Ciências ULisboa | Informática

Aplicações Distribuídas

2017/18

Projeto 4

1. Descrição geral

A avaliação da cadeira de aplicações distribuídas está divida em quatro projetos. O quarto projeto incide sobre medidas de desempenho e de segurança para o projeto anterior (projeto 3).

No terceiro projeto foi concretizado um serviço WEB para gerir um sistema simplificado de classificação de series de TV de utilizadores. Para esse efeito no servidor foi utilizada a *framework* de desenvolvimento WEB *Flask* [2] e o motor de base de dados SQL *sqlite* [3]. O programa cliente utiliza o módulo *requests* [4] para implementar a interação cliente/servidor baseada no HTTP.

2. Desempenho

Assumindo que o serviço é executado através do servidor WSGI embutido no *Flask* e que este executa num computador com múltiplos núcleos de processamento, os alunos deverão alterar o servidor do projeto 3 para que este responda eficientemente a múltiplos clientes. Além das alterações ao código deverão justificar o método escolhido no ficheiro README que será entregue.

Ainda no sentido de aumentar o desempenho, independentemente das capacidades ou configuração do servidor, o cliente desenvolvido através do módulo *requests* deverá fazer com que os vários pedidos feitos ao servidor sejam enviados na mesma ligação TCP (ligações persistentes no HTTP).

3. Autenticação, confidencialidade da comunicação e autorização

O protocolo SSL/TLS deverá ser utilizado com o *Flask* e com o módulo *requests* para que o cliente e o servidor verifiquem mutuamente a autenticidade do interlocutor e para que a comunicação seja confidencial (cifrada). Para este efeito, cliente e servidor deverão ter certificados de chave pública assinados por uma *Certificate Authority (CA)*. A implementação no *Flask* será feita através da classe `SSLContext` do módulo `ssl` [1] da biblioteca padrão do *Python*. O protocolo *OAuth2* [2][3] deverá ser utilizado com o *Flask* e com o módulo *requests-oauthlib* [4][5] para que o cliente possa pedir autorização para usar os recursos disponiblizados pelo servidor. Para este efeito, o URI do servidor deverá estar registado numa API de OAuth (google, facebook, twitter,...), de forma que o cliente possa ser autenticado e autorizado por esta API.

3.1. Criação de certificados

Para que os certificados do cliente e do servidor possam ser assinados por uma CA é necessário criar um certificado para a CA fictícia:

```
openssl genrsa -out root.key 2048
```
(cria a chave da CA)

```
openssl req -x509 -new -nodes -key root.key -sha256 -days 365 -out root.pem
```
(cria um certificado autoassinado para a CA)

De seguida serão geradas as chaves do servidor e do cliente de acordo com o seguinte padrão de comando:

```
openssl genrsa -out <nome do ficheiro>.key 2048
```

Para que a CA assine os certificados do cliente e do servidor ter-se-á que emitir um pedido de assinatura de certificado para cada um. Isso pode ser feito de acordo com o padrão de comando:

```
openssl req -new -nodes -key <nome do ficheiro>.key -sha256 -days 365 \
-out <nome do ficheiro>.csr
```

Na posse dos pedidos de assinatura a CA procede à geração dos certificados assinados por si. Isso pode ser feito para o cliente e para o servidor pelo seguinte padrão de comando:

```
openssl x509 -req -in <nome do ficheiro>.csr -CA root.pem -CAkey root.key \
-CAcreateserial -out <nome do ficheiro>.crt -days 365 -sha256
```

Através deste método cliente e servidor deverão, cada um, ter um par de ficheiros `<nome>.crt` e `<nome>.key`. Estes serão utilizados nos programas para que a autenticação e comunicação cifrada com o interlocutor sejam possíveis. Na implementação da autenticação os programas cliente e servidor terão que utilizar o certificado da CA (root.pem) para validar a assinatura do certificado apresentado pelo interlocutor.

4. Configurações de Segurança

Além das tarefas descritas acima, os alunos deverão definir regras da firewall *iptables* para proteger o servidor do sistema.

Antes de começar a realizar o projeto, estude a ferramenta *iptables* e efetue os exercícios do guião da aula PL.

Pretende-se que os alunos utilizem o comando *iptables* de modo a configurar a máquina segura onde será instalado o servidor.

A melhor maneira de garantir a segurança da máquina é reduzir os seus serviços ao mínimo indispensável e garantir a sua constante atualização. Neste contexto, a *firewall* deve ser configurada de modo a concretizar a seguinte política:

- Serviços suportados: *ping*, serviços necessários para o servidor e serviço SSH.
- Restrições: a máquina responde a *pings* com origem na máquina **nemo.alunos.di.fc.ul.pt (10.101.85.18)**, aceita ligações de clientes com qualquer origem para o servidor, e aceita ligações SSH da sua rede local.
- Serviços utilizados: DNS

Os alunos devem incluir no ficheiro README um relatório com o seguinte conteúdo:

- Regras do comando *iptables* que permitem concretizar a política; e
- explicação do método de teste utilizado e observações realizadas.

Observações:

i) o normal funcionamento dos computadores dos laboratórios depende do seu acesso às seguintes máquinas:
DCs: 10.121.53.14, 10.121.53.15, 10.121.53.16
Storage: 10.121.72.23
Iate/Falua: 10.101.85.6, 10.101.85.138
Nemo: 10.101.85.18
Gateway: 10.101.148.1

Proxy: 10.101.85.136, 10.101.85.137

Deste modo ao testarem as suas regras não devem impedir o acesso a estas máquinas.

ii) a opção –F do *iptables* não altera a política definida por omissão. Assim, a seguinte sequência de comandos bloqueará o computador (ver justificação na observação anterior):

```
$…
$ sudo iptables –P OUTPUT DROP
$ sudo iptables –F OUTPUT
```

iii) O tráfego do dispositivo de loopback não deve ser filtrado:

```
$ sudo iptables –A INPUT –i lo –j ACCEPT
$ sudo iptables –A OUTPUT –o lo –j ACCEPT
```

iv) O tráfego relacionado com uma ligação já estabelecida também deve ser aceite:

```
$ sudo iptables –A INPUT –m state –-state ESTABLISHED,RELATED  –j
    ACCEPT
$ sudo iptables –A OUTPUT –m state –-state ESTABLISHED,RELATED –j
    ACCEPT
```

5. Entrega

A entrega do projeto 4 consiste em colocar todos os ficheiros *.py* e *.db* do projeto, o ficheiro README e o ficheiro com o esquema da base de dados em SQL numa diretoria cujo nome deve seguir exatamente o padrão **grupoXX** (por exemplo grupo01 ou grupo23). Nesta diretoria será colocado o ficheiro README bem como três diretorias com os nomes client, server e certs. Nestas serão colocados os ficheiros referentes ao cliente (grupoXXX/client), ao servidor (grupoXXX/server/) e aos certificados (grupoXXX/certs/).

Juntamente com os ficheiros *.py* e *.db* deverá ser enviado um ficheiro de texto README.txt (não é .pdf nem .rtf nem .doc nem .docx) onde os alunos podem relatar a informação que acharem pertinente sobre a sua implementação do projeto (por exemplo, limitações). Neste ficheiro os alunos, também, colocarão um exemplo para cada operação que a aplicação cliente suporta. A diretoria será incluída num ficheiro ZIP cujo nome deve seguir exatamente o padrão **grupoXX.zip**. Esse ficheiro será submetido num recurso a disponibilizar para o efeito na página de AD no moodle da FCUL.

Note que a **entrega deve conter apenas os ficheiros .py, .sql, certificados e o ficheiro README.txt, qualquer outro ficheiro vai ser ignorado**.

O prazo de entrega é domingo, dia 20/05/2018, até às 22:00hs.

6. Bibliografia

[1] https://docs.python.org/2/library/ssl.html

[2] https://aaronparecki.com/oauth-2-simplified/

[3] https://oauth.net

[4] https://pypi.python.org/pypi/requests-oauthlib

[5] http://requests-oauthlib.readthedocs.io/en/latest/index.html