NOVA Faculdade de Ciências e Tecnologia

Projeto 1

Relatório submetido no âmbito da cadeira de Segurança de Sistemas de Computadores

Afonso de Jorge – 74015

Tiago Dias - 66131

MEI

1. Introdução

O projeto tinha como objetivo garantir o armazenamento seguro de ficheiros num servidor. Estes ficheiros são divididos em blocos, que de seguida são encriptados pelo cliente e enviados para o servidor.

2. Cliente

O cliente é a componente principal da arquitetura de segurança, sendo responsável por todas as operações criptográficas. O servidor nunca tem acesso a chaves ou quaisqueres dados em claro, atuando apenas como um repositório.

Ficheiro de Configuração

Através da classe complementar CryptoConfig conseguimos obter diferentes configurações sem necessidade de recompilação do cliente. No arranque, o cliente lê o ficheiro de configuração linha a linha e analisa cada campo especificado. No nosso caso, podem ser especificados os seguintes campos:

- Cifra;
- Tamanho da chave;
- Tamanho do vetor de inicialização (IV);
- Tamanho do Nonce;
- Algoritmo escolhido para o HMAC;
- Tamanho da chave utilizada no HMAC;
- Tamanho da *tag*;

Após a leitura, a classe determina automaticamente se a cifra é AEAD, como AES/GCM por exemplo. Caso seja AEAD, não é necessário adicionar qualquer tipo de verificação de integridade/autenticidade, visto já estar incluída na cifra. Caso seja não-AEAD (como AES/CBC), é adicionado sempre a lógica de HMAC. Isto poderia não ter sido implementado de forma automatizada, ou seja, qualquer cifra não-AEAD poderia funcionar de forma insegura, mas achamos mais correto garantir sempre a integridade/autenticidade de forma automática, adicionando o HMAC. Além disso, caso existam valores não especificados no ficheiro de configuração, existem sempre um conjunto de valores padrão para garantir o correto funcionamento do programa.

Password Based Encryption (PBE)

Após a configuração estar definida, o cliente precisa de obter as chaves necessárias para a cifra/decifra de blocos. Com o objetivo de não armazenar chaves em plain-text, utilizamos um mecanismo de PBE para proteger as chaves.

Quando o cliente é executado, é pedida uma password ao utilizador. Esta password é utilizada para derivar uma chave través da função PBKDF2WithHmacSHA256, combinando a password com um salt aleatório de 16 bytes (armazenado num ficheiro único). A utilização do salt impede ataques de rainbow tables e a execução intencionalmente de forma lente (65536 iterações) garante a mitigação de ataques de força-bruta, resultando assi numa chave robusta de 256 bits.

Esta chave de proteção derivada é então usada como password para uma KeyStore, onde se encontram as chaves de encriptação e de MAC.

3. Servidor

A função principal do servidor é persistir os dados que o cliente envia , sem ter a necessidade de os decifrar ou compreender. O servidor é multithreaded, ou seja, aceita conexões de múltiplos clientes, iniciando uma nova thread para cada um.

A lógica de armazenamento é centrada no comando STORE_BLOCK, onde o servidor lê um block_id enviado pelo client (que corresponde a um hash do nome do ficheiro mais um número sequencial de bloco para garantir que o nome do ficheiro original não é exposto) e, caso não um duplicado, ou seja, se o nome ainda não existir no mapa de metadados do servidor, armazena num diretório os dados já encriptados e autenticados pelo cliente.

Metadata

Associado a cada ficheiro, e enviados juntamento com o primeiro bloco, estão as chamadas keywords que são enviadas, já hashed, também do cliente e são mapeadas num ficheiro metadata, associadas ao hash do nome do ficheiro.

Esta abordagem garante que o servidor armazena o conteúdo de forma opaca. Algum administrador que inspecione o servidor verá apenas nomes de hashes e blocos indecifráveis, dos quais não pode tirar a mínima informação.

Execução e Testing

Para testar o projeto, adicionámos um ficheiro de teste que passa por todas as operações (cltest.txt) e basta compilar e executar o BlockStorageServer e o BlockStorageClient. Ou seja:

Em terminais diferentes,

javac BlockStorageServer.java

java BlockStorageServer

e,

javac BlockStorageClient.java

java BlockStorageClient

Quando o BlockStorageClient for executado o script corre automaticamente e podem ser verificados todos os ficheiros criados na diretoria.

É dada a opção do cliente funcionar com o ficheiro de teste ou com o input de um utilizador através das seguintes linhas no começo do ficheiro do cliente:

```
Scanner scanner = new Scanner(new File ( pathname: "cltest.txt"))
//Scanner scanner = new Scanner(System.in)
```

Utilizamos o ficheiro de requerimentos do projeto como ficheiro para teste, mas caso queira ser alterado basta modificar o ficheiro cltest.txt, bem como qualquer fluxo de comandos.