Laboratórios de Informática 2017–2018
DETI - Universidade de Aveiro

Trabalho de aprofundamento 2

Objetivos:

- Planeamento e preparação do trabalho de aprofundamento 2.

Este guião apresenta as regras o segundo trabalho de aprofundamento. Os exercícios sugeridos dão os primeiros passos para a concretização do trabalho recorrendo às ferramentas estudadas.

1.1 Regras

O trabalho deve ser realizado por um grupo de 2 e entregue, via a plataforma `https://elearning.ua.pt`, dentro do prazo lá indicado. A entrega deverá ser feita por **apenas um dos membros** do grupo e deve consistir de **um único arquivo** `.zip`, `.tgz` (TAR comprimido por `gzip`) ou `.tbz` (TAR comprimido por `bzip2`). O arquivo deve conter o código desenvolvido assim como o ficheiro PDF final do relatório, todos ficheiros com código fonte (`.tex`, `.bib`, etc.) e todas as imagens ou outros recursos necessários à compilação do código ou documento. Excluem-se aqui eventuais imagens de máquinas virtuais utilizadas.

Na elaboração do relatório recomenda-se a adopção do estilo e estrutura de relatório descrito nas aulas teórico-práticas e a utilização de recursos de escrita como: referências a fontes externas, referências a figuras e tabelas, tabela de conteúdo, resumo, conclusões, etc. O objectivo do relatório é descrever a implementação, apresentar testes que comprovem o seu funcionamento correto e analisar os resultados obtidos.

É obrigatório incluir uma secção "Contribuições dos autores" onde se descrevem resumidamente as contribuições de cada elemento do grupo e se avalia a percentagem de trabalho de cada um. Esta auto-avaliação poderá afetar a ponderação da nota a atribuir a cada elemento.

1.2 Avaliação

A avaliação irá incidir sobre:

1. **cumprimento dos objetivos através das funções suportadas**,
2. **qualidade do código produzido**,
3. **testes unitários realizados**,
4. **estrutura e conteúdo do relatório**,
5. **utilização das funcionalidades de tarefas do code.ua e git**.
6. **o suporte de segurança adicionado**

A soma das componentes 1 a 5 não será inferior a 16 valores.

Relatórios meramente descritivos sem qualquer descrição da aplicação, apresentação dos resultados obtidos, testes efetuados, ou discussão serão fracamente avaliados.

Só serão avaliados trabalhos enviados via a plataforma `https://elearning.ua.pt`. Ficheiros corrompidos ou inválidos não serão avaliados à posteriori e não será permitido o reenvio.

Deve ser utilizado um projeto na plataforma Code.UA, com um identificador segundo o formato `labi2018-ap2-gX`. **Substitua o caractere X pelo número 1. Se não for possível criar este projeto, incremente o número até que o resultado seja positivo.**

1.3 Tema Proposto

Os docentes instalaram uma sonda de temperatura, humidade e vento, sendo que esta possui a capacidade de enviar os dados através de uma rede sem fios. A aquisição é feita de forma automática e constante, emitindo um valor novo a cada 10 segundos.

O objetivo do trabalho é o de criar um cliente com a capacidade de aceder remotamente à sonda, registando os dados num ficheiro Comma Separated Values (CSV)[1]. Além disso, deverão ser apresentadas no terminal algumas indicações sobre a possibilidade (ou necessidade) de se levar t-shirt, casaco, gorro ou outras peças de roupa. O mecanismo de decisão e as regras ficam ao critério de cada grupo.

A comunicação com a sonda faz-se através de um `socket` Transmission Control Protocol (TCP)[2], na porta 8080 do servidor `xcoa.av.it.pt`, enviando-se comandos de texto e recebendo-se objectos JavaScript Object Notation (JSON)[3]. Tanto os comandos como os objetos `JSON` são compostos por uma única linha e terminados pelo caractere `\n`.

Para testar a interação com o servidor, tal como descrito nas aulas teórico-práticas, é possível utilizar os comandos `telnet` ou `nc`.

É importante que se dividam as operações em pequenos blocos de código (funções) de forma a que seja possível a realização de testes unitários. Na realização destes testes, deve-se validar a invocação das funções com valores dentro e fora do domínio de operação. As funções de registo dos valores e as utilizadas para elaboração das sugestões, cálculo do valor X ou cifra, são as melhores candidatas para a incorporação de testes.

As operações suportadas são as descritas de seguida:

`CONNECT <valor_inicial>`: Estabelecer uma ligação à sonda, indicando opcionalmente um valor inicial a utilizar (explicado mais abaixo). É devolvida uma mensagem de erro ou a mensagem seguinte:

```
{
    "TOKEN": "<string_token>",
}
```

O campo **`TOKEN`** possui um valor aleatório, único por ligação, que deve ser incluído em todos os outros pedidos.

`READ <string_token>`: Indica que o cliente pretende receber os valores da sonda, que passarão a ser enviados automáticamente e de forma periódica.

```
{
    "TEMPERATURE": <float>,
    "HUMIDITY": <float>,
    "WIND": <float>,
}
```

1.3.1 Segurança nas comunicações

O argumento **`valor_inicial`** corresponde aos valores necessários para uma troca de chaves usando o algoritmo Diffie-Hellman. O algoritmo é bastante simples e permite que duas entidades cheguem a acordo relativamente a uma chave, sem que esta seja enviada no canal de comunicações.

O processo baseia-se no facto de $(x^b)^a = (x^a)^b$ e funciona da seguinte forma:

- O cliente possui 2 valores públicos p e g, que por exemplo podem ser 23 e 5. Depois este gera um valor aleatório a e calcula $A = g^a \ mod \ p$. A operação **`mod`** corresponde ao resto da divisão inteira. Considerando que $a = 6$, em Python faria-se `A = pow(5,6,23)`, logo `A=8`.
- O cliente envia para o servidor o conjunto de valores **`A,p,g`** (ou seja **`8,23,5`**). Isto corresponde ao valor do argumento **`valor_inicial`**.
- O servidor recebe o valor inicial e gera um valor aleatório b, calculando depois $B = g^b \ mod \ p$, tal como o cliente fez, enviando B para o cliente.
- O cliente recebe B e calcula agora $X = B^a \ mod \ p$, enquanto por sua vez o servidor calcula $X = A^b \ mod \ p$.
- Se tudo tiver corrido bem, o valor X irá ser igual em ambas as entidades e é um segredo partilhado que pode ser usado num processo de cifra.

Utilizando este conceito, se for especificado o argumento **`valor_inicial`** a resposta será uma mensagem no seguinte formato:

```
{
    "TOKEN": "<string_token>",
    "B": <inteiro>,
}
```

Após isto, todas as mensagens enviadas **em ambos os sentidos** devem ser cifradas utilizando o algoritmo **`AES-128`**, com o resultado codificado para Base64[1]. Continua a considerar-se que as mensagens são terminadas com o caractere **`\n`**.

Como chave utilizam-se os primeiros octetos 16 do resultado em hexadecimal (**`hexdigest`**) da síntese **`MD5`** sobre `str(X).encode("utf-8")`.

[1] ver `https://docs.python.org/3/library/base64.html`

Glossário

CSV Comma Separated Values

JSON JavaScript Object Notation

TCP Transmission Control Protocol

Referências

[1] Y. Shafranovich, *Common Format and MIME Type for Comma-Separated Values (CSV) Files*, RFC 4180 (Informational), Internet Engineering Task Force, out. de 2005.

[2] J. Postel, *Transmission Control Protocol*, RFC 793 (Standard), Updated by RFCs 1122, 3168, 6093, Internet Engineering Task Force, set. de 1981.

[3] E. T. Bray, *The JavaScript Object Notation (JSON) Data Interchange Format*, RFC 7159, Internet Engineering Task Force, mar. de 2014.