



École Polytechnique de Montréal

INF8102 – Sécurité dans les environnements infonuagiques

Travail Pratique 1

Remis par:

Ilias Bettayeb - 2092408

Benoit Dambrine - 2075984

Soumis à : Armstrong Foundjem

Date de remise: 06 octobre 2023

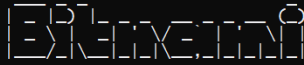
Exercise 1. (15 point)

1. Copy file securely from your local environment to the cloud and back

```
PS C:\Users\bd200\OneDrive\ecole\A2023\INF8102\lab1> ssh -i TP1.pem bitnami@ec2-18-232-93-60.compute-1.amazonaws.com
Linux ip-172-31-37-100 5.10.0-25-cloud-amd64 #1 SMP Debian 5.10.191-1 (2023-08-16) x86_64
```

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.



```
*** Welcome to the WordPress packaged by Bitnami 6.3.1-7 ***
*** Documentation: https://docs.bitnami.com/aws/apps/wordpress/ ***
***                 https://docs.bitnami.com/aws/ ***
*** Bitnami Forums: https://github.com/bitnami/vms/ ***
bitnami@ip-172-31-37-100:~$
```

```
PS C:\Users\bd200\OneDrive\ecole\A2023\INF8102\lab1> scp -i TP1.pem -v rufus.jpg bitnami@ec2-3-91-105-128.compute-1.amazonaws.com:rufus.jpg
Executing: program ssh.exe host ec2-3-91-105-128.compute-1.amazonaws.com, user bitnami, command scp -v -t rufus.jpg
openssh_for_windows.8.tpm, LibreSSL 3.0.2
debug1: Connecting to ec2-3-91-105-128.compute-1.amazonaws.com [3.91.105.128] port 22.
debug1: Connection established.
debug1: identity file TP1.pem type -1
debug1: identity file TP1.pem-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_for_Windows_8.1
debug1: Remote protocol version 2.0, remote software version OpenSSH_8.4p1 Debian-5+deb11u1
debug1: match: OpenSSH_8.4p1 Debian-5+deb11u1 pat OpenSSH compat 0x04000000
debug1: Authenticating to ec2-3-91-105-128.compute-1.amazonaws.com:22 as 'bitnami'
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: curve25519-sha256
debug1: kex: host key algorithm: ecdsa-sha2-nistp256
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: Server host key: ecdsa-sha2-nistp256 SHA256:bd1j20twnaGLt+c10xt+3nJ2iR0dXej5XVnpIThby8w
debug1: Host 'ec2-3-91-105-128.compute-1.amazonaws.com' is known and matches the ECDSA host key.
debug1: Found key in c:\Users\bd200\.ssh\known_hosts:3
debug1: rekey out after 134217728 blocks
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: rekey in after 134217728 blocks
debug1: pubkey prepare: ssh_get_authentication_socket: No such file or directory
debug1: Will attempt key: TP1.pem explicit
debug1: SSH2_MSG_EXT_INFO received
debug1: kex input ext info: server-sig-algs=ssh-ed25519,sk-ssh-ed25519@openssh.com,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,sk-ecdsa-sha2-nistp256@openssh.com,webauthn-sk-ecdsa-sha2-nistp256@openssh.com
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Trying private key: TP1.pem
debug1: Authentication succeeded (publickey).
Authenticated to ec2-3-91-105-128.compute-1.amazonaws.com ([3.91.105.128]:22).
debug1: channel 0: new (client-session)
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: pledge: network
debug1: client_input_global_request: rtype hostkeys-00@openssh.com want_reply 0
debug1: Remote: /home/bitnami/.ssh/authorized_keys:1: key options: agent-forwarding port-forwarding pty user-rc x11-forwarding
debug1: Sending command: scp -v -t rufus.jpg
Sending file modes: C0666 21602 rufus.jpg
Sink: C0666 21602 rufus.jpg
rufus.jpg
debug1: client_input_channel_req: channel 0 rtype exit-status reply 0
debug1: channel 0: free: client-session, nchannels 1
Transferred: sent 24092, received 2664 bytes, in 0.2 seconds
Bytes per second: sent 102803.9, received 11367.7
debug1: Exit status 0
PS C:\Users\bd200\OneDrive\ecole\A2023\INF8102\lab1> scp -i TP1.pem -v rufus.jpg bitnami@ec2-3-91-105-128.compute-1.amazonaws.com:rufus.jpg
```

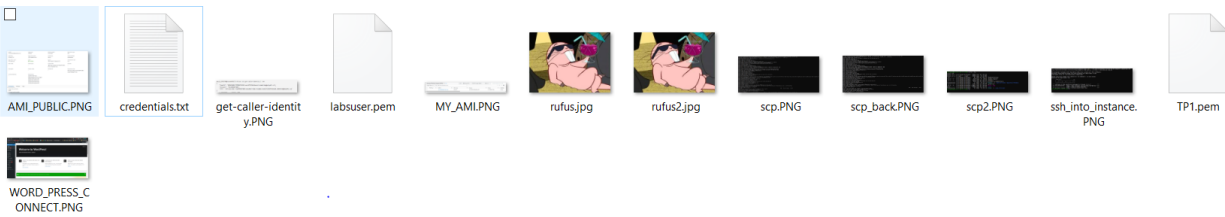
```
bitnami@ip-172-31-37-100:~$ ls -la
total 60
drwxr-xr-x 3 bitnami bitnami 4096 Sep 22 19:58 .
drwxr-xr-x 3 root root 4096 Sep 6 13:08 ..
-rw-r--r-- 1 bitnami bitnami 61 Sep 22 19:54 .bash_history
-rw-r--r-- 1 bitnami bitnami 220 Aug 4 2021 .bash_logout
-rw-r--r-- 1 bitnami bitnami 4220 Sep 22 19:29 .bashrc
-rw-r--r-- 1 bitnami bitnami 430 Sep 22 19:52 bitnami_credentials
lrwxrwxrwx 1 bitnami bitnami 27 Sep 6 13:11 htdocs -> /opt/bitnami/apache2/htdocs
-rw-r--r-- 1 bitnami bitnami 1501 Sep 22 19:29 .profile
-rw-r--r-- 1 bitnami bitnami 21602 Sep 22 19:58 rufus.jpg
drwx----- 2 bitnami bitnami 4096 Sep 22 19:29 .ssh
lrwxrwxrwx 1 bitnami bitnami 12 Sep 6 13:11 stack -> /opt/bitnami
bitnami@ip-172-31-37-100:~$
```

```

C:\Users\bd200\OneDrive\ecole\A2023\INF8102\lab1> scp -i TP1.pem -v bitnami@ec2-3-91-105-128.compute-1.amazonaws.com:rufus.jpg rufus2.jpg
Executing: program ssh.exe host ec2-3-91-105-128.compute-1.amazonaws.com, user bitnami, command scp -v -f rufus.jpg
OpenSSH_for_Windows_8.1p1, LibreSSL 3.0.2
debug1: connecting to ec2-3-91-105-128.compute-1.amazonaws.com [3.91.105.128] port 22.
debug1: Connection established.
debug1: identity file TP1.pem type -1
debug1: identity file TP1.pem-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_for_Windows_8.1
debug1: Remote protocol version 2.0, remote software version OpenSSH_8.4p1 Debian-5+deb11u1
debug1: match: OpenSSH_8.4p1 Debian-5+deb11u1 pat OpenSSH_compat 0x04000000
debug1: Authenticating to ec2-3-91-105-128.compute-1.amazonaws.com:22 as 'bitnami'
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: curve25519-sha256
debug1: kex: host key algorithm: ecdsa-sha2-nistp256
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: Server host key: ecdsa-sha2-nistp256 SHA256:bdlj20twnaGLt+c10xt+JnJ2iR0dXejSxVNPtHbY8w
debug1: Host 'ec2-3-91-105-128.compute-1.amazonaws.com' is known and matches the ECDSA host key.
debug1: Found key in C:\Users\bd200\ssh\known_hosts:3
debug1: rekey out after 134217728 blocks
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: rekey in after 134217728 blocks
debug1: pubkey prepare: ssh.get_authentication_socket: No such file or directory
debug1: Will attempt key: TP1.pem explicit
debug1: SSH2_MSG_EXT_INFO received
debug1: kex input_ext_info: server-sig-algs=<ssh-ed25519,sk-ssh-ed25519@openssh.com,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,sk-ecdsa-sha2-nistp256@openssh.com,webauthn-sk-ecdsa-sha2-nistp256@openssh.com>
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Trying private key: TP1.pem
debug1: Authentication succeeded (publickey).
Authenticated to ec2-3-91-105-128.compute-1.amazonaws.com ([3.91.105.128]:22).
channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: pledge: network
debug1: Client input global request: rtype hostkeys-00@openssh.com want_reply 0
debug1: Remote: /home/bitnami/.ssh/authorized_keys:1: key options: agent-forwarding port-forwarding pty user-rc x11-forwarding
debug1: Sending command: scp -v -f rufus.jpg
Sending file modes: C0644 21602 rufus.jpg
Sink: C0644 21602 rufus.jpg
rufus.jpg
debug1: Client input channel req: channel 0 rtype exit-status reply 0
debug1: Client input channel req: channel 0 rtype eof@openssh.com reply 0
debug1: channel 0: free: client-session, nchannels 1
Transferred: sent 2426, received 24360 bytes, in 0.2 seconds
Bytes per second: sent 11031.5, received 110678.6
debug1: Exit status 0

```

100% 21KB 584.4KB/s 00:00



2. Use the RSA algorithm to encrypt/decrypt files on your cloud and modify the content locally then send them back to the cloud and open the content, what do you see?

```
bitnami@ip-172-31-37-100: ~
GNU nano 5.4 example.txt
tp1
hello world

bitnami@ip-172-31-37-100: ~
bitnami@ip-172-31-37-100:~$ openssl enc -aes-256-cbc -salt -in
enc: Option -in needs a value
enc: Use -help for summary.
bitnami@ip-172-31-37-100:~$ l
-bash: l: command not found
bitnami@ip-172-31-37-100:~$ ls
bitnami_credentials  encrypted_file.bin  example.txt  httdocs  recipient2.pub  recipient.pub  rufus.jpg  stack
bitnami@ip-172-31-37-100:~$ openssl enc -aes-256-cbc -salt -in example.txt -out example.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bitnami@ip-172-31-37-100:~$ ls
bitnami_credentials  example.enc  httdocs  recipient2.pub  stack
encrypted_file.bin  example.txt  recipient2.pub  rufus.jpg
bitnami@ip-172-31-37-100:~$ nano example.enc
bitnami@ip-172-31-37-100:~$

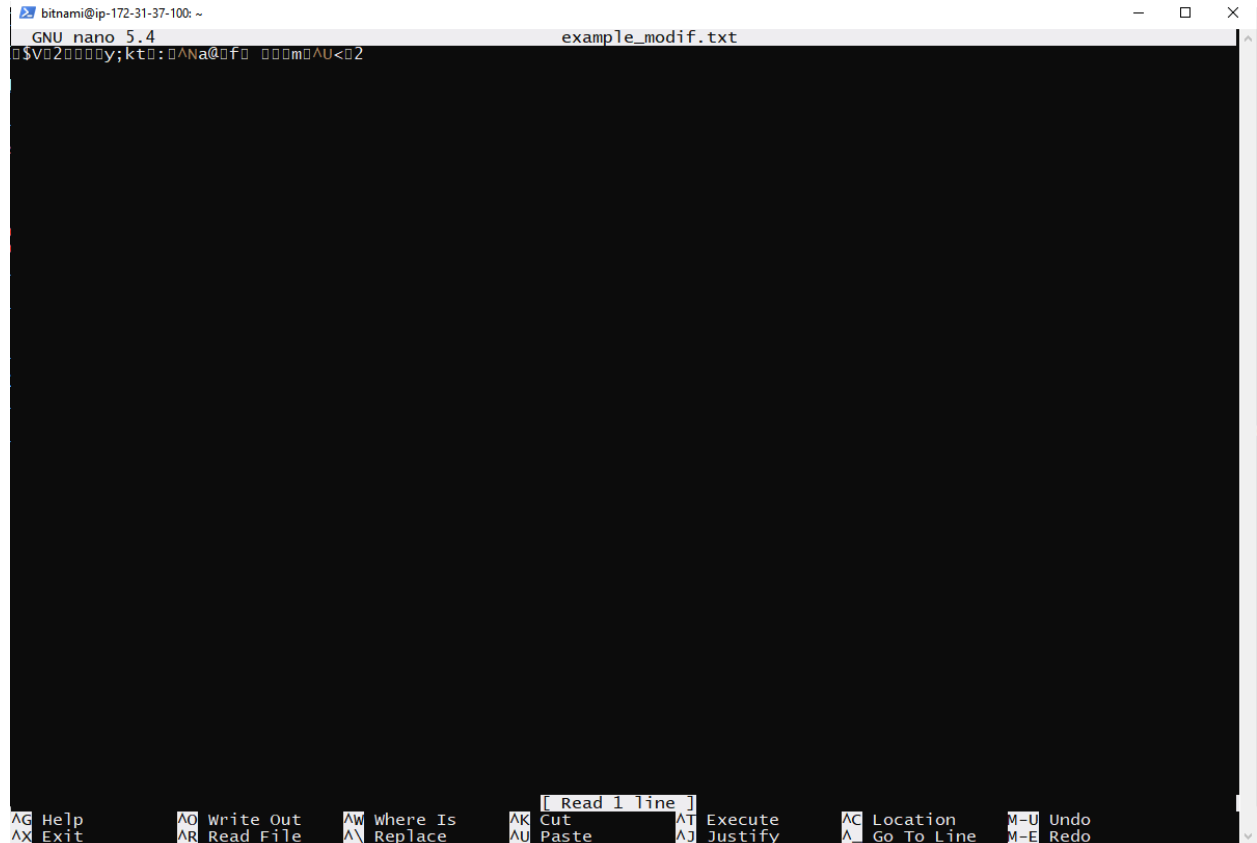
PS C:\Users\bd200\OneDrive\ecole\A2023\INF8102\lab1> scp -i TP1.pem bitnami@ec2-54-162-247-163.compute-1.amazonaws.com:
example.enc example.enc
example.enc
100% 48 0.8KB/s 00:00
PS C:\Users\bd200\OneDrive\ecole\A2023\INF8102\lab1> openssl
openssl: The command 'openssl' is not recognized as the name of a cmdlet, function, script file, or executable program.
```

```
MINGW64:/c:/Users/bd200/OneDrive/ecole/A2023/INF8102/lab1
bd200@LAPTOP-UL4HKL77 MINGW64 ~/OneDrive/ecole/A2023/INF8102/lab1
$ nano
bd200@LAPTOP-UL4HKL77 MINGW64 ~/OneDrive/ecole/A2023/INF8102/lab1
$ nano example.enc
bd200@LAPTOP-UL4HKL77 MINGW64 ~/OneDrive/ecole/A2023/INF8102/lab1
$ cat example.enc
Salted__$p__$oh__$Es/__$
__$?__$3__$>A9__$
bd200@LAPTOP-UL4HKL77 MINGW64 ~/OneDrive/ecole/A2023/INF8102/lab1
$ |
```

```
MINGW64:/c:/Users/bd200/OneDrive/ecole/A2023/INF8102/lab1
bd200@LAPTOP-UL4HKL77 MINGW64 ~/OneDrive/ecole/A2023/INF8102/lab1
$ cat example.enc
Salted__$p__$oh__$sgfsh__$Es/__$
__$?__$3__$>A9__$
bd200@LAPTOP-UL4HKL77 MINGW64 ~/OneDrive/ecole/A2023/INF8102/lab1
$ |
```

```
PS C:\Users\bd200\OneDrive\ecole\A2023\INF8102\lab1> scp -i TP1.pem example.enc bitnami@ec2-34-207-178-33.compute-1.amazonaws.com:example.enc
example.enc
PS C:\Users\bd200\OneDrive\ecole\A2023\INF8102\lab1>
100% 55 1.2KB/s 00:00
```

```
bitnami@ip-172-31-37-100:~$ openssl enc -d -aes-256-cbc -salt -in example.enc -out example_modif.txt -k helloworld
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bad decrypt
139996726134080:error:0606506D:digital envelope routines:EVP_DecryptFinal_ex:wrong final block length:../crypto/evp/ev
p_enc.c:599:
bitnami@ip-172-31-37-100:~$
```



Il est possible de constater que le fichier a bien été envoyé qu'il a été possible d'encrypter le fichier example.txt, le decrypter localement, le modifier et le renommer example_modif.txt et le renvoyer.

3. Explain the permission with 400 granted to the private key, what does it mean?

Ajouter la permission 400, permet de mettre la permission à lecture seulement pour le propriétaire du fichier et d'enlever tous les droits pour les groupes et les autres. Ce changement permet au propriétaire de lire le fichier, mais il ne peut pas l'exécuter ou le modifier. Il est utile de faire cela pour protéger la private key de ce faire modifier par erreur.

4. What is the purpose of the following keys?:

a. Aws_access_key_id

Cette première clé est utilisée afin d'authentifier l'utilisateur ou le rôle IAM. AWS aura alors le premier élément nécessaire pour identifier l'identité de l'utilisateur en "programmation calls". Cette variable spécifie une clé d'accès AWS associée à un compte IAM.

b. aws_secret_access_key

Cette seconde clé est aussi utilisée afin d'authentifier l'utilisateur ou le rôle IAM. Cette clé est nécessaire afin de se connecter à Amazon Keyspaces programmatically. Cette variable spécifie la clé secrète associée à la première clé ID, jouant alors d'une certaine manière le rôle d'un mot de passe.

c. Aws_session_token

Cette troisième clé est optionnelle pour l'authentification de l'utilisateur/rôle IAM. Cette variable spécifie la valeur du session token value nécessaire si l'utilisateur utilise des security credentials temporaire trouvé à partir du AWS Security Token Service operations.

5. Describe the model of cloud that your public images represent and state what level of responsibilities are involved by you (the cloud provider) and your customers?

Le modèle de notre image publique est plateforme as code. En effet, dans notre image l'infrastructure et l'environnement de production est fourni avec l'image. Le fournisseur de cloud est responsable de la plateforme, de l'environnement et de l'infrastructure. Le client est responsable de l'application et des données

Exercise 2. (5 points)

1. Describe one security vulnerability and one best practice associated with S3.

Mettre le S3 bucket publique par inadvertance avec des mauvais configuration. La meilleure pratique serait de bien configurer le S3.

2. Why is the policy.json file used when permission can be granted directly to objects?

Permet de centraliser le contrôle de l'accès, il est possible de contrôler l'accès de plusieurs ressources en même temps au lieu de contrôler l'accès un environnement à la fois.

Permet d'avoir un contrôle plus granulaire sur les ressources, ce qui n'est pas toujours faisable en gérer les permissions directement avec les objets.

Permet d'avoir une trace des droits qu'on a donné.

3. In the above policy.json file configuration, list two examples where the "aws" after the arn (i.e., arn:aws:...) is not/never used.

Le format typique pour les ARN est le suivant : arn:partition:service:region:account-id:resource
Ici, la partition est l'endroit où se trouve la ressource utilisée. Elle peut prendre les valeurs suivantes: aws (AWS Regions, plus commercial), aws-cn (China Regions) et aws-us-gov (AWS GovCloud (US) Regions). On peut donc comprendre que les régions sont regroupées en partitions. Ces partitions ont des instances de AWS IAM indépendantes. Alors, il est possible de conclure que le aws après le arn n'est pas utilisé si la ressource utilisée se trouve dans la partition aws-cn ou encore la partition aws-us-gov. Donc, les personnes possédant un compte AWS China peuvent utiliser les services de cette partition, mais ils ont besoin d'une licence de business. Un premier exemple est donc une compagnie chinoise nécessitant les services de AWS China. Les services de la partition aws-us-gov sont accessibles aux clients du gouvernement et aux entités commerciales répondants aux exigences d'AWS GovCloud (US). Un autre exemple serait alors le Département de la Justice américain qui utilise les services de cette partition.

4. Given the figure below can you describe what is going on? In terms of hosting a static page on the internet.

D'après l'image nous voyons un S3 bucket qui est disponible publiquement. Il est donc possible de faire des appels à l'adresse IP public du bucket pour accéder à des objets et les stocker. Dans le cas de la figure, notre bucket est public donc ce qui est accessible doit être des objets publics comme une page web static

Bonus 2 points

Explain why the following codes don't seem to work properly?

Cette première ligne de code nous a permis de créer un "Low-Level client": `client = boto3.client`. Puis, nous créons aussi un second "Low-Level client" représentant EC2. L'appel au filtre `region_name` de la méthode `describe_regions` permet de spécifier que la région active pour notre compte est la région "us-west-7". Or, ce code n'est associé à aucune des régions fournies par un compte AWS. Puisque, comme le dit AWS sur son site web, il est impossible d'accéder à des régions supplémentaires à partir d'un compte AWS. Il est donc probable que la région spécifiée n'existe donc pas, expliquant pourquoi le code ne semble pas fonctionner.

Il semble aussi que la configuration du client EC2 est écrasée par le second appel à la méthode `boto3.client`. Il faut s'assurer de créer de clients distincts, qui auront alors des noms de variables distinctes. On peut alors créer un `client_IAM` et un `client_EC2` afin de les différencier.

Dans la seconde partie du code, nous créons notre "Low-Level client" représentant AWS STS à partir de notre propre session, soit `STS_CLIENT`. Le client de type "ec2" est associé à la région "us-east-2", mais nous avons spécifié précédemment que la région associée à ce client était plutôt "us-east-7", ce qui explique l'erreur. Une correction est de changer la spécification initiale du nom de la région du client représentant EC2.