שאלה 2

Finding Summary

Threat	***Compromised clients data, exposure of sensitive information***
Affected component	*Client's saved files, client's encrypted data*
Module details	*Client and server software*
Vulnerability class	*Man in the middle, third party impersonation*
Description	*The network protocol is written in such a way that a client can not confirm the identity of other side as the server it's trying to communicate with and vice versa. As such, an attacker can hijack the communication, communicate with the client as the server and with the server as the client, while both sides believe they communicate directly with one another.* *As an example:* *An attacker can receive a client's key exchange request, send to the server the same request but with its own rsa encryption instead of the user given one, decrypt the aes key returned by the server and encrypt it with the user given public key before sending the result back to the user.* *At this point both the client and the server believe that they communicate with one another directly with no issues. Now, a request to back up file by the user can be processed by the attacker and the file decrypted, before encrypting it again and sending it to the server, or even changing it and sending a custom file to be saved on the server on the client's behalf.* *Given the perquisites, an attacker can completely negate the encryption used in the protocol and render any attempt to safeguard sensitive data within the protocol as useless, as such this is a high priority risk*
Result	*The result of a man in the middle attack as described could be unauthorized access to encrypted data sent through the communication, alongside corruption of files sent to be backed up on the server*
Perquisites	*To perform an impactful man in the middle attack on this communication method, an attacker has to know the flow of communication based on the protocol, insert itself as the server address for the client program and know the servers address.*
Proposed remediation	*To lower the potential of a man in the middle attack, several steps can be taken:* 1. *Client software could be sent out with an included rsa public key, fitting to an rsa private key created by the server, to allow communication between both sides to be encrypted from the start (albeit using a slower encryption model), before reaching the key exchange phase of the protocol, where both sides decide on an aes key and can switch to it for encryption for the rest of the communication.*

	2. *Use the encryption in each phase of the protocol to encrypt the protocol fields alongside the data, to make it harder for attackers to read the data sent and find the correct spot to hijack the communication.* 3. *Use credentials to identify the server as a valid one, and make it harder for attackers to impersonate the server in the eyes of the client.*
Threat	***Client impersonation***
Affected component	*Server's response modules, server's database*
Module details	*Server's registration module*
Vulnerability class	*Data sniffing*
Description	*the protocol as it stands is vulnerable to attackers sniffing the servers incoming and outgoing communication, specifically targeting the registration process.* *An attacker listening in on a registration request coming from a client and the fitting response sent by the server (identifiable by the code specified in the header, can impersonate the client after registration with ease, using the client's name sent in the request and the client's id sent in the response.* *Using said information, the attacker can completely interact with the server on behalf of the unsuspecting client.*
Result	*The result of such an attack could result in an attacker uploading files to the server on behalf of the client without either side realizing it. Such files could be legally incriminating or maliceful in other ways.*
Perquisites	*To perform such an attack, an attacker would need to sniff the incoming and outgoing communication of the server or a specific targeted client, catching the interaction surrounding the registration process.*
Proposed remediation	*In order to prevent such an attack from taking place, the protocol could be changed to start the communication between a potential client and the server with a key exchange (the client sending a public rsa key, alongside the client's name in the registration request for example – and the server responding with the client's id after registration encrypted using said pubic key, for the client to decrypt and save)* *After which, further communication should continue to send sensitive fields of the protocol, such as the client's id or a file's name encrypted, using an agreed upon method (for example, the rsa or aes methods already in use by the protocol).* *Thus preventing eavesdroppers from gathering enough information to send requests on behalf of the client.*
Threat	*Server's resources flooding*

Affected component	*The server and the machine running it*
Module details	*Server program*
Vulnerability class	*Denial of service*
Description	*The protocol as it stands does not impose restrictions on registration requests or send file requests, as such – a malicious user could flood the server with registration requests using different names to occupy a fairly large portion of the database.* *Another potential attack would be flooding the server with requests to back up large files, to restrict either the server's available memory, storage or both.*
Result	*An attack of this nature, might even be caused by natural communication with multiple clients without malicious intent, but could result in the server's resources being restricted or even denied for other users, or even a complete crash of the server's program and system running it.*
Perquisites	*Such an exploit requires only a client following the protocol as is, and thus should be addressed properly.*
Proposed remediation	*To lower the potential of DoS attacks exploiting the protocol:* 1. *The server could be changed to support a limited amount of registration requests sent from the same IP.* 2. *The server could impose a restriction of the total size of files saved by any given client.* 3. *The server could run a background process to remove registered clients suspected of malicious activity, or that are unactive for a set amount of time and delete from storage the files corresponding to those clients, allowing the space to be used by legitimate clients.*
Threat	***Brute force access to sensitive data***
Affected component	*Client's send file request, key exchange module on both sides*
Module details	*Aes and rsa encryptions*
Vulnerability class	*Brute force*
Description	*Although the protocol facilitates file transfers after encryption, the methods used by both ends of the communication could still be broken with enough computing power or circumvented by a third party.*
Result	*The result of such an attack would be exposure of sensitive client's data, sent to the server*
Perquisites	*To perform such an attack, an attacker would be required to have access to the data communicated between the client and the server.*

Proposed remediation	*To lower the potential of such a brute force attack:* *1. The encryption on both sides could be bolstered to use stronger methods or more bits per key, at a minimal additional cost to performance using modern computing power.* *2. The aes encryption used by the protocol could be changed to use a non static iv. The randomized iv value could be sent, encrypted, alongside the aes key during the response from the server to a key exchange request (response code 2102), thus making it harder for a third party to successfully guess the correct key combination.*
Threat	***Exposure of sensitive clients files by an unrelated intrusion of the server's computer***
Affected component	*Client's saved files*
Module details	*Server's send file request handling*
Vulnerability class	*Side channel attack*
Description	*According to the protocol, the server receives encrypted files from clients and then saves them on its system after decrypting them to their original form. An attacker attacking the machine running the server, or even a user snooping around in the machine's files could find their way to the location where files sent by clients are saved and gain unrestricted access to every single file and the data they contain.*
Result	*An unauthorized third party could have complete access to sensitive and private data sent by clients to the server.*
Perquisites	*An attacker would be required to have access to the machine running the server, either physically or remotely.*
Proposed remediation	*to prevent unauthorized access to files sent by clients on the servers machine, the protocol could be changed to facilitate saving client files with their encryption, after decrypting them temporarily to validate the crc value against the client.* *And to go a step even further, if the server is not intended to take control of client files and run them on its side – the aes key used to encrypt and decrypt the file could be deleted from the file's entry in the database after the file has been validated, to prevent an attacker from locating the correct key in the database file alongside the saved file on the system to decrypt the file regardless of it being saved encrypted – while still allowing the legitimate client to decrypt the file and regain the original file if the client saves the aes key sent by the server and the protocol allows files being sent back to clients.*