Segurança de Sistemas Computacionais

Trabalho Prático nº 2 (Versão inicial v 0.9)

Resumo

O objetivo do trabalho é implementar **um sistema seguro para acesso e gestão remota de ficheiros, com segurança de comunicação e garantias de privacidade de operação**. O sistema envolve desenhar e implementar os diversos módulos da solução, nomeadamente: um módulo de controlo de despacho de operações disponibilizadas ao cliente (através de um serviço que implementa um API que pode concretizar um serviço REST (um *endpoint* de serviço seguro REST suportado por HTPS/TLS) ou com base na implementação da API a partir de um protocolo REQUEST/REPLY suportado em *Sockets TLS* ou *WebSockets TLS*). A solução deve integrar um módulo de serviço de processamento e despacho das operações, um módulo de autenticação de utilizadores, um módulo de controlo de acessos (ou de gestão de permissões) e um módulo de acesso e gestão segura do armazenamento de ficheiros com garantias de preservação de privacidade.

Todas as comunicações entre todos os módulos do sistema devem ser protegidas por TLS, sendo os *endpoints* TLS parametrizáveis, permitindo configurar a versão do protocolo TLS, o modo de autenticação e estabelecimento dos parâmetros de associações de segurança do canal TLS (incluindo autenticação mútua cliente/servidor, ou autenticação unilateral do lado do servidor ou autenticação unilateral do lado do cliente. As parametrizações permitirão ainda controlar as *ciphersuites* (com referencia base às parametrizações consideradas seguras e que possam ser suportadas e habilitadas, dadas as versões do protocolo TLS (ex.; TLS 1.1, TLS 1.2 ou TLS 1.3).

O suporte acrescido de privacidade permitirá que o repositório de dados (ficheiros) gerido pelo módulo de armazenamento de ficheiros seja concebido de forma a poder ser futuramente *outsourced* num sistema replicado de armazenamento CLOUD, resistindo a um atacante do tipo HbC (Honest-But-Curious) com preservação de privacidade dos dados e operações sobre os mesmos, não obstante esta concretização ser opcional para efeitos valorativos da concepção e implementação do trabalho por parte dos alunos e não ter lugar nos requisitos obrigatórios do trabalho) e que emulará as garantais de privacidade através de um repositório para o sistema de ficheiros usado por aquele módulo.

1. introdução

A arquitetura do sistema e os vários módulos da solução, de acordo com a funcionalidade a suportar, serão desenvolvidos em duas fases:

- A primeira fase é obrigatória para efeitos de avaliação (sendo avaliada de 0 a 14 pontos em escala de 20)
- A segunda fase é opcional, valorizando o trabalho até 6 pontos de acordo com os requisitos implementados nessa fase. Assim, a valorização será adicionada à implementação completa e correta dos objetivos realizados na fase 1.

Cada fase e o seu completamento estão associados à implementação de objetivos específicos, como se descrevem a seguir.

2. Entrega do trabalho

A entrega do trabalho envolve:

a) entrega da implementação (código fonte do projeto de implementação, com todos os componentes necessários de modo a ser avaliado, testado e ensaiado em comprovação experimental), devendo o código ser disponibilizado em repositório de projeto GitHub, partilhado para acesso com a conta "henriquejoaolopesdomingos"
b) Entrega de formulário com respostas a questões sobre a caracterização da implementação e demais aspetos questionados

Para efeitos de metodologia de desenvolvimento do trabalho, as sucessivas fases, suas funcionalidades e os serviços de segurança associados, podem ser desenvolvidas progressivamente. Tal permitirá ir atendendo os requisitos de forma incremental até à data limite de entrega do trabalho para efeitos de avaliação.

3. Arquitetura de referência do sistema a desenvolver para Fase 1

O sistema deve ser concebido com base numa arquitetura distribuída tendo como referência o modelo cliente/servidor), representada na figura 1.O cliente (usado por utilizadores) deve interagir com os servidores (módulos de serviço da solução). Na arquitetura inicial (relativa à fase 1) existirão as seguintes entidades do modelo do sistema:

- **Client**e
    - Componente que será usado elos utilizadores do sistema
- **Servidor de acesso ao serviço de ficheiros (FServer)**.
    - O **FServer** (que implementa o ponto de acesso ao serviço) funciona como um *gateway* que implementa o serviço de acesso remoto a ficheiros
- Módulos do serviço de ficheiros sendo:
    - **um módulo de serviço de autenticação (FServerAuth)**
        - **Este módulo gere a autenticação dos utilizadores**
    - **um módulo de controlo de acesso (FServerAccessControl)**
        - Este módulo gere as permissões para pedidos de realização de operações no sistema por parte de utilizadores previamente autenticados pelo serviço de autenticação
    - **um módulo de gestão de armazenamento de dados (FServerStorage)**
        - Este módulo implementará o sistema de armazenamento dos ficheiros no sistema de ficheiros local da máquina e que executar.

- Todos os anteriores módulos da solução constituem para todos os efeitos servidores independentes e autónomos, podendo ser distribuídos em diferentes máquinas (por exemplo, em diferentes instâncias de sistemas operativos virtualizados, com base em ambiente VirtualBox, VMware.
- O deployment da solução deve ser flexível, de modo que os vários servidores possam ser colocados em operação na mesma ou e diferentes máquinas (ex., máquinas virtuais), mas podendo também executar na mesma máquina, para efeitos de desenvolvimento e comprovação experimental inicial.
- Como solução opcional alternativa, cada um dos módulos pode ser concebido para executar virtualmente de forma isolada em dockers independentes se os alunos assim pretenderem

Na arquitetura de sistema distribuído, os clientes e o **FServer** comunicam por TCP/IP, sendo as interações protegidas por TLS (ou SSL). A solução pode ser baseada em interações REST/HTTPS, com base em *sockets* TLS (usando o suporte Java JSSE) ou ainda com base em WebSockets suportados em TLS.

O suporte TLS constituirá uma camada básica de segurança transparente do protocolo do nível aplicação subjacente às interações entre os módulos que assim serão "tunneled protocols" no suporte TLS, propiciando as propriedades de segurança de transporte/sessão seguros com flexibilidade de parametrização, de acordo com as parametrizações de configuração permitidas pelo suporte Java JSSE, com todas as parametrizações que permitam que a interação se faça da seguinte forma:

- Todos os componentes (m´dulos da solução) devem ser suportados em interações TLS, com possibilidade de autenticação unilateral do lado servidor ou autenticação mútua, devendo conter as necessárias *keystores* ou *trusted stores*, de acordo com o *setup* para comprovação experimental.
- Os *endpoints* de configuração (do lado servidor) apenas permitirão estabelecer sessões a partir da configuração de um ficheiro que estipulará configurações da seguinte forma (exemplo):
-

TLS-PROT-ENF <tls protocol enforced>
TLS-AUTH: <auth-type>
CIPHERSUITES: <csuite options>

Em que <tls protocolo enforced> poderá ser: TLS-1.2 ou TLS-1.1
<auth-type> poderá ser: SERV ou MUTUAL
<csuite options> é uma lista de configurações expressadas na forma:
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

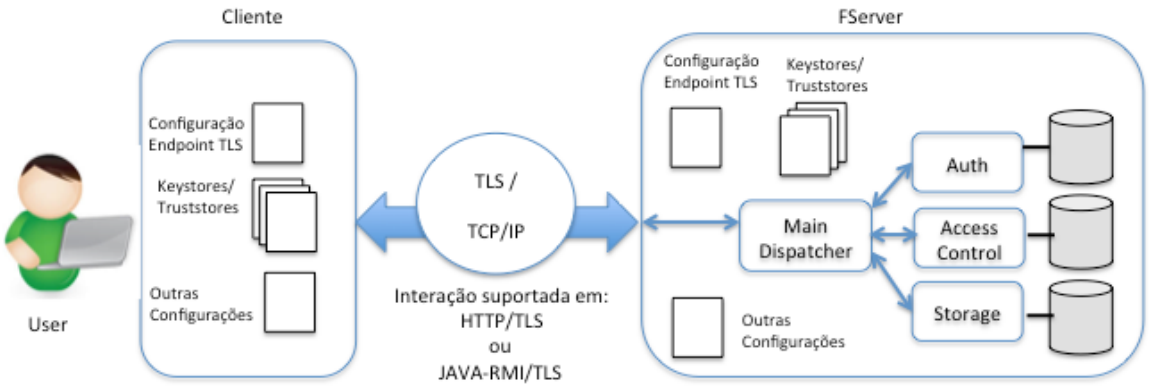

Fig. 1 Arquitetura e entidades do sistema (Fases 1 e 2)

A interação ao nível aplicação entre o cliente e o *endpoint* FServer (*Main Dispatcher*) e entre este e os restantes servidores, poderá ser suportada numa das seguintes opções (que podem ser selecionadas como alternativas por parte dos alunos):

- **Opção REST/HTTP/TLS**: corresponderá a desenvolver os servidores como *endpoints* de acesso REST suportados em TLS, com configuração sensível a ficheiros de configuração como indicados acima, de forma a impor as parametrizações de segurança que são requeridas na comunicação TLS.
    - A opção por uma implementação REST/HTTPS/TLS poderá ser suportada com base em tecnologia de um servidor aplicacional Web (REST), como por exemplo ysando tecnologia Spring (https://spring.io/, https://spring.io/projects/spring-boot). Esta opção é particularmente aconselhável para alunos que tenham já utilizado esta tecnologia para desenvolvimento de servidores aplicacionais Web em unidades curriculares anteriores. É também mais conselhável que esta opção seja considerada para a interação entre o cliente (utilizadores) e o serviço Main Dispatcher) que assim pode ser concretizado como um serviço Web/REST em moldes usuais. Uma questão interessante de exploração neste caso, é como conseguir suportar TLS nesta opção de implementação de odo a poder ser usado com possível modelo de autenticação mútua.

- **Opção Sockets TLS**: corresponderá a desenvolver os servidores como *endpoints* de acesso REST suportados em TLS, com configuração sensível a ficheiros de configuração como indicados acima. Neste caso a implementação pode recorrer ao suporte de programação em Java e com base no pacote Java JSSE (https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html sendo aqui relevante a prática de utilização de Sockets TLS, também considerada nos exercícios e observações experimentais nos materiais dos Laboratórios Lab-6 (emissão e gestão de certificados de chave pública com ferramentas tais como openssl (https://www.openssl.org/) , ou keytool (https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html)

- **Opção WebSockets TLS**: corresponderá a desenvolver os servidores como *endpoints* de acesso REST suportados em WebSockets também suportados em TLS, com configuração sensível a ficheiros de configuração como indicados acima. Informação sobre utilização e programação de Web Sockets pode ser vista em varias fontes, como por exemplo:
    - https://www.developer.com/web-services/intro-web-sockets/https://www.developer.com/web-services/intro-web-sockets/
    - https://docs.web3j.io/4.10.0/advanced/tls_http_websockets/
    - https://websockets.readthedocs.io/en/stable/topics/authentication.html,

- **Opção híbrida**: que corresponde a uma possível solução híbrida entre as anteriores, usada entre as diferentes interações de cada par dos módulos na arquitetura da solução.

Assim, entre as diversas alternativas, os alunos deverão adotar desde o início uma das opções acima para concepção e desenvolvimento do trabalho. Preferencialmente a opção escolhida deverá rentabilizar o conhecimento e experiência prática dos alunos obtida em programação de sistemas distribuídos ou aplicações de redes de computadores suportadas na pilha TCP/IP, para ser usada como referencia para toda a implementação do trabalho.

Deve notar-se que na arquitetura indicada na figura 1, o ficheiro de configuração de ciphersuites TLS nos endpoints dos módulos de serviço devem designar-se: **servertls.conf**., quando os módulos executam o papel de servidores e deve designa-se chamar **clienttls.conf** para efeitos de execução com papel de clientes. Notar que de acordo com a arquitetura, os módulos executam com algum destes papéis, nas iterações com os restantes módulos, a saber:

Interação Cliente – *Main Dispatcher*	o cliente atua como cliente TLS e o Main Dispatcher como servidor TLS
Interação *Main Dispatcher* – Serviço de Autenticação	O *Main Dispatche*r atua como cliente TLS e o Serviço de Autenticação como servidor TLS Interação *Main Dispatcher* – Serviço de Controlo de Acessos
Interação *Main Dispatcher* – Serviço de Controlo de Acessos	O *Main Dispatche*r atua como cliente TLS e o Serviço de de controlo de acessos como servidor TLS
Interação *Main Dispatcher* – *Serviço de Armazenamento de Ficheiros*	O *Main Dispatche*r atua como cliente TLS e o Serviço de de controlo de acessos como servidor TLS

Todos os restantes ficheiros de configuração que sejam necessários podem ter nomes escolhidos pelos alunos, sendo depois estes explicados na submissão do trabalho, sendo a flexibilidade e transparência das configurações para os diferentes modos de operação TLS um fator qualitativo de avaliação do trabalho.

2. Operação do sistema pra a Fase 1

Clientes

Os clientes permitem que os utilizadores possam ter acesso ao serviço de ficheiros remoto, permitindo autenticar os utilizadores e verificar o controlo de acesso e depois guardar, gerir e aceder a ficheiros no repositório de ficheiros remoto implementado no *filesystem* local do sistema onde executa o **FServer**. Os ficheiros serão organizados numa arquitetura hierárquica (de diretórios) a partir de uma raíz no sistema de ficheiros local do **FServer.** Nessa hierarquia, cada utilizador, corresponderá a uma subárvore, cuja raiz corresponde ao *username* de cada utilizador suportado.

Deste modo, do lado do **FServer**, qualquer ficheiro pode sempre ser designado remotamente na forma

/username/dir1/dir2/dir3/file

Em que *username* corresponde a um diretório *home* do utilizador "*username*", na raiz da hierarquia do sistema de ficheiros remoto gerido pelo **FServer**, através do servidor **Storage**.

Autenticação do cliente. Ao iniciar um cliente, cada utilizador deve autenticar-se com um par <"*username*","*password*"> ou <"email" "password"> sendo a autenticação dos clientes suportada pelo

módulo de autenticação do **FServer**. O protocolo de autenticação entre cliente e **AServer** (intermediado/reencaminhado de forma transparente pelo componente *MainDispatcher*) derá ser implementado em duas opções possíveis (sempre protegido por TLS):

- Uma opção corresponde a implementar um protocolo da fase de autenticação do sistema *Kerberos* (bastando que seja baseado no protocolo de autenticação base *Kerberos V5*). Notar que esta interação estará protegida por sua vez pelos serviços de segurança da camada TLS (SSL).
- O serviço de autenticação implementa um modelo de autenticação com passwords, com base numa tabela pré-definida de utilizadores previamente configurados para usarem o sistema, na forma:

  Username: SecureHashedPassword: Salt: Counter

- Complementarmente, o cliente vai usar, para além do envio da password, um protocolo com base num acordo *Diffie Hellman*, como a seguir se indica como linha inicial de orientação geral – mas que deve ser discutida pelos alunos no sentido do enriquecimento do protocolo:

  Cliente > FServer : username
  FServer > Cliente:
  $SecureRandom_1$ || assinatura de Yaserver // Nº público DH

  Cliente > AServer:
  { H (PWD || $SecureRandom_1$+1 } Ks || $SecureRandom_2$ || Assinatura de Ycliente
  ou, alternativamente usando proteção do autenticação via password-based encryption:
  { [ClienteID + Ext Client Attributes], || $SecureRandom_1$+1, Timestamp, Ycliente }$_{Kpbe}$

  AServer > Cliente: {$Assinatura_{AServer}$ (A || Ktoken1024 || TTL || other-credential-att)) || $SecureRandom_2$+1}Ks
  ou, por ex., com separação dos componentes:
  { A, Ktoken1024, TTL, || $SecureRandom_2$+1 }Ks || $Assinatura_{AServer}$ (A || Ktoken1024 || TTL || other-credential-att)

Em que:
Yaserver e Ycliente são parâmetros públicos de um acordo *Diffie-Hellman*, assinados (autenticados) pelo cliente e pelo **AServer**
Ks: uma chave gerada com base no acordo
Ktoken1024: Chave *token* para acesso ao **FServer (via o componenet Dispatcher)** assinado por **AServer**. Notar que este *token* será válido durante um TTL (from: data; to: data) como credencial autenticada de acesso por A, ao **FServer**, representando assim uma codificação de um cookie seguro e autenticado, com controlo de validade, que pode ser futuramente escrutinado por qualquer entidade backend do FServer.
other-credential-att: atributos adicionais que se entendam ser relevantes para a validação do autenticador no processamento das operações subsequentes.

Notar que independente de variantes do protocolo de autenticação (com propriedades de segurança semelhantes às requeridas), a interação do cliente só se faz com o componente Dispatcher que deve estar habilitado a verificar o autenticador (assinado pelo módulo AServer), em todas as operações solicitadas pelo cliente no contexto de uma sessão autenticada (na qual o autenticador apresentado é válido). Por outro lado, deve ter-se em conta que as sucessivas operações realizadas pelo Dispatcher nos componentes AccessControl e Storage, devem dar garantias a estes módulos que o Dispatcher está a intermediar corretamente o uma sessão previamente autenticada de um utilizador (com base na validação por parte do AccessControl do autenticador emitido pelo AServer) , e que a intermediação de operações pedidas ao módulo de Storage, podem ser escrutinadas por este com base n avaliação de que o pedido está a ser

feito a partir de uma verificação de controlo de acesso por parte do módulo AccessContro.

Assim, após autenticação bem sucedida de um cliente no protocolo de autenticação (e portanto estabelecida uma sessão autenticada (fim a fim) de um utilizador (nível aplicação) sobre os canais TLS (nível transporte/sessão ponto a ponto), será controlado o acesso do cliente pelo módulo *Access Control* e módulo de Storage, para todas as operações que o cliente pretende realizar e que estão a ser intermediadas corretamente pelo Dispatcher.

O cliente disponibilizará ao utilizador a seguinte funcionalidade:

	Operação	**Obrigatório FASE 1 ?**	**Opcional na Fase 1 ?**
Login username password	**Login de um utilizador no sistema.**	**SIM**	
ls username	Mostra ficheiros ou diretórios do utilizador *username* na sua home-root no repositório de ficheiros remotos	**SIM**	
ls username path	Mostra ficheiros ou diretórios do utilizador *username* na path indicada, sendo esta especificada na forma /a/b/c, a partir do directório home do utilizador.	**NÃO**	**SIM**
mkdir username path	Cria um directório no caminho path indicadp	**NÃO**	**SIM**
put username path/file	Coloca um ficheiro file no caminho path indicada SIM	**SIM**	
get username path/file	Obtém o ficheiro file no caminho path	**SIM**	
cp username path1/file1 path2/file2 (	Copia ficheiro file 1 no caminho path 1 para file 2 no caminho path 2	**SIM**	
rm username path/file	Remove/Apaga o ficheiro file no caminho path	**SIM**	
file	Mostra atributos do ficheiro file na path indicada, devolvendo o nome, se é um diretório ou se é um ficheiro, o tipo de ficheiro, data da criação, data da última modificação (seguir a semântica do comando file em ambiente Unix)	**NÃO**	**SIM**

Login username password (obrigatório fase 1)

// retorna em caso de sucesso uma credencial de autenticação emitida pelo módulos de autenticação, para que o cliente a apersente posteriormente ao módulo de controlo de acessos. Como inspiração, deve tentar suportar esta credencial com um formato inspirado numa estrutura similar à de um JSON Web Token, podendo inserir todos os elementos ou atributos de autenticação que considere convenientes. Notar que este token tera´que ser obtido assinado pelo serviço de autenticação

Ex: https://jwt.io/

Depois da credencial (token) de autenticação ser exibido ao serviço de controlo de acessos, e no caso de

obtenção de autorização para a operação pretendida pelo cliente, o servidor de controlo de acesso após verificar a validade da credencial de autenticação, deve por sua vez emitir uma credencial de autorização de acesso (em moldes similares à credencial anterior), agora assinada pelo serviço de controlo de acessos. Esta nova credencial será então posteriormente apresentada pelo cliente ao serviço de armazenamento de ficheiros. Este, após validação e aceitação da credencial de controlo de acessos, poderá então realizar a operação que o cliente pretende.
É importante notar que não existe interação direta entre o cliente e os serviços de autenticação, controlo de acesso e armazenamento, Todas estas interações são intermediadas pelo serviço *Main Dispatcher*, que é o único módulo que interage com o cliente, na arquitetura da solução.

FServerAuth

Este módulo gere uma tabela de autenticação (de estrutura inspirada no ficheiro /etc/passwd de um sistema UNIX, Linux ou Mac OSX). Cada entrada está associada a um utilizador, como se ilustra a seguir:

hj:hj@fct.unl.pt:Henrique Domingos:**********:TRUE

Os campos (neste caso separados por ":") possuem respetivamente a seguinte informação:

- Username:Email:Nome:**********:BOOLEAN
- O campo ********* possui a password (ou uma sua transformação, por exemplo uma síntese da mesma) cifrada por uma chave simétrica. Trata-se de um segredo partilhado com o cliente. No cliente, pode usar-se um esquema do tipo PBEEncryption para implementar o protocolo de autenticação.
- O valor booleano indica que:
    - TRUE: o utilizador pode ser autenticado
    - FALSE: o utilizador está bloqueado e não pode ser autenticado (não podendo assim usar o sistema)

Aspetos importantes a ter em conta:

- Os utilizadores que podem autenticar-se no sistema (e assim serem suportados ao nível do módulo de autenticação) são definidos manualmente, num ficheiro similar ao ficheiro de controlo de utilizadores locais para autenticação num sistema Unix.
- Mais informação de inspiração para os ficheiros de configuração de utilizadores e fatores de autenticação pode ser vista em:
    - https://manpages.ubuntu.com/manpages/trusty/en/man5/passwd.5.html
    - https://manpages.ubuntu.com/manpages/trusty/en/man5/shadow.5.html

    Notar que no contexto da gestão de passwords neste tipo de ficheiros, é particularmente relevante o uso de sínteses de passwords, de acordo com as discussões sobre sínteses seguras e resistentes a ataques de longa duração, usadas para passwords. Ver também uma discussão prática em: https://www.cyberciti.biz/faq/understanding-etcshadow-file/

    Um dos aspetos importantes no trabalho é desde logo a concepção e especificação por parte dos alunos (a partir das linhas de orientação acima) do suporte do módulo de serviço de autenticação, bem como o desenho do protocolo de autenticação que será depois jogado entre o cliente e o módulo de autenticação (através do entreposto Main Dispatcher).

FServerAccessControl
Este componente deve implementar uma política de controlo de acesso. Para tal usará um ficheiros de configuração que implementa uma tabela de controlo de acesso do tipo discricionário.

- Especificação do ficheiro access.conf:
  username1: deny // Utilizador não autorizado a ler ou escrever ficheiros
  username3: allow read // só pode ler ficheiros
  username4: allow read write // pode ler e escrever

- Qualquer *username* que não conste do ficheiro *access.conf*, não poderá ter acesso ao serviço. Neste ficheiro is usernames podem ser expressos pelos identificadores ou endereços Email, conforme estão configurados na tabela de autenticação gerida pelo módulo FServerAuth.
- Notar que o módulo **FServerAccessControl** implementará o conceito associado ao princípio de mediação completa, com a respetiva gestão de privilégios. Por outro lado, nenhum acesso a ficheiros por parte de utilizadores pode ser feito sem o necessário escrutínio da correta autenticação (via **FServerAuth**) e das reras de controlo de acesso (via FServerAccessControl), o que deverá ser coordenado pelo módulo **FServer**.

FServerStorage
Este módulo implementa o serviço de armazenamento de ficheiros.

- É o módulo que gere o filesystem remoto como pretendido. A sua função é criar - escrever ou ler ficheiros (ou criar diretórios), suportando as operações solicitadas pelos utilizadores remotos (operações solicitadas via o endpoint FServer).
- Tem um ficheiro local de configuração com tabelas que definem a raiz do *file-system* remoto (sob a qual estão os diretórios de ficheiros dos utilizadores numa estrutura hierárquica), tendo por base uma diretoria no filesystem.

3. Arquitetura de referência do sistema a desenvolver para Fase 2

A fase 1 envolve extensões à arquitetura inicialmente desenvolvida na fase 1. Na fase 2, para além da extensão da funcionalidade dos anteriores módulos na passagem da fase 1 para a fase 2, a arquitetura irá incorporar alguns módulos adicionais: um módulo de indexação no cliente que permitirºa que todos os ficheiros sejam armazenados no módulo de Storage de forma a que os ficheiros serão armazenados pelo módulo de storage com as seguintes garantias a fornecer aos clientes:

- Confidencialidde: os ficheiros geridos e armazenados, são mantidos e operados sempre cifrados;
- Integridade: os ficheiros estão armazenados de forma a ficar preservada e detectável a sua integridade;
- Autenticidade: os ficheiros serão armazenados com provas de assinatura digital do dono desses ficheiros

Por outro lado, o módulo de armazenamento será estendido por um módulo proxy (**FServerProxy**) que permitirá ao componente **FServerStorage** usar repositórios de dados outsourced - *Cloud* (ex., *Dropbox, AmazonEC3, GoogleDrive,* ou mesmo um ou mais serviçso de Cloud Storage) em vez do sistema de ficheiros local usado pelo módulo **FServerStorage**. Para esse efeito, o módulo tratará de replicar e manter os ficheiros em diferente repositórios, o que aumentará as garantias de disponibilidade e tolerância a falhas do armazenamento. Neste contexto apenas se prevê tolerar falhas por paragem (*fail-stop only model*) em alguma réplica do sistema de armazenamento

A figura 2 constitui uma referencia inicial da arquitetura para a concepção e implementação da fase 2 do trabalho.

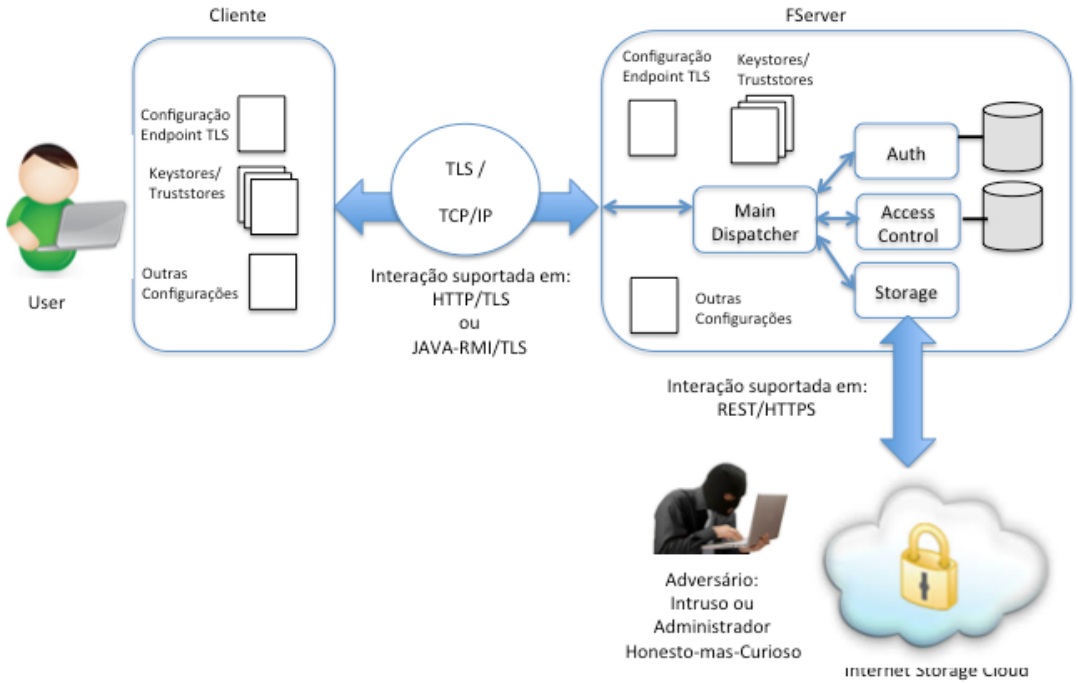

Figura 2. Arquitetura estendida para utilização de uma solução Internet Storage Cloud

4. Operação e demonstração do sistema com implementação na Fase 2

A operação e demonstração do sistema, conforme implementações por parte dos grupos será discutida em aula, consoante a concepção de cada grupo. Algumas linhas de orientação comuns ser incluídas na versão 1.0 do enunciado.

5. Critérios de avaliação

A referência de critérios de avaliação é a seguinte:

FASE 1: de 0 a 14 valores (sendo o máximo obtido com a completude, correção e qualidade do trabalho, bem como a sua entrega nos prazos estabelecidos e assim sem penalizações).

FASE 2: de 0 a 6 valores (sendo o máximo obtido com a completude, correção e qualidade do trabalho, bem como a sua entrega nos prazos estabelecidos e assim sem penalizações).

O resumo seguinte apresenta um sumário dos critérios orientadores de avaliação das soluções e pontuações de referência

FASE 1 0-14 valores (70% de 100 pontos)		Conformidade da arquitetura e suporte de interação entre os módulos	Correção da interação de segurança (TLS) entre os módulos da solução	Correção, conformidade e completude dos 5 módulos da solução	Funcionamento e interoperação de cada um dos módulos da arquitetura da fase 1	Flexibilidade e suporte de parametrizações de segurança nos *endpoints* TLS dos diversos módulos
	Implementação	0-10	0-10	0-10	0-10	0-10
	Teste/Verificação e Correção	0-10	0-10	0-10	0-10	0-10
	TOTAL					
	0-100	0-20	0-20	0-20	0-20	0-20

FASE 2	Ver sugestões de valorizações abaixo						
Sugestões:	a)	b)	c)	d)	e)	f)	g)
0-6 valores (30% de 100 pontos obtidos a partir e cada *feature* considerada	0-10 pontos	0-10 pontos	0-10 pontos	0-20 pontos	0-30 pontos	0-40 pontos	(a considerar em função da garantia em concreto)

Sugestões de aspetos de concepção e implementação que podem ser tidos em conta para avaliação de critérios da FASE 2

a) Implementação da funcionalidade (operações) consideradas opcionais para a Fase 1
b) Todos os ficheiros que ficam no sistema de armazenamento ficam, guardados sempre cifrados (garantindo-se confidencialidade), possuem prova de integridade e estão assinados pelo utilizador que lá os guardou. Isso garante que um adversário do tipo "HbC" não poderá alterar nem quebrar a autenticação do utilizador que lá os colocou nem consegue quebrar a confidencialidade ou integridade dos mesmos
c) Todos os ficheiro são guardados de modo que a sua dimensão é sempre igual. Isso não permite a um adversário do tipo HBC poder inferir nada sobre o tamanho dos mesmos
d) Todos os ficheiros, são guardados obedecendo a a), b) e c) e garantindo que os nomes dos ficheiros e diretorias são igualmente cifrados, pelo que um adversário do tipo HbC não poderá inferir nada sobre os nomes e a ligação dos mesmos aos eventuais conteúdos.
e) Todas as garantias a), b) , c) e d) com os ficheiros guardados particionados (ou fragmentados) em blocos (sempre do mesmo tamanho), sendo o armazenamento operado como se fossem conteúdos de blocos de ficheiros num disco.
f) Todas as garantias a), b), c) e d), e do lado do servidor um adversário do tipo HbC só vê uma única diretoria que é usada como uma tabela única (*Big Table*) de blocos. O adversário não consegue inferir nada sobre a estrutura hierárquica que possa estar a ser gerida pelo cliente
g) Outras garantias que possam ser propostas e implementadas pelos alunos coo fatores diferenciadores e valorativos do trabalho em conjugação ou em complementaridade com as anteriores