**Maman 15 Research Document – The Dangers of MITM Attacks**

**Written by: Ilai Azaria**

**The protocol**

When coming to write a server-client application, we must have for ourselves a protocol that will enable good, efficient, and secure communication between the two. In maman 15 we are introduced to one.

The protocol enables communication between a (thoretically) unlimited number of clients and a server. Clients can't communicate with each other.

- The protocol is binary and implemented over TCP.

- All numeric fields must have values greater than zero (**unsigned**) and represented as **little endian.**

- This protocol supports requests to the server and responses to the client. Requests or responses can contain a "message".

Each message has a header and a payload. In the header we can identify the message code, and what it intends to do. By exchanging messages the client and server allow a thorough transfer of a file, which the client chooses, to the server.

The following describes the order of events in the protocol:תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

התיאור נוצר באופן אוטומטי

**Vulnerabilities in the protocol**

The protocol has various significant vulnerablities that affect its security. In this document I will describe 5, possible exploits of those vulnerabilities and possible ways to effectively counter and prevent them.

**1.Lack of authentication of the server**

Perhaps the first vulnerability that we can identify in the protocol is the lack of mutual authentication. I split this vulnerability into two – client and server.

Looking at the protocol we can see that in no stage there is an attempt of the client to authenticate that it is communicating with the server. In all of the replies' documentation the client simply accepts the message so long it had come from the socket he is connected to.

Theoretically, this could be exploited in many ways, the main one is through a **man in the middle attack**: "A man-in-the-middle (MITM) attack is a cyberattack where an adversary secretly intercepts and potentially alters communications between two parties without their knowledge. This attacker can eavesdrop on the communication, manipulate data, and even impersonate one or both parties, effectively inserting themselves as an unauthorized intermediary in the communication channel, compromising the confidentiality and integrity of the data exchanged."

In our case such an attack could be disastrous. The attacker could impersonate as the server and get the client to register to it and then get the client's files and keys, or even worse, get it to login to the attacker's server and then with the client's login credentials it could login to the real server and get all the client's information.

An obvious fix for this problem would be to change the protocol so that a client could validate it is talking to the real server.

**2.Lack of authentication of the client**

In addition to a lack of authentication of the server, there is a lack of authentication of the client in the server. This can be seen as in all the requests' documentation it is not written how should the server make sure it is communicating with the real client, and we can infer that the protocol expects the server to accept any request as if it is coming from the real client. Such an assumption is extremely dangerous.

Here again attacker could exploit this with a sophisticated "man in the middle" attack. For example, the attacker impersonating a client could wait for the real client to register, then imporsonate the client when the server sends back the uuid of the client. After it gets the server created uuid, the attacker could continue registering from there and send private and public RSA keys to the server. From this stage on – the attacker basically has all the credentials it needs to login as this client and do whatever it wishes with the client's information later. Even if the client will eventually realize there is an attack and stop registering, the attacker now seems to the server as a real one with the name of a real one, and in the real world this could have grave impacts on the system.

Again, An obvious fix for this problem would be to change the protocol so that the server could validate it is talking to the client the session was originally initiated with.

**3.No Certificate authority (CA)**

The two weaknesses which I described above could both be Theoretically solved by using a certificate authority for mutual authentication. But I decided to include it as a different weakness category, as the implications of lacking a CA vary further:

"A Certificate Authority (CA) is a trusted entity responsible for issuing digital certificates that vouch for the authenticity and identity of entities, such as websites or individuals, in online communications. In the context of MITM attacks, CAs play a crucial role in ensuring secure connections by verifying the legitimacy of public keys, helping to prevent attackers from intercepting or altering data by impersonating trusted parties. MITM attacks can occur when trust in a compromised or rogue CA is exploited, allowing attackers to intercept and manipulate encrypted communications without detection."

So even though CA could help solve the two weaknesses above, lacking it implicates the security of key echange between the client and the server. In this protocol, the client itself creates two RSA keys, one private and one public, and the server creates an aes key. Such implementation is susceptible to different attacks that impersonate the client or server and via this create themselves keys for communication.

Creating a mutual authentication system would solve the first two weaknesses, but not neccesarily this one. In comparison, implementing a CA in the protocol with improved error handling would likely solve all the three! It would allow for a secure creation of keys and far more secure exchange of information, and vastly improve the protocol overall.

**4.easily compromised sending of critical information**

In the protocol, critical information such as: Keys (RSA public key), crc result, and more is sent as plain text. By doing so the protocol increases the ease of using this information if an attacker manages to sniff it through the network. Such attacks don't even require the attacker to impersonate the client or the server, but only to listen to their communication at the right place and the right time. This information could potentially be used to get the client's other keys, various files, and more.

The best way to prevent this is to simply encrypt that information before it is sent and decrypt at the receiving side.

**5.Lack of protection measures against DOS attacks**

The last substantial weakness in the protocol that I have found is the lack of protection against DOS attacks: "DOS attacks can flood the server with a high volume of requests or otherwise overwhelm the system, causing it to become unresponsive or unavailable to legitimate users."

In the protocol there are a few aspects that make it vulnerable to DOS attacks: Firstly, it lacks rate limiting. The protocol does not mention any rate-limiting or access control mechanisms for clients, which means that an attacker could potentially flood the server with requests, consuming resources and causing a DoS.

Secondly, there is no session management: There are no clear guidelines for session management or timeouts, so an attacker could potentially establish and maintain multiple connections from the same ip simultaneously, further straining server resources easily.

And lastly, there is insufficient Error Handling: Incomplete or ambiguous error handling mechanisms in this protocol can make it difficult to detect and respond to abnormal behavior caused by DoS attacks.

We could improve the protocol regarding DOS attacks by adding rate limiting, improved session management and improved error handling.

**Possible exploits in the protocol**

In this section I will summarize possible exploits in the protocol using exploit tables.

**Client impersonation**

Threat: Client impersonation

Affected component: Client management component

Module details: server.py

Vulnerability class: MITM attack

Description: The server has no way of verifing that it is working with the real client, and this makes a MITM attack much easier.

Result: An attacker can impersonate as the client communicating with the server using a MITM attack and steal his credentials or do unwanted actions as the client.

Prerequisties: The attacker has access to the server-client session.

Business impact: A malicious third party could access clients' personal information or make actions in their name, which is a violation of law.

Proposed Remedation: Change the protocol so that the server authenticates the client it communicates with.

Risk- Damage potential: 8

Reproducability: 6

Exploitability: 4

Affected users: 9

Discoverability: 5

Overall: 6.8

**Server impersonation**

Threat: Server impersonation

Affected component: Client management component

Module details: Client.cpp and Client.h

Vulnerability class: MITM attack

Description: The client has no way of verifing that it is working with the real server, and this makes a MITM attack much easier.

Result: An attacker can impersonate as the server communicating with the client using a MITM attack and steal his credentials or do unwanted actions as the client later.

Prerequisties: The attacker has access to the server-client session.

Business impact: A malicious third party could access clients' personal information or make actions in their name, which is a violation of law.

Proposed Remedation: Change the protocol so that the client authenticates the server it communicates with.

Risk- Damage potential: 9

Reproducability: 8

Exploitability: 4

Affected users: 7

Discoverability: 7

Overall: 7

**Exchanging illegitimate keys**

Threat: Exchanging illegitimate keys

Affected component: Client management component

Module details: Client.cpp, Client.h and server.py and database.py

Vulnerability class: MITM attack

Description: The server and client create keys themselves and send them to each other directly, which increases the risk and effectiveness of a MITM attack to get those keys.

Result: An attacker can create keys and send them to the client or server, as they are not verified. Using such MITM attack he can later steal a client's credentials or files.

Prerequisties: The attacker has access to the server-client session.

Business impact: A malicious third party could access clients' personal information, which is a violation of law.

Proposed Remedation: Create a CA to distribute the keys to the client and server, so that they can be verified.

Risk- Damage potential: 7

Reproducability: 8

Exploitability: 3

Affected users: 6

Discoverability: 7

Overall: 6.2

**Sniffing crucial information**

Threat: Sniffing crucial information

Affected component: Client management component

Module details: server.py, database.py, Client.cpp and Client.h

Vulnerability class: Sniffing

Description: Some crucial information is sent as plain text between the client and the server.

Result: An attacker that manages to listen to the client-server session can quite easily identify keys and crucial information sent on the network as it is sometimes sent in plain text, unencrypted.

Prerequisties: The attacker has access to the server-client session.

Business impact: A malicious third party could access clients' personal information, which is a violation of law.

Proposed Remedation: Send all crucial information encrypted.

Risk- Damage potential: 8

Reproducability: 7

Exploitability: 5

Affected users: 6

Discoverability: 7

Overall: 6.6

**Dos attack**

Threat: Dos attack

Affected component: Server flow component

Module details: server.py

Vulnerability class: Denial of service

Description: There is limited action taken in the protocol to prevent Dos attacks.

Result: An attacker could quite easily disable the server from working, sending it a big amount of requests.

Prerequisties: The attacker has limited knowledge in IT and a strong computer

Business impact: The server could be disabled at virutally any time, and clients would be forbidden from using the service, decreasing client interest in it.

Proposed Remedation: Adding rate limiting, session management mechanisms and improved error handling would significantly decrease the threat of a Dos attack.

Risk- Damage potential: 5

Reproducability: 9

Exploitability: 9

Affected users: 9

Discoverability: 2

Overall: 6.8

**Summary**

As we can see, the protocol we have been intruced to in Maman 15 has numerous flaws, and in the real world it probably would not be accepted nowadays. It is especially vulnerable to MITM attacks, and it may be a great example to teach how such attacks could work in the real world.

Despite all of this, it is important to clarify that the basis of this protocol is quite good. it implements various security measures that are popular nowadays such as symmetric and a-symmetric encryption, unique id for clients etc. Therefore, with some improvements it can become a viable protocol for file transferring server-client applications.