|  |  |
| --- | --- |
| Threat | User impersonation |
| Affected Component | Login component for existing user |
| Module details | Protocol for Maman 15, page 4 |
| Vulnerability Class | Identification bypass/flaw |
| Description | The identification for an existing user is based on a file on the user’s device and is therefore very deficient. The user can obtain all the ID’s and names of users, using command 120, and then can change his UUID on file to another users ID, thereby posing as a different user. |
| Result | A user can send their symmetric key to another user, when really the impersonator will gain access to it, by changing his ID, and then when the user messages his intended contact, the message will really be received and decrypted by the impersonator.  A user can be denied access to his messages, if someone impersonates him and constantly requests to pull messages (140). The user himself will never see the messages, because the server will delete them after sending to the impersonator. |
| Prerequisites | The attacker is signed up to the messaging service.  The attacker has access to the me.info file on his device |
| Business Impact | A malicious user can access the data and messages of a user, as well as impersonate him. This violates laws of privacy and ethics. |
| Proposed Remediation | The client shouldn’t have access to other user’s IDs. All client-side requests should be done with the usernames, and the server can do the mapping to the ID’s. That way the ID, which is the real identification of clients, will remain hidden from other clients.  The login design for an existing user should be more secure, for example requiring a user to login with his username and a password, and not just reading from a file. That way, provided the passwords are secure, he won’t be able to impersonate other users. |
| Risk | Damage potential- 9  Reproducibility- 5  Exploitability- 6  Affected users- 7  Discoverability- 7  Overall- 6.8 |

|  |  |
| --- | --- |
| Threat | Distributed denial of service (DDOS) |
| Affected Component | Service’s server |
| Module details | General vulnerability |
| Vulnerability Class | Denial of service |
| Description | User creates numerous accounts and repeatedly sends multiple requests and large files to the server. |
| Result | The server becomes flooded and overwhelmed with requests and data and is unable to handle real clients requests (or in reasonable time). Alternatively, the server might crash. |
| Prerequisites | The user has access to a large network of devices to log in from and send requests. (only one login per device/file ) |
| Business Impact | The service can be shut down for various periods of time, causing customer confusion and frustration |
| Proposed Remediation | Use content delivery networks, or multiple servers.  Use rate limiting to limit number of requests  Use selectors to limit unique client threads |
| Risk | Damage potential- 6  Reproducability- 3  Exploitability- 5  Affected users- 9  Discoverability- 7  Overall- 6 |

|  |  |
| --- | --- |
| Threat | Man in the middle attack (MITM) |
| Affected Component | Message sending component |
| Module details | Protocol for Maman 15, pages 5-7 |
| Vulnerability Class | Insecure security design |
| Description | Although messages and keys are encrypted using RSA/AES, a user could intercept a different users messages using tools like Wireshark etc., and establish connections while pretending to be a different user, or even pretending to be the server. Then the encryption would be pointless, because the malicious user would have the symmetric key/private key (his own). |
| Result | User’s messages are not secure and users can be impersonated. |
| Prerequisites | Malicious user is signed up to the service.  Malicious user can intercept data and manipulate it, which requires certain expertise. |
| Business Impact | A malicious user can access the data and messages of a user, as well as impersonate him. This violates laws of privacy and ethics. |
| Proposed Remediation | Use a certificate system such as SSL or TLS , and get a certificate from a CA. That will mitigate the MITM exploit. |
| Risk | Damage potential- 8  Reproducability- 3  Exploitability- 4  Affected users- 5  Discoverability- 5  Overall- 4 |

|  |  |
| --- | --- |
| Threat | Vulnerability of encrypted data |
| Affected Component | Encryption of messages sent by users |
| Module details | Maman 15 protocol, page 11 |
| Vulnerability Class | Cryptographic failures |
| Description | The RSA encryption is only 1024 bits, which is considered weak by modern standards.  The AES encryption is weak, because the IV bits are always 0, which makes it less effective  Both encryptions have a problem of lacking backward secrecy, namely that the keys are not changed, which increases the likelihood of them being discovered, and of past messages being compromised. |
| Result | User’s messages and keys can be compromised and decrypted with relative ease. |
| Prerequisites | Attacker has significant computing resources at his disposal to break the encryption.  Attacker can listen in on communications and read the messages (e.g. Wireshark) |
| Business Impact | A malicious user can access the data and messages of a user. This violates laws of privacy. |
| Proposed Remediation | For AES- Use a random IV for each encryption. The protocol should also use a mode like AES-GCM instead of AES-CBC.  For RSA- The key should be at least 2048 bits. |
| Risk | Damage potential- 6  Reproducability- 3  Exploitability- 3  Affected users- 5  Discoverability- 3  Overall- 4 |

|  |  |
| --- | --- |
| Threat | Messages Overflow |
| Affected Component | Protocol for sending messages to another user |
| Module details | Maman 15 protocol, page 8 |
| Vulnerability Class | Buffer overflow |
| Description | The length of the ‘content size’ field (in the message sending protocol) is 4 bytes, which is a maximum of approximately 4GB (2^32 bits), but there is no size limit on a message or a file. This means that if the user sends a file or message larger than 4GB, the content size field will be only 4GB, but the content sent will be more than 4GB, which results in a buffer overflow in the protocol. |
| Result | The user can send a file 4GB large and append malicious data/code at the end of the file. The server will only read the 4GB, and the trailing leftover data will be left over until the client sends his next request. That remaining data can be another user’s ID, and a symmetric key, and a malicious user can exploit this overflow to pose as another user and send messages maliciously. |
| Prerequisites | The user has access to a file 4GB large.  The user has gained access to other clients ID’s (request 120) |
| Business Impact | A malicious user can deviate from protocol and can utilize the overflow to send a message to another user, posing as someone else. This violates ethical considerations |
| Proposed Remediation | The protocol should limit the maximum size of a message or a file.  Alternatively, the protocol can add a field ‘content lengths length’ which will tell the server how many bytes is the ‘content length’ itself. It can be 4 bytes, and if needed-5, or even more. (5 bytes covers up to hundreds of GB, so that should be enough for almost all files) |
| Risk | Damage potential- 4  Reproducability- 3  Exploitability- 4  Affected users- 7  Discoverability- 6  Overall- 4.8 |

|  |  |
| --- | --- |
| Threat | Pull messages overflow |
| Affected Component | Protocol for pulling messages request |
| Module details | Protocol for Maman 15, page 10 |
| Vulnerability Class | Buffer overflow |
| Description | The user requests to pull messages and the server sends them all the messages. The messages can accumulate to more than 4GB (for example, 5 files of 1GB each). But the field of payload size is only 4 bytes, which can store up to 2^32 bits, which is 4GB. So, if there is more than 4GB of messages to be pulled, the payload size will only be 4GB, when payload is actually larger, and a buffer overflow occurs. |
| Result | The user requests to pull his messages, and reads exactly 4GB of data, as per the ‘payload size’ field. But the server really sent more than 4GB of data. So, the next time the client waits for a response from the server, the response will be corrupted and won’t be able to be processed by the client (instead of opcode and ID, or whatever the response is, it will contain the remaining messages.) |
| Prerequisites | The user received over 4GB of data, in files or messages |
| Business Impact | Users will receive corrupted and unreadable data, and will encounter error messages from the server, resulting in user frustration and lack of service. |
| Proposed Remediation | Insert a condition in the server pull function, whereas the server sends only up to 4GB of messages, and if the client has more than 4GB, the server only sends the first 4GB and also will send an error message or warning to the user.  Like the previous vuln., the protocol can provide a field of ‘payload size length’, which will inform the user how many bytes the ‘payload length’ will be. That way, the client can receive larger amounts of data, safely. |

|  |  |
| --- | --- |
| Risk | Damage potential- 3  Reproducability- 3  Exploitability- 2  Affected users- 4  Discoverability- 4  Overall- 3.2 |

|  |  |
| --- | --- |
| Threat | Unencrypted messages are lost |
| Affected Component | Protocol for receiving messages and pulling them |
| Module details | Maman 15 protocol, pages 3 and 5 |
| Vulnerability Class | Loss of information |
| Description | The server sends messages to the user that requested a pull, and once sending them, deletes them from server. But if a client couldn’t decrypt his messages, he has no way of restoring them from the server, and all the messages are lost to him. |
| Result | If a user communicated with someone with a symmetric key, and then logged out, he loses the key. (Acc. To Maman protocol). So when he logs back on, and only then pulls his messages, he will be unable to decrypt any of them. But since the server already deleted them, he won’t be able to restore them. This results in loss of unread messages every time a client logs out, because he doesn’t retain his symmetric keys with other clients. |
| Prerequisites | The user communicated with someone using a symmetric key.  The user logged out before reading all his messages. |
| Business Impact | General loss of messages is a flaw in the service and can result in user frustration and loss of users. |
| Proposed Remediation | The client-side service can print a message to the user before logging out (choosing 0) warning them of this problem, and recommending they pull their messages (option 140) before logging out.  Alternatively, the client can maintain a file where he stores all his symmetric keys with other users (similar to the file that stores his private key and UUID). That way, upon logging in, he can read all his messages he hadn’t read yet. |
| Risk | Damage potential- 8  Reproducability- 6  Exploitability- 1  Affected users- 8  Discoverability- 1  Overall- 4.8 |