Universidade do Minho

Mestrado Integrado em Engenharia Informática

# Engenharia de Segurança

**Trabalho TP2**

**Grupo 2**
Paulo Gameiro - A72067
Pedro Rodrigues - PG41092
Rafaela Soares - A79034

Braga, Portugal
9 de Março de 2020

# Conteúdo

# 1 Exercício 1:

## 1.1 Pergunta P1.1

Nesta pergunta pretende-se alterar o código fornecido para a experiência 1.2, de forma a simplificar o *input* e *output*.

### 1.1.1 Assinante

Para o ficheiro init-app.py é pretendido que este retorne o R' utilizado no processo, que corresponde ao ponto da curva aleatoriamente escolhido para cada pedido e, que através da execução de python init-app.py -init, os componentes initComponents e pRDashComponents sejam calculados e guardados num ficheiro sig.settings.

Posto isto, o utilizador primeiramente tem de correr o comando: python init-app.py -init para que o estado interno seja criado com o initComponents e pRDashComponents (R'). De seguida, deve correr o comando: python init-app.py e este irá ler do ficheiro sig.settings o componente R' e irá imprimi-lo no ecrã.

```
1
2  import sys
3  from eVotUM.Cripto import eccblind
4
5  settings_file = "./sig.settings"
6  initComponents = ""
7  pRDashComponents = ""
8
9  def printUsage():
10     print("Usage: python init.py")
11
12 def init():
13     initComponents, pRDashComponents = eccblind.initSigner()
14     f = open(settings_file, "w")
15     f.write(initComponents + "\n" + pRDashComponents) # Guarda as componentes no
       ficheiro settings
16
17 def load_settings():
18     f = open(settings_file, "r")
19     initComponents = f.readline()
20     pRDashComponents = f.readline()
21     return initComponents, pRDashComponents
22
23 def parseArgs():
24     if len(sys.argv) == 1: # Programa inicializado sem a op   o -init
25         initComponents,pRDashComponents = load_settings()
26         print("pRDashComponents: %s" % pRDashComponents)
27     elif sys.argv[1] == "-init": # Quando inicializado com o -init
28         init()
29     else:
30         printUsage()
31
32 if __name__ == "__main__":
33     parseArgs()
```

O ficheiro blindSignature-app.py é o que contém o código necessário para realizar a assinatura cega, para além disso é pretendido que sejam adicionadas as opções -key <chave_privada>, que permitirá o uso de um ficheiro com a chave privada do assinante e -bmsg blind-message, que corresponderá à mensagem que se pretende assinar, acabando por retornar ao utilizador a BlindSignature(s).

```
1 from eVotUM.Cripto import utils
2 import sys
3 from eVotUM.Cripto import eccblind
4
```

```
settings_file = "./sig.settings"

def printUsage():
    print("Usage: python blindSignature.py -key <private-key.pem> -bmsg <Blind
    message>")

def load_settings():
    f = open(settings_file, "r")
    initComponents = f.readline()
    pRDashComponents = f.readline()
    return initComponents, pRDashComponents

def parseArgs():
    if len(sys.argv) == 5 and sys.argv[1] == "-key" and sys.argv[3] == "-bmsg":
        eccPrivateKeyPath = sys.argv[2] # Caminho para a chave privada
        msg = sys.argv[4] # Blind Message
        main(eccPrivateKeyPath, msg)
    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, blindM):
    initComponents, pRDashComponents = load_settings() # Carrega do ficheiro os
    componentes
    pemKey = utils.readFile(eccPrivateKeyPath)
    print("Input")
    passphrase = raw_input("Passphrase: ")
    errorCode, blindSignature = eccblind.generateBlindSignature(pemKey, passphrase,
     blindM, initComponents)
    showResults(errorCode, blindSignature) # Retorna a Blind Signature

if __name__ == "__main__":
    parseArgs()
```

### 1.1.2 Requerente

O ficheiro ofusca-app.py gera a Blind Message (m'), imprimindo-a no ecrã, e guarda as Blind-Components e pRComponentes num ficheiro do Requerente (req.settings). Este programa tem como opções -msg <MENSAGEM_ORIGINAL> e -RDash <pRDashComponents> que são a mensagem a ofuscar e o R' que é o ponto aleatório da curva elíptica seleccionado na inicialização, respectivamente. Assim, as alterações essenciais ao código foram:

```
import sys
from eVotUM.Cripto import eccblind

settings_file = "./req.settings"

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

def parseArgs():
    if len(sys.argv) == 5 and sys.argv[1] == "-msg" and sys.argv[3] == "-RDash":
        main(sys.argv[2], sys.argv[4])
    else:
        printUsage()
```

```
14
15 def showResults(errorCode, result):
16     print("Output")
17     if (errorCode is None):
18         blindComponents, pRComponents, blindM = result
19         print("Blind message: %s" % blindM) # Imprime a blind message no ecra
20         f = open(settings_file, "w")
21         f.write(blindComponents + "\n" + pRComponents) # Guarda num ficheiro as
       components
22         f.close()
23     elif (errorCode == 1):
24         print("Error: pRDash components are invalid")
25
26 def main(data, pRDashComponents):
27     errorCode, result = eccblind.blindData(pRDashComponents, data)
28     showResults(errorCode, result)
29
30 if __name__ == "__main__":
31     parseArgs()
```

O programa desofusca-app.py gera a signature(s') através da blind signature e do R', em que estes são passados como parâmetros, -s <BlindSignature> e -RDash <pRDashComponents>, sendo estes o resultado da BlindSignature retornada pelo programa blindSignature-app.py e o R' que é o ponto aleatório da curva elítica selecionado na inicialização.

```
1  import sys
2  from eVotUM.Cripto import eccblind
3
4  settings_file = "./req.settings"
5
6  def printUsage():
7      print("Usage: python desofusca.py -s <Blind Signature> -RDash <pRDashComponents
       >")
8
9  def load_settings():
10     f = open(settings_file, "r")
11     blindComponents = f.readline()
12     pRComponents = f.readline()
13     return blindComponents, pRComponents
14
15 def parseArgs():
16     if len(sys.argv) == 5 and sys.argv[1] == "-s" and sys.argv[3] == "-RDash":
17         main(sys.argv[2], sys.argv[4])
18     else:
19         printUsage()
20
21 def showResults(errorCode, signature):
22     print("Output")
23     if (errorCode is None):
24         print("Signature: %s" % signature) # Imprime a assinatura
25     elif (errorCode == 1):
26         print("Error: pRDash components are invalid")
27     elif (errorCode == 2):
28         print("Error: blind components are invalid")
29     elif (errorCode == 3):
30         print("Error: invalid blind signature format")
31
32 def main(blindSignature, pRDashComponents):
33     print("Input")
34     blindComponents, pRComponents = load_settings() # Carrega do ficheiro do
       Requerente as components
35     errorCode, signature = eccblind.unblindSignature(blindSignature,
       pRDashComponents, blindComponents)
36     showResults(errorCode, signature)
37
38 if __name__ == "__main__":
```

```
39      parseArgs()
```

### 1.1.3 Verificador

O programa verify-app.py irá validar se a assinatura obtida através do programa desofusca-app.py é válida ou não. Para esse efeito serão necessários as seguintes opções disponibilizadas:

- -cert <certificado do assinante>
- -msg <mensagem original a assinar>
- -sDash <Signature>
- -f <ficheiro do requerente>

Estas opções são o certificado com a chave pública do Assinante, a mensagem do Requerente sem ser assinada, a assinatura devolvida pelo desofusca-app.py e o ficheiro das settings do requerente (req.settings), respetivamente.

```
1  import sys
2  from eVotUM.Cripto import eccblind
3  from eVotUM.Cripto import utils
4
5  def printUsage():
6      print("Usage: python verify.py -cert <certificado do assinante> -msg <mensagem
       original a assinar> -sDash <Signature> -f <ficheiro do requerente>")
7
8  def load_settings(settings_file):
9      f = open(settings_file, "r")
10     blindComponents = f.readline()
11     pRComponents = f.readline()
12     return blindComponents, pRComponents
13
14 def parseArgs():
15     if len(sys.argv) == 9 and sys.argv[1] == "-cert" and sys.argv[3] == "-msg" and
       sys.argv[5] == "-sDash" and sys.argv[7] == "-f":
16         eccPublicKeyPath = sys.argv[2]
17         main(eccPublicKeyPath, sys.argv[4], sys.argv[6], sys.argv[8])
18     else:
19         printUsage()
20
21 def showResults(errorCode, validSignature):
22     print("Output")
23     if (errorCode is None):
24         if (validSignature):
25             print("Valid signature")
26         else:
27             print("Invalid signature")
28     elif (errorCode == 1):
29         print("Error: it was not possible to retrieve the public key")
30     elif (errorCode == 2):
31         print("Error: pR components are invalid")
32     elif (errorCode == 3):
33         print("Error: blind components are invalid")
34     elif (errorCode == 4):
35         print("Error: invalid signature format")
36
37 def main(eccPublicKeyPath, data, signature, req_file):
38     pemPublicKey = utils.readFile(eccPublicKeyPath) # Leitura do certificado
39     blindComponents, pRComponents = load_settings(req_file) # Leitura das
       componentes
40     errorCode, validSignature = eccblind.verifySignature(pemPublicKey, signature,
       blindComponents, pRComponents, data)
41     showResults(errorCode, validSignature)
```

```
42
43 if __name__ == "__main__":
44     parseArgs()
```

# 2 Exercício 2:

## 2.1 Experiência P2.1

Através da utilização do **SSL Labs** para efetuar o *SSL Server Test* no *site* **Governo Português**, foi possível obter informações em termos de segurança relativas ao nível dos protocolos e certificados utilizados pelo *site*.

O SSL Labs utiliza um *rating* para classificar estes *sites*, sendo que o utilizado neste exemplo tem um *rating* de A+, que representa o nível máximo.

Apresenta também uma funcionalidade experimental relativa à nova versão do TLS (1.3), e o *Server Name Indication* (SNI), que é uma extensão do TLS que permite ao utilizador "escolher"qual o *hostname* a que está a tentar aceder durante o *handshake*, ou seja, o *server* tem a possibilidade de apresentar diferentes certificados, consoante a página que for acedida.

Estes diferentes certificados podem ser visualizados no SSL Labs, abaixo de um pequeno resumo sobre os dados recolhidos acerca do *site* que foi monitorizado.

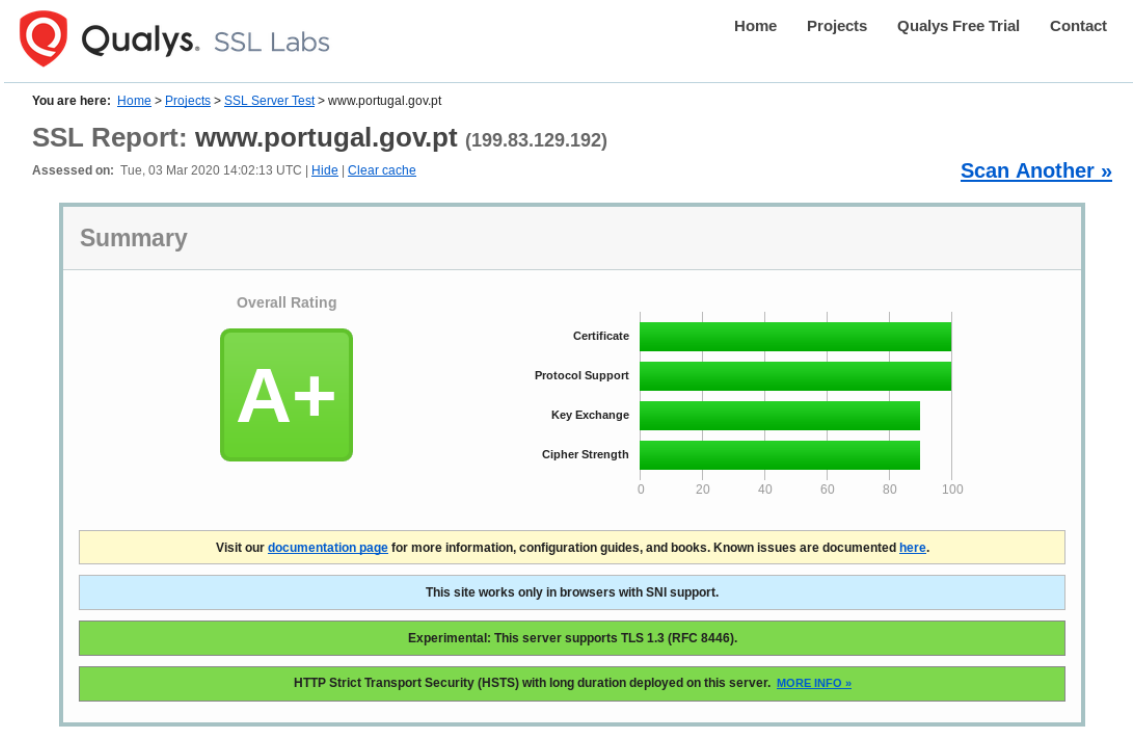



Figura 1: *SSL Server Test* no *site* **Governo Português**

## 2.2 Pergunta P2.1

Para a realização deste exercício foram escolhidas duas universidades europeias para que os seus sites fossem analisados repetindo a experiência realizada anteriormente, sendo que essas universidades foram a **Universidade de Amsterdão** e a **Universidade de Bolohna**.

Os resultados completos dos testes poderão ser visualizados na pasta criada neste repositório de nome **"Resultados SSL Universidades"**.

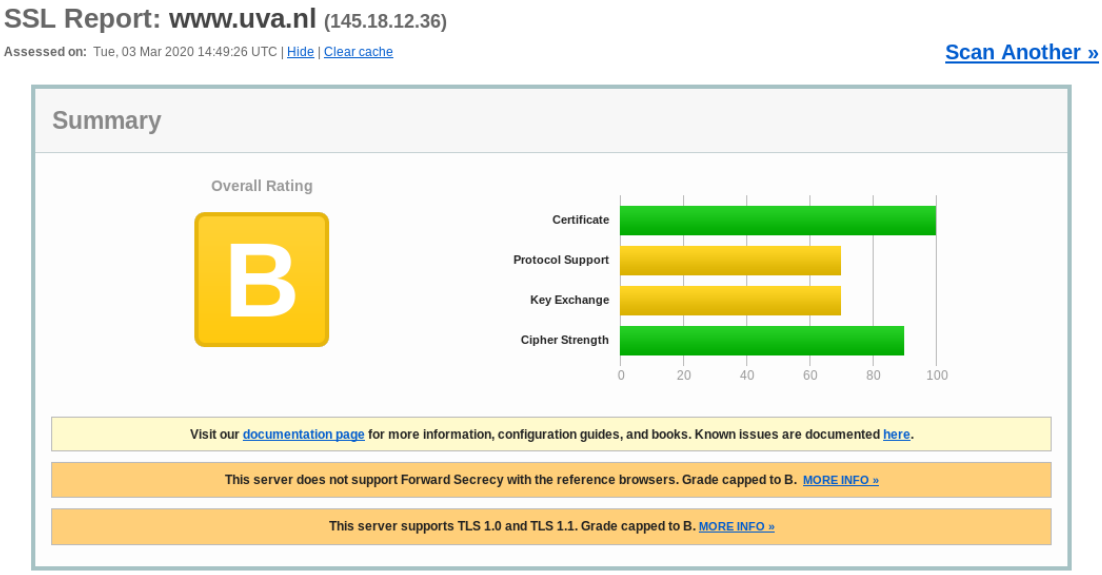



Figura 2: *SSL Server Test* relativo *site* com pior *rating* **Universidade de Amsterdão**

Após a análise dos resultados, foi possível observar que o *site* com o pior *rating* é o da Universidade de Amsterdão (www.uva.nl/en). Apesar de ambos terem ***rating* B**, este ao contrário do da Universidade de Bolonha, não contém ***Forward Secrecy***, ou seja, no caso das chaves serem comprometidas, todas as chaves de sessões passadas estão também comprometidas.

Relativamente à versão do TLS, o *site* utilizado pela Universidade encontra-se ainda a utilizar a versão 1.0 e 1.1, que vão brevemente ser removidos dos *browsers*, tendo em conta que não há formas de resolver os problemas existentes no SSL nessas versões e, portanto, é requerido que os servidores contenham a versão do TLS mais recente possível.

***"HTTP Strict Transport Security (HSTS) with long duration deployed on this server."***

O *HTTP Strict Transport Security* (HSTS) é um mecanismo utilizado em páginas *web* que ajuda na proteção contra ataques do género *Man-In-The-Middle* (MITM), porque permite que os servidores garantam que os *browsers* podem apenas interagir com o *site* utilizando ligações HTTPS, que contém TLS/SSL, impossibilitando, assim, os atacantes de forçarem a ligação através de HTTP.

O aviso apresentado significa que o método está a demorar mais tempo do que o usual para ser executado, ou seja, abre espaço para que o ataque MITM ocorra.

# 3 Exercício 3:

## 3.1 Pergunta P3.1

Em primeira instância, como sugerido no enunciado, configurou-se uma conta em **Shodan**. De seguida, no intento da escolha de dois servidores ssh de Universidades Europeias, não Portuguesas, realizou-se, neste *site*, as seguintes pesquisas: **port:22 org:"University of Cambridge"** e **port:22 org:"University of Manchester"**.

Importante referir que o método de seleção dos servidores baseou-se na escolha do primeiro servidor apresentado como resultado de pesquisa - **koha.dar.cam.ac.uk** (Universidade de *Cambridge*) e **winssh.physics.ox.ac.uk** (Universidade de *Manchester*) - como pode ser evidenciado nas imagens abaixo.

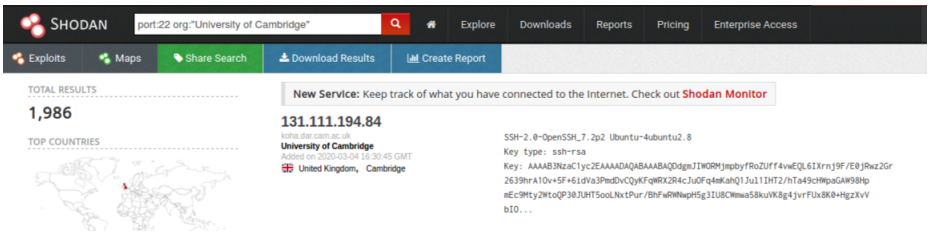



Figura 3: Servidor Universidade de *Cambridge*

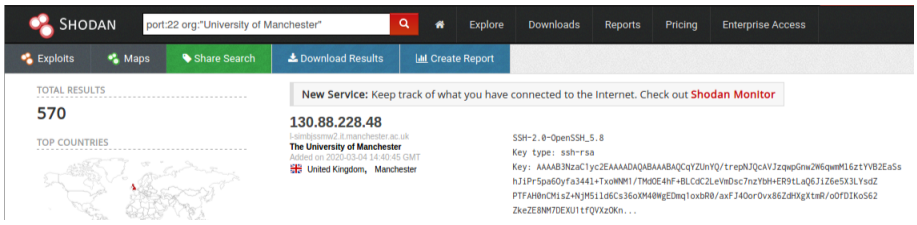



Figura 4: Servidor Universidade de *Manchester*

Relativamente aos resultados da execução do ssh-audit perante os dois servidores, estes foram os seguintes:


Figura 5: Resultado da execução do ssh-audit perante o servidor da Universidade de *Cambridge*

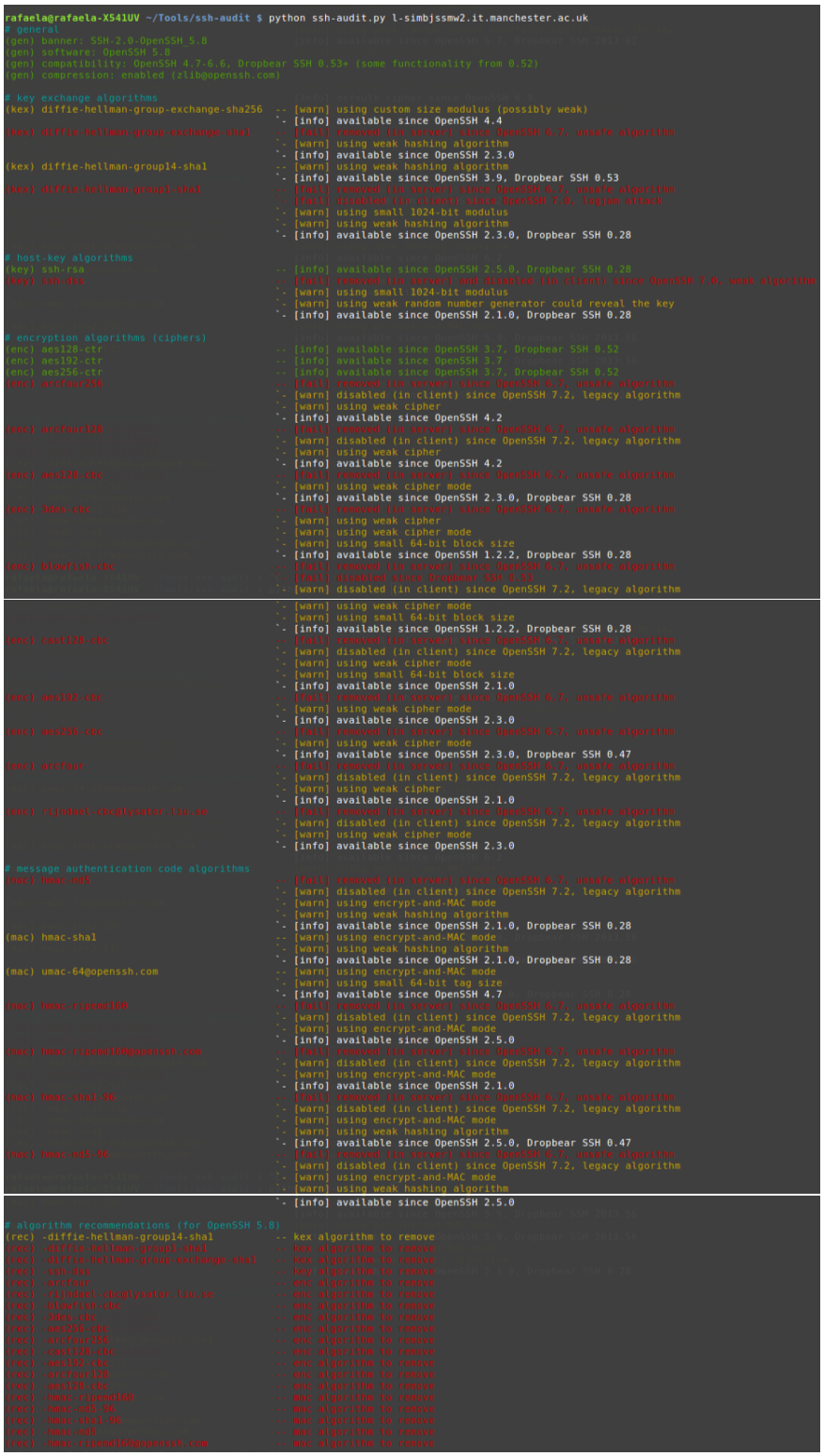
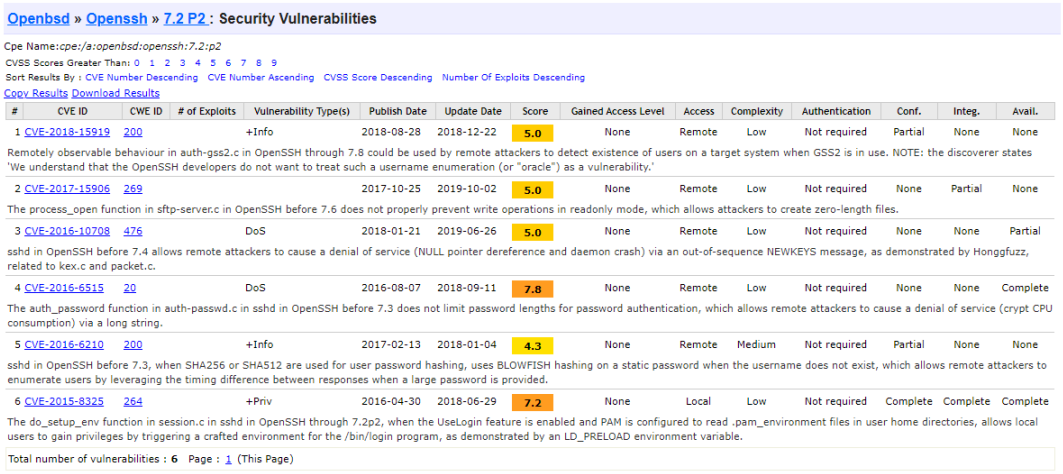



Figura 6: Resultado da execução do ssh-audit perante o servidor da Universidade de *Manchester*

Através destes, pode-se inferir que o *software* e correspondente versão utilizada pelos servidores são:

- **servidor koha.dar.cam.ac.uk (Universidade de *Cambridge*)**: OpenSSH 7.2p2
- **servidor winssh.physics.ox.ac.uk (Universidade de *Manchester*)**: OpenSSH 5.8

Destas versões de *software*, a que apresenta um maior número de vulnerabilidades, de acordo com **CVE Details**, é a OpenSSH 5.8, como pode ser verificado nas figuras abaixo.



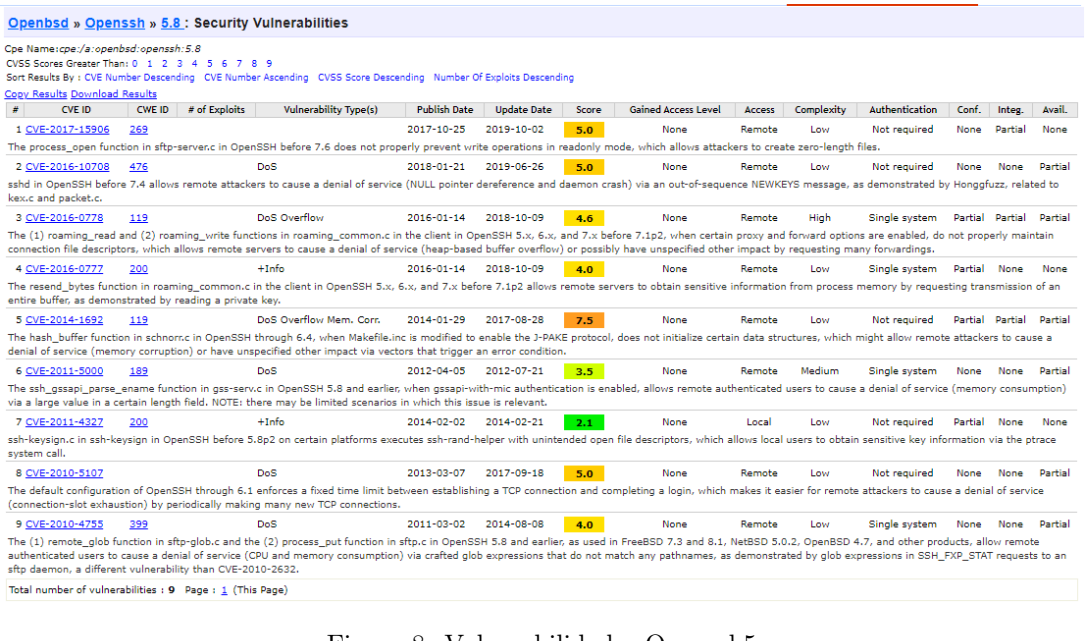
Openbsd » Openssh » 7.2 P2 : Security Vulnerabilities

Cpe Name:*cpe:/a:openbsd:openssh:7.2:p2*
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-2018-15919 | 200 | | +Info | 2018-08-28 | 2018-12-22 | 5.0 | None | Remote | Low | Not required | Partial | None | None |
| Remotely observable behaviour in auth-gss2.c in OpenSSH through 7.8 could be used by remote attackers to detect existence of users on a target system when GSS2 is in use. NOTE: the discoverer states 'We understand that the OpenSSH developers do not want to treat such a username enumeration (or "oracle") as a vulnerability.' | | | | | | | | | | | | | | |
| 2 | CVE-2017-15906 | 269 | | | 2017-10-25 | 2019-10-02 | 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. | | | | | | | | | | | | | | |
| 3 | CVE-2016-10708 | 476 | | DoS | 2018-01-21 | 2019-06-26 | 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. | | | | | | | | | | | | | | |
| 4 | CVE-2016-6515 | 20 | | DoS | 2016-08-07 | 2018-09-11 | 7.8 | None | Remote | Low | Not required | None | None | Complete |
| The auth_password function in auth-passwd.c in sshd in OpenSSH before 7.3 does not limit password lengths for password authentication, which allows remote attackers to cause a denial of service (crypt CPU consumption) via a long string. | | | | | | | | | | | | | | |
| 5 | CVE-2016-6210 | 200 | | +Info | 2017-02-13 | 2018-01-04 | 4.3 | None | Remote | Medium | Not required | Partial | None | None |
| sshd in OpenSSH before 7.3, when SHA256 or SHA512 are used for user password hashing, uses BLOWFISH hashing on a static password when the username does not exist, which allows remote attackers to enumerate users by leveraging the timing difference between responses when a large password is provided. | | | | | | | | | | | | | | |
| 6 | CVE-2015-8325 | 264 | | +Priv | 2016-04-30 | 2018-06-29 | 7.2 | None | Local | Low | Not required | Complete | Complete | Complete |
| The do_setup_env function in session.c in sshd in OpenSSH through 7.2p2, when the UseLogin feature is enabled and PAM is configured to read .pam_environment files in user home directories, allows local users to gain privileges by triggering a crafted environment for the /bin/login program, as demonstrated by an LD_PRELOAD environment variable. | | | | | | | | | | | | | | |

Total number of vulnerabilities : 6 Page : 1 (This Page)

Figura 7: Vulnerabilidades Openssh 7.2p2



Openbsd » Openssh » 5.8 : Security Vulnerabilities

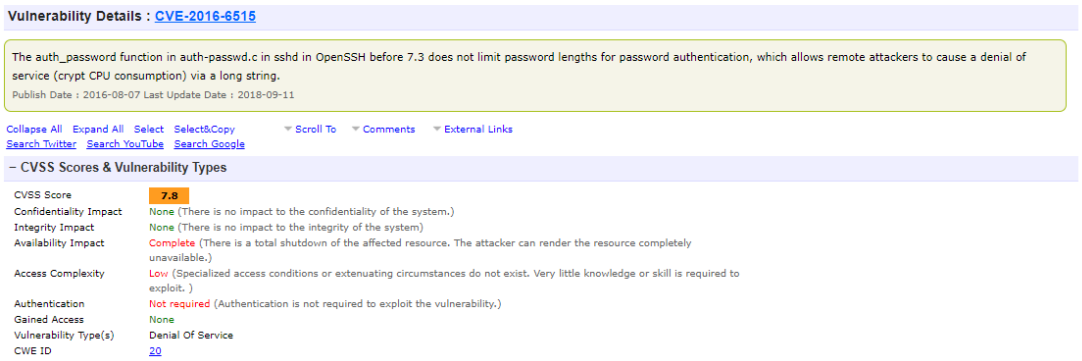
Cpe Name:*cpe:/a:openbsd:openssh:5.8*
CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending
Copy Results Download Results

| # | 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 | 269 | | | 2017-10-25 | 2019-10-02 | 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 | 2019-06-26 | 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-2016-0778 | 119 | | DoS Overflow | 2016-01-14 | 2018-10-09 | 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. | | | | | | | | | | | | | | |
| 4 | CVE-2016-0777 | 200 | | +Info | 2016-01-14 | 2018-10-09 | 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. | | | | | | | | | | | | | | |
| 5 | 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. | | | | | | | | | | | | | | |
| 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. | | | | | | | | | | | | | | |

Total number of vulnerabilities : 9 Page : 1 (This Page)

Figura 8: Vulnerabilidades Openssh5

Contudo, a versão que apresenta a vulnerabilidade com *Base Score* mais elevado é a OpenSSH 7.2p2.



**Vulnerability Details : CVE-2016-6515**

The auth_password function in auth-passwd.c in sshd in OpenSSH before 7.3 does not limit password lengths for password authentication, which allows remote attackers to cause a denial of service (crypt CPU consumption) via a long string.

Publish Date : 2016-08-07 Last Update Date : 2018-09-11

Collapse All Expand All Select Select&Copy Scroll To Comments External Links
Search Twitter Search YouTube Search Google

**– 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 | 20 |

Figura 9: Vulnerabilidade CVE-2016-6515

Esta vulnerabilidade, identificada como CVE-2016-6515, possibilita o *exploit Denial of Service* (DoS). Este tipo de ataque consiste em tornar o serviço ou uma máquina indisponível para o utilizador, por um determinado período.

Tal é possível devido à não implementação de restrições relativas ao tamanho de *passwords* na função responsável pela autenticação (auth_password), o que pode levar a que os mecanismos responsáveis por este processo utilizem recursos significativos, levando a que o sistema fique também indisponível para os restantes utilizadores.

Apesar de esta vulnerabilidade não implicar o ganho de acesso ao sistema e de não implicar impacto na confidencialidade e integridade, tem um impacto considerável na disponibilidade. Para além disso, trata-se de um ataque de complexidade baixa, uma vez que não é necessário autenticação nem interação com um utilizador em específico.

Posto isto, acredita-se que é justificável a sua classificação como uma vulnerabilidade grave.