ATRAK - DOS

DEFACEMMENT ATTACK

MEMORIA RAM

MARCOS | DANIEL | PEDRO

CONTENIDO

Herramienta utilizada

Para analizar la memoria RAM vamos a utilizar el programa volatility encargado de analizar la memoria RAM. Vamos a usar el siguiente comando para poder instalarlo en nuestro sistema Linux.

COMANDO

```
$ git clone https://github.com/volatilityfoundation/volatility.git
$ cd volatility/tools/linux
$ make
$ cd ~/volatility
$ chmod +x vol.py
```

Una vez esté instalado y tengamos el perfil creado, analizaremos la información que contiene la memoria RAM, indicando la RAM, el PERFIL RAM y el COMANDO a ejecutar para comprobar que todo funciona correctamente..

COMANDO

```
$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin'
--profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_pstree
```

FRAGMENTO DE SALIDA

```
Name                 Pid          Uid
systemd              1
.systemd-journal     413
.lvmetad             442
.systemd-udevd       471
.systemd-timesyn     560          100
.dhclient            991
.iscsid              1140
```

NETSTAT

Para analizar el estado de la red, las estadísticas de los protocolos y las conexiones que se realizan con la máquina deberemos utilizar el siguiente COMANDO.

COMANDO

```
$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin'
--profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_netstat
```

Como se puede observar en la salida del NETSTAT, existe un apartado en el que se realiza una conexión mediante el servicio sshd a la máquina con la IP 23.226.128.37

FRAGMENTO DE SALIDA

```
UNIX 110294         (sd-pam)/9060
UNIX 110301         (sd-pam)/9060
TCP 172.31.47.60 : 22 23.226.128.37:42760 ESTABLISHED    sshd/9118
UNIX 110219                 sshd/9118
UNIX 110387                 sshd/9118
```

COMANDO DE BASH

Para analizar los comandos de bash utilizados en la terminal del sistema utilizaremos el siguiente COMANDO.

COMANDO

```
$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin'
--profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_bash
```

Como se puede observar en el output de los COMANDOS BASH, se ejecutaron bastantes comandos relacionados al servicio Apache y Mysql el mismo día, a la misma hora, minuto y segundo, por lo que podemos suponer que se ejecutó un script automático.

FRAGMENTO DE SALIDA

```
Pid            Name             CommandTime                Command
-------- ------------------- ---------------------------- -------
9126           bash          2018-07-24 05:24:19 UTC+0000   grep Listen ../*
9126           bash          2018-07-24 05:24:19 UTC+0000   sudo vi ../ports.conf
9126           bash          2018-07-24 05:24:19 UTC+0000   sudo a2enmod ssl
9126           bash          2018-07-24 05:24:19 UTC+0000   sudo chown -R www-data:www-data
wordpress

9126           bash          2018-07-24 05:24:19 UTC+0000   sudo letsencrypt --apache -d
ganga.site -d www.ganga.site

9126           bash          2018-07-24 05:24:19 UTC+0000   history
9126           bash          2018-07-24 05:24:19 UTC+0000   ifconfig
9126           bash          2018-07-24 05:24:19 UTC+0000   cd /etc/apache
9126           bash          2018-07-24 05:24:19 UTC+0000   sudo apt install apache2
libapache2-mod-php php-mysql

9126           bash          2018-07-24 05:24:19 UTC+0000   sudo apt upgrade
9126           bash          2018-07-24 05:24:19 UTC+0000   mysql -uadmin -p
-hganga.ctmbcxcdb3us.eu-central-1.rds.amazonaws.com ganga

9126           bash          2018-07-24 05:24:19 UTC+0000   sudo service apache2 restart
9126           bash          2018-07-24 05:24:19 UTC+0000   sudo apt update
9126           bash          2018-07-24 05:24:19 UTC+0000   sudo letsencrypt --apache -d
ganga.site -d www.ganga.site

9126           bash          2018-07-24 05:24:19 UTC+0000   apt-cache search certbot
9126           bash          2018-07-24 05:24:19 UTC+0000   mysql -uadmin -p
-hganga.ctmbcxcdb3us.eu-central-1.rds.amazonaws.com ganga

9126           bash          2018-07-24 05:24:19 UTC+0000   sudo letsencrypt --apache -d
ganga.site -d www.ganga.site

9126           bash          2018-07-24 05:24:19 UTC+0000   cd /etc/apache2
9126           bash          2018-07-24 05:24:19 UTC+0000   certbot
```

ARCHIVOS DEL SISTEMA

Para averiguar todos los archivos que se encontraban en el sistema utilizaremos el siguiente comando.

COMANDO

```
$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin'
--profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_find_file --listfiles
```

Como se puede observar en el OUTPUT de los ARCHIVOS DEL SISTEMA, dentro de los archivos del sistema se encuentran varios ficheros relacionados con los logs del sistema y de varios servicios, por lo que lo extraemos para obtener más información detallada de lo que sucedió en el sistema.

FRAGMENTO DE SALIDA

```
Inode      Number      Inode            File Path
-----    ----------- ------------------ ---------
                      fff8800369a54a8    /var/log
          50604       0xffff880005865d08 /var/log/syslog
          186         0xffff880004c691a8 /var/log/syslog.1
          50684       0xffff8800058658d8 /var/log/syslog.2.gz
          51220       0xffff880004c6bfb8 /var/log/auth.log
----------------      0x0                /var/log/auth.log.1
          50938       0xffff880004c6bb88 /var/log/kern.log
----------------      0x0                /var/log/kern.log.1
          515273      0xffff88000dce58d8 /var/log/apache2
          519916      0xffff8800058654a8 /var/log/apache2/error.log
          519914      0xffff880004c68518 /var/log/apache2/error.log.1
          515593      0xffff880005864c48 /var/log/apache2/access.log
          519913      0xffff880004c680e8 /var/log/apache2/access.log.1
          515594      0xffff880005864818 /var/log/apache2/error.log.2.gz
          519915      0xffff8800058611a8 /var/log/apache2/access.log.2.gz
```

LOGS DEL SISTEMA

Para obtener los logs del sistema tendremos que indicar con el siguiente comando el Inode y el output del archivo para exportarlo a nuestro equipo para su análisis.

COMANDO

```
$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin'
--profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_find_file -i
0xffff880005865d08 -O sysfile
```

Como podemos comprobar en el archivo SYS FILE del sistema, un segundo antes de ejecutar los comandos de bash, se crea una sesión con el usuario Ubuntu en el sistema.

FRAGMENTO DEL ARCHIVO

```
Jul 24 05:24:18 ip-172-31-47-60 systemd[1]: Created slice User Slice of ubuntu.
Jul 24 05:24:18 ip-172-31-47-60 systemd[1]: Starting User Manager for UID 1000...
Jul 24 05:24:18 ip-172-31-47-60 systemd[1]: Started Session 289 of user ubuntu.
Jul 24 05:24:18 ip-172-31-47-60 systemd[9057]: Reached target Sockets.
Jul 24 05:24:18 ip-172-31-47-60 systemd[9057]: Reached target Timers.
Jul 24 05:24:18 ip-172-31-47-60 systemd[9057]: Reached target Paths.
Jul 24 05:24:18 ip-172-31-47-60 systemd[9057]: Reached target Basic System.
Jul 24 05:24:18 ip-172-31-47-60 systemd[9057]: Reached target Default.
Jul 24 05:24:18 ip-172-31-47-60 systemd[9057]: Startup finished in 12ms.
Jul 24 05:24:18 ip-172-31-47-60 systemd[1]: Started User Manager for UID 1000.
Jul 24 05:26:45 ip-172-31-47-60 kernel: [332180.597488] lime: loading out-of-tree module
taints kernel.
Jul 24 05:26:45 ip-172-31-47-60 kernel: [332180.597519] lime: module verification failed:
signature and/or required key missing - tainting kernel
```

LOGS DE AUTH

Para obtener los logs del AUTH que nos indica todas las actividades de autorización del sistema tendremos que indicar con el siguiente comando el Inode y el output del archivo para exportarlo a nuestro equipo para su análisis.

COMANDO	$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin' --profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_find_file -i fff880004c6bfb8 -O auth.log

Como podemos observar, al mismo tiempo que se creó la sesión del usuario Ubuntu, se realizó una conexión mediante el servicio SSH desde la misma IP que encontramos en el análisis de Netstat, accediendo directamente con un Usuario Root del sistema, debido a que el UID es 0.

FRAGMENTO DEL ARCHIVO

```
Jul 24 05:24:18 ip-172-31-47-60 sshd[9055]: Accepted publickey for ubuntu from
23.226.128.37 port 42760 ssh2: RSA SHA256:Q27pW6dDYPJ8N0mBX6L8SO8OQ7LVSdNdm1xxzyBT23Y
Jul 24 05:24:18 ip-172-31-47-60 sshd[9055]: pam_unix(sshd:session): session opened for
user ubuntu by (uid=0)
Jul 24 05:24:18 ip-172-31-47-60 systemd: pam_unix(systemd-user:session): session opened
for user ubuntu by (uid=0)
Jul 24 05:24:18 ip-172-31-47-60 systemd-logind[1166]: New session 289 of user ubuntu.
```

LOGS DE APT

Para obtener los logs del APT que nos indica todos los comandos utilizados con apt tendremos que indicar con el siguiente comando el Inode y el output del archivo para exportarlo a nuestro equipo para su análisis.

COMANDO	`$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin' --profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_find_file -i 0xffff8800160bcc48 -O history.log`

Como podemos observar, se han realizado algunas instalaciones de paquetes en el sistema con el usuario Ubuntu, pero estos se realizaron antes de la conexión realizada por SSH y la ejecución de comandos de BASH, por lo que no encontramos información relevante al caso.

FRAGMENTO DEL ARCHIVO

```
Start-Date: 2018-07-20  09:29:25
Commandline: apt-get install python-letsencrypt-apache
Requested-By: ubuntu (1000)
Install: python-augeas:amd64 (0.5.0-1, automatic), libaugeas0:amd64 (1.4.0-0ubuntu1.1,
automatic), augeas-lenses:amd64 (1.4.0-0ubuntu1.1, automatic),
python-letsencrypt-apache:amd64 (0.4.1-1)
End-Date: 2018-07-20  09:29:26

Start-Date: 2018-07-20  09:40:10
Commandline: apt install mysql-client-core-5.7
Requested-By: ubuntu (1000)
Install: libaio1:amd64 (0.3.110-2, automatic), mysql-client-core-5.7:amd64
(5.7.22-0ubuntu0.16.04.1)
End-Date: 2018-07-20  09:40:11

Start-Date: 2018-07-24  05:24:53
Commandline: apt install make
Requested-By: ubuntu (1000)
Install: make:amd64 (4.1-6)
End-Date: 2018-07-24  05:24:54
```

LOGS DE APACHE

Para obtener los logs del APACHE ACCESS tendremos que indicar con el siguiente comando el Inode y el output del archivo para exportarlo a nuestro equipo para su análisis.

COMANDO

```
$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin'
--profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_find_file -i
0xffff880005864c48 -O access.log
```

Podemos comprobar que antes de realizar la conexión SSH, la misma dirección IP se conectó a la página web, tanto en la ruta principal como en la ruta favicon.ico, también podemos observar que las peticiones que realizó se hicieron con GET y podemos obtener información tanto del sistema operativo que utilizó como del navegador.

FRAGMENTO DEL ARCHIVO

```
109.94.177.117 - - [24/Jul/2018:04:42:00 +0000] "GET / HTTP/1.0" 200 942 "-" "-"
[24/Jul/2018:05:18:49 +0000] ganga.site 23.226.128.37 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256
GET / HTTP/1.1 191819 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36"
[24/Jul/2018:05:18:51 +0000] ganga.site 23.226.128.37 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256
GET /favicon.ico HTTP/1.1 286 "https://ganga.site/" "Mozilla/5.0 (Windows NT 6.1; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36"
::1 - - [24/Jul/2018:05:19:11 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.18
(Ubuntu) OpenSSL/1.0.2g (internal dummy connection)"
```

En este caso la conexión se realizó desde un sistema operativo Windows 7 x64 bits utilizando un navegador Chrome.

NAVEGADOR Y SISTEMA OPERATIVO

REGLAS YARA

En este caso utilizaremos con volatility las reglas YARA para detectar en sistema si existe algún artefacto relacionado con las familias de malware más comunes que se pudiesen dejar en el sistema, para ello, primero instalaremos el plugin de Yara para volatility con los siguientes comandos.

COMANDO

```
$ tar -zxf yara-4.0.1.tar.gz
$ cd yara-4.0.1
$ ./bootstrap.sh
$ sudo apt-get install automake libtool make gcc pkg-config
$ ./configure
$ make
$ sudo make install
$ make check
$ pip install yara-python
```

El segundo paso será crear las reglas Yara.

CREACIÓN DE REGLAS YARA

```
#!/usr/bin/env python
# encoding: utf-8

import os
import shutil

def get_rules_from_git():
 shutil.rmtree("./rules")
 os.system("git clone https://github.com/Yara-Rules/rules.git")

def list_yara_files():
 all_yara_files = []
 for root, directories, filenames in os.walk("./rules/malware"):
  print ("Processing " + root)
  filenames.sort()
  for file_name in filenames:
   rule_filename, rule_file_extension = os.path.splitext(file_name)
   if rule_file_extension == ".yar" or rule_file_extension == ".yara":
            all_yara_files.append(os.path.join(root, file_name))
 return all_yara_files

def remove_incompatible_imports(files):
 filtered_files = []
 for yara_file in files:
  with open(yara_file, 'r') as fd:
   yara_in_file = fd.read()
   if not (("import \"math\"" in yara_in_file) or ("import \"cuckoo\"" in yara_in_file) or ("import \"hash\"" in yara_in_file) or ("imphash" in yara_in_file)):
            filtered_files.append(yara_file)
 return filtered_files

def fix_duplicated_rules(files):
 filtered_files = []
 first_elf = True
 to_delete = False
 for yara_file in files:
  print ("Processing " + yara_file)
  with open(yara_file, 'r') as fd:
   yara_in_file = fd.readlines()
   for line in yara_in_file:
            if line.strip() == "private rule is_elf {":
            if first_elf:
            first_elf = False
            else:
            to_delete = True
            if not to_delete:
            filtered_files.append(line)
            if (not first_elf) and line.strip() == "}":
            to_delete = False
   filtered_files.append("\n")
 return filtered_files

def merge_rules(all_rules):
```

```
 with open("malware_rules.yar", 'w') as fd:
   fd.write(''.join(all_rules))

def main():
 get_rules_from_git()
 all_yara_files = list_yara_files()
 all_yara_filtered_1 = remove_incompatible_imports(all_yara_files)
 all_yara_filtered_2 = fix_duplicated_rules(all_yara_filtered_1)
 merge_rules(all_yara_filtered_2)

# Main body
if __name__ == '__main__':
 main()
```

Trás crear el archivo con las reglas Yara lo ejecutaremos para que se nos guarde en el sistema la combinación de las reglas Yara obtenidas de Yara Rules Project, para así poder ejecutar el análisis en la memoria RAM.

COMANDO	`$ python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin' --profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_yarascan -y malware_rules.yar`

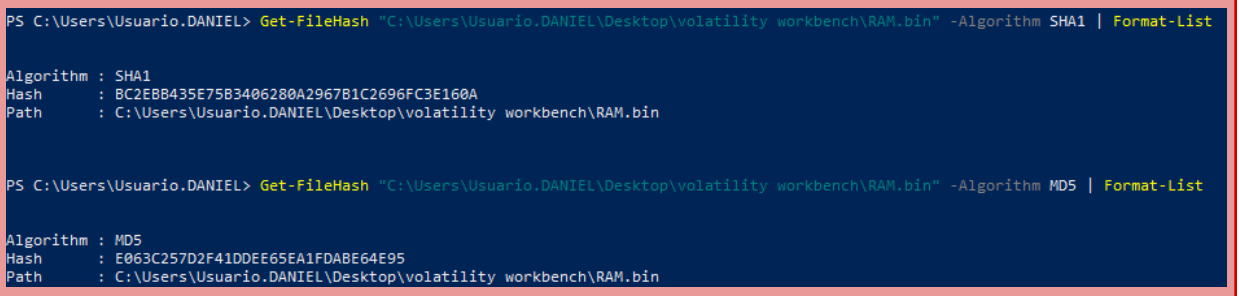
Como podemos comprobar en el OUTPUT, en este caso no ha encontrado nada, por lo que podemos suponer que no hay ningún malware de las familias más conocidas ejecutándose en la memoria RAM.

FRAGMENTO DE SALIDA

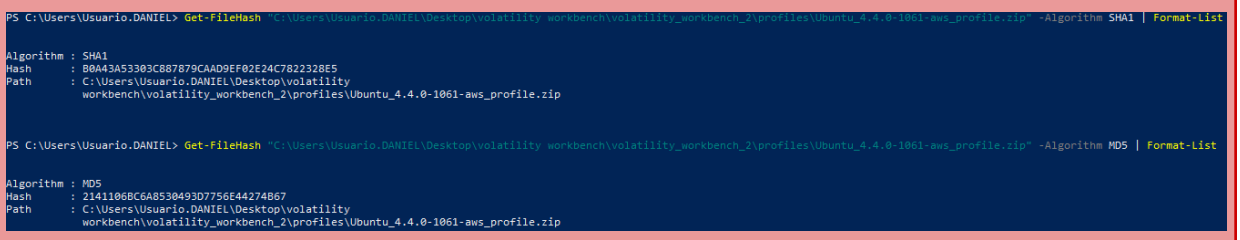
```
Volatility Foundation Volatility Framework 2.6.1
```

ANEXO

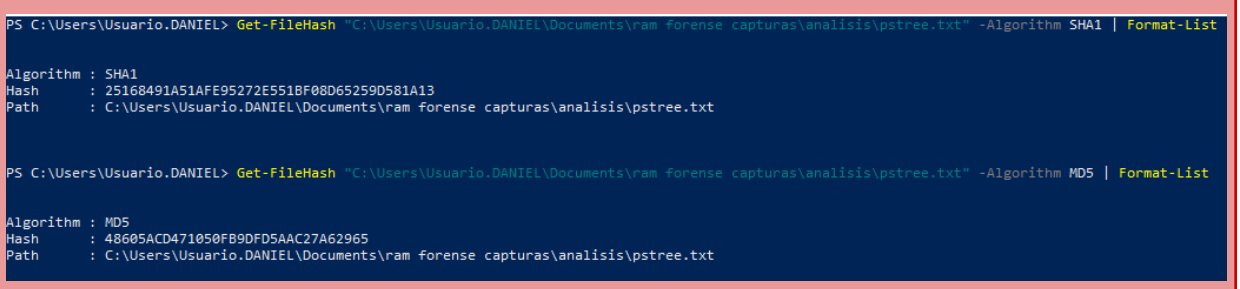
MEMORIA RAM

RAM	https://drive.google.com/file/d/1a32mvxnNhVIJDvtfJxji5mvcgLQMuhDx/view?usp=sharing	
CAPTURA		
HASH	SHA1	bc2ebb435e75b3406280a2967b1c2696fc3e160a
	MD5	e063c257d2f41ddee65ea1fdabe64e95

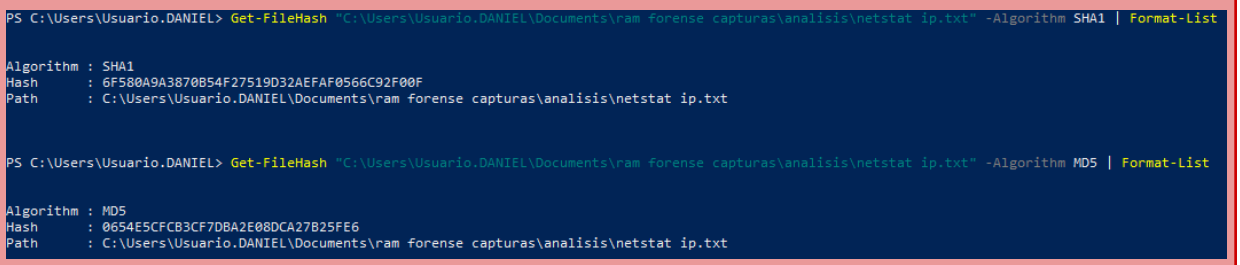
PERFIL RAM

PERFIL	https://drive.google.com/file/d/1aet1uvfbV9l7TGAwmN6Vny_DuPL-eA9c/view?usp=share_link	
CAPTURA		
HASH	SHA1	b0a43a53303c887879caad9ef02e24c7822328e5
	MD5	2141106bc6a8530493d7756e44274b67

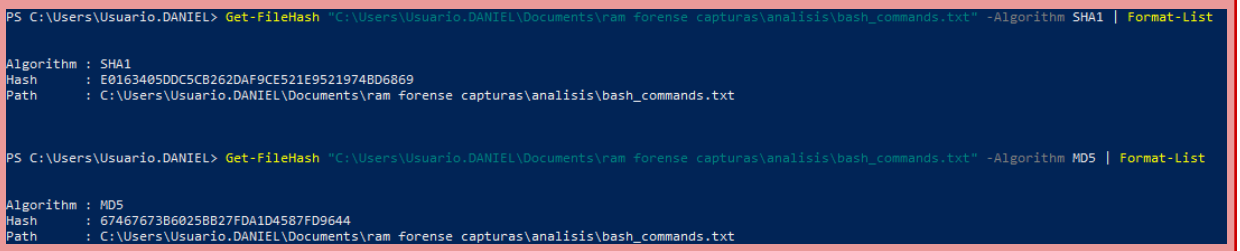
PSTREE

PSTREE	https://drive.google.com/file/d/1jYErjPZIciggAkdnFkE3abJniNCFRQzK/view?usp=share_link	
CAPTURA		
HASH	SHA1	25168491A51AFE95272E551BF08D65259D581A13
	MD5	48605ACD471050FB9DFD5AAC27A62965

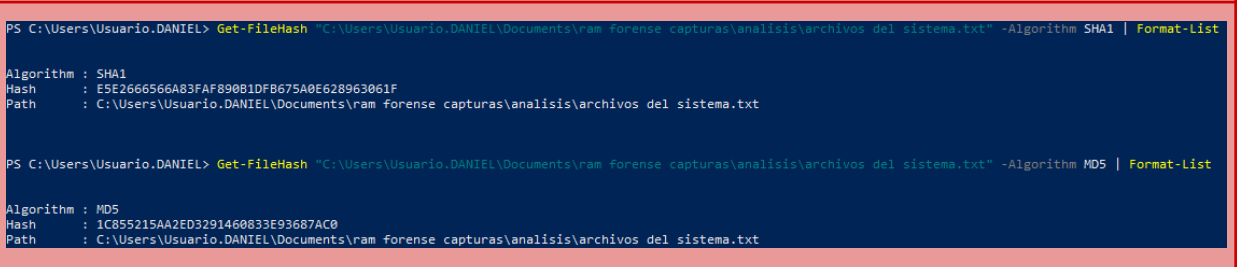
NETSTAT

NETSTAT	https://drive.google.com/file/d/1_4odItztwHF1y9xMhkSbrpp1Yn72IMxk/view?usp=share_link	
CAPTURA		
HASH	SHA1	6F580A9A3870B54F27519D32AEFAF0566C92F00F
	MD5	0654E5CFCB3CF7DBA2E08DCA27B25FE6

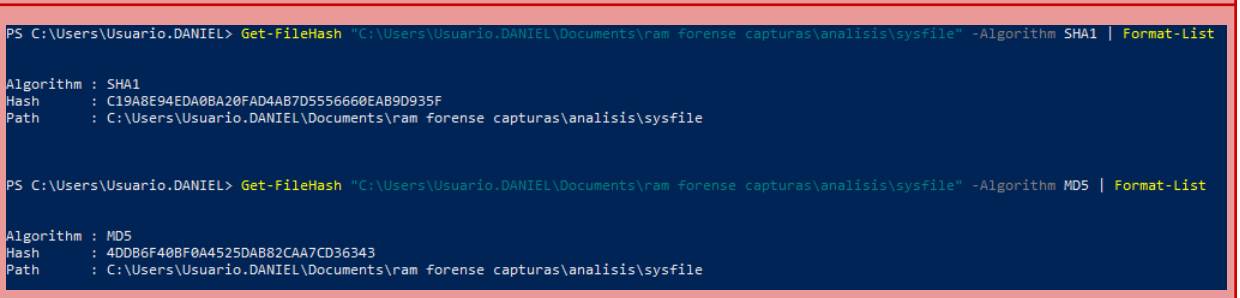
BASH_COMMANDS

BASH_COMMANDS	https://drive.google.com/file/d/1TIK9hsrsu615uG8kqfCnIxkgwcUCP4-1/view?usp=share_link	
CAPTURA		
HASH	SHA1	E0163405DDC5CB262DAF9CE521E9521974BD6869
	MD5	67467673B6025BB27FDA1D4587FD9644

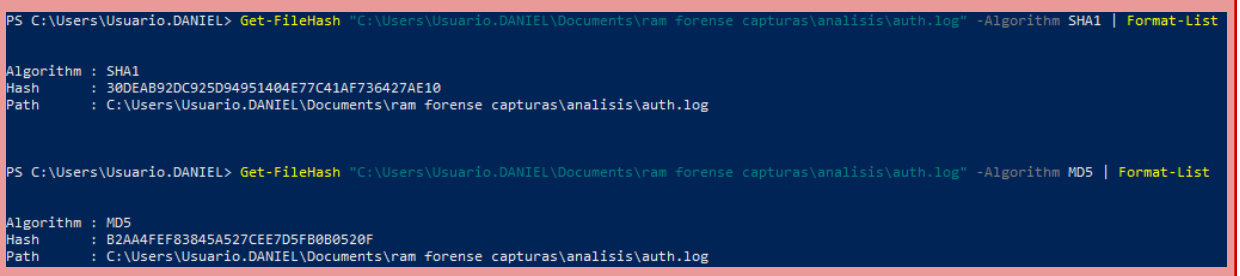
ARCHIVOS DEL SISTEMA

ARCHIVO DEL SISTEMA	https://drive.google.com/file/d/1x1W4LQSzWPGOjy6zNOfpuXs77codNDII/view?usp=share_link	
CAPTURA		
HASH	SHA1	E5E2666566A83FAF890B1DFB675A0E628963061F
	MD5	1C855215AA2ED3291460833E93687AC0

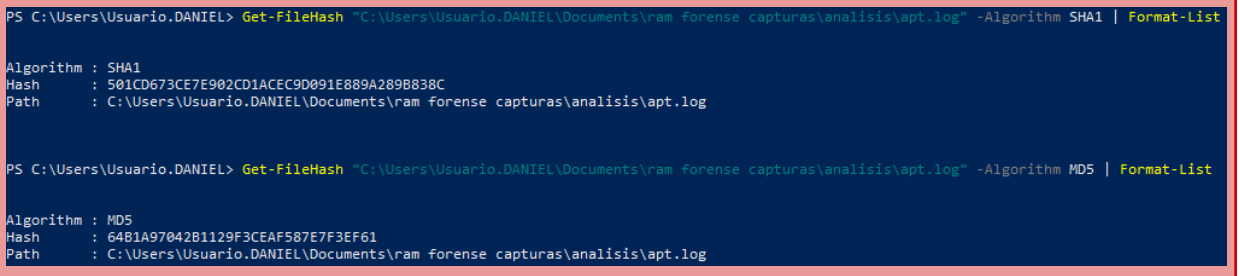
SYSFILE

SYSFILE	https://drive.google.com/file/d/1aBQp1M1JmKAjb2x78i05AKGd6C7SqHne/view?usp=share_link	
CAPTURA		
HASH	SHA1	C19A8E94EDA0BA20FAD4AB7D5556660EAB9D935F
	MD5	4DDB6F40BF0A4525DAB82CAA7CD36343

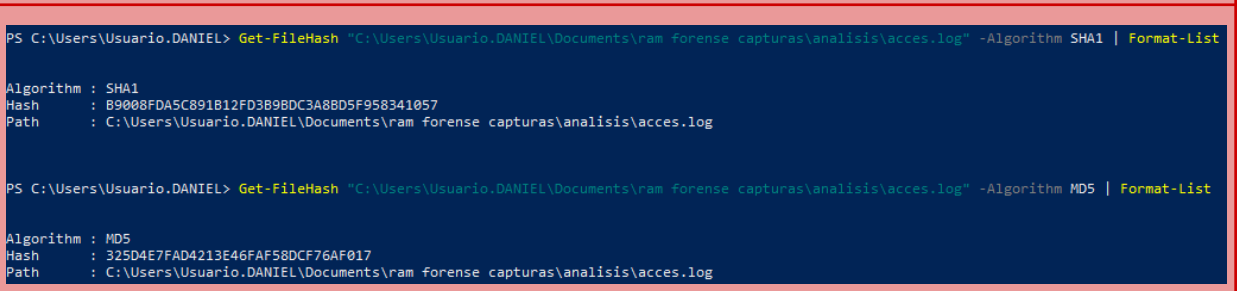
AUTH

AUTH	https://drive.google.com/file/d/1wVq7U8ZJTTsGlay6y25tb1xnxmptkEvL/view?usp=share_link	
CAPTURA		
HASH	SHA1	30DEAB92DC925D94951404E77C41AF736427AE10
	MD5	B2AA4FEF83845A527CEE7D5FB0B0520F

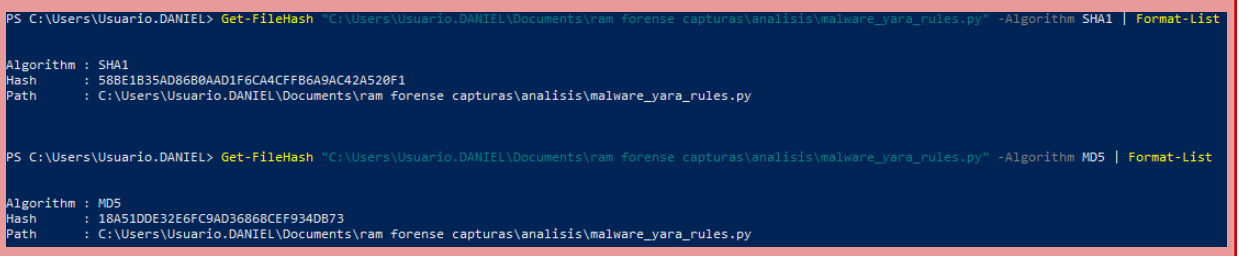
APT

APT	https://drive.google.com/file/d/105o691sAHJKQqVwuNXyB5FHCEDdepsbA/view?usp=share_link	
CAPTURA		
HASH	SHA1	501CD673CE7E902CD1ACEC9D091E889A289B838C
	MD5	64B1A97042B1129F3CEAF587E7F3EF61

APACHE ACCESS

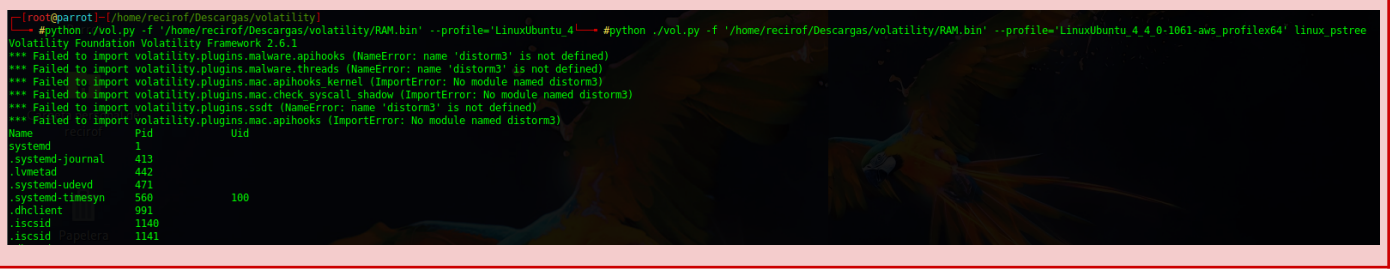
ACCESS		https://drive.google.com/file/d/1luGGogCl9nk07qQsMzE25_nom5LlvR_c/view?usp=share_link
CAPTURA		
HASH	SHA1	B9008FDA5C891B12FD3B9BDC3A8BD5F958341057
	MD5	325D4E7FAD4213E46FAF58DCF76AF017

YARA_RULES

YARA RULES		https://drive.google.com/file/d/11Eg_-dHZ6qniB55Wc6k0095j3NHuaQzh/view?usp=share_link
CAPTURA		
HASH	SHA1	58BE1B35AD86B0AAD1F6CA4CFFB6A9AC42A520F1
	MD5	18A51DDE32E6FC9AD36868CEF934DB73

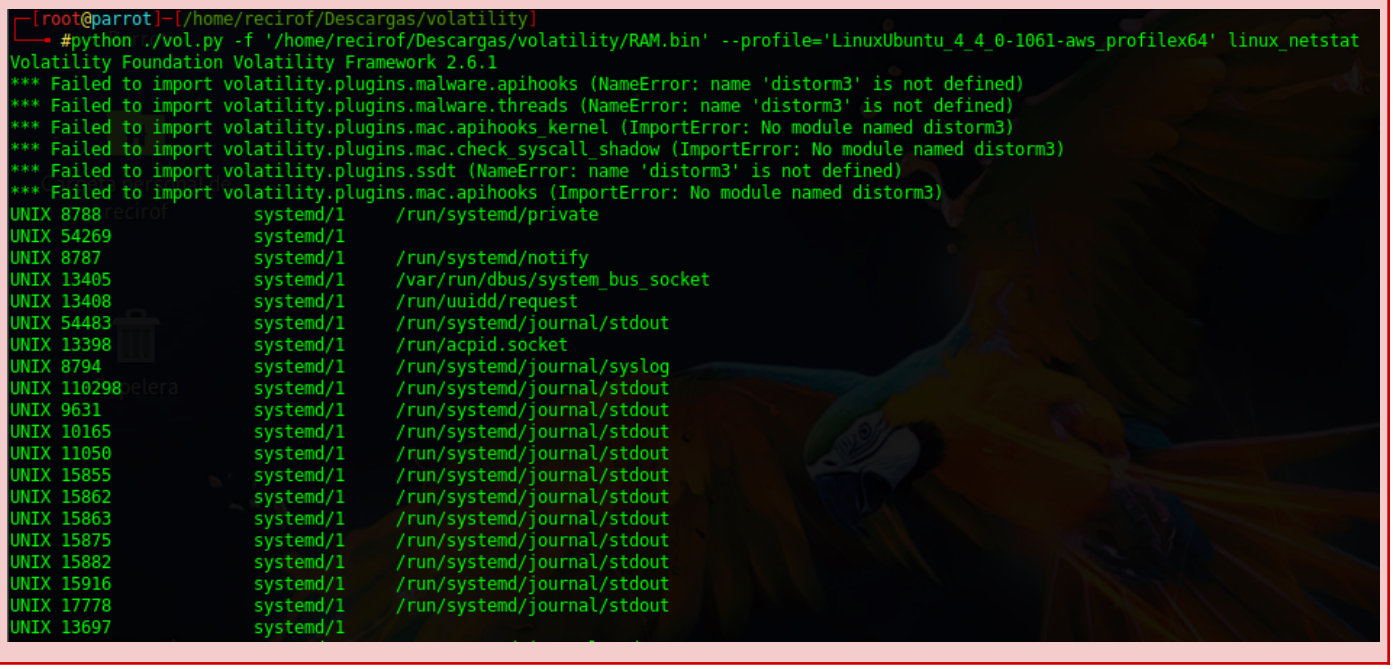
CAPTURA DE COMANDO DE PRUEBA

CAPTURA



CAPTURA DE NETSTAT

CAPTURA



CAPTURA DE COMANDOS DE BASH

CAPTURA

```
┌─[root@parrot]─[/home/recirof/Descargas/volatility]
└──╼ #python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin' --profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_bash
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.ssdt (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
Pid      Name                 Command Time                   Command
-------- -------------------- ------------------------------ -------
    9126 bash                 2018-07-24 05:24:19 UTC+0000   grep Listen ../*
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo vi ../ports.conf
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo a2enmod ssl
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo chown -R www-data:www-data wordpress
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo letsencrypt --apache -d ganga.site -d www.ganga.site
    9126 bash                 2018-07-24 05:24:19 UTC+0000   history
    9126 bash                 2018-07-24 05:24:19 UTC+0000   ifconfig
    9126 bash                 2018-07-24 05:24:19 UTC+0000   cd /etc/apache
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo apt install apache2 libapache2-mod-php php-mysql
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo apt upgrade
    9126 bash                 2018-07-24 05:24:19 UTC+0000   mysql -uadmin -p -hganga.ctmbcxcdb3us.eu-central-1.rds.amazonaws.com ganga
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo service apache2 restart
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo apt update
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo letsencrypt --apache -d ganga.site -d www.ganga.site
    9126 bash                 2018-07-24 05:24:19 UTC+0000   apt-cache search certbot
    9126 bash                 2018-07-24 05:24:19 UTC+0000   mysql -uadmin -p -hganga.ctmbcxcdb3us.eu-central-1.rds.amazonaws.com ganga
    9126 bash                 2018-07-24 05:24:19 UTC+0000   sudo letsencrypt --apache -d ganga.site -d www.ganga.site
    9126 bash                 2018-07-24 05:24:19 UTC+0000   cd /etc/apache2
```

CAPTURA ARCHIVOS DEL SISTEMA

CAPTURA

```
┌─[root@parrot]─[/home/recirof/Descargas/volatility]
└──╼ #python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin' --profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_find_file --listfiles
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.ssdt (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
Inode Number          Inode File Path
--------------- ------------------ ---------
              2 0xffff88003d5202e8 /dev
            320 0xffff880036489248 /dev/vhost-net
            319 0xffff88003cb48fb8 /dev/mqueue
            318 0xffff88003cb482e8 /dev/hugepages
            317 0xffff8800370efb48 /dev/initctl
            316 0xffff8800370eebe8 /dev/log
            315 0xffff8800367714d8 /dev/shm
            314 0xffff880036771248 /dev/autofs
            312 0xffff8800364d7b48 /dev/btrfs-control
            310 0xffff8800370a2be8 /dev/vcsa6
            308 0xffff8800370a1f18 /dev/vcs6
            306 0xffff8800370a1c88 /dev/vcsa5
            304 0xffff8800370a0fb8 /dev/vcs5
            302 0xffff8800370a0d28 /dev/vcsa4
            300 0xffff8800370a02e8 /dev/vcs4
            298 0xffff8800364d78b8 /dev/vcsa3
            296 0xffff8800364d6be8 /dev/vcs3
            294 0xffff8800364d6958 /dev/vcsa2
            292 0xffff8800364d5f18 /dev/vcs2
            178 0xffff880036fa3b48 /dev/disk
            181 0xffff8800371bf628 /dev/disk/by-uuid
            182 0xffff8800371bf108 /dev/disk/by-uuid/4a67ec61-9cd5-4a26-b00f-9391a34c8a29
            179 0xffff880036fa38b8 /dev/disk/by-label
            180 0xffff880036fa3628 /dev/disk/by-label/cloudimg-rootfs
```

CAPTURA YARA

CAPTURA

```
┌──[✗]─[root@parrot]─[/home/recirof/Descargas/volatility]
└──╼ #python ./vol.py -f '/home/recirof/Descargas/volatility/RAM.bin' --profile='LinuxUbuntu_4_4_0-1061-aws_profilex64' linux_yarascan
 -Y malware_rules.yar
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.ssdt (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
```