Security I Intermediate Project

"I wonder if people will ever say, 'Let's hear about The Silmailion and The Long Night of the Intermediate Project' And they'll say, 'Yes, that's one of my favorite projects. Froggins, Samgee, Pook, and Harrison, were really sleep-deprived, weren't they, Dad?' 'Yes, m'boy, the most deprived of sleeps. And that's saying a lot.'"

-Inspired from The Two Towers *(1954)*

TEAM:	**The Silmailion** (formerly Secur(s)ed)
PROFESSOR:	Steven Bellovin
MEMBERS:	Serena "Froggins Baggins" Cheng (stc2137)
(and fellow	William "Samgee Gamgee" Chiu (wnc2105)
LoTR fans)	Stacy "Pook Took" Tao (syt2111)
	Meribuck "Harrison Wang" Brandybuck (hbw2118)

Assumptions

- Attackers cannot sudo
- Mailbox directories are secured via permissions such that the contents (i.e. messages) cannot be tampered with, therefore no HMAC integrity checking is needed
- All users must log in once to generate certificates before being able to receive messages
- Only one message (to possibly multiple recipients) is sent at a time sendmsg
- Messages are limited to 1 MB in size; anything larger will be truncated
- No concurrent client requests

Sandboxing

- Sandboxing is intended to restrict server privileges to just the subdirectories that it needs to access when handling client requests. In particular, the CA store and mail directory, which contains all of the users' mailboxes, are sandboxed separately. Beyond providing security in the case of a compromised server, this simulates a system in which the CA and server are on different machines and where the server has to send requests to the CA. For simplicity, our implementation assumes the CA and server are on the same machine.
- Server has lots of privileged functions to access certificate store and mailboxes
    - sendmsg/recvmsg will need to access restricted mailboxes
    - getcert/changepw will need to access stored certificates and credentials
- Use chroot() to sandbox client programs' to client/public or client/tmp
- Use chroot() to sandbox server components according to the below specifications

Certificates

- Each user has at most one active certificate with appropriate configuration to act as a TLS web-client, file encrypting, and file signing certificate
- The CA revokes a user's old certificate after changepw is handled successfully and a new certificate is generated for the user

Server

- **oneprogramtorulethemail**
    - Parent process Pemithor fork()s, and the firstborn child process execl()s Boromail. Parent process Pemithor fork()s again, and the lastborn child process execl()s Faramail

- Faramail and Boromail each handle certain HTTP endpoints requested by the client:

  "A Chance For Faramail, Ca-pem Of Gondor, To Show His Security."
    - **Faramail**: Accepts TLS connections without client-side certificate
      - /getcert: login(), getcert()
      - /changepw: login(), checkmail(), changepw(), getcert()

  "One Does Not Simply Connect to My Server Without a Valid Certificate."
    - **Boromail**: Accepts only TLS connections with CA-issued client-side certificate (issued by getcert), which is verified with openssl
      - /getusercert: getusercert()
      - /sendmsg: sendmessage()
      - /recvmsg: recvmessage()

- **nineformortalmendoomedtodie**
  - Server side functions used by Faramail and Boromail to handle client requests
  - Each function is sandboxed according to its purpose, the files and directories that it needs to access, and the principle of least privilege
  - Faramail uses:
    - login() - verifies username/password against hash password stored in server/credentials/<username>.hashedpw
      - Sandboxed in server/credentials
    - checkmail() - checks if there are any pending messages in a user's mailbox
      - Sandboxed in server/mail/<username>
    - changepw() - updates the user's hashed password in server/credentials/username.hashedpw
      - Sandboxed in server/credentials
    - addcsr() - writes a CSR to server/ca/intermediate/csr/<username>.req.pem to prepare getcert()
      - Sandboxed in server/ca
    - getcert() - fulfills a CSR, stores the resulting certificate in server/ca/intermediate/certs/<username>.cert.pem
      - Sandboxed in server/ca
  - Boromail uses:
    - getusercert() - retrieves a user's (either a sender's or a receiver's) stored certificate from server/ca/intermediate/certs/<username>.cert.pem
      - Sandboxed in server/ca/intermediate/certs

        - sendmessage() - writes an encrypted and signed message to a recipient's mailbox with filename ##### (numbering scheme is the same as in HW3, e.g. 00001, 00002, etc.)
            - Sandboxed in server/mail
        - getOldestFileName() - retrieves the path of the oldest pending message in a recipient's mailbox
            - Sandboxed in server/mail
        - recvmessage() - sends the oldest pending message from the recipient's mailbox back to the client and deletes it from the mailbox (the recipient is parsed from the peer client certificate used to establish the TLS connection)
            - Sandboxed in server/mail
    - Sandboxing is performed by Faramail and Boromail with chroot() before calling these functions

<u>**Client**</u>

- getcert
    - Input username and private key file
    - Prompt user for password
    - Create CSR for user with private key
    - Connect to Faramail server via TLS without a client-side certificate
    - Send HTTP request for /getcert with username, password, and CSR
    - Receive the client-side certificate from Faramail if successful otherwise display appropriate error
        - The first time getcert is called, a new certificate will be returned
        - On all subsequent calls of getcert, the same certificate will always be returned no matter the key provided
    - Prompt user for a path to write the resulting client-side certificate to and write it
- changepw
    - Input username and (possibly new) private key file
    - Prompt user for password and new password
    - Create CSR for user with private key
    - Connect to Faramail server via TLS without a client-side certificate
    - Send HTTP request for /changepw with username, password, newpassword, and CSR
    - Receive the client-side certificate from Faramail if successful otherwise display appropriate error

    - Prompt user for a path to write the resulting client-side certificate to
    - Note that if the user had sent messages to others that remain unread before changepw, then the recipients will no longer be able to receive these messages because the signature will not be verifiable against the sender's new certificate. In this case, the message will simply be deleted from the recipient's mailbox on recvmsg. We call this feature 'unsend' :)
- sendmsg
    - Input certificate file, private key file, and message file
        - Messages are expected to begin with a properly formatted "MAIL FROM:<sender>" line and at least one "MAIL TO:<recipient>" line
    - Verify private key against certificate and connect to Boromail server via TLS after handshake with the client-side certificate
    - Parse recipients from the message
    - For each recipient, send HTTP request for /getusercert with recipient name
        - If recipient is invalid or has not generated a certificate yet, Boromail will respond with proper error code
        - Otherwise, receive the recipient's certificate from Boromail and write it to a temporary <recipient>.cert.pem file
    - For each recipient, encrypt the message with the recipient's certificate and then sign the encrypted message with the true sender's certificate
        - The true sender (i.e. the user who owns the provided certificate and its corresponding private key) may be different from the sender specified in the message's "MAIL FROM:<sender>" header. This will cause a signature verification error in recvmsg, but sendmsg does not check this case
    - For each recipient, send HTTP request for /sendmsg with encrypted and signed message and recipient name
    - Remove all temporary files
- recvmsg
    - Input certificate file and private key file
    - Verify private key against certificate and connect to Boromail server via TLS after handshake with the client-side certificate
    - Send HTTP request for /recvmsg
        - If no message to read, then do nothing
        - Otherwise receive an encrypted and signed message back from Boromail and write it to a temporary file
        - Note that the recipient name is not included in the request because otherwise an attacker could bypass the client to send an HTTP

        request for /recvmsg to Boromail with any user as the recipient field. Although the attacker may not be able to decrypt the returned message, the message would be deleted from the server, which is not acceptable. Instead, the correct recipient name is parsed from the client certificate by Boromail.

  - Parse the sender name from the message
  - Send HTTP request for /getusercert with sender name
    - If recipient is invalid or has not generated a certificate yet, Boromail will respond with proper error code and client will exit
    - Receive the sender's certificate from Boromail and write it to a temporary <sender>.cert.pem file
  - Verify the signature on the message against the retrieved sender's certificate
    - The client must do this verification because it cannot trust the server to verify. The server simply provides a certificate for the client to check against.
    - Write the verified message to a temporary file
  - Prompt user for a path to write the decrypted message to
  - Decrypt the verified message with the recipient's private key and write the plaintext message to the user-specified path
  - Remove all temporary files

File Layout and Permissions

The server directory and all server files are accessible to root only (permissions are omitted after the first line of the directory below), and the server's CA store contains all generated client certificates. The client programs escalate permission level to group `ring` which gives access to the CA chain (root and intermediate certificates) in the client/private directory. Permissions on files written by the client for the user (private key, certificate, received message) are not specifically set, except for on the private key which is set to `rwx------`.

```
tree
├── [drwxr-xr-x root     ring    ]  client
│   ├── [drwxr-sr-x root     ring    ]  bin
│   │   ├── [-rwxr-sr-x root     ring    ]  genkey.sh
│   │   ├── [-rwxr-sr-x root     ring    ]  makecsr.sh
│   │   ├── [-rwxr-S--x root     ring    ]  recvmsg
│   │   └── [-rwxr-S--x root     ring    ]  sendmsg
│   ├── [-rwxr--r-- root     ring    ]  ca.cert.pem
│   ├── [-rwxr--r-- root     ring    ]  imopenssl.cnf
│   ├── [-rwxr--r-- root     ring    ]  intermediate.cert.pem
│   └── [drwxrwx--- root     ring    ]  tmp
├── [-rwx------ serena   serena  ]  creds.txt
└── [drwx------ root     root    ]  server
    ├── [drwx------ root     root    ]  bin
    │   ├── [-rwx------ root     root    ]  boromail
    │   ├── [-rwxr-xr-x root     root    ]  boromailutils
    │   ├── [-rwx------ root     root    ]  faramail
    │   ├── [-rwxr-xr-x root     root    ]  faramailutils
    │   ├── [-rwx------ root     root    ]  getcert.sh
    │   └── [-rwx------ root     root    ]  pemithor
    ├── [drwx------ root     root    ]  ca
    │   ├── [drwx------ root     root    ]  certs
    │   │   └── [-rwx------ root     root    ]  ca.cert.pem
    │   ├── [-rwx------ root     root    ]  index.txt
    │   ├── [-rwx------ root     root    ]  index.txt.attr
    │   ├── [-rwx------ root     root    ]  index.txt.old
    │   ├── [drwx------ root     root    ]  intermediate
    │   │   ├── [drwx------ root     root    ]  certs
    │   │   │   ├── [-rw-r--r-- root     root    ]  addleness.cert.pem
    │   │   │   ├── [-rwx------ root     root    ]  boromail.cert.pem
    │   │   │   ├── [-rwx------ root     root    ]  ca-chain.cert.pem
    │   │   │   ├── [-rwx------ root     root    ]  faramail.cert.pem
    │   │   │   └── [-rwx------ root     root    ]  intermediate.cert.pem
    │   │   ├── [drwx------ root     root    ]  csr
    │   │   │   ├── [-rw-r--r-- root     root    ]  addleness.req.pem
    │   │   │   ├── [-rwx------ root     root    ]  boromail.csr.pem
    │   │   │   ├── [-rwx------ root     root    ]  faramail.csr.pem
    │   │   │   └── [-rwx------ root     root    ]  intermediate.csr.pem
    │   │   ├── [-rw-r--r-- root     root    ]  index.txt
    │   │   ├── [-rw-r--r-- root     root    ]  index.txt.attr
    │   │   ├── [-rwx------ root     root    ]  index.txt.attr.old
    │   │   ├── [-rwx------ root     root    ]  index.txt.old
    │   │   ├── [drwx------ root     root    ]  newcerts
    │   │   │   ├── [-rwx------ root     root    ]  1000.pem
    │   │   │   ├── [-rwx------ root     root    ]  1001.pem
    │   │   │   └── [-rw-r--r-- root     root    ]  1002.pem
│   │   ├──── [drwx------ root    root   ]  private
│   │   │   ├──── [-rwx------ root    root   ]  boromail.key.pem
│   │   │   ├──── [-rwx------ root    root   ]  faramail.key.pem
│   │   │   └──── [-rwx------ root    root   ]  intermediate.key.pem
│   │   ├──── [-rw-r--r-- root    root   ]  serial
│   │   └──── [-rwx------ root    root   ]  serial.old
│   ├──── [drwx------ root    root   ]  newcerts
│   │   └──── [-rwx------ root    root   ]  1000.pem
│   ├──── [drwx------ root    root   ]  private
│   │   └──── [-rwx------ root    root   ]  ca.key.pem
│   ├──── [-rwx------ root    root   ]  serial
│   └──── [-rwx------ root    root   ]  serial.old
├──── [drwx------ root    root   ]  credentials
│   ├──── [-rwx------ root    root   ]  addleness.hashedpw
  .
  .
  .
│   └──── [-rwx------ root    root   ]  whaledom.hashedpw
├──── [-rwx------ root    root   ]  imopenssl.cnf
├──── [drwx------ root    root   ]  mail
│   ├──── [drwx------ root    root   ]  addleness
│   │   └──── [-rwx------ root    root   ]  00001
  .
  .
  .
│   └──── [drwx------ root    root   ]  whaledom
```

```
                                                                        root   root other
server/                                                                r w x  - - - - - -
      imopenssl.cnf
      bin/
            oneprogramtorulethemail
            boromail
            faramail
            mail-in
            mail-out
                                                                        root  loot  other
            login                                                       r w x - - s - - -
                                                                        root  root  other
            check-mail                                                  r w x - - - - - -
```

```
                                                                  root  poot  other
      change-pw                                                   r w x - - s - - -
                                                                  root  goot  other
      get-cert                                                    r w x - - s - - -
                                                                  root  root  other
      send-to                                                     r w x - - - - - -
                                                                  root  mint  other
      msg-in                                                      r w x - - s - - -
                                                                  root  mout  other
      msg-out                                                     r w x - - s - - -
                                                                  root  vsin  other
      verify-sign                                                 r w x - - s - - -
                                                                  root  vuse  other
      verify-cert                                                 r w x - - s - - -
                                                                  root   root other
mail/                                                            r w x  - - - - - -
      billy/
            00001-stacy
            00002-stacy
            00003-harrison
      harrison/
      serena/
      stacy/
ca/
      certs/
            ca.cert.pem
      intermediate/
            certs/
                  addleness.cert.pem
                  ca-chain.cert.pem
                  intermediate.cert.pem
                  faramail.cert.pem
                  boromail.cert.pem

            csr/
            newcerts/
            private/
                  intermediate.key.pem
                  faramail.key.pem
                  boromail.key.pem
      newcerts/
      private/
```

```
                ca.key.pem

      credentials/
            addleness.hashedpw
                                                                      root ring other
client/                                                             r w x  r - x r - x
      imopenssl.cnf
      bin/                                                          r w x  r - x r - x
            get-cert                                                r w x  r - s - - x
            change-pw                                               r w x  r - s - - x
            send-msg                                                r w x  r - s - - x
            recv-msg                                                r w x  r - s - - x
      public/
            ca-chain.cert.pem
      tmp/
```

Test Plan

Basic Functionality Tests

1. getcert
    a. Verify output sent to user matches with file stored in private
    b. String sanitization and size of password
2. changepass
    a. Attempt login with old password, verify that it fails
    b. Check string sanitization for password
    c. Check size of password
3. sendmsg
    a. Verify missing or invalid certificate does not write to mailbox
    b. Verify valid certificate writes to mailbox and verify return code
    c. Verify invalid recipient names are rejected
    d. Verify message encryption only matches one recipient's encryption cert
    e. Check string sanitization for message
    f. Check size of message input and recipient names input
4. rcvmsg
    a. Verify missing or incorrect certificate does not return a message
    b. Verify correct certificate returns a message, deletes from mailbox (check size of mailbox)
    c. Verify message signature only matches one sender's signing cert
    d. Verify message is not decryptable with incorrect decryption key
    e. Verify message is decryptable with correct decryption key

Security Tests

1. Attempt to read mailbox as different group (incorrect group ID), should fail, so no one can modify mailboxes directly
2. Any fake users taking any action should be rejected
3. Assuming a valid user, ensure they cannot read/write/execute from a mailbox (need certificate)

Volume / DOS tests

1. Try to call sendmsg thousands of times
2. Try to spam getcert with thousands of login attempts
3. Try to send very very large message
4. Try to fill up a single mailbox with more than 99999 messages

Network tests

We are still awaiting the provided code for the networking part.

1. Try to connect to Boromail without a valid certificate
2. Kill network connection prematurely (e.g. using ^D to put server or client in an indeterminate state)

"Why couldn't the eagles just carry this entire project?"