITY 구조로 복사? 구조 다음으로 복사?
    // CopySid() : SID(security identifier)의 버퍼를 복사
    CopySid(SECURITY_MAX_SID_SIZE, Identity->Value.AccountSid.Data, tokenInfoBuffer.token
User.User.Sid);

    //SID 크기를 지정하고 WINB를 할당
    Identity->Value.AccountSid.Size = GetLengthSid(tokenInfoBuffer.tokenUser.User.Sid);
    Identity->Type = WINBIO_ID_TYPE_SID;

E_Exit:

    if (tokenHandle != NULL) {
        CloseHandle(tokenHandle);
    }
    return hr;
}


string DataBases() {
    HRESULT hr = S_OK;
    PWINBIO_STORAGE_SCHEMA storageSchemaArray = NULL;
    SIZE_T storageCount = 0;
    SIZE_T index = 0;
```

```
    string guid = "";

    hr = WinBioEnumDatabases(
        WINBIO_TYPE_FINGERPRINT,
        &storageSchemaArray,
        &storageCount
    );
    if (FAILED(hr)) {
        wprintf_s(L"₩nWinBioEnumDatabases failed. hr = 0x%x₩n", hr);
        WinBioFree(storageSchemaArray);
        storageSchemaArray = NULL;
    }
    guid = SaveGuid(&storageSchemaArray[0].DatabaseId);

    return guid;
}


string SaveGuid(__in PWINBIO_UUID Guid) {
    char S_Guid[16];
    char buffer[64];

    sprintf(
        buffer,
        "%08x%04x%04x%02x%02x%02x%02x%02x%02x%02x%02x",
        Guid->Data1,
        Guid->Data2,
        Guid->Data3,
        Guid->Data4[0],
        Guid->Data4[1],
        Guid->Data4[2],
        Guid->Data4[3],
        Guid->Data4[4],
        Guid->Data4[5],
        Guid->Data4[6],
        Guid->Data4[7]
    );
    string guid(buffer);

    return guid;
}
```

```
// 생체인식 센서
int Verify(WINBIO_BIOMETRIC_SUBTYPE subFactor) {

    HRESULT hr = S_OK;
    WINBIO_SESSION_HANDLE sessionHandle = NULL;
    WINBIO_UNIT_ID unitId = 0;
    WINBIO_REJECT_DETAIL rejectDetail = 0;
    WINBIO_IDENTITY identity = { 0 };
    BOOLEAN match = FALSE;

    int result = 2;
    int option = 2;
    string guid = "";

    // 사용자 identity 찾기
    hr = GetCurrentUserIdentity(&identity);
    if (FAILED(hr)) {
        wprintf_s(L"User Identity not found. hr = 0x%x₩n", hr);
        goto E_Exit;
    }

    // 시스템에 연결
    hr = WinBioOpenSession(
        WINBIO_TYPE_FINGERPRINT,
        WINBIO_POOL_SYSTEM,
        WINBIO_FLAG_DEFAULT,
        NULL,
        0,
        NULL,
        &sessionHandle
    );
    if (FAILED(hr)) {
        wprintf_s(L"₩nWinBioOpenSession failed. hr = 0x%x₩n", hr);
        goto E_Exit;
    }

    // 생체 인식 확인
    // 생체 인식 샘플을 캡쳐하고 사용자 ID에 해당하는지 확인
    wprintf_s(L"₩n** Calling WinBioVerify - Swip finger on sersor ... **₩n");
    hr = WinBioVerify(
        sessionHandle,
```

```
            &identity,
            subFactor,
            &unitId,
            &match,
            &rejectDetail
        );
        wprintf_s(L"₩nSwipe Processed _ Unit ID : %d₩n", unitId);

        if (FAILED(hr)) {
            if (hr == WINBIO_E_NO_MATCH) {
                wprintf_s(L"₩nNO MATCH _ Identity verification failed... ₩n");
            }
            else if (hr == WINBIO_E_BAD_CAPTURE) {
                wprintf_s(L"₩nBAD CAPTURE _ Reason : %d₩n", rejectDetail);
            }
            else {
                wprintf_s(L"₩nWinBioVerify failed ... hr = 0x%x₩n", hr);
            }
            option = 2; // option 2는 지문인식 실패
            guid = DataBases();
            result = Finet(option, guid);

            goto E_Exit;
        }

        option = 1; // option 1은 지문인식 성공
        wprintf_s(L"₩nFingerprint Verified : %d ₩n", unitId);
        guid = DataBases();
        result = Finet(option, guid);

        return result;


E_Exit:
        if (sessionHandle != NULL) {  // 세션이 이미 열려 있으면
            WinBioCloseSession(sessionHandle);    // 세션 닫고
            sessionHandle = NULL;  // 세션 초기화
        }
        wprintf_s(L"₩nClose..... bye₩n");

        return 2;
}
```

(Rasp Raspbian) SocketServer/finet.cpp

```
#include "finet.h"
#include <pthread.h>
#include <fstream>
#include <sstream>
#include <iomanip>


int finet(char *port){
	int servSock;
	int clntSock;

	sockaddr_in servAddr;
	sockaddr_in clntAddr;
	socklen_t clntAddrSize;

	pthread_t thread_t;

	servSock=socket(PF_INET, SOCK_STREAM, 0);
	if(servSock == -1)
		ErrorHandling("socket() error");

	memset(&servAddr, 0, sizeof(servAddr));
	servAddr.sin_family=AF_INET;
	servAddr.sin_addr.s_addr=htonl(INADDR_ANY);
	servAddr.sin_port=htons(atoi(port));

	if(bind(servSock, (sockaddr*) &servAddr, sizeof(servAddr))==-1)
		ErrorHandling("bind() error");

	if(listen(servSock, 5)==-1)
		ErrorHandling("listen() error");

	clntAddrSize = sizeof(clntAddr);
	MultipleArg *multiple_arg;

	multiple_arg = (MultipleArg *)malloc(sizeof(MultipleArg));
	while(1)
	{
		int th_id = 0;
		clntSock=accept(servSock, (sockaddr*) &clntAddr, &clntAddrSize);
		multiple_arg->a = clntSock;
```

```
            multiple_arg->b = clntAddr;
            if(clntSock==-1)
                ErrorHandling("accept() error");
            th_id = pthread_create(&thread_t, NULL, crauth, (void*)multiple_arg);
            if(th_id != 0){
                ErrorHandling("Tread Create Error");
            }
            pthread_detach(thread_t);
            //free(multiple_arg);
        }
        close(servSock);
        free(multiple_arg);
}

std::string to_format(const int num) {
        std::stringstream ss;
        ss << std::setw(2) << std::setfill('0') << num;
        return ss.str();
}

void *wtrustip(void *clntip){
        std::cout << "wtrustip!!!!" << std::endl;
        char *ip = (char*)clntip;
        //std::cout << ip[0] << ip[1] << ip[2] << std::endl;
        //std::cout << ip << std::endl;
        std::string ip_s(ip);
        //std::cout << ip_s << std::endl;

        std::ofstream fs;
        std::ifstream fs_r;
        std::ofstream fs_w;
        std::string path = "trust_ip.txt";
        fs.open(path, std::ios::out | std::ios::app);
        if(fs.fail())
            throw std::ios_base::failure(strerror(errno));
        fs.exceptions(fs.exceptions() | std::ios::failbit | std::ifstream::badbit);
        fs << ip_s << std::endl;
        fs.close();
        std::cout << ip_s << " - trust_ip APPEND(10sec)!" << std::endl;
        unsigned int microsecond = 1000000;
        usleep(10 * microsecond);//sleeps for 3 second
        std::cout << ip_s << " - trust_ip EXPIRED" << std::endl;
```

```
    fs_r.open(path, std::ios::in);
    fs_w.open(path, std::ios::out);
    std::string line = "";
    while(getline(fs_r, line)){
        line.replace(line.find(ip_s), ip_s.length(), "");
        fs_w << line << std::endl;
    }
    fs_r.close();
    fs_w.close();
    return NULL;
}

void *crauth(void *multiple_arg){

    printf("=================Client Socket Thread Start!=================₩n
");
    u_char rbuffer[136] = {0x00, };
    u_char wbuffer[136] = {0x00, };
    FINETP* rfinetp;
    FINETP* wfinetp;
    MultipleArg *my_multiple_arg = (MultipleArg *)multiple_arg;
    int clntSock_t = my_multiple_arg->a;
    sockaddr_in clntAddr_t = my_multiple_arg->b;
    int th_id_r = 0;
    pthread_t thread_t_r;

    // for log
    std::ofstream fs_logw;
    std::string path = "finet.log";
    fs_logw.open(path, std::ios::out | std::ios::app);
    if(fs_logw.fail())
        throw std::ios_base::failure(strerror(errno));
    fs_logw.exceptions(fs_logw.exceptions() | std::ios::failbit | std::ifstream::badbit);
    std::string log_content = "";

    char cip[16]={0, };
    inet_ntop(AF_INET, &clntAddr_t.sin_addr, cip, INET_ADDRSTRLEN);
    std::string ip_s(cip);
    time_t curr_time;
    struct tm *curr_tm;
    curr_time = time(NULL);
    curr_tm = localtime(&curr_time);
```

```
log_content += ip_s;
log_content += "-";
log_content += to_format(curr_tm->tm_year + 1900);
log_content += "/";
log_content += to_format(curr_tm->tm_mon + 1);
log_content += "/";
log_content += to_format(curr_tm->tm_mday);
log_content += "-";
log_content += to_format(curr_tm->tm_hour);
log_content += ":";
log_content += to_format(curr_tm->tm_min);
log_content += ":";
log_content += to_format(curr_tm->tm_sec);

// recieve request packet
memset(rbuffer, 0x00, sizeof(rbuffer));
memset(wbuffer, 0x00, sizeof(wbuffer));
recv(clntSock_t, rbuffer, sizeof(FINETP), 0);
rfinetp = (FINETP*)rbuffer;
wfinetp = (FINETP*)wbuffer;
printf("0. FINET - Received Packet ₩n");
srand((unsigned)time(NULL));
u_int16_t challenge = (rand() % 65534);
std::cout << rfinetp->header << " " << FINETP_HDR << std::endl;
if(ntohs(rfinetp->header) == FINETP_HDR){
    printf("1. FINET - Valid HEADER ₩n");
    if(ntohs(cal_chksum(rfinetp)) == rfinetp->checksum){
        printf("2. FINET - Valid CHECKSUM ₩n");
        if(rfinetp->id == FLOW_REQ){
            printf("3. FINET - Valid ID (1) Authentication Request ₩n");
            if(ntohs(rfinetp->option) == FINETP_SUCCESS){
                printf("4. FINET - Valid Option - SUCCESS ₩n");
                u_char sender_id[64] = {0x00,};
                memcpy(&sender_id[0], &rfinetp->senId[0], sizeof(sender_id));
                if(chkavailable_database(sender_id) == 1){
                    wfinetp->header = htons(FINETP_HDR);
                    wfinetp->id = FLOW_CREQ;
                    wfinetp->option = htons(challenge);
                    memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                    memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                    wfinetp->checksum = htons(cal_chksum(wfinetp));
                    send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
```

```
                        printf("5. FINET - Send C/R Challenge Packet ₩n");
                    }else{
                        wfinetp->header = htons(FINETP_HDR);
                        wfinetp->id = FLOW_RES;
                        wfinetp->option = htons(FINETP_FAIL);
                        memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                        memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                        wfinetp->checksum = htons(cal_chksum(wfinetp));
                        send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                        printf("5. FINET - Send Response(fail) Packet ₩n");
                        log_content += ERR_FSCHK;
                        fs_logw << log_content;
                        printf("6. LOGGING SUCCESS ₩n");
                        printf("=================Client Socket Thread Finish!=====
============₩n");
                        fs_logw.close();
                        close(clntSock_t);
                        return NULL;
                    }
                }else if(ntohs(rfinetp->option) == FINETP_FAIL){
                    printf("4. FINET - Valid Option - FAIL ₩n");
                    u_char sender_id[64] = {0x00,};
                    memcpy(&sender_id[0], &rfinetp->senId[0], sizeof(sender_id));
                    writefail_database(sender_id);
                    wfinetp->header = htons(FINETP_HDR);
                    wfinetp->id = FLOW_RES;
                    wfinetp->option = htons(FINETP_FAIL);
                    memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                    memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                    wfinetp->checksum = htons(cal_chksum(wfinetp));
                    send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                    printf("5. FINET - Send Response(fail) Packet ₩n");
                    log_content += ERR_FSCHK;
                    fs_logw << log_content;
                    printf("6. LOGGING SUCCESS ₩n");
                    printf("=================Client Socket Thread Finish!========
=========₩n");
                    fs_logw.close();
                    close(clntSock_t);
                    return NULL;
                }else{
                    printf("Invalid Option₩n");
```

```
                    wfinetp->header = htons(FINETP_HDR);
                    wfinetp->id = FLOW_RES;
                    wfinetp->option = htons(FINETP_FAIL);
                    memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                    memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                    wfinetp->checksum = htons(cal_chksum(wfinetp));
                    send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                    printf("4. FINET - Send Response(fail) Packet ₩n");
                    log_content += ERR_OPTION;
                    fs_logw << log_content;
                    printf("5. LOGGING SUCCESS ₩n");
                    printf("=================Client Socket Thread Finish!========
=========₩n");
                    fs_logw.close();
                    close(clntSock_t);
                    return NULL;
                }
            }else{
                printf("Invalid ID₩n");
                wfinetp->header = htons(FINETP_HDR);
                wfinetp->id = FLOW_RES;
                wfinetp->option = htons(FINETP_FAIL);
                memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                wfinetp->checksum = htons(cal_chksum(wfinetp));
                send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                printf("3. FINET - Send Response(fail) Packet ₩n");
                log_content += ERR_FLOW;
                fs_logw << log_content;
                printf("4. LOGGING SUCCESS ₩n");
                printf("=================Client Socket Thread Finish!==========
=======₩n");
                fs_logw.close();
                close(clntSock_t);
                return NULL;
            }
        }else{
            printf("Invalid CHECKSUM₩n");
            wfinetp->header = htons(FINETP_HDR);
            wfinetp->id = FLOW_RES;
            wfinetp->option = htons(FINETP_FAIL);
            memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
```

```
                memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                wfinetp->checksum = htons(cal_chksum(wfinetp));
                send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                printf("2. FINET - Send Response(fail) Packet ₩n");
                log_content += ERR_CHKSUM;
                fs_logw << log_content;
                printf("3. LOGGING SUCCESS ₩n");
                printf("=================Client Socket Thread Finish!=============
====₩n");
                fs_logw.close();
                close(clntSock_t);
                return NULL;
            }
        }else{
            printf("Invalid HEADER₩n");
            printf("=================Client Socket Thread Finish!================
=₩n");
            fs_logw.close();
            close(clntSock_t);
            return NULL;
        }
        printf("=====================================₩n");
        recv(clntSock_t, rbuffer, sizeof(FINETP), 0);
        rfinetp = (FINETP*)rbuffer;
        wfinetp = (FINETP*)wbuffer;
        printf("0. FINET - Received Packet ₩n");
        if(ntohs(rfinetp->header) == FINETP_HDR){
            printf("1. FINET - Valid HEADER ₩n");
            if(cal_chksum(rfinetp) == ntohs(rfinetp->checksum)){
                printf("2. FINET - Valid CHECKSUM ₩n");
                if(rfinetp->id == FLOW_CRES){
                    printf("3. FINET - Valid ID (3) C/R Response ₩n");
                    if(ntohs(rfinetp->option) == FINETP_FIDO || ntohs(rfinetp->option) == FINE
TP_IDPW){
                        printf("4. FINET - Valid Option - SUCCESS ₩n");
                        u_char sender_id[64] = {0x00,};
                        u_char response[64] = {0x00,};
                        memcpy(&sender_id[0], &rfinetp->senId[0], sizeof(sender_id));
                        memcpy(&response[0], &rfinetp->data[0], sizeof(response));
                        int auth_type = 0;
                        if(ntohs(rfinetp->option) == FINETP_FIDO) auth_type = 1;
                        else if(ntohs(rfinetp->option) == FINETP_IDPW) auth_type = 2;
```

```
                    if(cmp_cresponse(sender_id, response, challenge, auth_type) == 1){
                        printf("5. FINET - Valid Response Data₩n");
                        wfinetp->header = htons(FINETP_HDR);
                        wfinetp->id = FLOW_RES;
                        wfinetp->option = htons(FINETP_SUCCESS);
                        memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                        memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                        wfinetp->checksum = htons(cal_chksum(wfinetp));
                        send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);

                        th_id_r = pthread_create(&thread_t_r, NULL, wtrustip, cip);
                        if(th_id_r != 0){
                            ErrorHandling("Tread Create Error");
                        }
                        pthread_detach(thread_t_r);
                        printf("6. FINET - Send Response(success) Packet ₩n");
                        u_char sender_id[64] = {0x00,};
                        memcpy(&sender_id[0], &rfinetp->senId[0], sizeof(sender_id));
                        writesuccess_database(sender_id, auth_type);
                        log_content += ERR_NOERR;
                        fs_logw << log_content;
                        printf("7. LOGGING SUCCESS ₩n");
                    }else{
                        printf("5. FINET - Invalid Response Data₩n");
                        wfinetp->header = htons(FINETP_HDR);
                        wfinetp->id = FLOW_RES;
                        wfinetp->option = htons(FINETP_FAIL);
                        memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                        memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                        wfinetp->checksum = htons(cal_chksum(wfinetp));
                        send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                        printf("6. FINET - Send Response(fail) Packet ₩n");
                        u_char sender_id[64] = {0x00,};
                        memcpy(&sender_id[0], &rfinetp->senId[0], sizeof(sender_id));
                        writefail_database(sender_id, auth_type);
                        log_content += ERR_CRCHK;
                        fs_logw << log_content;
                        printf("7. LOGGING SUCCESS ₩n");
                        printf("=================Client Socket Thread Finish!=====
============₩n");
                        fs_logw.close();
                        close(clntSock_t);
```

```
                        return NULL;
                    }
                }else{
                    printf("Invalid Option₩n");
                    wfinetp->header = htons(FINETP_HDR);
                    wfinetp->id = FLOW_RES;
                    wfinetp->option = htons(FINETP_FAIL);
                    memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                    memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                    wfinetp->checksum = htons(cal_chksum(wfinetp));
                    send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                    printf("4. FINET - Send Response(fail) Packet ₩n");
                    log_content += ERR_OPTION;
                    fs_logw << log_content;
                    printf("5. LOGGING SUCCESS ₩n");
                    printf("=================Client Socket Thread Finish!========
=========₩n");
                    fs_logw.close();
                    close(clntSock_t);
                    return NULL;
                }
            }else{
                printf("Invalid ID₩n");
                wfinetp->header = htons(FINETP_HDR);
                wfinetp->id = FLOW_RES;
                wfinetp->option = htons(FINETP_FAIL);
                memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                wfinetp->checksum = htons(cal_chksum(wfinetp));
                send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                printf("3. FINET - Send Response(fail) Packet ₩n");
                log_content += ERR_FLOW;
                fs_logw << log_content;
                printf("4. LOGGING SUCCESS ₩n");
                printf("=================Client Socket Thread Finish!==========
=======₩n");
                fs_logw.close();
                close(clntSock_t);
                return NULL;
            }
        }else{
            printf("Invalid CHECKSUM₩n");
```

```
                wfinetp->header = htons(FINETP_HDR);
                wfinetp->id = FLOW_RES;
                wfinetp->option = htons(FINETP_FAIL);
                memset(&wfinetp->senId[0], 0x00, sizeof(wfinetp->senId));
                memset(&wfinetp->data[0], 0x00, sizeof(wfinetp->data));
                wfinetp->checksum = htons(cal_chksum(wfinetp));
                send(clntSock_t, (char*)wfinetp, sizeof(FINETP), 0);
                printf("2. FINET - Send Response(fail) Packet ₩n");
                log_content += ERR_CHKSUM;
                fs_logw << log_content;
                printf("3. LOGGING SUCCESS ₩n");
                printf("=================Client Socket Thread Finish!=============
====₩n");
                fs_logw.close();
                close(clntSock_t);
                return NULL;
            }
        }else{
            printf("Invalid HEADER₩n");
            printf("=================Client Socket Thread Finish!================
=₩n");
            fs_logw.close();
            close(clntSock_t);
            return NULL;
        }
        printf("=================Client Socket Thread Finish!=================₩n
");
        fs_logw.close();
        close(clntSock_t);
        return NULL;
}
```

(Rasp Raspbian) netfiler/main.cpp

```
#include "header.h"
#include "finet.h"
#include <vector>
using namespace std;
/*static vector<int> ipc;
static vector<vector<int>> trust_ip;*/
static int cmp = 0;
```

```
void dump(unsigned char* buf, int size) {
	int i;
	for (i = 0; i < size; i++) {
		if (i % 16 == 0)
			printf("₩n");
		printf("%02x ", buf[i]);
	}
}

int police(u_char *data){ //indent remove
	cout << "hey police!!!" << endl;
	libnet_ipv4_hdr *iph = (struct libnet_ipv4_hdr*)(&data[0]);
	if(iph->ip_p != IPPRO_TCP) //TCP CHECK
		return -1;
	fstream fs;
	string tmpbuf = "";
	string path = "trust_ip.txt";
	fs.open(path, ios::in);
	if(fs.fail()){
		cout << "trust_ip.txt file open error!";
		return -1;
	}
	//row.resize(1000000);
	//const char * str = "192.168.1.2";
	int c[4];
	while(!fs.eof()){
		getline(fs, tmpbuf);
		cout << "trust_ip : " << tmpbuf << endl;
		memset(&c[0], 0x00, 4);
		sscanf(tmpbuf.c_str(), "%d.%d.%d.%d", &c[0], &c[1], &c[2], &c[3]);
		//cout << c[0] << c[1] << c[2] << c[3]<<endl;
		if(c[0] == iph->ip_src[0] && c[1] == iph->ip_src[1] && c[2] == iph->ip_src[2] &&
c[3] == iph->ip_src[3]){
			cout << "pass!" << endl;
			fs.close();
			return 0;
		}
	}
	fs.close();
	return -1;
}
```

```
/* returns packet id */
static u_int32_t print_pkt (struct nfq_data *tb)
{
	int id = 0;
	struct nfqnl_msg_packet_hdr *ph;
	struct nfqnl_msg_packet_hw *hwph;
	u_int32_t mark,ifi;
	int ret;
	unsigned char *data;

	ph = nfq_get_msg_packet_hdr(tb);
	if (ph) {
		id = ntohl(ph->packet_id);
		printf("hw_protocol=0x%04x hook=%u id=%u ",
			ntohs(ph->hw_protocol), ph->hook, id);
	}

	hwph = nfq_get_packet_hw(tb);
	if (hwph) {
		int i, hlen = ntohs(hwph->hw_addrlen);

		printf("hw_src_addr=");
		for (i = 0; i < hlen-1; i++)
			printf("%02x:", hwph->hw_addr[i]);
		printf("%02x ", hwph->hw_addr[hlen-1]);
	}

	mark = nfq_get_nfmark(tb);
	if (mark)
		printf("mark=%u ", mark);

	ifi = nfq_get_indev(tb);
	if (ifi)
		printf("indev=%u ", ifi);

	ifi = nfq_get_outdev(tb);
	if (ifi)
		printf("outdev=%u ", ifi);
	ifi = nfq_get_physindev(tb);
	if (ifi)
		printf("physindev=%u ", ifi);
```

```
	ifi = nfq_get_physoutdev(tb);
	if (ifi)
		printf("physoutdev=%u ", ifi);

	ret = nfq_get_payload(tb, &data);
	dump(data, ret);
	cmp = police(data);
	printf("₩n cmp : %d ₩n", cmp);
	//
	if (ret >= 0)
		printf("payload_len=%d ", ret);

	fputc('₩n', stdout);

	return id;
}


static int cb(struct nfq_q_handle *qh, struct nfgenmsg *nfmsg,
		struct nfq_data *nfa, void *data)
{
	UNUSED(data);
	UNUSED(nfmsg);
	u_int32_t id = print_pkt(nfa);

	printf("entering callback₩n");
	int v = cmp == 0 ? NF_ACCEPT : NF_DROP;
	return nfq_set_verdict(qh, id, v, 0, NULL);
}

int main(int argc, char **argv)
{

	struct nfq_handle *h;
	struct nfq_q_handle *qh;
	struct nfnl_handle *nh;
	UNUSED(nh);
	int fd;
	int rv;
	char buf[4096] __attribute__ ((aligned));

	printf("opening library handle₩n");
```

```
    h = nfq_open();
    if (!h) {
        fprintf(stderr, "error during nfq_open()₩n");
        exit(1);
    }

    printf("unbinding existing nf_queue handler for AF_INET (if any)₩n");
    if (nfq_unbind_pf(h, AF_INET) < 0) {
        fprintf(stderr, "error during nfq_unbind_pf()₩n");
        exit(1);
    }

    printf("binding nfnetlink_queue as nf_queue handler for AF_INET₩n");
    if (nfq_bind_pf(h, AF_INET) < 0) {
        fprintf(stderr, "error during nfq_bind_pf()₩n");
        exit(1);
    }

    printf("binding this socket to queue '0'₩n");
    qh = nfq_create_queue(h,  0, &cb, NULL);
    if (!qh) {
        fprintf(stderr, "error during nfq_create_queue()₩n");
        exit(1);
    }

    printf("setting copy_packet mode₩n");
    if (nfq_set_mode(qh, NFQNL_COPY_PACKET, 0xffff) < 0) {
        fprintf(stderr, "can't set packet_copy mode₩n");
        exit(1);
    }

    fd = nfq_fd(h);

    for (;;) {
        if ((rv = recv(fd, buf, sizeof(buf), 0)) >= 0) {
            printf("pkt received₩n");
            nfq_handle_packet(h, buf, rv);

            continue;
        }
        if (rv < 0 && errno == ENOBUFS) {
            printf("losing packets!₩n");
```

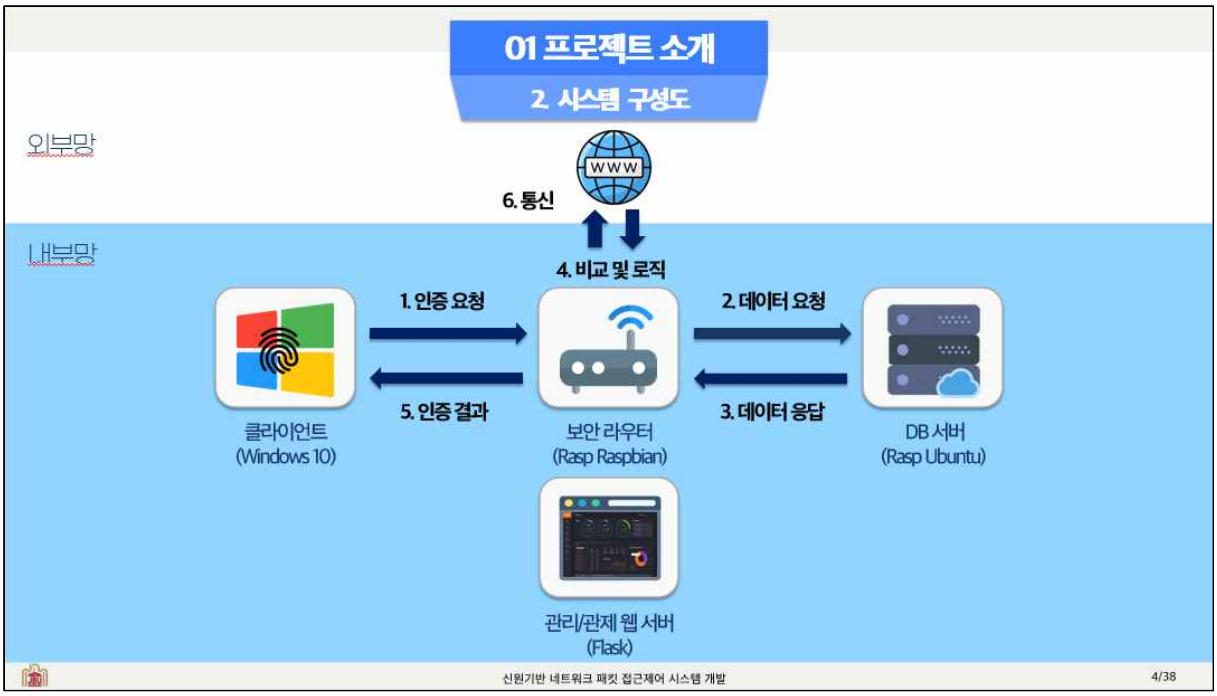
```
                continue;
            }
            perror("recv failed");
            break;
        }

        printf("unbinding from queue 0₩n");
        nfq_destroy_queue(qh);

#ifdef INSANE
        /* normally, applications SHOULD NOT issue this command, since
         * it detaches other programs/sockets from AF_INET, too ! */
        printf("unbinding from AF_INET₩n");
        nfq_unbind_pf(h, AF_INET);
#endif

        printf("closing library handle₩n");
        nfq_close(h);

        exit(0);
}
```

5.2 발표자료

신원기반 네트워크 패킷
접근제어 시스템 개발
김현진, 조재현, 허송이

01
프로젝트 개요
1. 주제
2. 시스템 구성도
02
프로젝트 내용
1. 설계
2. 개발
3. 시연
03
프로젝트 결과
1. 논문
2. 수상

01 프로젝트 개요
1. 주제
신원기반 네트워크 패킷 접근제어 시스템 개발
지문인식을 통한
사용자 인증 기능
인증 실패 로그 확인,
사용자 등록 등
패킷 및 사용자 관리 기능
신원기반 네트워크 패킷 접근제어 시스템 개발
3/38


01 프로젝트 소개
2. 시스템 구성도
외부망
6. 통신
내부망
4. 비교 및 로직
1. 인증 요청
2. 데이터 요청
5. 인증 결과
3. 데이터 응답
클라이언트
(Windows 10)
보안 라우터
(Rasp Raspbian)
DB 서버
(Rasp Ubuntu)
관리/관제 웹 서버
(Flask)
신원기반 네트워크 패킷 접근제어 시스템 개발
4/38

02 프로젝트 내용

1. 설계

1. DB 설계
2. C/R 인증 자체 프로토콜 설계
3. 보안 요소

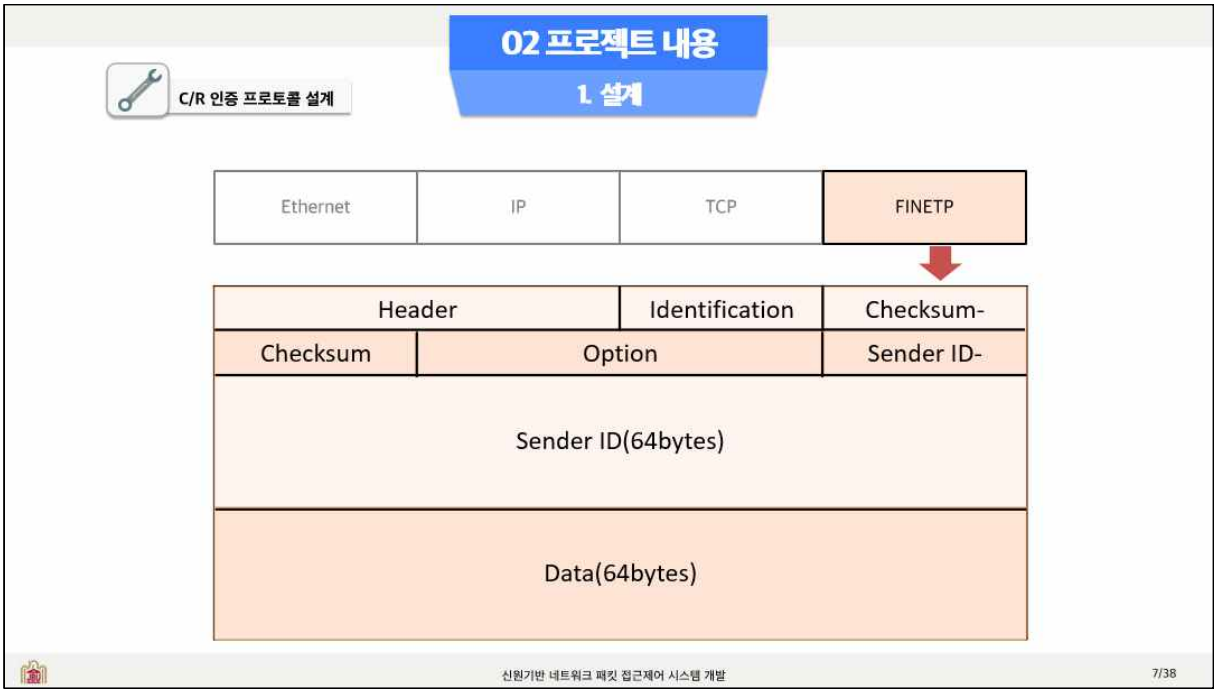
02 프로젝트 내용

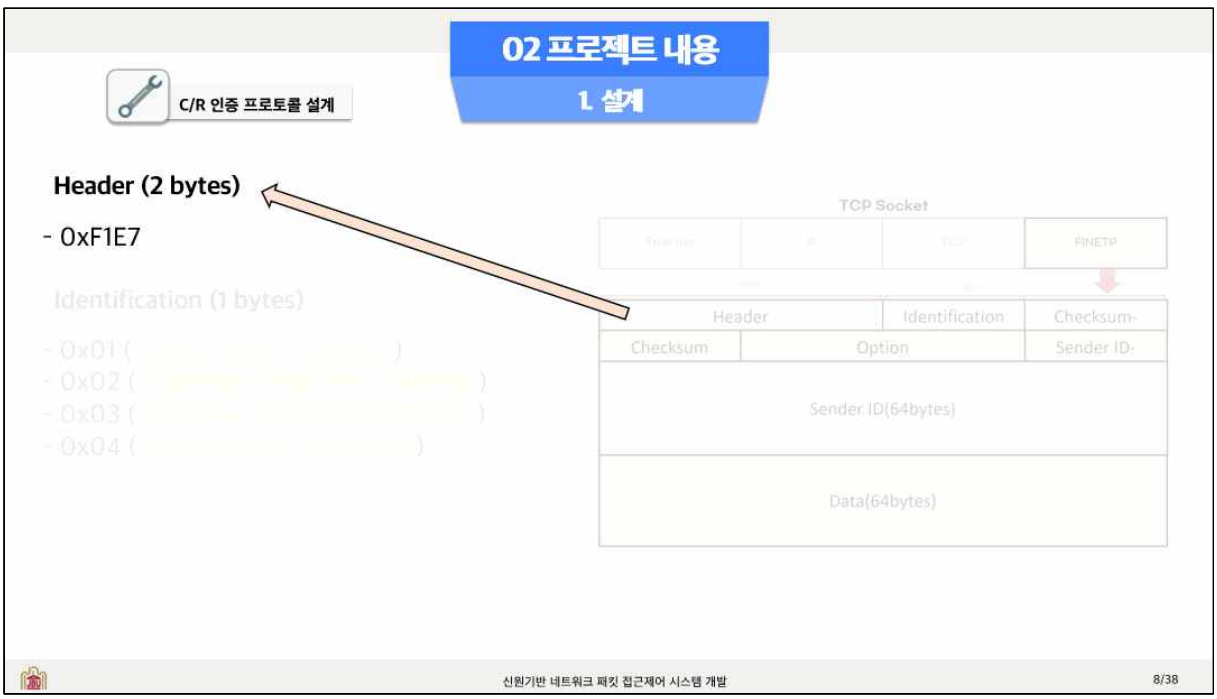
1. 설계

DB 설계

Finetdb - Member_tb

컬럼명	타입	옵션	정보
member_idx	int(10)	PRI, NOT NULL, AUTO_INCREMENT	기본키, 인덱스
member_id	varchar(64)	UNIQ, NOT NULL	Client MAC Addr (SHA-256)
member_guid	varchar(64)	UNIQ, NOT NULL	Client GUID (SHA-256)
member_subid	varchar(20)	UNIQ, NOT NULL	Client ID (SHA-256)
member_subpw	varchar(64)	UNIQ, NOT NULL	Client PW (SHA-256)


02 프로젝트 내용
1. 설계
C/R 인증 프로토콜 설계
Ethernet
IP
TCP
FINETP
Header
Identification
Checksum-
Checksum
Option
Sender ID-
Sender ID(64bytes)
Data(64bytes)
신원기반 네트워크 패킷 접근제어 시스템 개발
7/38


02 프로젝트 내용
1. 설계
C/R 인증 프로토콜 설계
Header (2 bytes)
- 0xF1E7
Identification (1 bytes)
- 0x01 (
- 0x02 (
- 0x03 (
- 0x04 (
TCP Socket
Header
Identification
Checksum-
Checksum
Option
Sender ID-
Sender ID(64bytes)
Data(64bytes)
신원기반 네트워크 패킷 접근제어 시스템 개발
8/38

02 프로젝트 내용

1. 설계

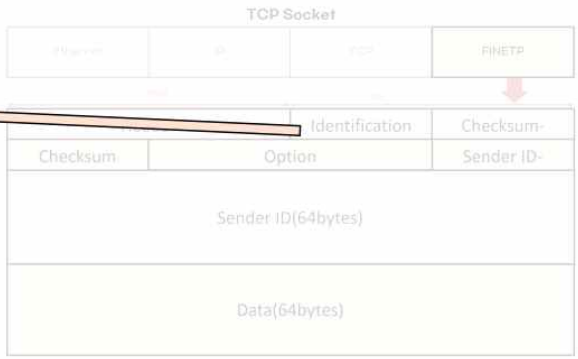
C/R 인증 프로토콜 설계

Header (2 bytes)

- 0xF1E7

Identification (1 bytes)

- 0x01 (Authentication Request)
- 0x02 (Challenge-Response Challenge)
- 0x03 (Challenge-Response Response)
- 0x04 (Authentication Response)



02 프로젝트 내용

1. 설계

C/R 인증 프로토콜 설계

Header (2 bytes)

- 0xF1E7

Identification (1 bytes)

- 0x01 (Authentication Request)
- 0x02 (Challenge-Response Challenge)
- 0x03 (Challenge-Response Response)
- 0x04 (Authentication Response)

Checksum (2 bytes)

- ~(FINETP[0]+…+FINETP[134](Except [3],[4]))&0x0000FFFF

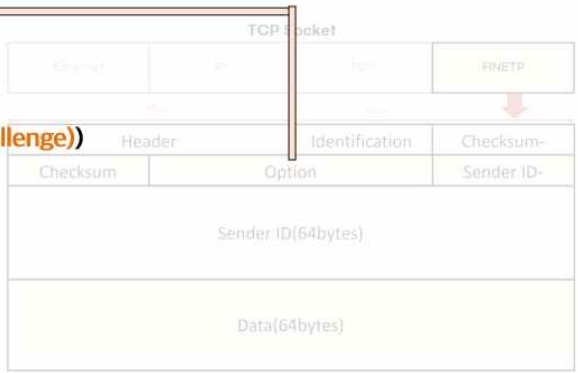
02 프로젝트 내용

1. 설계

C/R 인증 프로토콜 설계

Option (2 bytes)

- 0x0001 (Success, When ID(0x03) FIDO Auth)
- 0x0002 (Fail, When ID(0x03) ID/PW Auth)
- 0xXXXX (When ID(0x02) Random Number(Challenge))



02 프로젝트 내용

1. 설계

C/R 인증 프로토콜 설계

Sender ID (64 bytes)

- 0x0000…0000 (Padding)
- 0xXXXX…XXXX (When ID(0x03)/Option(0x0001) Client MAC(Hashed))
- 0xXXXX…XXXX (When ID(0x03)/Option(0x0002) User ID)

02 프로젝트 내용

1. 설계

C/R 인증 프로토콜 설계

Option (2 bytes)

- 0x0001 (Success, When ID(0x01) FIDO Auth)
- 0x0002 (Fail, When ID(0x01) ID/PW Auth)
- 0xXXXX (When ID(0x02) Random Number(Challenge))

Sender ID (64 bytes)

- 0x0000…0000 (Padding)
- 0xXXXX…XXXX (
- 0xXXXX…XXXX (When ID(0x03)/Option(0x0002) User ID)

Data (64 bytes)

- 0x0000…0000 (Padding)
- 0xXXXX…XXXX (When ID(0x03)/Option(0x0001) GUID + Challenge(Hashed))
- 0xXXXX…XXXX (When ID(0x03)/Option(0x0002) User PW + Challenge(Hashed))

02 프로젝트 내용

1. 설계

C/R 인증 프로토콜 설계

Option (2 bytes)

- 0x0001 (Success,)
- 0x0002 (Fail,)
-)

초기 인증 성공 후 요청

Header	Identification	Checksum	Option	SenderID	Data
0xF1E7	0x01	0x53ab	0x0001	0x000…	0x000…

초기 인증 실패 후 요청

Header	Identification	Checksum	Option	SenderID	Data
0xF1E7	0x01	0x6f7c	0x0002	0x000…	0x000…

02 프로젝트 내용

1. 설계

C/R 인증 프로토콜 설계

Challenge-Response Challenge(2)

Option (2 bytes)

- 0xXXXX (When ID(0x02) Random Number(Challenge))

Header	Identification	Checksum	Option	SenderID	Data
0xF1E7	0x02	0x1bac	0x7b31	0x000···	0x000···

02 프로젝트 내용

1. 설계

C/R 인증 프로토콜 설계

Challenge-Response Response(3)

Option (2 bytes)

- 0x0001 (When ID(0x03) FIDO Auth)
- 0x0002 (When ID(0x03) ID/PW Auth)

Sender ID (64 bytes)

- When ID(0x03)/Option(0x0001) Client MAC(Hashed)
- When ID(0x03)/Option(0x0002) User ID

Data (64 bytes)

- When ID(0x03)/Option(0x0001) GUID + Challenge(Hashed)
- When ID(0x03)/Option(0x0002) User PW + Challenge(Hashed)

FIDO 방식 선택

Header	Identification	Checksum	Option	SenderID	Data
0xF1E7	0x03	0x7dfe	0x0001	0xb22a···	0xfe19···

ID/PW 방식 선택

Header	Identification	Checksum	Option	SenderID	Data
0xF1E7	0x03	0x32a7	0x0002	test12	0x3b51···

02 프로젝트 내용

1. 설계

C/R 인증 프로토콜 설계

Authentication Response(4)

Option (2 bytes)

- 0x0001 (Success,)
- 0x0002 (Fail,)
-)

C/Response 인증 성공

Header	Identification	Checksum	Option	SenderID	Data
0xF1E7	0x04	0x193b	0x0001	0x000···	0x000···

C/Response 인증 실패

Header	Identification	Checksum	Option	SenderID	Data
0xF1E7	0x04	0xc471	0x0002	0x000···	0x000···

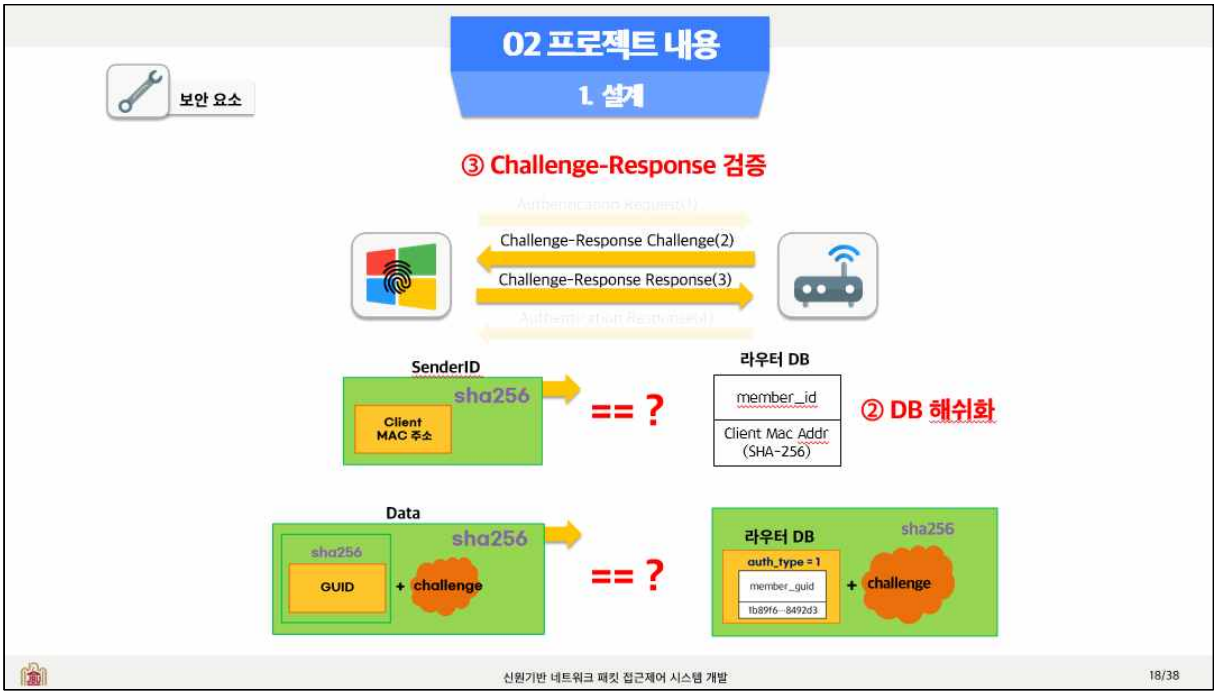
02 프로젝트 내용

1. 설계

보안 요소

① 패킷 검증

Header	Checksum	Identification	Option
0xF1E7	0x193b	0x04	0x0001


02 프로젝트 내용
1. 설계
보안 요소
③ Challenge-Response 검증
Challenge-Response Challenge(2)
Challenge-Response Response(3)
SenderID
sha256
Client MAC 주소
== ?
라우터 DB
member_id
Client Mac Addr (SHA-256)
② DB 해쉬화
Data
sha256
sha256
GUID
+ challenge
== ?
라우터 DB
sha256
auth_type = 1
member_guid
+ challenge
신원기반 네트워크 패킷 접근제어 시스템 개발
18/38

02 프로젝트 내용
1. 설계
보안 요소
④ 로깅
0 (ERR_NOERR) : 성공 (에러 없음)
1 (ERR_FLOW) : 플로우 에러
2 (ERR_CHKSUM) : 체크섬 에러
3 (ERR_OPTION) : 옵션 에러
4 (ERR_SENID) : SenID 에러
5 (ERR_DATA) : Data 에러
6 (ERR_FSCHK) : 초기인증에러
7 (ERR_CRCHK) : C/R 인증 에러
root@kali /var/log cd /var/log/
root@kali /var/log ls -l finet.log
-rw-r--r-- 1 root root 1296 May 18 11:48 finet.log
root@kali /var/log cat finet.log
172.31.13.132-2021/04/14-15:23:27-0
172.31.13.132-2021/04/14-15:23:42-0
172.31.13.132-2021/04/14-15:30:39-0
172.31.13.132-2021/04/14-15:30:50-7
172.31.13.132-2021/04/14-15:33:46-7
172.31.13.132-2021/04/14-15:33:58-0
172.31.13.132-2021/04/14-15:34:07-7
172.31.13.132-2021/04/14-15:34:58-3
172.31.13.132-2021/04/14-15:35:40-1
172.31.13.132-2021/04/14-15:36:17-2
172.31.13.132-2021/04/14-15:37:08-6
172.31.13.132-2021/04/26-16:29:47-2
172.31.13.132-2021/05/10-16:29:47-1
172.31.13.132-2021/05/11-16:29:47-3
172.31.13.132-2021/05/12-16:29:47-0
신원기반 네트워크 패킷 접근제어 시스템 개발
20/38

02 프로젝트 내용

1. 설계

보안 요소

⑤ 관리자 페이지에서의 패킷 관제 및 사용자 등록

패킷 관제	사용자 등록
로그 분석하여 이상한 움직임 감지	관리자 페이지에서 사용자의 MAC, guid 직접 등록

02 프로젝트 내용

2. 개발

02 프로젝트 내용

2. 개발

지문인증, ID/PW

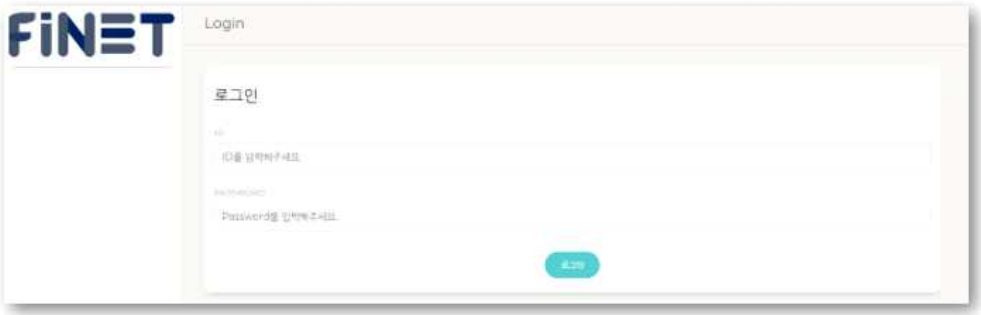
3번 이상 실패시

계정 잠금

02 프로젝트 내용

2. 개발

① 관리자 로그인 페이지



02 프로젝트 내용

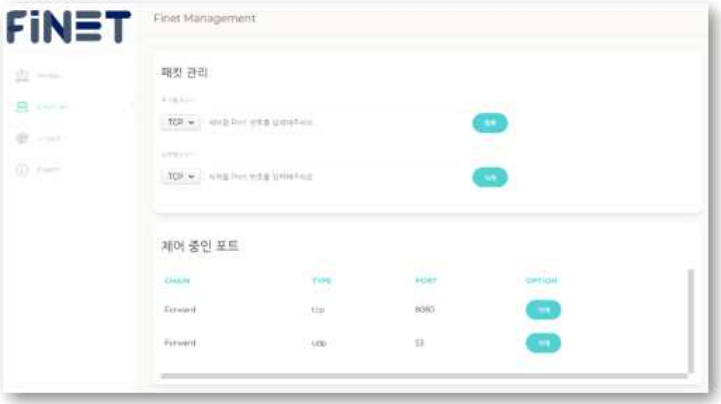
2. 개발

② 요청량 및 실패로그 등 모니터링 페이지

02 프로젝트 내용

2. 개발

③ 포트 관리 페이지



02 프로젝트 내용

2. 개발

④ 기기 등록 페이지

02 프로젝트 내용

2. 개발

⑤ 사용자 관리 페이지

02 프로젝트 내용

2. 개발

⑥ FINET 프로그램 소개 페이지

02 프로젝트 내용

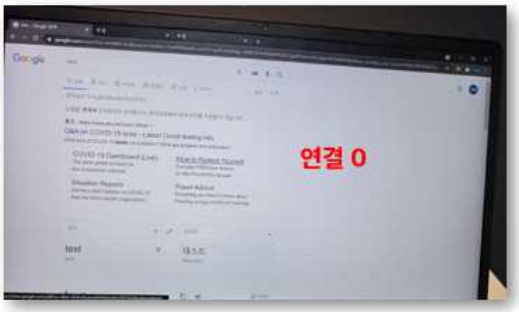
3. 시연

왼쪽: 라즈베리파이, 오른쪽: 사용자 pc 환경

02 프로젝트 내용

3. 시연

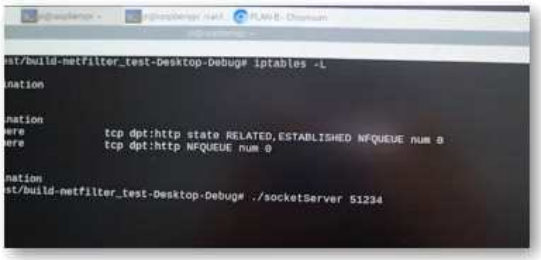
사용자 pc FINET-AP에 연결



02 프로젝트 내용

3. 시연

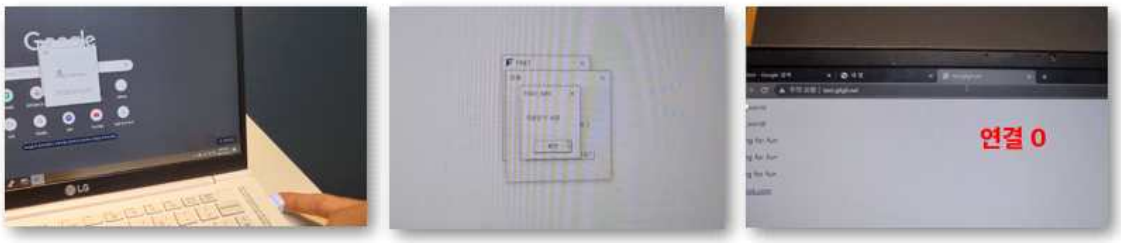
방화벽 설정을 통해 80번 포트 차단



02 프로젝트 내용

3. 시연

FINET 프로그램을 이용하여 사용자 인증 수행



02 프로젝트 내용

3. 시연

등록된 지문이 아니라면 사용자 인증 실패

02 프로젝트 내용

3. 시연

성공 실패 로그 기록

03 결과

1. 논문

논문 작성 및 발표

한국정보보안학회 2019년 하계학술대회 논문집

신원 기반 네트워크 패킷 접근제어 시스템 개발

조재현[1], 김현진[1], 허송이[1]
중부대학교[1]

Development of identity-based network packet access control system

Jaehyeon Cho[1], Hyeonjin Kim[1], Songyi Heo[1]

요약 : 재택근무 환경에서 직원들이 내부망으로 접근하기 위해 반드시 본인인증 등과 같은 다양한 인증을 거쳐야 접속한다는 불편함이 존재한다. 본 논문에선 이러한 문제를 해결하기 위해 지문 인식 기반 사용자 인증을 안전하게 수행할 프로토콜을 정의하고, 보안 라우터를 통해 기업 내/외부망에 접근하는 네트워크 패킷을 제어하며, 전체 웹 서버를 통해 기업 내/외부망에 접근하는 인증 성공, 실패 관련 로그를 모니터링 및 관리할 수 있는 시스템을 개발하고자 한다.

Key Words : Fingerprint authentication, Protocol Design, Packet management, Network Separation

1. 서 론

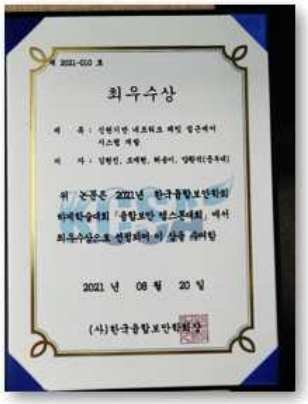
코로나19 바이러스의 장기화로 기업에서는 비대면 업무가 일상화되고 있다. 이에 따라 RDP, SDP, VDI를 이용해 스마트워크 환경을 구축하고 있다[1]. 이들 VDI는 가상 데스크톱 환경을 지원하는 기술로 공공기관에서 많이 도입하여 쓰고 있다. 이 기술은 SSL VPN을 이용하여 사용 전전 인증을 받고 또한 2-Factor 인증을 요구하기도 한다[2]. 이로 인해 원격 근무를 하는 사용자는 기술과 결합하여 보안을 더욱 강화하는 연구가 진행되고 있다.

지문 인증은 또 다른 인증 수단인 공인인증서, ID/PW 입력 등과 비교했을 때 다양한 강점을 지니고 있다. 비밀번호의 경우 사용자들은 기억하기 쉬운 번호를 사용하여 재사용되기도 한다. 따라서 노출되기 쉬운 단점이 있다. 반면 공인인증서의 경우 보안성은 충지만, 개발 및 운영비용이 증가한다는 단점이 있다.

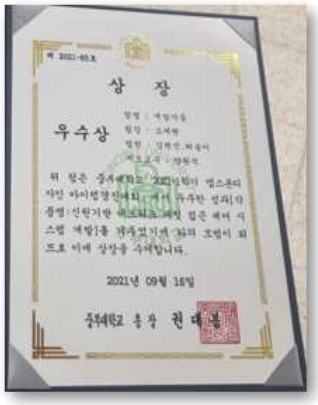
03 결과

2. 수상

융합보안 캡스톤대회 최우수상



캡스톤디자인 아이템경진대회 우수상



감사합니다.