2.4 Edition

Development build and wiki:
github.com/volatilityfoundation

Download a stable release:
volatilityfoundation.org

Read the book:
artofmemoryforensics.com

Development Team Blog:
http://volatility-labs.blogspot.com

(Official) Training Contact:
voltraining@memoryanalysis.net

Follow: @volatility
Learn: www.memoryanalysis.net

Basic Usage

Typical command components:
`# vol.py -f [image] --profile=[profile] [plugin]`

Display profiles, address spaces, plugins:
`# vol.py --info`

Display global command-line options:
`# vol.py --help`

Display plugin-specific arguments:
`# vol.py [plugin] --help`

Load plugins from an external directory:
`# vol.py --plugins=[path] [plugin]`

Specify a DTB or KDBG address:
`# vol.py --dtb=[addr] --kdbg=[addr]`

Specify an output file:
`# vol.py --output-file=[file]`

Image Identification

Get profile suggestions (OS and architecture):
imageinfo

Find and parse the debugger data block:
kdbgscan

Processes Listings

Basic active process listing:
pslist

Scan for hidden or terminated processes:
psscan

Cross reference processes with various lists:
psxview

Show processes in parent/child tree:
pstree

Process Information

Specify –o/--offset=OFFSET or -p/--pid=1,2,3

Display DLLs:
dlllist

Show command line arguments:
cmdline

Display details on VAD allocations:
vadinfo [--addr]

Dump allocations to individual files:
vaddump --dump-dir=PATH [--base]

Dump all valid pages to a single file:
memdump --dump-dir=PATH

Display open handles:
handles

-t/--object-type=TYPE	Mutant, File, Key, etc…
-s/--silent	Hide unnamed handles

Display privileges:
privs

-r/--regex=REGEX	Regex privilege name
-s/--silent	Explicitly enabled only

Display SIDs:
getsids

Display environment variables:
envars

PE File Extraction

Specify -D/--dump-dir to any of these plugins to identify your desired output directory.

Dump a kernel module:
moddump

-r/--regex=REGEX	Regex module name
-b/--base=BASE	Module base address

Dump a process:
procdump

-m/--memory	Include memory slack

Dump DLLs in process memory:
dlldump

-r/--regex=REGEX	Regex module name
-b/--base=BASE	Module base address

Injected Code

Specify –o/--offset=OFFSET or -p/--pid=1,2,3

Find and extract injected code blocks:
malfind

-D/--dump-dir=PATH	Dump findings here

Cross-reference DLLs with memory mapped files:
ldrmodules

Scan a block of code in process or kernel memory for imported APIs:
impscan

-p/--pid=PID	Process ID
-b/--base=BASE	Base address to scan
-s/--size=SIZE	Size to scan from start of base

Logs / Histories

Recover event logs (XP/2003):
evtlogs

-S/--save-evt	Save raw event logs
-D/--dump-dir=PATH	Write to this directory

Recover command history:
cmdscan and consoles

Recover IE cache/Internet history:
iehistory

Show running services:
svcscan

-v/--verbose	Show ServiceDll from registry

Networking Information

Active info (XP/2003):
connections and sockets

Scan for residual info (XP/2003):
connscan and sockscan

Network info for Vista, 2008, and 7:
netscan

Kernel Memory

Display loaded kernel modules:
modules

Scan for hidden or residual modules:
modscan

Display recently unloaded modules:
unloadedmodules

Display timers and associated DPCs:
timers

Display kernel callbacks, notification routines:
callbacks

Audit the SSDT
ssdt

-v/--verbose	Check for inline API hooks

Audit the IDT and GDT:
idt (x86 only)
gdt (x86 only)

Audit driver dispatch (IRP) tables:
driverirp

-r/--regex=REGEX	Regex driver name

Display device tree (find stacked drivers):
devicetree

Print kernel pool tag usage stats:
pooltracker

-t/--tags=TAGS	List of tags to analyze
-T/--tagfile=FILE	pooltag.txt for labels

Kernel Objects

Scan for driver objects:
driverscan

Scan for mutexes:
mutantscan
```
  -s/--silent    Hide unnamed mutants
```

Scan for used/historical file objects:
filescan

Scan for symbolic link objects (shows drive mappings):
symlinkscan

Registry

Display cached hives:
hivelist

Print a key's values and data:
printkey
```
  -o/--hive_offset=OFFSET   Hive address (virtual)
  -K/--key=KEY              Key path
```

Dump userassist data:
userassist

Dump shellbags information:
shellbags

Dump the shimcache:
shimcache

Timelines

To create a timeline, create output in body file format. Combine the data and run sleuthkit's mactime to create a CSV file.

```
timeliner --output=body > time.txt
shellbags --output=body >> time.txt
mftparser --output=body >> time.txt

mactime –b [time.txt] [-d] > csv.txt
```

Volshell

List processes:
```
>>> ps()
```

Switch contexts by pid, offset, or name:
```
>>> cc(pid = 3028)
>>> cc(offset = 0x3eb31340, physical=True)
>>> cc(name = "explorer.exe")
```

Acquire a process address space after using cc:
```
>>> process_space =
proc().get_process_address_space()
```

Disassemble data in an address space
```
>>> dis(address, length, space)
```

Dump bytes, dwords or qwords:
```
>>> db(address, length, space)
>>> dd(address, length, space)
>>> dq(address, length, space)
```

Display a type/structure:
```
>>> dt("_EPROCESS", recursive = True)
```

Display a type/structure instance:
```
>>> dt("_EPROCESS",  0x820c92a0)
```

Create an object in kernel space:
```
>>> thread = obj.Object("_ETHREAD", offset =
0x820c92a0, vm = addrspace())
```

Dump Conversion

Create a raw memory dump from a hibernation, crash dump, firewire acquisition, virtualbox, vmware snapshot, hpak, or EWF file:
imagecopy –O/--output-image=FILE

Convert any of the aforementioned file types to a Windows crash dump compatible with Windbg:
raw2dmp –O/--output-image=FILE

API Hooks

Scan for API hooks:
apihooks
```
  -R/--skip-kernel     Don't check kernel modules
  -P/--skip-process    Don't check processes
  -Q/--quick           Scan faster
```

Yara Scanning

Scan for Yara signatures:
yarascan
```
  -p/--pid=PID               Process IDs to scan
  -K/--kernel                Scan kernel memory
  -Y/--yara-rules=RULES      String, regex, bytes, etc.
  -y/--yara-file=FILE        Yara rules file
  -W/--wide                  Match Unicode strings
  -s/--size                  Size of preview bytes
```

File System Resources

Scan for MFT records:
mftparser
```
  --output=body    Output body format
  -D/--dump-dir    Dump MFT-resident data
```

Extract cached files (registry hives, executables):
dumpfiles
```
  -D/--dump-dir=PATH     Output directory
  -r/--regex=REGEX       Regex filename
```

Parse USN journal records:
usnparser (github.com/tomspencer)

GUI Memory

Sessions (shows RDP logins):
sessions

Window stations (shows clipboard owners):
wndscan

Desktops (find ransomware):
Deskscan

Display global and session atom tables:
atoms and atomscan

Dump the contents of the clipboard:
clipboard

Detect message hooks (keyloggers):
messagehooks

Take a screen shot from the memory dump:
screenshot --dump-dir=PATH

Display visible and hidden windows:
windows and wintree

Strings

Use GNU strings or Sysinternals strings.exe:
```
strings -a -td FILE > strings.txt
strings -a -td -el FILE >> strings.txt (Unicode)

strings.exe -q -o > strings.txt (Windows)
```

Translate the string addresses:
strings
```
  -s/--string-file=FILE    Input strings.txt file
  -S/--scan
```

Password Recovery

Dump LSA secrets:
lsadump

Dump cached domain hashes:
cachedump

Dump LM and NTLM hashes:
hashdump (x86 only)

Extract OpenVPN credentials:
openvpn (github.com/Phaeilo)

Extract RSA private keys and certificates:
dumpcerts
```
  -s/--ssl     Parse certificates with openssl
```

Disk Encryption

Recover cached TrueCrypt passphrases:
truecryptpassphrase

Triage TrueCrypt artifacts:
truecryptsummary

Extract TrueCrypt master keys
truecryptmaster

Malware Specific

Dump Zeus/Citadel RC4 keys:
zeusscan and citadelscan

Find and decode Poison Ivy configs:
poisonivyconfig

Decode Java RAT config:
javaratscan (github.com/Rurik)

General Investigations	
Dump the system's raw registry hive files	dumpfiles -p 4 --regex='(config|ntuser)' --ignore-case --name -D ./
Create a Graphviz diagram of processes	psscan --output=dot --output-file=graph.dot
Create a color coded diagram of processes memory	vadtree -p PID --output=dot --output-file=graph.dot
Translate an account SID to user name	printkey -K "Microsoft\\Windows NT\\CurrentVersion\\ProfileList\\[SID]" | grep ProfileImagePath
List run keys for HKLM and all users	printkey -K "Microsoft\\Windows\\CurrentVersion\\Run" printkey -K "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
Find Unicode hostnames or URLs	yarascan -Y "/(www|http).+\.(com|net|org)/" --wide [--kernel]
Find null-terminated ASCII dot quad IP addresses	yarascan -Y "/([0-9]{1,3}\.){3}[0-9]{1,3}\x00/" --wide [--kernel]
Locate and extract the HOSTS file to local directory	filescan | egrep hosts$ | awk '{print $1}' 0x0000000005e3c6d8 dumpfiles -Q 0x0000000005e3c6d8 --name -D ./
Extract the admin password hash	hashdump | grep Administrator > admin.txt
Malicious Code	
Check if a process has domain or enterprise admin	getsids | egrep '(Domain|Enterprise)'
Identify processes with raw sockets	handles -t File | grep "\\Device\\RawIp\\0"
Look for explicit enabled debug privilege	privs --silent --regex=debug
Identify alternate data streams	mftparser | grep "DATA ADS"
Dump MFT-resident batch scripts	mftparser -D output/ file output/* | grep "DOS batch file"
Determine what is spying on the clipboard	wndscan | grep ClipViewer
Dump injected code and focus on executables	malfind -D output/ file output/* | grep PE
Trace API hooks through memory	apihooks -p PID --quick | grep 'Hook address' 0x1da654f echo "dis(0x1da654f, length = 512)" | volshell -p PID
Scan for a specific mutex on the system	mutantscan | grep [-i] [MUTANT NAME]
Dump injected DLL, fix image base + IDA import labels	dlldump --base=ADDR -p PID -D./ --fix –memory impscan --base=ADDR -p PID --output=idc > labels.idc
Find binaries loaded from temporary directories	envars -p PID | grep TEMP | awk '{print $5}' C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp Filter dlllist and modules output for the specified path
User Activity	
Detect remote mapped shares	handles -t File | egrep "\\Device\\(LanmanRedirector|Mup)"
Files on Truecrypt volumes	filescan | grep TrueCryptVolume
Extract ASCII and Unicode clipboard content	clipboard | grep TEXT
Brute force search for command history	yarascan -Y "/C:\\\\.+>/" --wide [--kernel]
Recently clicked applications and shortcuts	userassist | grep REG_BINARY
Find prefetch files (recently executed programs)	mftparser | grep \.pf$ | awk '{print $NF}'
Kernel Memory	
Identify hooked driver dispatch tables	driverirp --regex=tcpip | grep IRP | egrep -vi '(tcpip|ntos)'
Look for hooked SSDT functions	ssdt | egrep –vi '(ntos|win32k) '
Malicious kernel callbacks and timers	callbacks | grep UNKNOWN (same with timers)
Locate hidden thread-based kernel rootkits	threads -F OrphanThread | grep StartAddress
Speed Enhancements	
Find and set the kernel DTB	psscan | grep System | awk '{print $5}' 0x00319000 (Now use --dtb=0x00319000)
Find and set the KDBG on XP-7 and 32-bit 8	kdbgscan | grep Offset | grep V | uniq Offset (V) : 0xf80002803070 (add to --kdbg)
Find and set the KDBG on 64-bit 8 and 2012	kdbgscan --profile=[PROFILE] | grep KdCopyDataBlock KdCopyDataBlock (V) : 0xf80281ff5ea0 (add to --kdbg)
Volshell Scripting	
Create a process ID lookup table	by_pid = dict((p.UniqueProcessId, p) for p in getprocs()) parent_name = by_pid[PID].ImageFileName

Scan process memory and print a hex dump	needles = ["abc123", "def456"]
	for hit in proc().search_process_memory(needles):
	    db(hit)

Extract a chunk of kernel memory to disk	data = addrspace().zread(ADDR, SIZE)
	with open("output.bin", "wb") as handle:
	    handle.write(data)

Translate a kernel address and seek to it (raw dumps only)	echo "addrspace().vtop(0x98dfd9c8)" | volshell -f [MEMDUMP] 597989832 xxd -s 597989832 [MEMDUMP]
Kernel modules with embedded PE signatures	signed = [mod for mod in getmods() if mod.sec_dir()]

Linux Commands

Processes Listings

Basic active process listing:
linux_pslist

List processes and threads:
linux_pidhashtable

Cross reference processes with various lists:
linux_psxview

Show processes in parent/child tree:
linux_pstree

Process Information

Specify –o/--offset=OFFSET or -p/--pid=1,2,3

Display shared libraries:
linux_library_list

List threads:
linux_threads

Show command line arguments:
linux_psaux

Display details on memory ranges:
linux_proc_maps

Dump allocations to individual files:
linux_dump_map
```
      -D/--dump-dir=PATH
      --vma=ADDR     Range to dump
```

Display open handles:
linux_lsof

Display environment variables:
linux_psenv and linux_bash_env

ELF File Extraction

Specify -D/--dump-dir to any of these plugins to identify your desired output directory.

Dump a kernel module:
linux_moddump
```
   -r/--regex=REGEX   Regex module name
   -b/--base=BASE     Module base address
```

Dump a process:
linux_procdump

Dump shared libraries in process memory:
linux_librarydump
```
   -r/--regex=REGEX   Regex module name
   -b/--base=BASE     Module base address
```

Injected Code

Specify –o/--offset=OFFSET or -p/--pid=1,2,3

Find and extract injected code blocks:
linux_malfind

Cross-reference shared libraries with memory-mapped files:
linux_ldrmodules

Check for process hollowing:
linux_process_hollow
```
    -b/--base    Base address of ELF file in memory
    -P/--path    Path of known good file on disk
```

Command History

Recover command history:
linux_bash

Recover executed binaries:
linux_bash_hash

Networking Information

Active info:
linux_netstat

Interface information:
linux_ifconfig

Raw sockets:
linux_list_raw

Routing cache:
linux_route_cache
```
    -R/--resolve    DNS resolve destination IPs
```

Netfilter entries:
linux_netfilter

ARP cache:
linux_arp

Kernel Memory

Display loaded kernel modules:
linux_lsmod

Check for system call hooks:
linux_check_syscall

Check for network stack hooks:
linux_check_afinfo

Check for credential copying:
linux_check_creds

Check for file operations hooking:
linux_check_fop

Check for inline kernel hooks:
linux_check_inline_kernel

Check for hidden modules:
linux_check_modules
linux_hidden_modules

Check for TTY hooks:
linux_check_tty

Check for malicious keyboard callbacks:
linux_keyboard_notifiers

Print the kernel debug buffer:
linux_dmesg

Audit the IDT:
linux_idt (x86 only)

Userland API Hooks

Scan for API hooks:
linux_apihooks
```
    -a/--all        Check hooked PLT entries
```

Scan for GOT/PLT hooks:
linux_plthook
```
    -a/--all       List all PLT entries
    -i/--ignore    Libraries to ignore in processing
```

Yara Scanning

Scan for Yara signatures:
linux_yarascan
```
   -p/--pid=PID              Process IDs to scan
   -K/--kernel               Scan kernel memory
   -Y/--yara-rules=RULES     String, regex, bytes, etc.
   -y/--yara-file=FILE       Yara rules file
   -W/--wide                 Match Unicode strings
   -s/--size                 Size of preview bytes
```

File System Resources

List mount points:
linux_mount

Enumerate files:
linux_enumerate_files

Extract cached files:
linux_find_file
```
   -F/--find=FILE        Path of file to find
   -i/--inode=INODE      Address of inode to dump
   -L/--listfiles        Lists files in cache
   -O/--outputfile       File path to write
```

Disk Encryption

Recover cached Truecrypt passphrases:
linux_truecryptpassphrase

Strings

Translate extracted strings:
linux_strings
```
   -s/--string-file=FILE    Input strings.txt file
```

Mac OS X Commands

Processes Listings

Basic active process listing:
mac_pslist

List PID hash table:
mac_pid_hash_table

List tasks:
mac_tasks

Cross reference processes with various lists:
mac_psxview

Show processes in parent/child tree:
mac_pstree

Process Information

Specify –o/--offset=OFFSET or -p/--pid=1,2,3

Display shared libraries:
mac_dyld_maps

Show command line arguments:
mac_psaux

Display details on memory ranges:
mac_proc_maps

Dump allocations to individual files:
```
mac_dump_map
    -D/--dump-dir=PATH
   --map_address=ADDR
```

Display open handles:
mac_lsof

Display environment variables:
mac_psenv and mac_bash_env

Display login sessions:
mac_list_sessions

Mach-O File Extraction

Specify -D/--dump-dir to any of these plugins to identify your desired output directory.

Dump a kernel module:
```
mac_moddump
  -r/--regex=REGEX   Regex module name
  -b/--base=BASE     Module base address
```

Dump a process:
mac_procdump

Dump shared libraries in process memory:
```
mac_librarydump
  -b/--base=BASE     Module base address
```

Injected Code

Specify –o/--offset=OFFSET or -p/--pid=1,2,3

Find and extract injected code blocks:
mac_malfind

Cross-reference shared libraries with memory-mapped files:
mac_ldrmodules

Command History

Recover command history:
mac_bash

Recover executed binaries:
mac_bash_hash

Networking Information

Active info:
mac_netstat

Active info from network stack:
mac_network_conns

Interface Information:
mac_ifconfig

ARP cache:
mac_arp

Route table:
mac_route

Socket filters:
mac_socket_filters

IP filters:
mac_ip_filters

Kernel Memory

Display loaded kernel modules:
mac_lsmod

Check for kernel API hooks:
mac_apihooks_kernel

Check for system call hooks:
mac_check_syscalls

Check for shadow system call table:
mac_check_syscall_shadow

Check sysctl handlers:
mac_check_sysctl

Check the trap table:
mac_check_trap_table

Check the mig table:
mac_check_mig_table

Check for file operations hooking:
mac_check_fop

Check for inline kernel hooks:
mac_check_inline_kernel

Check for hidden modules:
mac_lsmod_iokit
mac_lsmod_kext_map

Check for TrustedBSD hooks:
mac_trustedbsd

Print the kernel debug buffer:
mac_dmesg

API Hooks

Scan for API hooks:
```
mac_apihooks
  -R/--skip-kernel     Don't check kernel modules
  -P/--skip-process    Don't check processes
  -Q/--quick           Scan faster
```

Check for process hollowing:
```
mac_process_hollow
   -b/--base    Base address of ELF file in memory
   -P/--path    Path of known good file on disk
```

Scan for GOT/PLT hooks:
```
mac_plthook
   -a/--all      List all PLT entries
   -i/--ignore   Libraries to ignore in processing
```

Yara Scanning

Scan for Yara signatures:
```
mac_yarascan
  -p/--pid=PID               Process IDs to scan
  -K/--kernel                Scan kernel memory
  -Y/--yara-rules=RULES      String, regex, bytes, etc.
  -y/--yara-file=FILE        Yara rules file
  -W/--wide                  Match Unicode strings
  -s/--size                  Size of preview bytes
```

Disk Encryption

Recover possible Keychain keys:
mac_keychaindump

File System Resources

List mount points:
mac_mount

List cached files and their vnode addresses:
mac_list_files

Extract cached files:
```
mac_dump_file
   -q/--file_offset    Offset of vnode to dump
   -O/--outputfile     File path to write
```

Strings

Translate extracted string:
```
mac_strings
  -s/--string-file=FILE    Input strings.txt file
```

User Activity

Recover Adium messages, including OTR chat:
mac_adium

Recover Calendar entries:
mac_calendar

Recover contacts:
mac_contacts