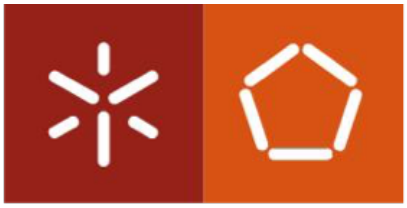


Universidade do Minho
Departamento de Informática
Mestrado Integrado em Engenharia Informática

Engenharia de Segurança: TP2

Aula 3

Diana Lopes, nº a74944
Gabriela Vaz, nº a74899

Conteúdo

Capítulo 1

Assinaturas cegas (*Blind signatures*)

baseadas no Elliptic Curve Discrete Logarithm Problem (ECDLP)

1.1 E1.1

Esta experiência tem como objetivo gerar um par de chaves e o certificado, utilizando o openssl.

- openssl ecparam -name prime256v1 -genkey -noout -out key.pem
- openssl req -key key.pem -new -x509 -days 365 -out key.crt

1.2 E1.2

- **Inicialização:** python initSigner-app.py
- **Ofuscação:** python generateBlindData-app.py
- **Assinatura:** python generateBlindSignature-app.py key.pem
- **Desofuscação:** python unblindSignature-app.py
- **Verificação:** python verifySignature-app.py key.crt

1.3 P1.1

Alteramos todas as funções de acordo com o que era exigido no enunciado, como se pode verificar nos apêndices.

Capítulo 2

Protocolo SSL/TLS

2.1 P2.1

- Grupo 2 - Escolha quatro sites de Universidades Europeias, não Portuguesas.
  i. Anexe os resultados do *SSL Server test* à sua resposta.

  1. University of Glasgow(Anexo A)
  2. Imperial College London (Anexo B)
  3. LMU Munich (Anexo C)
  4. The University of Edinburgh (Anexo D)

  ii. Analise o resultado do *SSL Server test* relativo ao site escolhido com pior rating. Que comentários pode fazer sobre a sua segurança. Porquê?

  O pior rating encontrado foi o da Imperial College London (Anexo B).
  O seu rating tem este valor devido ao facto de permitir várias cifras fraca e até algumas inseguras (usando o RC4 como cifra insegura). Para além disto, é também indicada uma vulnerabilidade *Return Of Bleichenbacher's Oracle Threat (ROBOT)*, avisando que o seu nível de segurança pode mesmo baixar para **F** no caso de não a solucionarem.

  iii. É natural que tenha sido confrontado com a seguinte informação: *"HTTP Strict Transport Security (HSTS) with long duration deployed on this server."*. O que significa, para efeitos práticos?

  O HTTPS serve para garantir o tráfego para um site, tornando muito difícil para um atacanre intercetar, modificar ou falsificar o tráfego entre um usuário e o site. Posto isto, se esta a demorar demasiado tempo a ser implantado, quando visitamos um dado site, a nossa máquina vai estar mais vulnerável a ataques *main-in-the-midle*.

Capítulo 3

Protocolo SSH

Inicialmente, procedeu-se à instalação do **ssh-audit**, executando os comandos que se apresentam de seguida.

- cd
- mkdir Tools
- cd Tools
- git clone
- https://github.com/arthepsy/ssh-audit
- cd ssh-audit
- python ssh-audit.py

3.1 E3.1

Foi pedido, a título de experiência, que se executasse o comando

- python ssh-audit.py algo.paranoidjasmine.com

Pode consultar-se, na Figura 3.1, o resultado da execução deste comando.

3.2 P3.1 (Grupo 2)

Foi criada uma conta em `https://www.shodan.io/`, de forma a fazer pesquisas sobre serviços disponíveis na Web. Este grupo (grupo 2) tem como objetivo escolher **dois servidores ssh de Universidades Europeias, não Portuguesas**, e dar resposta aos seguintes pontos:

1. Anexe os resultados do ssh-audit à sua resposta.

```
user@CSI:~/Tools/ssh-audit$ python ssh-audit.py algo.paranoidjasmine.com
# general
(gen) banner: SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u2
(gen) software: OpenSSH 7.4p1
(gen) compatibility: OpenSSH 7.3+, Dropbear SSH 2016.73+
(gen) compression: enabled (zlib@openssh.com)

# key exchange algorithms
(kex) curve25519-sha256              -- [warn] unknown algorithm
(kex) curve25519-sha256@libssh.org   -- [info] available since OpenSSH 6.5, Dropbear SSH 2013.62
(kex) diffie-hellman-group16-sha512  -- [info] available since OpenSSH 7.3, Dropbear SSH 2016.73
(kex) diffie-hellman-group18-sha512  -- [info] available since OpenSSH 7.3
(kex) diffie-hellman-group14-sha256  -- [info] available since OpenSSH 7.3, Dropbear SSH 2016.73

# host-key algorithms
(key) ssh-rsa                        -- [info] available since OpenSSH 2.5.0, Dropbear SSH 0.28
(key) rsa-sha2-512                   -- [info] available since OpenSSH 7.2
(key) rsa-sha2-256                   -- [info] available since OpenSSH 7.2

# encryption algorithms (ciphers)
(enc) chacha20-poly1305@openssh.com  -- [info] available since OpenSSH 6.5
                                     `- [info] default cipher since OpenSSH 6.9.
(enc) aes128-ctr                     -- [info] available since OpenSSH 3.7, Dropbear SSH 0.52
(enc) aes192-ctr                     -- [info] available since OpenSSH 3.7
(enc) aes256-ctr                     -- [info] available since OpenSSH 3.7, Dropbear SSH 0.52
(enc) aes128-gcm@openssh.com         -- [info] available since OpenSSH 6.2
(enc) aes256-gcm@openssh.com         -- [info] available since OpenSSH 6.2

# message authentication code algorithms
(mac) umac-128-etm@openssh.com       -- [info] available since OpenSSH 6.2
(mac) hmac-sha2-256-etm@openssh.com  -- [info] available since OpenSSH 6.2
(mac) hmac-sha2-512-etm@openssh.com  -- [info] available since OpenSSH 6.2

# algorithm recommendations (for OpenSSH 7.4)
(rec) +ssh-ed25519                   -- key algorithm to append
```

Figura 3.1: Resultado da execução de *python ssh-audit.py algo.paranoidjasmine.com*.

2. Indique o software e versão utilizada pelos servidores ssh.
3. Qual dessas versões de software tem mais vulnerabilidades?
4. E qual tem a vulnerabilidade mais grave (de acordo com o CVSS score identificado no CVE details)?
5. Para efeitos práticos, a vulnerabilidade indicada no ponto anterior é grave? Porquê?

Foram escolhidas as universidades de Ioannina (`http://www.uoi.gr/`) e de São Petersburgo (`http://spbu.ru/`). Em resposta ao **ponto 1**, os resultados do ssh-audit da universidade de Ioannina e da universidade de São Petersburgo podem ser consultados nos ficheiros **Ioannina.txt** e **SP.txt** que se encontram no GitHub.
Para facilitar a escrita e compreensão, consideremos que o **servidor ssh 1** diz respeito ao servidor ssh da universidade de Ioannina e o **servidor ssh 2** diz respeito ao servidor ssh da universidade de São Petersburgo.

Ponto 2

- Software usado pelo **servidor ssh 1**: OpenSSH 6.7p1
- Software usado pelo **servidor ssh 2**: OpenSSH 5.3

Ponto 3

Em `https://www.cvedetails.com/vulnerability-list/vendor_id-97/product_id-585/version_id-188833/Openbsd-Openssh-6.7.html` vemos que a versão 6.7p1 apresenta 5 vulnerabilidades. Em `https://www.cvedetails.com/vulnerability-list/vendor_id-97/product_id-585/version_id-121223/Openbsd-Openssh-5.3.html` vemos que a versão 5.3 apresenta 10 vulnerabilidades.

Ponto 4

Nas Figuras 3.2 e 3.3 podemos ver as vulnerabilidades de OpenSSH 6.7p1 e de OpenSSH 5.3, respetivamente.

Openbsd » Openssh » 6.7 P1 : Security Vulnerabilities

Cpe Name:*cpe:/a:openbsd:openssh:6.7:p1*
CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending
Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2016-8858	399		DoS	2016-12-09	2018-02-03	7.8	None	Remote	Low	Not required	None	None	Complete
	** DISPUTED ** The kex_input_kexinit function in kex.c in OpenSSH 6.x and 7.x through 7.3 allows remote attackers to cause a denial of service (memory consumption) by sending many duplicate KEXINIT requests. NOTE: a third party reports that "OpenSSH upstream does not consider this as a security issue."													
2	CVE-2016-10708	476		DoS	2018-01-21	2018-02-08	5.0	None	Remote	Low	Not required	None	None	Partial
	sshd in OpenSSH before 7.4 allows remote attackers to cause a denial of service (NULL pointer dereference and daemon crash) via an out-of-sequence NEWKEYS message, as demonstrated by Honggfuzz, related to kex.c and packet.c.													
3	CVE-2017-15906	275			2017-10-25	2018-01-31	5.0	None	Remote	Low	Not required	None	Partial	None
	The process_open function in sftp-server.c in OpenSSH before 7.6 does not properly prevent write operations in readonly mode, which allows attackers to create zero-length files.													
4	CVE-2016-0778	119		DoS Overflow	2016-01-14	2017-02-16	4.6	None	Remote	High	Single system	Partial	Partial	Partial
	The (1) roaming_read and (2) roaming_write functions in roaming_common.c in the client in OpenSSH 5.x, 6.x, and 7.x before 7.1p2, when certain proxy and forward options are enabled, do not properly maintain connection file descriptors, which allows remote servers to cause a denial of service (heap-based buffer overflow) or possibly have unspecified other impact by requesting many forwardings.													
5	CVE-2016-0777	200		+Info	2016-01-14	2017-11-20	4.0	None	Remote	Low	Single system	Partial	None	None
	The resend_bytes function in roaming_common.c in the client in OpenSSH 5.x, 6.x, and 7.x before 7.1p2 allows remote servers to obtain sensitive information from process memory by requesting transmission of an entire buffer, as demonstrated by reading a private key.													

Total number of vulnerabilities : 5 Page : 1 (This Page)

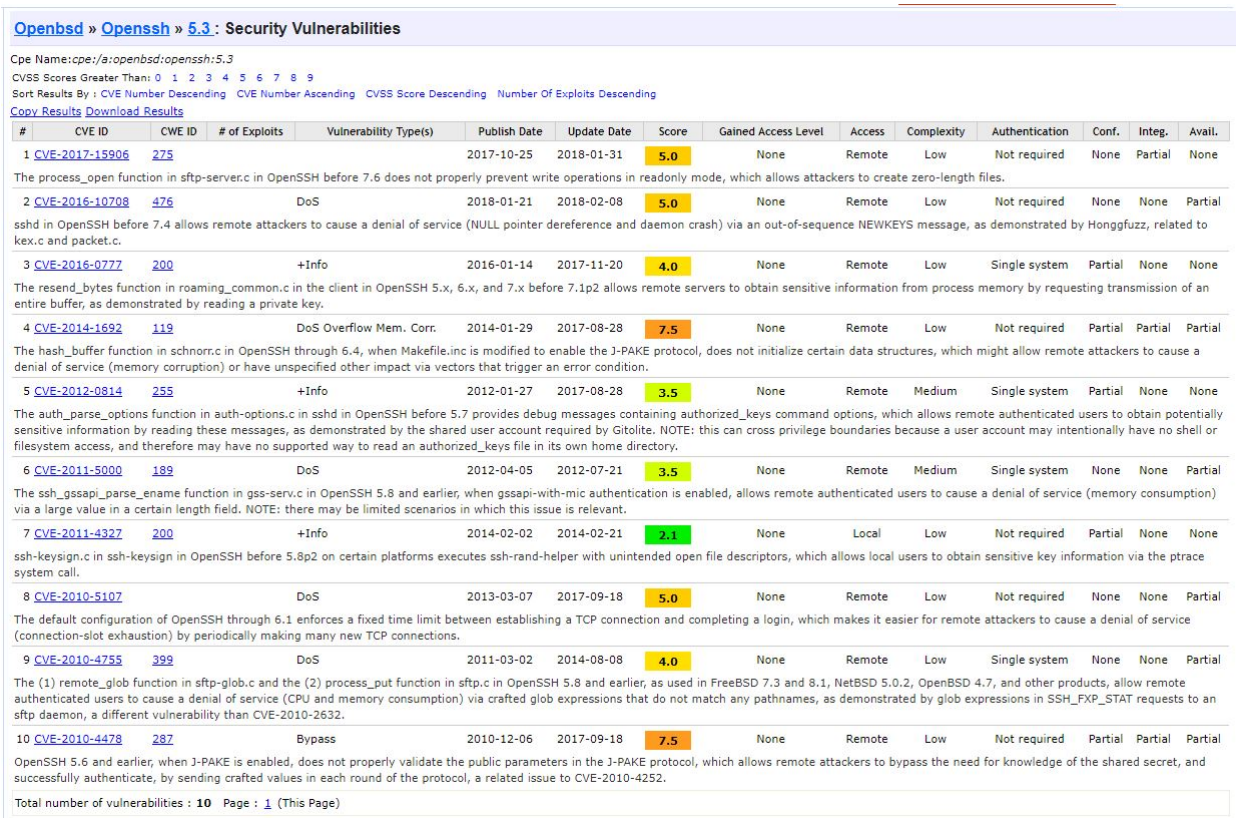
Figura 3.2: Vulnerabilidades de OpenSSH 6.7p1.

Na Figura 3.4 podemos consultar a vulnerabilidade mais grave, com *score* de 7.8.

Ponto 5

Podemos consultar, na Figura 3.5, a vulnerabilidade mais grave apresentada em detalhe.

A vulnerabilidade apresentada não tem implicações na confidencialidade nem na integridade do sistema. No entanto, não é necessário muito conhecimento para explorar esta vulnerabilidade, que consiste no facto de atacantes remotos poderem negar o serviço. Apesar do seu *score*, esta vulnerabilidade não foi considerada um problema de segurança.



Openbsd » Openssh » 5.3 : Security Vulnerabilities

Cpe Name:*cpe:/a:openbsd:openssh:5.3*

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2017-15906	275			2017-10-25	2018-01-31	5.0	None	Remote	Low	Not required	None	Partial	None
The process_open function in sftp-server.c in OpenSSH before 7.6 does not properly prevent write operations in readonly mode, which allows attackers to create zero-length files.														
2	CVE-2016-10708	476		DoS	2018-01-21	2018-02-08	5.0	None	Remote	Low	Not required	None	None	Partial
sshd in OpenSSH before 7.4 allows remote attackers to cause a denial of service (NULL pointer dereference and daemon crash) via an out-of-sequence NEWKEYS message, as demonstrated by Honggfuzz, related to kex.c and packet.c.														
3	CVE-2016-0777	200		+Info	2016-01-14	2017-11-20	4.0	None	Remote	Low	Single system	Partial	None	None
The resend_bytes function in roaming_common.c in the client in OpenSSH 5.x, 6.x, and 7.x before 7.1p2 allows remote servers to obtain sensitive information from process memory by requesting transmission of an entire buffer, as demonstrated by reading a private key.														
4	CVE-2014-1692	119		DoS Overflow Mem. Corr.	2014-01-29	2017-08-28	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
The hash_buffer function in schnorr.c in OpenSSH through 6.4, when Makefile.inc is modified to enable the J-PAKE protocol, does not initialize certain data structures, which might allow remote attackers to cause a denial of service (memory corruption) or have unspecified other impact via vectors that trigger an error condition.														
5	CVE-2012-0814	255		+Info	2012-01-27	2017-08-28	3.5	None	Remote	Medium	Single system	Partial	None	None
The auth_parse_options function in auth-options.c in sshd in OpenSSH before 5.7 provides debug messages containing authorized_keys command options, which allows remote authenticated users to obtain potentially sensitive information by reading these messages, as demonstrated by the shared user account required by Gitolite. NOTE: this can cross privilege boundaries because a user account may intentionally have no shell or filesystem access, and therefore may have no supported way to read an authorized_keys file in its own home directory.														
6	CVE-2011-5000	189		DoS	2012-04-05	2012-07-21	3.5	None	Remote	Medium	Single system	None	None	Partial
The ssh_gssapi_parse_ename function in gss-serv.c in OpenSSH 5.8 and earlier, when gssapi-with-mic authentication is enabled, allows remote authenticated users to cause a denial of service (memory consumption) via a large value in a certain length field. NOTE: there may be limited scenarios in which this issue is relevant.														
7	CVE-2011-4327	200		+Info	2014-02-02	2014-02-21	2.1	None	Local	Low	Not required	Partial	None	None
ssh-keysign.c in ssh-keysign in OpenSSH before 5.8p2 on certain platforms executes ssh-rand-helper with unintended open file descriptors, which allows local users to obtain sensitive key information via the ptrace system call.														
8	CVE-2010-5107			DoS	2013-03-07	2017-09-18	5.0	None	Remote	Low	Not required	None	None	Partial
The default configuration of OpenSSH through 6.1 enforces a fixed time limit between establishing a TCP connection and completing a login, which makes it easier for remote attackers to cause a denial of service (connection-slot exhaustion) by periodically making many new TCP connections.														
9	CVE-2010-4755	399		DoS	2011-03-02	2014-08-08	4.0	None	Remote	Low	Single system	None	None	Partial
The (1) remote_glob function in sftp-glob.c and the (2) process_put function in sftp.c in OpenSSH 5.8 and earlier, as used in FreeBSD 7.3 and 8.1, NetBSD 5.0.2, OpenBSD 4.7, and other products, allow remote authenticated users to cause a denial of service (CPU and memory consumption) via crafted glob expressions that do not match any pathnames, as demonstrated by glob expressions in SSH_FXP_STAT requests to an sftp daemon, a different vulnerability than CVE-2010-2632.														
10	CVE-2010-4478	287		Bypass	2010-12-06	2017-09-18	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
OpenSSH 5.6 and earlier, when J-PAKE is enabled, does not properly validate the public parameters in the J-PAKE protocol, which allows remote attackers to bypass the need for knowledge of the shared secret, and successfully authenticate, by sending crafted values in each round of the protocol, a related issue to CVE-2010-4252.														

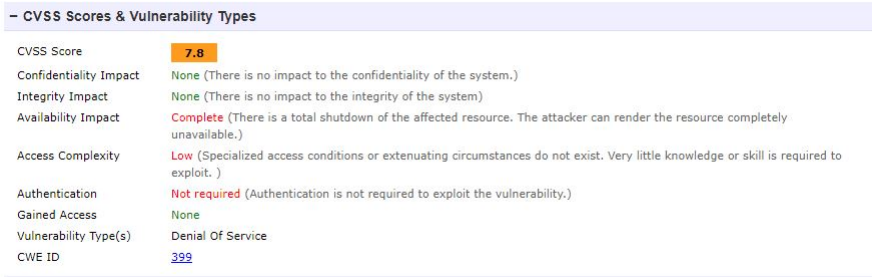
Total number of vulnerabilities : **10** Page : 1 (This Page)

Figura 3.3: Vulnerabilidades de OpenSSH 5.3.

3	CVE-2016-8858	399		DoS	2016-12-09	2018-02-03	7.8	None	Remote	Low	Not required	None	None	Complete

** DISPUTED ** The kex_input_kexinit function in kex.c in OpenSSH 6.x and 7.x through 7.3 allows remote attackers to cause a denial of service (memory consumption) by sending many duplicate KEXINIT requests. NOTE: a third party reports that "OpenSSH upstream does not consider this as a security issue."

Figura 3.4: Vulnerabilidade mais grave.



– CVSS Scores & Vulnerability Types

CVSS Score	7.8
Confidentiality Impact	None (There is no impact to the confidentiality of the system.)
Integrity Impact	None (There is no impact to the integrity of the system)
Availability Impact	Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Denial Of Service
CWE ID	399

Figura 3.5: Vulnerabilidade mais grave em detalhe.

Apêndice A

Inicialização - *init-app.py*

```
1  import sys
2  from eVotUM.Cripto import eccblind
3
4  def printUsage():
5      print("Usage: python init-app.py or python init-app.py -init
       ")
6
7  def parseArgs():
8      if (len(sys.argv) > 1):
9          if(sys.argv[1]=="-init"):
10             main()
11         else:
12             printUsage()
13     else:
14         imprimeR()
15
16 def imprimeR():
17     initComponents, pRDashComponents = eccblind.initSigner()
18     print (pRDashComponents);
19
20 def main():
21     initComponents, pRDashComponents = eccblind.initSigner()
22     file = open("assinante.txt", "w")
23     file.write(initComponents)
24     file.write("\n")
25     file.write(pRDashComponents)
26     file.write("\n")
27     file.close()
28
29 if __name__ == "__main__":
30     parseArgs()
```

Apêndice B

Ofuscação - *ofusca-app.py*

```
import sys
from eVotUM.Cripto import eccblind


def printUsage():
    print("Usage: python ofusca-app.py -msg <mensagem a assinar>
        -RDash <pRDashComponents>")

def parseArgs():
    if (len(sys.argv) < 5):
        printUsage()
    else:
        if(sys.argv[1]=="-msg" and sys.argv[3]=="-RDash"):
            main()
        else:
            printUsage()

def showResults(errorCode, result):
    print("Output")
    if (errorCode is None):
        blindComponents, pRComponents, blindM = result
        print("Blind message: %s" % blindM)
        file = open("requerente.txt", "w")
        file.write(blindComponents)
        file.write("\n")
        file.write(pRComponents)
        file.write("\n")
    elif (errorCode == 1):
        print("Error: pRDash components are invalid")

def main():
    print("Input")
    data = sys.argv[2]
    pRDashComponents = sys.argv[4]
```

```
34      errorCode, result = eccblind.blindData(pRDashComponents,
            data)
35      showResults(errorCode, result)
36
37  if __name__ == "__main__":
38      parseArgs()
```

Apêndice C

Assinatura - *blindSignature-app.py*

```
from eVotUM.Cripto import utils
import sys
from eVotUM.Cripto import eccblind

def printUsage():
    print("Usage: python blindSignature-app.py -key <chave
        privada> -bmsg <Blind message>")

def parseArgs():
    if (len(sys.argv) < 5):
        printUsage()
    else:
        if(sys.argv[1]=="-key" and sys.argv[3]=="-bmsg"):
            eccPrivateKeyPath = sys.argv[2]
            main(eccPrivateKeyPath)
        else:
            printUsage()

def showResults(errorCode, blindSignature):
    print("Output")
    if (errorCode is None):
        print("Blind signature: %s" % blindSignature)
    elif (errorCode == 1):
        print("Error: it was not possible to retrieve the
            private key")
    elif (errorCode == 2):
        print("Error: init components are invalid")
    elif (errorCode == 3):
        print("Error: invalid blind message format")

def main(eccPrivateKeyPath):
    pemKey = utils.readFile(eccPrivateKeyPath)
    print("Input")
```

```
32      passphrase = raw_input("Passphrase: ")
33      blindM = sys.argv[4]
34      file = open("assinante.txt", "r")
35      initComponents = file.readline()[:-1]
36      errorCode, blindSignature = eccblind.generateBlindSignature(
            pemKey, passphrase, blindM, initComponents)
37      showResults(errorCode, blindSignature)
38
39  if __name__ == "__main__":
40      parseArgs()
```

Apêndice D

Desofuscação - *desofusca-app.py*

```
1  import sys
2  from eVotUM.Cripto import eccblind
3
4
5  def printUsage():
6      print("Usage: python desofusca-app.py -s <Blind Signature> -
           RDash <pRDashComponents>")
7
8  def parseArgs():
9      if (len(sys.argv) < 5):
10         printUsage()
11     else:
12         if(sys.argv[1]=="-s" and sys.argv[3]=="-RDash"):
13             main()
14         else:
15             printUsage()
16
17 def showResults(errorCode, signature):
18     print("Output")
19     if (errorCode is None):
20         print("Signature: %s" % signature)
21     elif (errorCode == 1):
22         print("Error: pRDash components are invalid")
23     elif (errorCode == 2):
24         print("Error: blind components are invalid")
25     elif (errorCode == 3):
26         print("Error: invalid blind signature format")
27
28 def main():
29     print("Input")
30     file = open("requerente.txt", "r")
31     blindSignature = sys.argv[2]
32     blindComponents = file.readline()[:-1]
33     pRDashComponents = sys.argv[4]
```

```
34      errorCode, signature = eccblind.unblindSignature(
            blindSignature, pRDashComponents, blindComponents)
35      showResults(errorCode, signature)
36
37  if __name__ == "__main__":
38      parseArgs()
```

Apêndice E

Verificação - *verify-app.py*

```
1 import sys
2 from eVotUM.Cripto import eccblind
3 from eVotUM.Cripto import utils
4
5 def printUsage():
6     print("Usage: python verify-app.py -cert <certificado do
          assinante> -msg <mensagem original a assinar> -sDash <
          Signature> -f <ficheiro do requerente>")
7
8 def parseArgs():
9     if (len(sys.argv) < 9):
10         printUsage()
11     else:
12         if(sys.argv[1]=="-cert" and sys.argv[3]=="-msg" and sys.
              argv[5]=="-sDash" and sys.argv[7]=="-f"):
13             main()
14         else:
15             printUsage()
16
17 def showResults(errorCode, validSignature):
18     print("Output")
19     if (errorCode is None):
20         if (validSignature):
21             print("Valid signature")
22         else:
23             print("Invalid signature")
24     elif (errorCode == 1):
25         print("Error: it was not possible to retrieve the public
              key")
26     elif (errorCode == 2):
27         print("Error: pR components are invalid")
28     elif (errorCode == 3):
29         print("Error: blind components are invalid")
30     elif (errorCode == 4):
31         print("Error: invalid signature format")
```

```
32
33 def main():
34     pemPublicKey = utils.readFile(sys.argv[2])
35     print("Input")
36     data = sys.argv[4]
37     signature = sys.argv[6]
38     ficheiro = sys.argv[8]
39     file = open(ficheiro, "r")
40     blindComponents = file.readline()[:-1]
41     pRComponents = file.readline()[:-1]
42     errorCode, validSignature = eccblind.verifySignature(
           pemPublicKey, signature, blindComponents, pRComponents,
           data)
43     showResults(errorCode, validSignature)
44
45 if __name__ == "__main__":
46     parseArgs()
```