Universidade do Minho

Mestrado Integrado em Engenharia Informática

Engenharia de Segurança

Aula 2 - 17/Fev/2020

João de Macedo
A76268

João Aloísio
A77953

Nelson Gonçalves
A78173

24 de Fevereiro de 2020

1 Pergunta P1.1

A primeira conclusão que podemos retirar da execução destes comandos é que, em ‘dev/-random‘, com o aumento do tamanho requerido de bytes aleatórios, existe também um aumento do tempo necessário para que o mesmo seja gerado e apresentado.

A segunda conclusão que podemos retirar da execução do último comando é que, em ‘dev/urandom‘, a geração do número pseudoaleatório, independentemente do tamanho, é feita de imediato, visto que o sistema não espera pela coleta da entropia necessária para gerar uma sequência de bytes pseudoaleatórios criptograficamente forte. Isto implica que, apesar de ser um método mais eficiente em termos de desempemho, pode por vezes não ser seguro logo, ser teoricamente suscetível a ataques criptográficos.

2 Pergunta P1.2

Após a execução dos comandos apresentados, é possível concluir que, com o _daemon_ do **haveged** ativo, a coleta da entropia necessária para a geração de bytes aleatórios é bastante mais rápida pelo que, mesmo pedindo a geração dos mesmos através de ‘/dev/random‘, esta é feita de forma imediata. Esta conclusão pode ser retirada de dois factos

- O **haveged** é um _daemon_ que coleta entropia apartir de efeitos indiretos,como por exemplo o uso do contador do clock cycle do processador , que ocorrem no sistema, ou seja, é capaz de coletar bastante mais entropia do que seria normalmente o caso, o que permite gerar mais rapidamente os bytes aleatórios.

- A realização de uma experiência feita pelo grupo, ao executar o comando ‘head -c 10000000 /dev/random | openssl enc -base64‘, pedindo a geração de dez milhões de bytes aleatórios, que resultou apenas no tempo de geração dos número necessário de bytes e não uma espera excessiva pela coleta de entropia. Servindo como indicador, a execução completa deste comando, desde a geração do primeiro byte ao último demorou cerca de 6 segundos apenas, o que para números desta grandeza é compreensível.

3 Pergunta P2.1

A. Após a análise dos 3 ficheiros indicados e a necessária geração da chave privada e do seu certificado, gerados numa diretoria criada para o propósito com o nome de ‘key_p21‘, foi necessário executar o seguinte comando de forma a dividir o segredo com as propriedades definidas no enunciado: ‘python2 createSharedSecret-app.py 8 5 g12 key_p21/mykey.pem‘. De seguida, introduzimos a palavra-chave definida no ato da criação da chave privada e inserimos o segredo indicado no enunciado. Para reconstruir o segredo a partir de um quorum de 5 intervenientes, foi necessário executar o seguinte comando: ‘python2 recoverSecretFromComponents-

app.py 5 g12 key_p21/mykey.crt‘. Como é possível ver no segundo comando mostrado, é necessário indicar ao programa o número de intervenientes necessários (quorum) para que seja possível recuperar o segredo. O programa de seguida, pede a inserção de 5 componentes das 8 previamente geradas ao criar o segredo.

B. A diferença entre os dois programas reside no facto de que um deles apenas precisa das componentes de um número de intervenientes definidos como o quorum (neste caso são 5) para conseguir recuperar o segredo (‘recoverSecretFromComponents-app.py‘), enquanto que o outro necessita das componentes de todos os intervenientes inicialmente definidos (neste caso são 8), para conseguir recuperar o segredo (‘recoverSecretFromAllComponents-app.py‘). Uma das situações em que poderemos precisar de todos os intervenientes será, por exemplo, se quisermos alterar o segredo, visto que para esse efeito deve ser dada permissão de todos os intervenientes e, assim, o segredo alterar-se-ia e novas chaves seriam distribuídas pelo número de intervenientes necessários.

Pergunta P3.1

4 Pergunta P4.1

Tendo nos sido atribuído o grupo 8, no site https://webgate.ec.europa.eu/tl-browser/ foi possível obter o conteúdo necessário para descobrir o algoritmo e o tamanho da chave usado. Resultado do comando *openssl x509 -in cert.crt -text -noout*:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5d:f5:55:01:8c:89:45:56:59:8d:cf:d9:13:3b:87:ab
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = PT, O = "ACIN-iCloud Solutions, Lda",
        OU = Global Trusted Sign,
        CN = Global Trusted Sign Root Certification Authority 01
        Validity
            Not Before: Aug 11 15:40:09 2017 GMT
            Not After : Aug 11 15:40:09 2023 GMT
        Subject: C = PT, O = "ACIN iCloud Solutions, Lda",
        OU = Global Trusted Sign,
        CN = Global Trusted Sign Certification Authority 01
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
```

```
                    00:c0:4c:71:be:42:f6:ba:fa:f5:47:97:c2:b0:b4:
                    3e:fc:c1:f9:a9:c8:8a:8e:03:24:65:79:5d:0b:55:
                    8f:f2:35:06:f8:20:4c:2a:b2:0b:81:1c:7d:d2:74:
                    1a:cc:32:45:df:fc:9d:d7:32:02:20:3f:e9:bf:78:
                    f5:fa:dd:fc:48:66:3f:68:6c:c9:6a:0f:53:82:4f:
                    b9:71:b4:e7:aa:e8:90:7e:c5:9f:be:01:c7:55:33:
                    41:05:df:af:91:8f:e9:a2:de:32:3a:b2:9a:aa:23:
                    f3:0b:30:e9:a3:f5:cd:4e:07:47:74:07:38:41:e8:
                    46:cb:2a:c5:ea:82:58:29:89:61:6e:09:53:3d:cf:
                    77:87:2f:e1:80:aa:ab:3d:6c:da:27:35:10:d3:7e:
                    51:af:6b:9d:5f:a0:b8:e1:f2:22:e3:1c:11:60:a5:
                    e3:70:31:5c:7d:1b:a5:91:8e:79:7e:83:ec:5e:81:
                    e9:7c:60:22:b4:30:72:ba:32:f2:be:17:f2:52:00:
                    52:95:4a:d0:ed:93:fb:81:88:64:de:bd:f2:23:40:
                    5a:ca:f3:1f:b8:e9:95:cd:13:ed:4d:83:4a:e5:7c:
                    4a:51:56:ab:36:77:50:b5:a1:a0:a0:ef:77:ea:df:
                    6b:86:29:16:ba:e3:7b:e3:b5:71:94:91:04:32:75:
                    5e:05
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Subject Key Identifier:
        E6:95:6F:01:2D:F9:79:22:E7:00:39:19:88:15:B9:0A:8B:45:99:8B
            X509v3 Authority Key Identifier:
    keyid:A6:98:7C:A0:91:84:D9:DC:5E:DB:4E:2F:82:02:33:95:D3:74:2E:41

            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.50302.1.1.1.1.1.0
                  CPS: https://pki.globaltrustedsign.com/index.html

            Authority Information Access:
    CA Issuers - URI:https://pki.globaltrustedsign.com/root/gts_root.crt

            X509v3 CRL Distribution Points:

                Full Name:
      URI:https://pki.globaltrustedsign.com/root/gts_root_crl.crl
```

```
                Full Name:
URI:https://pki02.globaltrustedsign.com/root/gts_root_crl.crl

Signature Algorithm: sha256WithRSAEncryption
     35:5a:c9:6b:56:ec:59:93:c2:fc:ee:bf:6c:b6:00:b1:d2:c4:
     0f:6e:ad:0f:71:3b:36:4f:fd:3a:ef:1b:1d:96:52:93:91:b4:
     9f:31:05:b8:58:7f:4f:1e:42:c9:bc:36:ba:6d:98:04:c1:f2:
     04:fe:31:cc:04:77:7e:60:7f:c0:6f:88:2e:7a:87:46:e2:f3:
     fd:87:e9:42:a0:08:df:b8:ef:84:cf:35:2b:2a:50:56:b2:1a:
     90:a9:23:5f:e1:27:22:42:83:93:9a:1f:28:82:dc:24:b6:0b:
     f0:6a:c7:29:7b:3d:86:7c:78:5e:60:06:db:d8:27:c0:ee:1c:
     10:5f:aa:6e:7a:1f:c6:86:31:ab:ac:cf:47:ba:68:87:15:1f:
     d0:c8:fa:df:6b:a9:ac:d3:92:9a:8a:ec:30:50:28:4a:8f:84:
     53:9e:f3:fd:c8:62:57:c8:a8:64:d4:14:7f:e2:03:c9:48:22:
     b3:e6:f6:2a:cc:26:d3:0a:99:18:4c:a9:c9:ee:6c:fe:3a:51:
     e3:db:e4:dc:fc:a3:d1:16:b8:ce:70:33:65:bd:b1:9c:79:d5:
     7d:34:d0:6c:ed:9e:37:d6:85:6d:88:a3:b4:cf:41:24:3a:04:
     3d:16:4c:89:b5:10:a7:4c:b5:50:97:d4:00:62:78:3e:cf:dd:
     aa:88:c9:bc:e3:3a:30:e5:31:3a:80:40:ef:75:8f:98:a0:62:
     ac:21:ea:4f:04:bb:1f:9a:97:ec:3b:88:fc:3e:00:a5:e6:18:
     87:02:a5:a2:28:be:4b:f1:55:d4:01:eb:78:24:fc:2a:25:d0:
     f1:e7:fd:93:0d:00:f6:f7:df:4b:dd:31:7b:0e:23:15:f9:85:
     59:3c:6d:e0:bf:c9:b4:22:23:2a:4c:64:32:82:c2:86:a6:85:
     ef:6e:9d:8f:fd:40:51:52:4e:b2:e3:cf:60:2a:c9:cb:16:d7:
     2f:6b:40:ff:a1:fb:d2:91:97:f1:ae:96:df:0c:42:00:75:88:
     ca:9d:10:8e:73:c7:1e:4a:85:40:08:e2:61:da:a4:57:8c:3b:
     27:ea:d2:d6:ad:69:3c:59:1f:0b:18:21:65:03:54:9c:6c:b9:
     2c:b2:e7:d3:b0:aa:07:90:91:24:e7:5b:7f:54:9c:87:1a:f5:
     ce:ec:0c:ba:f6:ea:6e:65:8d:d7:cb:a8:4b:ed:19:a9:36:df:
     f5:1b:f7:0d:d0:e7:e6:9c:4a:1f:c5:65:57:64:13:77:51:1a:
     b3:f6:84:b9:98:d6:aa:99:84:50:3d:b5:ff:99:73:77:cc:9f:
     0f:d9:3b:cd:4f:e7:59:c5:1a:11:43:e2:5b:5f:4c:bb:c4:7b:
     d1:4d:55:46:49:df:38:c3
```

Como se pode observar recorre ao uso de SHA-256 com RSA-2048, sendo que este é suficientemente forte para curto e longo prazo. Caso fosse necessário a assinatura durar 30 ou mais anos, seria necessário usar algo mais forte do que 2048-bit RSA, mas para este caso não é.