Universidade do Minho

Mestrado em Engenharia Informática

Engenharia de segurança

Ficha de exercício 7

Grupo 1

Rui Carlos Azevedo Carvalho - PG47633
Daniel Barbosa Miranda - PG47123
Ana Luísa Lira Tomé Carneiro - PG46983

Protocolo TLS

Pergunta P1.1

i.

As universidades escolhidas foram a universidade do Minho e a universidade de Aveiro.

O teste SSL efetuado para o site da primeira `www.uminho.pt` pode-se encontrar no seguinte link. Já para a segunda universidade, o relatório pode-se ser verificado neste link.

ii.

O teste com menor *rating* é o da universidade do minho, com uma nota B. Este teste revelou várias inseguranças, tal como a utilização de parâmetros fracos no acordo de chaves com Diffie-Hellman, não há suporte para renegociação segura e as versões suportadas de TLS são a 1.0 e 1.1.

Para além disto, é possível verificar que em termos das cifras utilizadas o servidor usa de preferência a cifra "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256" que é uma cifra segura. No entanto as cifras que se seguem na lista de preferências são consideradas inseguras. Em geral, esta lista apresenta ao todo mais protocolos inseguros ou protocolos seguros mas com parâmetros inseguros do que utilizações corretas de cifras seguras. Em relação aos certificados é possível verificar que ambos utilizam como algoritmo de assinatura o RSA com SHA384 com chaves de 4096 bits, o que é considerado seguro. Para além disto, ambos os certificados não estão revogados e ambos são considerados confiáveis.

iii.

O "Zoombie Poodle" é uma vulnerabilidade que permite explorar *stacks* de servidores que se comportam de forma diferente quando recebem *records* TLS com endereços MAC válidos tendo estes um *padding* inválido. Estes servidores ao receberem um *record* TLS que contenha um *padding* não uniforme enviam um "TLS alert" que indica o tamanho correto do *padding*, ou seja, são autênticos oráculos, pois permite enviar vários ciphertexts com *paddings* diferentes no MAC e o servidor continuará a responder se este *padding* está bem formado ou não, o que permite a atacantes através de várias tentativas, enviando vários *records* com alterações do *padding* até que descubram quase byte a byte os bytes decifrados do último bloco. Assim sendo o campo presente no relatório indica se o servidor onde o website está alocado é um destes oráculos,

sendo que, se o for então é mais fácil para um atacante conseguir decifrar *ciphertexts* intercetados [1].

Protocolo SSH

Pergunta P2.1

Os servidores escolhidos para a realização desta pergunta foram **search7sci.di.uminho.pt**, da Universidade do Minho e **climetua.fis.ua.pt** da Universidade de Aveiro. Esses servidores foram obtidos em https://www.shodan.io/.

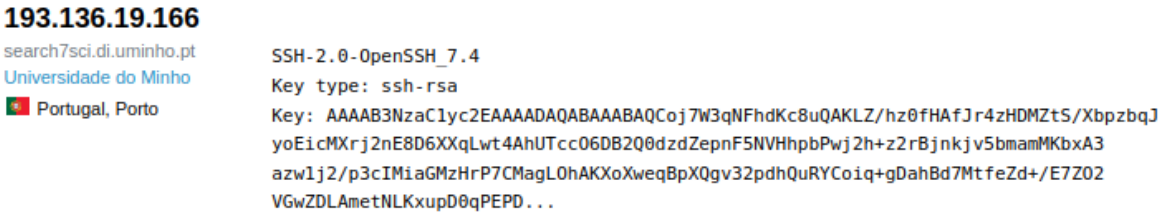

Figure 2.1: Servidor escolhido da Universidade do Minho

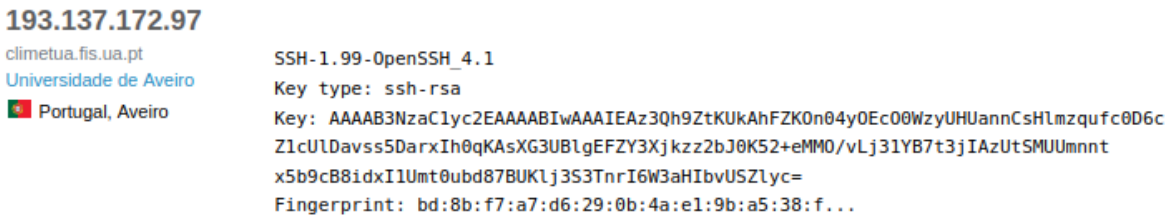

Figure 2.2: Servidor escolhido da Universidade de Évora

Ponto 1

Assim sendo, com auxílio do comando `ssh-audit` para a Universidade do Minho e para a Universidade de Aveiro obtiveram-se os resultados presentes has hiperligações https://github.com/uminho-mei-engseg-21-22/Grupo1/blob/main/Pratica%201/TP7/P2.1/uminho.txt e https://github.com/uminho-mei-engseg-21-22/Grupo1/blob/main/Pratica%201/TP7/P2.1/uaveiro.txt, respetivamente.

Ponto 2

Nas hiperligações mencionadas e, também nas 2 figuras acima, é possível observar que a versão utilizada pelo servidor da Universidade do Minho é `OpenSSH 7.4` e a da Universidade de Aveiro é `OpenSSH 4.1`.

Ponto 3

Nas hiperligações OpenSSH 7.4 e OpenSSH 4.1 estão listadas as respetivas vulnerabilidades sendo que a primeira apresenta 1 vulnerabilidade e a segunda 9, logo o servidor da Universidade de Aveiro apresenta mais vulnerabilidades.

Ponto 4

Quanto a qual possui a vulnerabilidade mais grave, também é `OpenSSH 4.1`, possuindo 2 com *CVSS Score* de 7.5, sendo estas `CVE-2007-4752` e `CVE-2010-4478`.

Ponto 5

Na primeira vulnerabilidade, ocorria um tratamento incorreto de cookies pelo que um atacante poderia ignorar a política esperada, de modo a elevar os seus privilégios e portanto ser tratado como confiável.

Na segunda, alguns parâmetros de comunicação não eram validados de forma correta, no protocolo `J-PAKE`, utilizado em acordos de chave autenticado por senha, pelo que, um atacante não necessitava do segredo partilhado podendo simplesmente realizar a autenticação.

Ambas estas vulnerabilidades apresentam caraterísticas semelhantes, na medida em que, ocorre *Information Disclosure* considerável, o atacante pode modificar algumas informações e também poderão ocorrer problemas de performance no sistema, reduzindo a disponibilidade do mesmo.

Assim sendo, pela observação dessas caraterísticas é possível concluir que ambas as vulnerabilidades são graves, já que haveria um comprometimento do sistema no que diz respeito a confidencialidade, integridade e disponibilidade.

TOR (The Onion Router)

Pergunta P3.1

Parte 1

A utilização do comando `sudo anonsurf start` não implica uma troca de localização para os EUA.

Parte 2

Segundo a sua descrição, este comando inicia o *tunneling* anônimo em todo o sistema sob o proxy TOR através de tabelas de IPs. Ora, para tal, o sistema obtém uma lista de nodos Tor de um servidor da diretoria e, dessa lista, seleciona um conjunto aleatório de nodos sobre o qual estabelece circuitos e gere as conexões das aplicações do utilizador. Dado que a seleção de nodos é anónima, também o IP de saída do circuito o será pelo que não se pode garantir que a execução do comando em questão irá trocar a localização para EUA, ou até mesmo mantê-la.

Pergunta P3.2

Parte 1

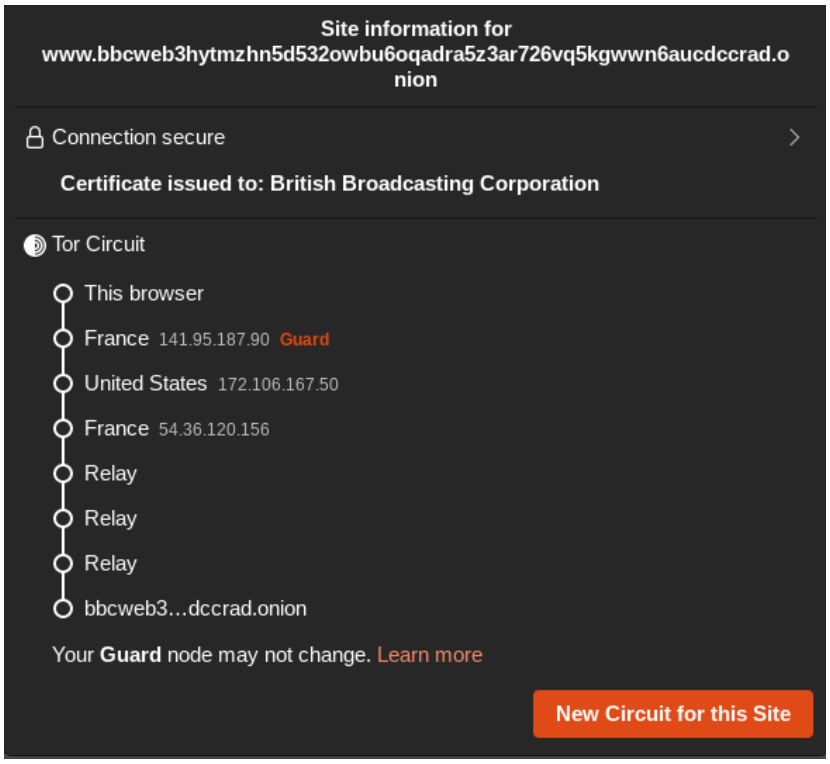

Figure 3.3: Circuito TOR em BBC

Parte 2

Porque existem 6 "saltos" até ao site Onion, sendo que 3 deles são "relay"? Utilize características do protocolo TOR para justificar.

Pelo protocolo TOR, os primeiros 3 saltos correspondem ao circuito TOR gerado pelo o utilizador, sendo que este circuito está conectado ao *Rendezvous Point*, enquanto que os 3 restantes, os *Relays*, correspondem àqueles gerados pelo servidor, estando também conectados ao *Rendezvous Point*. Estes saltos têm como objetivo proteger tanto o cliente como o servidor, na medida em que são anonimizados um face ao outro.

Parte 3

O *Rendez-vous Point* é um *Onion Router* escolhido pelo utilizador, aginda como um ponto de comunicação entre os circuitos gerados pelo utilizador e os circuitos gerados pelo servidor.

Blockchain

Pergunta P4.1

Esta pergunta consistia em alterar o código fornecido pelo docente de forma a criar um Genesis Block, cujo o *timestamp* fosse a data de hoje e que o dado incluído nesse bloco fosse "Bloco Inicial da KoreCoin". Para isso alterou se a função *createGenesisBlock* que tem como objetivo criar o bloco Genesis segundo as regras impostas neste. Abaixo encontra-se a função alterada segundo o que era pedido na pergunta.

```
createGenesisBlock(){

    // *** PROBLEMA 4.1 ***

    // Determina a data do dia de hoje
    var today = new Date();
    // Gera a data segundo o formato DD/MM/AAAA
    var date = today.getDate() + "/" + (today.getMonth()+1)
                + "/" + today.getFullYear();
    // Gera o bloco de block Genesis com a data criada
    //       e o dado 'Bloco inicial da koreCoin'
    return new Block(0, date, "Bloco inicial da koreCoin", "0");
}
```

Com esta alteração conseguimos obter o *output*, tal como se encontra na imagem abaixo.

Figure 4.4: *Blockchain* obtida segundo os requisitos pedidos

Pergunta P4.2

Esta pergunta consistia em adicionar alguns blocos simulando várias transações em cada um. Para isso foi necessário acrescentar mais blocos utilizando para isso o construtor *Block* e a função *addBlock*, ambas da classe *BlockChain*, que constrói e adiciona um novo bloco à *blockchain*, respetivamente. Abaixo encontra-se o código alterado de forma a adicionar e simular transações nos novos blocos da *blockchain*.

```
let koreCoin = new Blockchain();

// Blockchain original
//koreCoin.addBlock(new Block (1, "01/01/2018", {amount: 20}));
//koreCoin.addBlock(new Block (2, "02/01/2018", {amount: 40}));
//koreCoin.addBlock(new Block (3, "02/01/2018", {amount: 40}));

// Novos Blocks Adicionados
koreCoin.addBlock(new Block (1, "03/05/2022",
                             [{transaction: "T2", amount: 15},
                             {transaction: "T3", amount: 10, string: "Value"},
```

```
                                {transaction: "T4", amount: 20}]));
koreCoin.addBlock(new Block (2, "03/05/2022",
                                [{transaction: "T1", amount: 5},
                                {transaction: "T3", amount: 10, string: "Value"}]));
koreCoin.addBlock(new Block (3, "03/05/2022",
                                {transaction: "T1", amount: 5}));


console.log('Is Blockchain valid? ' + koreCoin.isChainValid());

console.log(JSON.stringify(koreCoin, null, 4));
```

Com esta alteração conseguimos obter o *output*, tal como se encontra nas imagens abaixo.

Figure 4.5: *Blockchain* obtida segundo os requisitos pedidos

Pergunta P4.3

Esta pergunta consiste em alterar a dificuldade de minerar para 2, 3, 4 e 5 e assim analisar o tempo que demora o processo de mineração para cada uma destas dificuldades. Para se alterar a dificuldade é necessário a alteração no construtor da classe *Blockchain* tal como está no código abaixo.

```
constructor(){
    this.chain = [this.createGenesisBlock()];
    this.difficulty = 2; // 3, 4, ou 5
}
```

De seguida, acrescentou-se ao código a função *now* da classe *performance* de forma a medir o tempo que se leva a minerar 3 blocos com cada uma das dificuldades, tal como está no código abaixo. Os tempos medidos para cada uma das dificuldades encontram-se na tabela 4.1.

```
let koreCoin = new Blockchain();

var startTime = performance.now()

console.log('Mining block 1...');
koreCoin.addBlock(new Block (1, "01/01/2018", {amount: 20}));
console.log('Mining block 2...');
koreCoin.addBlock(new Block (2, "02/01/2018", {amount: 40}));
console.log('Mining block 3...');
```

```
koreCoin.addBlock(new Block (3, "02/01/2018", {amount: 40}));
var endTime = performance.now()

console.log(`Call to doSomething took ${endTime - startTime} milliseconds`)
```

Dificuldade	Medição 1	Medição 2	Medição 3	Média
2	30.47 ms	30.87 ms	33.03 ms	32.46 ms
3	225.06 ms	223.58 ms	223.61 ms	224.08 ms
4	1188.76 ms	1200.43 ms	1190.29 ms	1193.16 ms
5	16991.06 ms	17201.62 ms	17230.88 ms	17141.19 ms

Table 4.1: Tempo medidos para as várias dificuldades

Um processo de mineração da *blockchain* corresponde à resolução de um puzzle que implica usar muito poder computacional de forma a criar um novo bloco para a *blockchain* que cumpra com os requisitos estabelecidos no puzzle. No programa fornecido pelo docente é pretendido que os *miners* gerem uma *hash* com determinada quantidade de zeros, sendo que essa quantidade vai definir a dificuldade do puzzle. Assim, dificuldade de dois significa que os *miners* terão de criar um bloco cuja a *hash* comece com 2 zeros. Como os *miners* não podem influenciar na *hash*, terão de usar muito poder computacional para conseguirem gerar várias *hashs* de forma a encontrar uma que comece com dois zeros.

Tal como podemos ver na tabela, quanto maior a dificuldade maior o tempo necessário para realizar o processo de mineração. Isto acontece pois, conseguir gerar uma *hash* com pelo menos dois zeros iniciais leva menos tempo do que tentar gerar uma *hash* com cerca de 5 zeros iniciais. Assim, a probabilidade de encontrar uma *hash* com pelo menos 2 zeros é muito maior do que encontrar uma *hash* com pelo menos 5 zeros, daí demorar mais tempo quando aumentamos a dificuldade da mineração.

Pergunta P4.4

1.

O algoritmo *proof of work* presente no código da experiência é o seguinte:

- Criar uma variável incrementor que, inicialmente, contém o valor da última prova incrementado em uma unidade.
- Incrementar esta variável até que se encontre um valor que seja divisível por 9 e pelo valor da última prova: incrementor % 9 == 0 $\wedge$ incrementor % last_proof == 0
- Este valor é o valor da *proof of work* deste *miner*

2.

Não é um algoritmo adequado para minerar. Um dos aspetos mais importantes a ter em conta nestes algoritmos é que o facto de um miner resolver um dos problemas não deveria afetar a probabilidade nem a dificuldade de outros *miners* fazerem o mesmo no futuro, pois os *puzzles* escolhidos devem ser independentes.

Ora, neste caso como se começa sempre com o valor da última prova incrementado em 1 unidade, o *puzzle* não é independente dos outros. Para além disto, torna-se cada vez mais dificil encontrar um valor que seja divisível por 9 e pelo valor da última prova com o aumento do valor da última prova.

Por estas duas razões é possível concluir que não é um algoritmo adequado para minerar.

Bibliography

[1] "What is Zombie POODLE?" [online]. Disponível em: https://www.tripwire.com/state-of-security/vert/zombie-poodle/ [Acedido em abril de 2022].