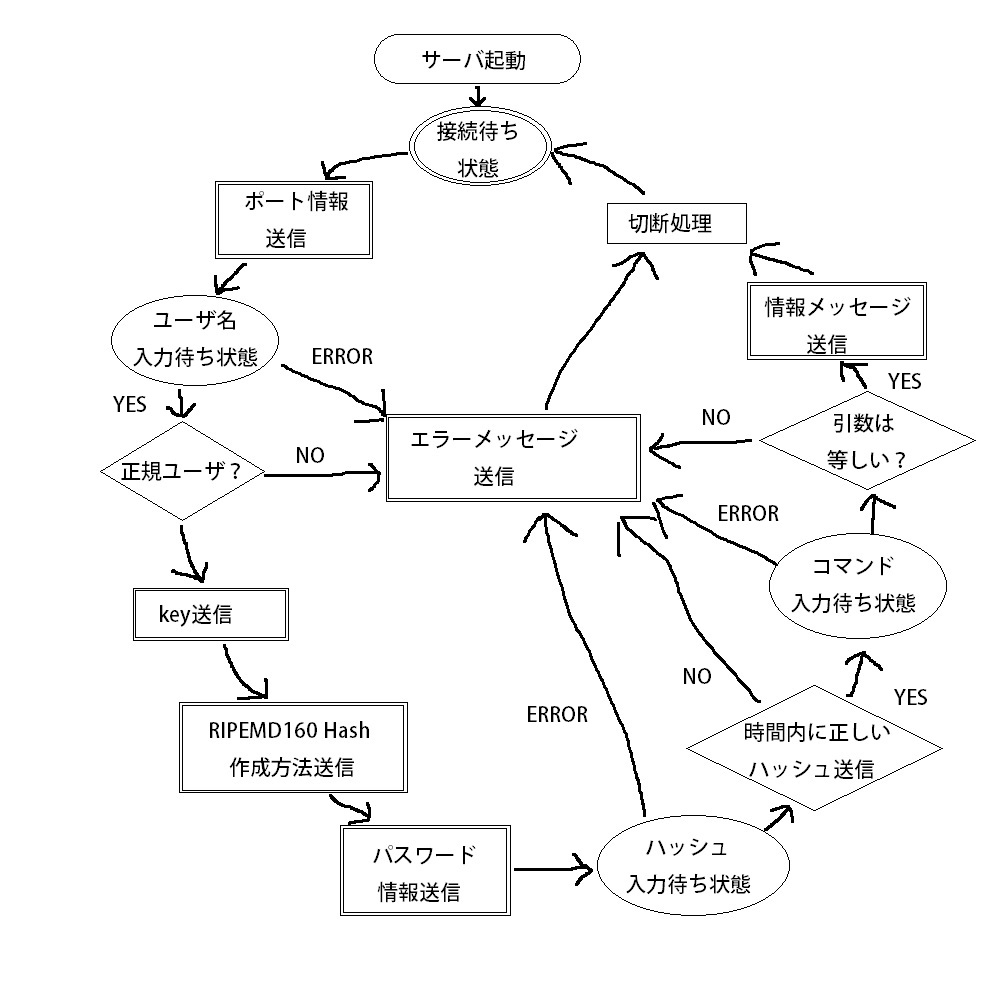
2018年7月11日

調査結果報告書

BP14003 秋山和哉

◆仕組まれていた謎の全貌



◆謎解明の手順

サーバのIPアドレスは172.29.144.28であり、ポート番号が不明であった。

ここの場面ではポート番号をまず探しだす必要がある。int型の変数を1回目の正しいポート番号だと仮定し、まず10000を入れておく。その10000からwhile文でUSRID bp14003を送り、正しいメッセージ（文字列）のINFOEXP1-MIYOSHI-1803\_1\_Ver11.03が返ってきたら、ポート番号特定である。

接続するとサーバから英語でメッセージが届き

（<http://www.geocities.jp/iexp1m/18iy3k8tijp56m>）というURLも届く。

Your key: kxCmeA82p8mw4412

のパスワードについては実行毎にランダムで変わっている。URLに接続するとRIPEMD160の暗号化について説明されたページにつながるようになっている。そしてそこでダウンロードしたファイルと関数を使用し、このパスワードをRIPEMD160に変換する。

変換すると

RIPEMD160 4899efe2925b403dd9dea41d311c24fe945747de

となる。この変換を自動化し3秒以内にサーバへ送信するとクリアである。

サーバからこのメッセージが届く。

Congratulations!!

Your next code: zaKTOvyw

Next info: <http://www.geocities.jp/iexp1m/18ghbn58tsh3tm>

URLに接続すると、新しい接続先のIPアドレスが表示される。

**情報源サーバ： 172.29.144.29**

今回もポート番号は不明であるので、int型の変数に10000をいれ、10000からwhileループで反応をみる、そしてINFOEXP1-MIYOSHI-1803\_2\_Ver11.02が返ってきたら正しいポート番号の特定になる。

そしてサーバの接続をする、

前回、クリアしたコードを

CLEDEPASSE zaKTOvyw

このような形にして送信して認証する。

認証に成功すると以下のメッセージが届く。

INFOEXP1-MIYOSHI-1803\_2\_Ver11.02

Welcome bp14003, you are correctly authenticated.

Your protocol is: 20000

プロトコル（通信規約or通信手順）についてはこれもまた、実行毎に変わっているが、桁数は固定されている。

今回プロトコル番号が[20000]なので、各プロトコルごとの処理を左から順に実行していく。

【プロトコル２】

サーバより整数値x,整数値yが送信される。

受信した整数値x,整数値yでを計算し、計算結果を

REPONSE （結果）

で返す、成功すると

Excellent. Go to next protocol

P0 start!!

のメッセージとともにプロトコル０開始のメッセージも表示される。

【プロトコル０】

サーバより8進数のxが送信される。

受信した8進数xを10進数に変換した数値をサーバに返す

このときもまた

REPONSE （結果）

で返す。そしてOKという反応を送ることで承認され、次のプロトコル（プロトコルC）へ移動する。

そして

FERMER 学籍番号の逆順

の形で送信する。

FERMER 30041pb

Congratulations!!

Your clear code: e7a78be5b1b1e5928ce59389

Clear time: 2C1E6 2EF36

P2 start!!

Go to the following URL: http://www.minet.se.shibaura-it.ac.jp/iexp1\_2018/

Go to the following URL: <http://www.minet.se.shibaura-it.ac.jp/iexp1_2018/>

というメッセージが届く、URLに接続し、クリアコードを入力すると

**情報実験I 2018 ファイナルステージへようこそ**

[FC2無料カウンターFC2無料カウンターFC2無料カウンターFC2無料カウンター](http://counter.fc2.com/)<img src="//counter1.fc2.com/counter\_img.php?id=89470004" />  
現在の閲覧者数: [FC2無料カウンターFC2無料カウンターFC2無料カウンターFC2無料カウンター](http://counter.fc2.com/)<img src="//counter1.fc2.com/counter\_now.php?id=89470004" />

**クリアコードを確認した。よくやったぞ、秋山和哉さん！！**

さて、いよいよファイナルステージだ。  
2018年度情報実験I（三好）の最後を飾るにふさわしい舞台を用意しておいたぞ。  
  
そう、それは・・・

**クイズサーバ Again!!**

君に取っておきのクイズを用意した。  
今回のバトルフィールドは大宮キャンパス5号館だ！！  
3問正解して、最後の（本当に最後の）メッセージを取得してくれ。

クイズサーバの場所は以下のとおりだ。  
  
　　**IPアドレス： 172.29.144.26 (iexp100)**  
　　**ポート番号： 34403**  
  
ユーザ名とパスワードは、前回課題と同じだ。  
秋山和哉さんの健闘を祈ってるよ！

というメッセージがとどき、クイズアゲインである。

上記のIPとポート番号を使い、サーバに接続をするとクイズが始まる。

３つクイズが出題される、まず

QUIZ 【問題5】 <http://www.minet.se.shibaura-it.ac.jp/iexp1_2018/05yl.html>

と表示されURLにアクセスすると

5422-2

と出るので、5422-2に行き、問題文を探す。

問題文は具体的には覚えていないが、

lolのUrgotのリワーク前の名前

を答える問題であった、この答えは

**ハイパーキネティックポジションリバーサー**

である。次の問題へ進む。

QUIZ 【問題3】 <http://www.minet.se.shibaura-it.ac.jp/iexp1_2018/0354262.html>

次の問題、URLへ行くと

5426-2

と出ていたので、5466-2へ行く、問題は

三好研究室にあるスリッパに描かれているキャラが登場する漫画の名前

であり、キャラクターの描いてあるスリッパが２つあり、困ったので三好研にいる方々にスリッパの質問をし、スリッパの位置を教えてもらい、無事答えにたどり着いた。

答えは

ポプテピピック

であった、次の問題へ進むと

QUIZ 【問題8】 <http://www.minet.se.shibaura-it.ac.jp/iexp1_2018/08bb2.html>

であり、掲示板の写真が出てくる。

5号館4階に問題文があった

カリギュラODの帰宅部の人数

という問題である、公式サイトのページでは帰宅部の欄にキャラクターが12人表示されていたが、答えが

11

であった。

クイズに3問正解すると、

If you want hidden message, input "GET MESSAGE".

というメッセージを受信するので、GET MESSAGEと送信すると

Response: 人間にとって大切なのは、この世に何年生きているかということではない。この世でどれだけの価値のあることをするかである。（Ｏ・ヘンリー）

と最後のメッセージを受信することが出来た。

QUITでプログラムを終了する。

調査結果報告は以上である。

（以下プログラムをclient\_ver1、client\_ver2、client\_ver3の３つと、ポートスキャンで使用した２つのプログラムport、port\_ver2を貼り付けます。）

Client\_ver1.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <arpa/inet.h>

#include "rmd160.h"

#define SOCK\_NAME "./socket"

#define PORT\_NUM 11188

#define RMDsize 160

extern byte \*RMD(byte \*);

int main()

{

struct sockaddr\_in saddr;

int soc;

char buf[1024];

int state=0;

char key[200];

byte \*hashcode;

char output[RMDsize/4+1];

int i;

int check =0;

if ( ( soc = socket( AF\_INET, SOCK\_STREAM, 0 ) ) < 0 ) {

perror( "socket" );

exit( 1 );

}

memset( (char \*)&saddr, 0, sizeof( saddr ) );

saddr.sin\_family = AF\_INET;

saddr.sin\_addr.s\_addr=inet\_addr("172.29.144.28");

saddr.sin\_port=htons(PORT\_NUM);

if ( connect( soc, ( struct sockaddr \* )&saddr, ( socklen\_t )sizeof( saddr ) ) < 0 ) {

perror( "connect" );

exit( 1 );

}

fprintf( stderr, "Connection established: socket %d used.\n", soc );

while( 1 ) {

if(state==0)strcpy(buf,"USRID bp14003");

else if(state>=1){

usleep(1000);

fgets(buf,1024,stdin);

//strcpy(buf,key);

}

if ( buf[strlen(buf)-1] == '\n' ) buf[strlen(buf)-1] = '\0';

write( soc, buf, strlen( buf )+1 );

usleep(1000);

fsync( soc );

read( soc, buf, 1024 );

usleep(1000);

fprintf( stdout, "%s\n", buf );

if(state==0){

fsync( soc );

read(soc , buf ,1024); //key

fprintf( stdout, "%s\n", buf );

strcpy(key,&buf[10]);

//printf("%s\n",key);

fsync( soc );

usleep(1000);

read( soc, buf, 1024 ); //パスワードのヒント

usleep(1000);

fprintf( stdout, "%s\n", buf );

usleep(1000);

//read(soc , buf ,1024);

//fprintf( stdout, "%s\n", buf );

//usleep(1000);

//write( soc, key, strlen( key )+1 );

//usleep(1000);

//fsync( soc );

//read(soc , buf ,1024);

//usleep(1000);

//fprintf( stdout, "%s\n", buf );

hashcode=RMD((byte \*)key);

for (i=0; i<RMDsize/8; i++)

sprintf(output+2\*i, "%02x", hashcode[i]);

printf("hashcode: %s\n", output);

strcpy(key,"RIPEMD160 ");

strcat(key,output);

//strcat(key,"miss");

printf("%s\n", key);

usleep(1000);

write(soc,key,strlen(key)+1);

usleep(1000);

fsync( soc );

printf("OK? %s\n",buf);

//check=read(soc , buf ,1024);

printf("%d \n",check);

//if(check==-1)printf("残念!!\n");

printf("OK!!\n");

usleep(1000);

buf[0]='\0';

fprintf( stdout, "%s\n", buf );

}

buf[0]='\0';

if(state==5)break;

state++;

}

close( soc );

return 0;

}s

Client\_ver2

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <arpa/inet.h>

#define SOCK\_NAME "./socket"

int main(int argc,char\* argv[])

{

struct sockaddr\_in saddr;

int soc;

char buf[1024];

int state=0;

int n;

char proto[5];

int i;

if ( ( soc = socket( AF\_INET, SOCK\_STREAM, 0 ) ) < 0 ) {

perror( "socket" );

exit( 1 );

}

memset( (char \*)&saddr, 0, sizeof( saddr ) );

saddr.sin\_family = AF\_INET;

saddr.sin\_addr.s\_addr=inet\_addr("172.29.144.29");

n=atoi(argv[1]);

saddr.sin\_port=htons(n);

//printf("%d\n",n);

if ( connect( soc, ( struct sockaddr \* )&saddr, ( socklen\_t )sizeof( saddr ) ) < 0 ) {

perror( "connect" );

exit( 1 );

}

fprintf( stderr, "Connection established: socket %d used.\n", soc );

fsync( soc );

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

while( 1 ) {

if(state==0){

strcpy(buf,"CLEDEPASSE zaKTOvyw");

}

else if(state>=1)fgets( buf, 1024, stdin );

//else if(state==1)strcpy(buf,"OK");

//fgets( buf, 1024, stdin );

if ( buf[strlen(buf)-1] == '\n' ) buf[strlen(buf)-1] = '\0';

write( soc, buf, strlen( buf )+1 );

fsync( soc );

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

if(state==0){

fsync( soc );

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

strcpy(proto,&buf[18]);

//printf("%s\n",proto);

//fsync( soc );

//read( soc, buf, 1024 );

//fprintf( stdout, "%s\n", buf );

}

strcpy(buf,"OK");

write( soc, buf, strlen( buf )+1 );

for(i=0;i<6;i++){

if(proto[i]=='0'){

printf("P0 start!!\n");

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

sprintf(buf,"REPONSE %d",strtol(buf,NULL,8));

write( soc, buf, strlen( buf )+1 );

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

write(soc,"OK",3);

usleep(1000);

}

if(proto[i]=='1'){

printf("P1 start!!\n");

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

int num=atoi(buf);

num=2\*num\*num\*num-num\*num+7\*num-2;

sprintf(buf,"REPONSE %d",num);

write( soc, buf, strlen( buf )+1 );

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

write(soc,"OK",3);

write( soc, buf, strlen( buf )+1 );

write(soc,"OK",3);

usleep(1000);

}

if(proto[i]=='2'){

printf("P2 start!!\n");

strcpy(buf,"COMMENCER PROTO\_2");

write( soc, buf, strlen( buf )+1 );

read( soc, buf, 1024 );

int x=atoi(buf);

fprintf( stdout, "%s\n", buf );

read( soc, buf, 1024 );

int y=atoi(buf);

fprintf( stdout, "%s\n", buf );

sprintf(buf,"REPONSE %d",2\*x\*x-y\*y);

write( soc, buf, strlen( buf )+1 );

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

write(soc,"OK",3);

}

if(proto[i]=='3'){

printf("P3 start!!\n");

strcpy(buf,"BONSOIR, CA VA BIEN?");

write( soc, buf, strlen( buf )+1 );

usleep(1000);

strcpy(buf,"COMMENCER PROTO\_3");

write( soc, buf, strlen( buf )+1 );

usleep(1000);

fsync( soc );

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

int x=atoi(buf);

read( soc, buf, 1024 );

fprintf( stdout, "%s\n", buf );

int y=atoi(buf);

if(4\*x<7\*y)sprintf(buf,"REPONSE %d",7\*y);

else sprintf(buf,"REPONSE %d",4\*x);

write( soc, buf, strlen( buf )+1 );

}

}

//if(state==8)break;

state++;

}

close( soc );

return 0;

}

Client\_ver3

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define SOCK\_NAME "./socket"

int main()

{

struct sockaddr\_in saddr;

int soc;

char buf[1024], cmd[1024]; // Buffer, Command

if ( ( soc = socket( AF\_INET, SOCK\_STREAM, 0 ) ) < 0 ) {

perror( "socket" );

exit( 1 );

}

memset( (char \*)&saddr, 0, sizeof( saddr ) );

saddr.sin\_family = AF\_INET;

saddr.sin\_addr.s\_addr = inet\_addr("172.29.144.26");

saddr.sin\_port = htons(34403);

// strcpy( saddr.sin\_path, SOCK\_NAME );

if ( connect( soc, ( struct sockaddr \* )&saddr, ( socklen\_t )sizeof( saddr ) ) < 0 ) {

perror( "connect" );

exit( 1 );

}

fprintf( stderr, "Connection established: socket %d used.\n", soc );

fprintf( stderr, "\n◆◆ Program started ◆◆\n\n QUIT: Quit, STAT: Status.\n(Enter into ANSR,after logging in.)\n\n" );

////////// Phase0, 1: LogIn

fprintf( stderr, "[LogIn]\n" );

while(1){

// Phase.1: User

fprintf( stderr, "USER " );

while( fgets( buf, 1024, stdin ) ){

if ( buf[strlen(buf)-1] == '\n' ) buf[strlen(buf)-1] = '\0';

if( strcmp( buf, "QUIT" ) == 0 || strcmp( buf, "STAT" ) == 0 )

// Quit, Status

write( soc, buf, strlen( buf )+1 );

else{

strcpy(cmd, buf); // Buf -> Cmd

sprintf(buf, "USER %s", cmd); // Cmd+ -> Buf

write( soc, buf, strlen( buf )+1 );

}

fsync( soc );

read( soc, cmd, 1024 );

fprintf( stdout, "Response: %s\n", cmd );

if( strcmp(cmd, "OK")==0 ){ // Go on to the next phase -> PASS

printf("Permittion.\n");

break;

}

else if( strcmp(cmd, "NG")==0 || strcmp(cmd, "ERROR")==0 )

printf("Try again. User?-> ");

else if( strcmp(cmd, "GOOD BYE")==0 ){

printf("Good Bye.\n");

exit(1);

}

else

// STAT

printf("User? -> ");

}

// Phase.1: Password

fprintf( stderr, "PASS " );

fgets( buf, 1024, stdin );

if ( buf[strlen(buf)-1] == '\n' ) buf[strlen(buf)-1] = '\0';

if( strcmp( buf, "QUIT" ) == 0 || strcmp( buf, "STAT" ) == 0 )

// Quit, Status

write( soc, buf, strlen( buf )+1 );

else{

strcpy(cmd, buf); // Buf -> Cmd

sprintf(buf, "PASS %s", cmd); // Cmd+ -> Buf

write( soc, buf, strlen( buf )+1 );

}

fsync( soc );

read( soc, cmd, 1024 );

fprintf( stdout, "Response: %s\n", cmd );

if( strcmp(cmd, "OK")==0 ){ // Go on to the next phase -> QUIZ

printf("Permittion. Starting Quiz!...\n\n");

break;

}

else if( strcmp(cmd, "NG")==0 || strcmp(cmd, "ERROR")==0 );

else if( strcmp(cmd, "GOOD BYE")==0 ){

printf("Good Bye.\n");

exit(1);

}

else

// STAT

printf("USER ");

}

////////// Phase.2, 3: QUIZ

int crct=0; // Correct

fprintf( stderr, "[Quiz]\n" );

while(1){

if(crct == 3){

fprintf( stdout, "Completed.\n\n" );

break;

}

// Phase.2: QUIZ

sprintf(buf, "QUIZ %d", crct);

write( soc, buf, strlen( buf )+1 );

fsync( soc );

read( soc, cmd, 1024 );

fprintf( stdout, "Correct: %d.\n", crct );

fprintf( stdout, "QUIZ %s\nANSR ", cmd );

// Phase.3: ANSR

fgets( buf, 1024, stdin );

if ( buf[strlen(buf)-1] == '\n' ) buf[strlen(buf)-1] = '\0';

if( strcmp( buf, "QUIT" ) == 0 || strcmp( buf, "STAT" ) == 0 )

// Quit, Status

write( soc, buf, strlen( buf )+1 );

else{

strcpy(cmd, buf); // Buf -> Cmd

sprintf(buf, "ANSR %s", cmd); // Cmd+ -> Buf

write( soc, buf, strlen( buf )+1 );

}

fsync( soc );

read( soc, cmd, 1024 );

fprintf( stdout, "Response: %s\n", cmd );

if( strcmp(cmd, "OK")==0 ) crct++;

else if( strcmp(cmd, "NG")==0 || strcmp(cmd, "ERROR")==0 );

else if( strcmp(cmd, "GOOD BYE")==0 ){

printf("Good Bye.\n");

exit(1);

}

else printf("Go on next quiz.\n"); // STAT

}

////////// Phase.03: MESSAGE

fprintf( stderr, "[Message]\n" );

fprintf( stderr, "If you want hidden message, input \"GET MESSAGE\".\n" );

while(1){

printf("Command? -> ");

fgets( buf, 1024, stdin );

if ( buf[strlen(buf)-1] == '\n' ) buf[strlen(buf)-1] = '\0';

write( soc, buf, strlen( buf )+1 );

fsync( soc );

read( soc, cmd, 1024 );

if( strcmp(cmd, "GOOD BYE")==0 ){

printf("Good Bye.\n");

exit(1);

}

else if( strcmp(cmd, "NG")==0 || strcmp(cmd, "ERROR")==0 );

else fprintf( stdout, "Response: %s\n", cmd );

}

close( soc );

return 0;

}

port

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define SOCK\_NAME "./socket"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

struct sockaddr\_in saddr;

int soc;

char buf[1024], cmd[1024]; // Buffer, Command

int u=10000; // Port Number

int u2=10000;

int i=0, j=15; // PWD1 Counter

char user[99] = "USERID bp14003"; // USERID

char pwd2[99]; // Password No.2

char pwd2\_send[99]; // Sending Password No.2

// char code[99]; // CODE

char code\_send[99]; // Sending CODE

char phase[99]; // Protocol[0-3]

int x=0, y=0; // x, y

int send=0; // Sending x

struct timeval tv; // Time Out

tv.tv\_sec = 3;

tv.tv\_usec = 0;

memset( ( char\* )&saddr, 0, sizeof( saddr ) );

saddr.sin\_family = AF\_INET;

saddr.sin\_addr.s\_addr = inet\_addr( "172.29.144.28" );

/////////////////////// Phase.01 ///////////////////////

printf( "[Phase.01 ---------> Searching Available Port]\n" );

u=10000;

while(1){

setsockopt( soc, SOL\_SOCKET, SO\_RCVTIMEO, (char\*)&tv, sizeof(tv) );

if( u==65535 ) break;

if ( ( soc = socket( AF\_INET, SOCK\_STREAM, 0 ) ) < 0 ) {

perror( "socket" );

exit(1);

}

saddr.sin\_port = htons(u);

if ( connect( soc, ( struct sockaddr \* )&saddr, ( socklen\_t )sizeof( saddr ) ) < 0 ) {

close( soc );

u++;

}

else{

printf( "Available port Found: No.%d -----\n", u );

fprintf( stderr, "Connection Established: socket %d used.\n", soc );

/////////////////////// Phase.02 ///////////////////////

write( soc, user, 1024 );

fsync( soc );

if( read( soc, buf, 1024 ) < 0 ||

strcmp( buf, "INFOEXP1-MIYOSHI-1803\_1\_Ver11.03") != 0 ){

close( soc );

u++;

}

else if( strcmp(buf, "INFOEXP1-MIYOSHI-1803\_1\_Ver11.03") == 0 ){

read( soc, cmd, 1024 );

printf( "\n" );

}

}

}

return 0;

}

**Port\_ver2**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define SOCK\_NAME "./socket"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

struct sockaddr\_in saddr;

int soc;

char buf[1024], cmd[1024]; // Buffer, Command

int u=10000; // Port Number

int u2=10000;

int i=0, j=15; // PWD1 Counter

char user[99] = "USERID bp14003"; // USERID

char pwd2[99]; // Password No.2

char pwd2\_send[99]; // Sending Password No.2

// char code[99]; // CODE

char code\_send[99]; // Sending CODE

char phase[99]; // Protocol[0-3]

int x=0, y=0; // x, y

int send=0; // Sending x

struct timeval tv; // Time Out

tv.tv\_sec = 3;

tv.tv\_usec = 0;

memset( ( char\* )&saddr, 0, sizeof( saddr ) );

saddr.sin\_family = AF\_INET;

saddr.sin\_addr.s\_addr = inet\_addr( "172.29.144.29" );

/////////////////////// Phase.01 ///////////////////////

printf( "[Phase.01 ---------> Searching Available Port]\n" );

u=10000;

while(1){

setsockopt( soc, SOL\_SOCKET, SO\_RCVTIMEO, (char\*)&tv, sizeof(tv) );

if( u==65535 ) break;

if ( ( soc = socket( AF\_INET, SOCK\_STREAM, 0 ) ) < 0 ) {

perror( "socket" );

exit(1);

}

saddr.sin\_port = htons(u);

if ( connect( soc, ( struct sockaddr \* )&saddr, ( socklen\_t )sizeof( saddr ) ) < 0 ) {

close( soc );

u++;

}

else{

printf( "Available port Found: No.%d -----\n", u );

fprintf( stderr, "Connection Established: socket %d used.\n", soc );

/////////////////////// Phase.02 ///////////////////////

write( soc, user, 1024 );

fsync( soc );

if( read( soc, buf, 1024 ) < 0 ||

strcmp( buf, "INFOEXP1-MIYOSHI-1803\_2\_Ver11.02") != 0 ){

close( soc );

u++;

}

else if( strcmp(buf, "INFOEXP1-MIYOSHI-1803\_2\_Ver11.02") == 0 ){

read( soc, cmd, 1024 );

printf( "\n" );

}

}

}

return 0;

}