



## Segurança de Sistemas Informáticos

### Guia para Aula Laboratorial 3

2º Ciclo em Engenharia Informática

2º Ciclo em Eng. Eletrotécnica e de Computadores

2º Ciclo em Matemática e Aplicações

## Computer Systems Security

### Guide for Laboratory Class 3

M.Sc. in Computer Science and Engineering

M.Sc. in Electrical and Computer Engineering

M.Sc. in Mathematics and Applications

#### Sumário

Exploração de funcionalidades avançadas da ferramenta OpenSSL: geração de números pseudo-aleatórios, derivação de chaves de cifra a partir de palavras-passe e teste de comunicações TLS.

#### Summary

*Exercises for revision of concepts concerning advanced functionalities of the OpenSSL toolkit: pseudo random number generation, password based key derivation functions and testing of TLS /SSL communications.*

#### Pré-requisitos:

A maior parte das tarefas enunciadas em baixo requerem a utilização da ferramenta OpenSSL (<http://www.openssl.org/>). Um ambiente linha de comandos Linux também facilita a execução de algumas dessas tarefas. Sugere-se, assim, o uso de uma distribuição comum de Linux, onde todas estas condições estarão provavelmente preenchidas ou pressupõe-se o acesso a um sistema com a possibilidade de instalar o *software* necessário.

## 1 Geração de Números Pseudo Aleatórios

### *Pseudo Random Number Generator*

A geração de sequências de bits aleatórias ou pseudo-aleatórias é de extrema importância em quase todos os mecanismos criptográficos. O OpenSSL implementa uma funcionalidade que permite obter uma sequência de bits pseudo-aleatória com tamanho à escolha.

#### Tarefa 1 Task 1

Escreva e emita o comando que lhe permite obter, no ecrã, uma sequência de 10 bytes pseudo-aleatórios em hexadecimal.

```
openssl rand -hex 10
```

Corra o comando mais do que uma vez e verifique que o material devolvido é diferente de cada vez que é executado.

**Q1.: Qual deve ser o valor da entropia calculada**

#### para bytes produzidos com o comando anterior?

- ☒ Deve ser próximo do máximo.
- ☐ Deve ser próximo do mínimo.
- ☐ Deve ser 0.
- ☐ Deve ser negativo.

## 2 Tratamento e Armazenamento de Palavras-Passe

### *Management and Storage of Passwords*

No sistema operativo linux, as *passwords* dos utilizadores costumam estar guardadas, de uma forma mais ou menos segura, no ficheiro `\etc\shadow`.

#### Tarefa 2 Task 2

Faça um `cat` ao ficheiro `\etc\shadow` e procure perceber quantos utilizadores têm a *password* configurada.

No ficheiro antes mencionado deve haver uma entrada parecida com:

```
root:$1$h0DZpn2J$fffGRrpV60/8tF2XuCSrM1:...
```

No *output* anterior, os cifrões são delimitadores de valores. O primeiro valor com interesse é logo a primeira palavra (*root*), que se refere ao utilizador. O \$1 indica que está a ser usado o algoritmo 1 para tratamento e armazenamento da *password* (algoritmo baseado no *hash* MD5)). O terceiro valor é o *salt* utilizado e o último é o valor que o sistema operativo guarda da *password*.

### Q2.: Sabe a *password* de *root* do computador do laboratório?

- ☐ Sim, é
- ☐ Não, mas já estou a ver que vou precisar dela...

### Tarefa 3 Task 3

Usando a *password* do utilizador de *root* e o *salt* que observa no ficheiro `\etc\shadow`, construa o comando `openssl` que lhe permite obter o valor que é guardado no ficheiro referido em representação da *password*. O comando deve começar com `openssl passwd` ....

```
openssl passwd -6 -salt YTEdZjE3Q labredespcc
```

Caso não esteja neste momento com acesso a uma máquina do laboratório, assuma que o valor que a entrada que lá encontrava para o utilizador *root* era:

```
root:$6$YTEdZjE3Q/z$g0fFhnBGZdtWt/DBVwhWAf...
```

Procure saber como é que o dito algoritmo funciona, para descargo de consciência.

Repare, o que este exercício mostra é que existe uma forma simples de conseguir obter uma representação para uma *password* tão segura como a que é usado no sistema Linux. Se quisesse implementar um sistema de autenticação numa aplicação local, poderia usar esse comando (ou integrar a sua funcionalidade) para guardar e verificar as credenciais de um utilizador.

Para ficar bem assente, emita o comando que permite criar a representação da *password* *superman* com o *salt* *LoisLane*.

### Q3.: Partindo do que aqui foi feito, conseguia, quase certamente, desenhar um método arcaico de *password cracking*, ou não?

- ☒ Sim! :) ☐ Não! :(
- Elabore um pouco na forma de o conseguir:

```
openssl passwd -6 -salt LoisLan superman
```

## 3 Teste de Comunicações TLS

### Testing TLS Communications

O `OpenSSL` tem uma ferramenta bastante útil para testar comunicações *Secure Socket Layer* (SSL) *Transport Layer Security* (TLS), e.g., sites `https`. Para ser seguro, o `HTTP` pode ser executado sobre uma sessão `TLS` que, por sua vez, corre sobre *Transport Control Protocol* (TCP), que corre sobre *Internet Protocol* (IP). Se quisermos testar uma ligação `HTTP`, normal, que corre sobre `TCP`, pode-se fazer *telnet* para a máquina remota e emitir comandos `HTTP` nessa ligação. O *telnet* trata de colocar os pedidos `HTTP` em pacotes `TCP`. No caso do `HTTPS`, precisamos de estabelecer uma ligação `SSL` ou `TLS` com o servidor, e não só uma ligação `TCP`.

### Tarefa 4 Task 4

Procure a opção `OpenSSL` que permite estabelecer e testar uma ligação `SSL`, e emita um comando que estabeleça essa ligação com o site da google (`www.google.com`).

```
openssl s_client www.google.com:443
```

### Q4.: Qual o tamanho do segredo mestre (*Master-Key*) trocado no estabelecimento da ligação?

- ☒ Entre 20 a 40 bytes. ☐ Entre 40 a 60 bytes.  
☐ Entre 60 a 80 bytes. ☐ Entre 80 a 100 bytes.

### Q5.: Qual é o algoritmo de cifra utilizado para efeitos de confidencialidade da ligação?

- ☐ AES. ☐ 3DES. ☐ RSA. ☒ Salsa20. ☐ RC4.

```
openssl s_client -tls1_3 www.google.com:443
```

### Q6.: Qual é o algoritmo usado para troca de chaves de sessão?

- ☐ Diffie-Hellman.  
☒ Diffie-Hellman sobre curvas elípticas.  
☐ RSA.  
☐ Pré-distribuição de chaves.

**Q7.: Qual é a função de *hash* usada para cálculo do código da origem da informação?**

- ☐ MD5.    ☐ SHA1.    ☐ SHA256.    ☐ SHA512.  
☐ SHA\_\_\_\_\_

**Q8.: Quantos certificados fazem parte da cadeia de certificação deste site?**

- ☐ 1.    ☐ 2.    ☒ 3.    ☐ 4.    ☐ 5.

**Q9.: Quantos certificados fazem parte da cadeia de certificação do site da UBI (www.ubi.pt:443)?**

- ☐ 1.    ☐ 2.    ☐ 3.    ☒ 4.    ☐ 5.

```
openssl s_client -tls1_3 www.ubi.pt:443
```

#### Tarefa 5 Task 5

##### Atenção:

Em versões mais recentes do OpenSSL, este usa os certificados contidos em `/etc/ssl/`, `/etc/pki/` ou `/etc/ca-certificates` automaticamente. Nesse caso, para se poder tirar total proveito do exercício, deve-se mudar (com cuidado) o nome padrão da diretoria temporariamente. Os comandos serão semelhantes a:

```
> sudo mv /etc/ssl/ /etc/ssl2/
```

Fazer os exercícios desta tarefa e emitir novamente

```
> sudo mv /etc/ssl2/ /etc/ssl/
```

Quando usa o `s_client`, o OpenSSL verifica automaticamente a cadeia de certificação associada a um site. Emita novamente o comando que permite estabelecer a sessão TLS para o site da UBI e procure a informação do estado da verificação de certificados. Procure responder às questões que se seguem. **Q10.: Os certificados foram verificados com sucesso?**

- ☐ Sim, foram.  
☐ Não, obtive alguns erros.

**Q11.: Caso tenha respondido “não” à questão anterior, qual acha ser a causa do insucesso? Por outras palavras, qual foi o primeiro erro que obteve?**

- ☐ Não consegue encontrar a raiz da confiança localmente.  
☐ Confiança a mais.  
☐ Nome inválido.  
☐ Prazo de validade ultrapassado.

**Q12.: Caso tenha respondido “Falta de confiança” na questão anterior, experimentou fornecer ao `openssl` uma localização dos certificados que considera como sendo de confiança?**

- ☐ Eh... não, não tentei. Mas era para tentar?  
☐ Tentei pois...

**Nota:** tente usar a opção `-CApath`

#### Tarefa 6 Task 6

Depois de fazer uma ligação com um site usando `> openssl s_client -connect`, pode fazer pedidos HTTP, usando a sintaxe correta do protocolo:

```
GET /Default.aspx HTTP/1.1
```

```
Host: www.ubi.pt
```

Experimente inserir as duas linhas anteriores no terminal, depois de ter estabelecido a ligação SSL com o site da UBI.

**Q13.: Verificou (observação humana) que a página foi, de facto, descarregada após inserir a instrução anterior?**

- ☐ A página foi descarregada.  
☐ A página não foi descarregada.

#### Tarefa 7 Task 7

Verifique se também conseguia obter a página por `telnet` (e.g., com um comando semelhante a `> telnet www.ubi.pt 80`).

A página da UBI pode ser obtida sem ligação segura?

- ☐ Sim, consegue.  
☐ Mais ou menos...  
☐ Mas é claro que não!

#### Tarefa 8 Task 8

Use a opção `-msg` para verificar quantas mensagens do protocolo SSL/TLS (experimentar com TLS1.2 e TLS1.3) são utilizadas para estabelecer a sessão e o material criptográfico:

- ☐ 3.    ☐ 4.    ☐ 5.    ☐ 6.    ☐ 7.    ☐ 8.    ☐ 9.    ☐ 10.

**Q14.: O número de mensagens bate certo com o que é especificado, e.g., na wikipedia em [http://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security)?**

- ☐ Sem tirar nem por.  
☐ Que estranho. Não bate certo...

## 4 Simular um Servidor TLS

### *Simulating a TLS Server*

Na parte anterior, foi sugerida a utilização do comando `s_client` do `OpenSSL` para testar um servidor SSL/TLS. Neste laboratório sugere-se que simule também o servidor.

#### Tarefa 9 *Task 9*

Abra dois terminais, e experimente emitir o seguinte comando num deles:

```
> openssl s_server -nocert -cipher  
ADH:@SECLEVEL=0
```

**Q15.: Dos três tipos de autenticação suportados pelo SSL/TLS, qual o tipo que é forçado pelo comando antes inserido?**

- ☐ Ligação Anónima (sem autenticação).
- ☐ Autenticação do Servidor.
- ☐ Autenticação Mútua.

**Q16.: Qual a porta em que o servidor que colocou a correr fica a correr?**

- ☐ 143.
- ☐ 1443.
- ☐ 443.
- ☐ 4433.

#### Tarefa 10 *Task 10*

Desligue a *firewall* da sua máquina, anote o seu endereço *Internet Protocolo* (IP) e peça a um(a) colega que faça a ligação SSL/TLS à sua máquina com o comando `OpenSSL s_client`.

**Q17.: Consegue comunicar (tipo *chat*) com o seu(u)a colega?**

- ☐ Parece que sim.
- ☐ Sim, consigo.

#### Tarefa 11 *Task 11*

Experimente correr o `tcpdump` para ver o conteúdo dos pacotes que está a trocar com o(a) colega. Para tornar a experiência mais sólida, considere seguir o procedimento descrito em baixo:

1. Emita o comando `tcpdump` no terminal, de modo a mostrar todo o tráfego que está a sair da sua máquina (use a opção `-X`).
2. Caso não saiba qual é o nome da interface que deve escutar, faça `> ifconfig` ou `ip add e`

analise o resultado do comando.

3. Abra um *browser* e navegue até uma página simples, sem `https` (e.g., `http://www.di.ubi.pt:80`).
4. Verifique, simultaneamente à execução da tarefa anterior, que o output do `tcpdump` mostra informação em texto limpo acerca do site (e.g., apresenta o URL).
5. Feche o *browser*, e comece a enviar mensagens usando o `openssl`.

**Q18.: As mensagens enviadas usando `openssl` vão cifradas ou em texto limpo?**

- ☐ Vão cifradas (pelo menos eu não as consigo ver no terminal usando o `tcpdump`).
- ☐ Não, não vão cifradas, porque eu vejo-as no terminal usando o `tcpdump`.

#### Tarefa 12 *Task 12*

**Q19.: Se lhe pedissem a sua opinião de especialista, diria que o `OpenSSL` é capaz de fornecer as funcionalidades de servidor `HTTPS`?**

- ☐ E já agora também cantava como se fosse o *Bono* dos U2?
- ☐ Dá.

A experiência anterior não tinha certificados digitais envolvidos, nem quaisquer páginas HTML transmitidas sobre o protocolo SSL/TLS. Para começar, crie um certificado auto-assinado para garantir a autenticação do servidor aquando das ligações. O procedimento que vai tomar é parecido ao que deveria tomar se fosse criar um servidor `https`:

```
> openssl req -new -x509 -extensions v3_ca  
-keyout serverkey.pem -out server.pem -days  
1825
```

Repare que o comando sugerido cria um certificado no ficheiro `server.pem` e guarda chaves RSA em `serverkey.pem`. O `OpenSSL` pede algumas informações ao utilizador interativamente, incluindo uma palavra-passe, que usa para cifrar o ficheiro das chaves RSA. Depois disto feito, procure criar o ficheiro `index.html`, que mostre apenas o texto “*Olá Mundo!*”.

Tendo todos os requisitos apontados anteriormente satisfeitos, construa o comando que simule, na máquina que opera, um servidor `http`. Depois, tome as providências que achar necessárias para responder

às questões incluídas a seguir. Aponte os comandos que utilizar:

---

---

**Q20.: Já experimentou aceder com o comando `openssl s_client`? Funciona?**

- ☐ Sim e sim.    ☐ Sim e não.  
☐ Não e sim.    ☐ não e não.

**Q21.: É possível aceder à página index.html (e site) usando um *browser*?**

- ☐ Isso é que era bom! Não tenho nenhum servidor http instalado...  
☐ Mas é claro que sim!

**Q22.: Qual é a opção do OpenSSL `s_server` que lhe permite devolver uma página depois de obter um pedido HTTPS válido através do `openssl s_client` (ou e.g., de um *browser*)?**

- ☐ -WWW    ☐ -ssl2    ☐ -CAfile    ☐ -HTTP