

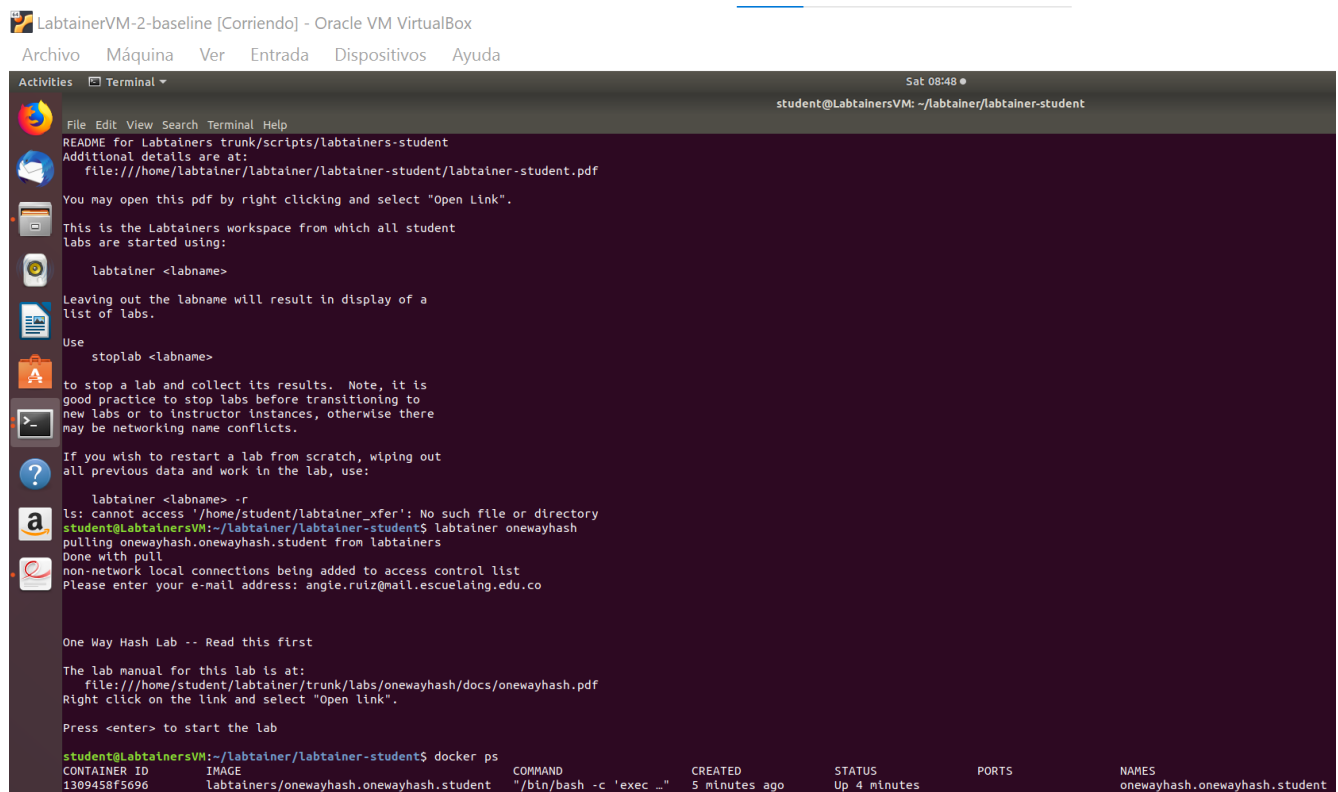
Crypto Lab: función hash unidireccional y MAC

1. Objetivos Generales:

- Conocer y entender las funciones hash unidireccionales
- Conocer y entender el funcionamiento del código de autenticación (MAC)
- Aprender a utilizar las herramientas que nos brinda Labtainer
- Aprender a escribir programas para generar valor hash unidireccional para un mensaje dado
- Aprender a escribir programas para generar MAC para un mensaje dado
- Aprender la diferencia entre la propiedad unidireccional y la propiedad libre de colisiones.

2. Pasos de realización del laboratorio onewayhash

Primero para ingresar al ambiente del laboratorio usamos el comando "labtainer onewayhash", luego nos pide ingresar un correo electrónico, en este caso usé mi correo institucional, al teclear el "enter" se nos abre la consola del laboratorio y desde la consola principal con "docker ps" vi que hace uso de Docker. Después de ello, abrimos el enlace del pdf que nos sale en la consola y este documento nos indica las tareas a realizar dentro de este laboratorio.



```
LabtainerVM-2-baseline [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
Activities  Terminal
Sat 08:48
student@LabtainersVM: ~/labtainer/labtainer-student
File Edit View Search Terminal Help
README for Labtainers trunk/scripts/labtainers-student
Additional details are at:
file:///home/labtainer/labtainer/labtainer-student/labtainer-student.pdf
You may open this pdf by right clicking and select "Open Link".
This is the Labtainers workspace from which all student
labs are started using:
labtainer <labname>
Leaving out the labname will result in display of a
list of labs.
Use
stoplab <labname>
to stop a lab and collect its results. Note, it is
good practice to stop labs before transitioning to
new labs or to instructor instances, otherwise there
may be networking name conflicts.
If you wish to restart a lab from scratch, wiping out
all previous data and work in the lab, use:
labtainer <labname> -r
ls: cannot access '/home/student/labtainer_xfer': No such file or directory
student@LabtainersVM:~/labtainer/labtainer-student$ labtainer onewayhash
pulling onewayhash.onewayhash.student from labtainers
Done with pull
non-network local connections being added to access control list
Please enter your e-mail address: angte.rutz@mail.escuelatng.edu.co

One Way Hash Lab -- Read this first

The lab manual for this lab is at:
file:///home/student/labtainer/trunk/labs/onewayhash/docs/onewayhash.pdf
Right click on the link and select "Open link".

Press <enter> to start the lab

student@LabtainersVM:~/labtainer/labtainer-student$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS          NAMES
1309458f5696   labtainers/onewayhash.onewayhash.student  "/bin/bash -c 'exec ..."  5 minutes ago  Up 4 minutes  -              onewayhash.onewayhash.student
```

LabtainerVM-2-baseline [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Activities Document Viewer

Sat 06:53 onewayhash.pdf 249.76%

3 Lab Tasks

3.1 Task 1: Generating Message Digest and MAC

In this task, we will play with various one-way hash algorithms. You can use the following `openssl dgst` command to generate the hash value for a file. To see the manpages, you can type `man openssl` and `man dgst`.

```
% openssl dgst dgsttype filename
```

Please replace the `dgsttype` with a specific one-way hash algorithm, such as `-md5`, `-sha1`, `-sha256`, etc. And replace `filename` with `filetodigest.txt`, which is in your home directory. In this task, you should try at least 3 different algorithms, and describe your observations. You can find the supported one-way hash algorithms by typing "`openssl dgst -h`" NOTE: the list of algorithms included in the manpages is not correct.

- Tarea 1: Generación de resumen de mensajes y MAC:

Con el comando “`openssl dgst`” podemos generar el valor hash para un archivo y dado el caso que queramos ver el manual, podemos usar el comando “`man openssl`” y “`man dgst`”. Usaremos desde la consola del laboratorio 3 veces (con algoritmos hash unidireccionales diferentes) este comando “`openssl dgst dgsttype filename`”, donde reemplazamos el `dgsttype` primero por `-md5`, luego por `-sha256` y por último `-sha1`, y reemplazamos el `filename` por el nombre del archivo con `filetodigest.txt`, que se encuentra dentro del material del laboratorio.

```
ubuntu@onewayhash:~$ openssl dgst -md5 filetodigest.txt
MD5(filetodigest.txt)= a01e40d042b0b1f3e6824b019700ba96
ubuntu@onewayhash:~$ openssl dgst -sha256 filetodigest.txt
SHA256(filetodigest.txt)= 065e85473455f7ed562db943ded0b98d33530caf39706e1214ec5cd931fea0f1
ubuntu@onewayhash:~$ openssl dgst -sha1 filetodigest.txt
SHA1(filetodigest.txt)= 3f9810c940ea130df077175dc9987c3e368cb792
```

Observaciones:

- MD5: Podemos ver que la longitud del hash es de 32 (128 bits).
- SHA256: Podemos ver que la longitud del hash es de 64 (256 bits).
- SHA1: Podemos ver que la longitud del hash es de 40 (60 bits).

- Tarea 2: Hash con clave y HMAC

Ahora vamos a generar un hash con clave (MAC) para un archivo. Usaremos para ello el comando “`openssl dgst -alg -hmac "holi" nombre de archivo`” donde “holi” es la clave, alg es el algoritmo (los mismos de la tarea anterior) y `filename` es el nombre del archivo, “`filetodigest.txt`” en este caso:


```
ubuntu@onewayhash: ~  
File Edit View Search Terminal Help  
00000000 05 73 20 75 6E 61 20 70 72 75 65 62 61 0A es una prueba.  
00000014  
00000028  
0000003C  
00000050  
00000064  
00000078  
0000008C  
000000A0  
000000B4  
000000C8  
000000DC  
000000F0  
00000104  
00000118  
0000012C  
00000140  
00000154  
00000168  
0000017C  
00000190  
000001A4  
000001B8  
000001CC  
-- edit-this-file.txt --0x0/0xE-----
```

```
ubuntu@onewayhash: ~  
File Edit View Search Terminal Help  
00000000 66 73 20 75 6E 61 20 70 72 75 65 62 61 0A fs una prueba.  
00000014  
00000028  
0000003C  
00000050  
00000064  
00000078  
0000008C  
000000A0  
000000B4  
000000C8  
000000DC  
000000F0  
00000104  
00000118  
0000012C  
00000140  
00000154  
00000168  
0000017C  
00000190  
000001A4  
000001B8  
000001CC  
-- edit-this-file.txt --0x1/0xF-----
```

```
ubuntu@onewayhash:~$ vi edit-this-file.txt  
ubuntu@onewayhash:~$ openssl dgst -md5 edit-this-file.txt  
MD5(edit-this-file.txt)= c21ad03851847863b143456354db8526  
ubuntu@onewayhash:~$ hexedit edit-this-file.txt  
[1]+ Stopped hexedit edit-this-file.txt  
ubuntu@onewayhash:~$ hexedit edit-this-file.txt  
[2]+ Stopped hexedit edit-this-file.txt  
ubuntu@onewayhash:~$ hexedit edit-this-file.txt  
ubuntu@onewayhash:~$ hexedit edit-this-file.txt  
[3]+ Stopped hexedit edit-this-file.txt  
ubuntu@onewayhash:~$ openssl dgst -md5 edit-this-file.txt  
MD5(edit-this-file.txt)= 204fa4a203b9ab6a7ebe4b688a4ed122  
ubuntu@onewayhash:~$ openssl dgst -sha256 edit-this-file.txt  
SHA256(edit-this-file.txt)= f82a3fcf9f9090ce3a4ac7661814f1b8883602ff54849fda56107f983a218cb7  
ubuntu@onewayhash:~$ hexedit edit-this-file.txt  
ubuntu@onewayhash:~$ openssl dgst -sha256 edit-this-file.txt  
SHA256(edit-this-file.txt)= 2a6070f95f5844632a5f9f12f58583ed0b6b502bae50e2f799cf427a7676fd0f
```

No son similares en ningún carácter en ambos casos, se ve una diferencia total en los hashes.

- **Tarea 4: Propiedad unidireccional versus propiedad libre de colisiones**

Para finalizar nos piden usar el método de fuerza bruta para ver cuánto tiempo lleva romper cada una de estas propiedades por medio de un código propio para invocar las funciones de resumen de mensajes de la biblioteca de cifrado de openssl, usando cualquier función hash unidireccional, pero solo teniendo en cuenta los primeros 24 bits del valor hash por cuestión de que nos llevara años romper con fuerza bruta si usamos más bits. Pero no entendí el código de guía dado en el enlace del pdf del laboratorio y no soy buena en C, entonces me dediqué a investigar y verlo por el lado matemático.

- ¿Cuántas pruebas le tomará romper la propiedad unidireccional utilizando el método de fuerza bruta? Depende del valor del hash, según el habrá muchos mensajes que satisfagan el mismo valor mas no que sean toda la cadena original, lo cual lo hace mucho más complicado de descifrar.
- ¿Cuántas pruebas le tomará romper la propiedad libre de colisiones utilizando el método de fuerza bruta? En el mejor de los casos encontrar una colisión toma menos de $2^{n/2}$. Explicación en la última pregunta.
- Según su observación, ¿qué propiedad es más fácil de romper utilizando el método de fuerza bruta?: El de colisión, porque supongamos que queremos romper la unidireccionalidad de un hash y nos dan el valor de 294, pero no podemos saber con ese valor que cadena es la original porque, por ejemplo 'abc' y 'cba' (y muchos otros) dan el mismo valor hash. Entonces es mucho más difícil de romper y se deben tener en cuenta todos los mensajes que puedan dar ese valor, en comparación con encontrar la colisión y su explicación está a continuación.
- ¿Puede explicar matemáticamente la diferencia en su observación?: La paradoja del cumpleaños; se dice que para tener confianza en encontrar dos mensajes con el mismo resumen $H(M)$ (probabilidad del 50%) no habrá que buscar en 2^n y bastará una búsqueda en el espacio $2^{n/2}$. En el peor de los casos será mayor la cantidad de operaciones a realizar.

¿En qué consiste la paradoja del cumpleaños?:

- Problema: Cuál será la confianza (probabilidad mayor que el 50%) de que, en un aula con 365 personas, donde no se tiene en cuenta el día 29 de febrero de los años bisiestos, dos de ellas al azar cumplan años en la misma fecha.
- Solución: Se escribe en la pizarra los 365 días del año y entran al aula de uno en uno, borrando el día de su cumpleaños de la pizarra. Para alcanzar esa confianza, basta que entren 23 personas al aula, un valor muy bajo, en principio inimaginable y de allí el nombre de paradoja.
- Explicación: El primero en entrar tendrá una probabilidad de que su número no esté borrado igual a $n/n = 1$, el segundo de $(n-1)/n$, etc. La probabilidad p de no

coincidencia será igual a $p = n!/(n-k)nk$. Para $k = 23$ se tiene $p = 0,493$ y entonces la probabilidad de coincidencia es 0,507.

3. Conclusiones obtenidas

-SHA256 es el más seguro vs MD5 y SHA1, porque la probabilidad de buscar y encontrar el mismo valor MD5 es de 2^{64} , por ende, es menos seguro que SHA1 y SHA256, y la probabilidad de buscar y encontrar el mismo valor SHA256 es de 2^{128} , por ende, es más seguro que SHA1, la probabilidad de buscar y encontrar el mismo valor SHA1 es de 2^{80} .

-HMAC nos brinda una mayor seguridad contra los ataques de fuerza bruta evitando colisiones, por ejemplo, HMAC-MD5 es más seguro que MD5 porque no sufre las debilidades que tiene MD5 solo.

- Al cambiar tan solo un solo bit se ve una diferencia total en los valores hash, pero matemáticamente si se modifica un solo bit del mensaje M , $h(M)$ debería cambiar aproximadamente la mitad de sus bits.

- La propiedad unidireccional es una función matemática en la cual se conoce un procedimiento de cálculo eficiente y rápido para computar esa función, pero que no se conoce un procedimiento eficiente para realizar ese mismo cálculo, pero a la inversa, por ende, le dicen "fácil de calcular, difícil de invertir".

- La propiedad libre de colisiones es una función que debe hacer casi imposible encontrar que dos entradas "x" & "z" (Cualquier mensaje de n longitud) que resulten en la misma salida "y" (hash).