A Toolset for Answering the Question: What Changed on Disk?

Stuart Maclean

Applied Physics Laboratory
University of Washington
stuart@apl.uw.edu

Open Source Digital Forensics Conference, 2013

Outline

1. Motivation
2. VMMount, Exposing Virtual Disk Content To The Host
3. TSK4J, A Java Binding For Sleuthkit
4. Armour, A Shell For File System Differencing
5. Conclusion

Motivation

The Question

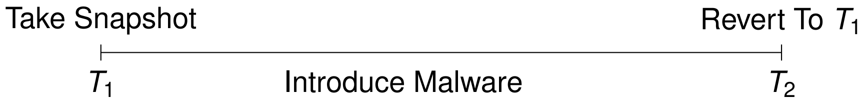
What impact does `nastyMalware.exe` have on my machine were I to run it, knowingly or otherwise?

More Generally

If I run my computer from time T_1 to time T_2, what are the impacts on the system in that time interval?

- Memory/process interaction
- Network activity
- Disk changes

Malware Analysis And Virtualization



Take Snapshot

Revert To T_1

T_1

Introduce Malware

T_2

Observation

The virtual machine's entire disk contents at times T_1, T_2 are recorded.

Introducing VMMount

What Is It?

- A tool to expose virtual machine disk content to a host file system.

What Can It Do?

- Understands virtual machine snapshots (disk part).
- Provides full seek, read, write(!) capability.

How It Is Built?

- FUSE-based, and uses existing FUSE4J Java-C bridge.
- Implemented for VirtualBox (.vdi) and VMWare (.vmdk) disk files.
- Uses no code from the virtualization engine itself.

VMMount In Action

```
$ cd /path/to/my/virtualMachines; mkdir mount
$ vmmount xpCuckoo xpRef mount
```

```
mount/xpCuckoo/sda   -> xpCuckoo/Snapshots/child2.vdi
mount/xpCuckoo/1/sda -> xpCuckoo/Snapshots/child1.vdi
mount/xpCuckoo/0/sda -> xpCuckoo/xpCuckoo.vdi
mount/xpRef/sda      -> xpRef/xpRef.vdi
mount/xpRef/sdb      -> xpRef/xpRefOther.vdi
```

- Handles multiple virtual machines.
- Exposes all disks.
- Exposes all snapshots/generations.
- Unix-style names for exposed virtual devices (arbitrary).
- Exposes whole disks, not partitions (others do this better).

VMMount In Action

```
$ cd /path/to/my/virtualMachines; mkdir mount
$ vmmount xpCuckoo xpRef mount

mount/xpCuckoo/sda   -> xpCuckoo/Snapshots/child2.vdi
mount/xpCuckoo/1/sda -> xpCuckoo/Snapshots/child1.vdi
mount/xpCuckoo/0/sda -> xpCuckoo/xpCuckoo.vdi
mount/xpRef/sda      -> xpRef/xpRef.vdi
mount/xpRef/sdb      -> xpRef/xpRefOther.vdi
```

- Handles multiple virtual machines.
- Exposes all disks.
- Exposes all snapshots/generations.
- Unix-style names for exposed virtual devices (arbitrary).
- Exposes whole disks, not partitions (others do this better).

VMMount In Action

```
$ cd /path/to/my/virtualMachines; mkdir mount
$ vmmount xpCuckoo xpRef mount

mount/xpCuckoo/sda   -> xpCuckoo/Snapshots/child2.vdi
mount/xpCuckoo/1/sda -> xpCuckoo/Snapshots/child1.vdi
mount/xpCuckoo/0/sda -> xpCuckoo/xpCuckoo.vdi
mount/xpRef/sda      -> xpRef/xpRef.vdi
mount/xpRef/sdb      -> xpRef/xpRefOther.vdi
```

- Handles multiple virtual machines.
- Exposes all disks.
- Exposes all snapshots/generations.
- Unix-style names for exposed virtual devices (arbitrary).
- Exposes whole disks, not partitions (others do this better).

VMMount In Action

```
$ cd /path/to/my/virtualMachines; mkdir mount
$ vmmount xpCuckoo xpRef mount

mount/xpCuckoo/sda   -> xpCuckoo/Snapshots/child2.vdi
mount/xpCuckoo/1/sda -> xpCuckoo/Snapshots/child1.vdi
mount/xpCuckoo/0/sda -> xpCuckoo/xpCuckoo.vdi
mount/xpRef/sda      -> xpRef/xpRef.vdi
mount/xpRef/sdb      -> xpRef/xpRefOther.vdi
```

- Handles multiple virtual machines.
- Exposes all disks.
- Exposes all snapshots/generations.
- Unix-style names for exposed virtual devices (arbitrary).
- Exposes whole disks, not partitions (others do this better).

VMMount In Action

```
$ cd /path/to/my/virtualMachines; mkdir mount
$ vmmount xpCuckoo xpRef mount
```

```
mount/xpCuckoo/sda   -> xpCuckoo/Snapshots/child2.vdi
mount/xpCuckoo/1/sda -> xpCuckoo/Snapshots/child1.vdi
mount/xpCuckoo/0/sda -> xpCuckoo/xpCuckoo.vdi
mount/xpRef/sda      -> xpRef/xpRef.vdi
mount/xpRef/sdb      -> xpRef/xpRefOther.vdi
```

- Handles multiple virtual machines.
- Exposes all disks.
- Exposes all snapshots/generations.
- Unix-style names for exposed virtual devices (arbitrary).
- Exposes whole disks, not partitions (others do this better).

VMMount In Action

```
$ cd /path/to/my/virtualMachines; mkdir mount
$ vmmount xpCuckoo xpRef mount

mount/xpCuckoo/sda   -> xpCuckoo/Snapshots/child2.vdi
mount/xpCuckoo/1/sda -> xpCuckoo/Snapshots/child1.vdi
mount/xpCuckoo/0/sda -> xpCuckoo/xpCuckoo.vdi
mount/xpRef/sda      -> xpRef/xpRef.vdi
mount/xpRef/sdb      -> xpRef/xpRefOther.vdi
```

- Handles multiple virtual machines.
- Exposes all disks.
- Exposes all snapshots/generations.
- Unix-style names for exposed virtual devices (arbitrary).
- Exposes whole disks, not partitions (others do this better).

Basic Operations On Virtual Disk Content

```
$ vmmount vmName mount

// Inspect the master boot record
$ xxd -l 512 mount/vmName/sda

// Extract 1000'th sector
$ dd if=mount/vmName/sda skip=1000 count=1

// Compare disk content over time, likely changed!
$ md5sum mount/vmName/0/sda mount/vmName/sda
```

Basic Operations On Virtual Disk Content

```
$ vmmount vmName mount

// Inspect the master boot record
$ xxd -l 512 mount/vmName/sda

// Extract 1000'th sector
$ dd if=mount/vmName/sda skip=1000 count=1

// Compare disk content over time, likely changed!
$ md5sum mount/vmName/0/sda mount/vmName/sda
```

Basic Operations On Virtual Disk Content

```
$ vmmount vmName mount

// Inspect the master boot record
$ xxd -l 512 mount/vmName/sda

// Extract 1000'th sector
$ dd if=mount/vmName/sda skip=1000 count=1

// Compare disk content over time, likely changed!
$ md5sum mount/vmName/0/sda mount/vmName/sda
```

Basic Operations On Virtual Disk Content

```
$ vmmount vmName mount

// Inspect the master boot record
$ xxd -l 512 mount/vmName/sda

// Extract 1000’th sector
$ dd if=mount/vmName/sda skip=1000 count=1

// Compare disk content over time, likely changed!
$ md5sum mount/vmName/0/sda mount/vmName/sda
```

Virtual Disk Differencing With Sleuthkit

Sleuthkit command line tools can infer the disk *structure*…

```
$ vmmount winxp mount

// volume systems: difference these outputs...
$ mmls mount/winxp/0/sda
$ mmls mount/winxp/sda

// file systems: difference these outputs...
$ fls -o 63 -r -m / mount/winxp/0/sda > T1.bodyfile
$ fls -o 63 -r -m / mount/winxp/sda   > T2.bodyfile
```

But how to compare? SQL?

Virtual Disk Differencing With Sleuthkit

Sleuthkit command line tools can infer the disk *structure*…

```
$ vmmount winxp mount

// volume systems: difference these outputs...
$ mmls mount/winxp/0/sda
$ mmls mount/winxp/sda

// file systems: difference these outputs...
$ fls -o 63 -r -m / mount/winxp/0/sda > T1.bodyfile
$ fls -o 63 -r -m / mount/winxp/sda   > T2.bodyfile
```

But how to compare? SQL?

Virtual Disk Differencing With Sleuthkit

Sleuthkit command line tools can infer the disk *structure*...

```
$ vmmount winxp mount

// volume systems: difference these outputs...
$ mmls mount/winxp/0/sda
$ mmls mount/winxp/sda

// file systems: difference these outputs...
$ fls -o 63 -r -m / mount/winxp/0/sda > T1.bodyfile
$ fls -o 63 -r -m / mount/winxp/sda   > T2.bodyfile
```

But how to compare? SQL?

Identifying Volume System Changes

Comparing `mmls` outputs will highlight any major disk alterations:

- New partitions
- Deleted partitions
- Resized partitions

It does not read partition content, so could not discover e.g.

- a Master Boot Record edit.
- malware hiding data in unallocated space.

Need a different tool for that. Everyone loves Java, so . . .

Introducing TSK4J

Using new Java binding to the Sleuthkit C library, walk the volume system of a virtual machine disk at times T_1, T_2 and compare content.

```
VolSystem vsT1=new VolSystem("mount/vmName/0/sda");
VolSystem vsT2=new VolSystem("mount/vmName/sda");
List<Partition> psT1 = vsT1.getPartitions();
List<Partition> psT2 = vsT2.getPartitions();
for( int i = 0; i < psT1.size(); i++ ) {
  Partition pT1 = psT1.get(i);
  if( pT1.isAllocated() )       // has a file system
    continue;
  Partition pT2 = psT2.get(i);
  InputStream isT1 = pT1.getInputStream();
  InputStream isT2 = pT2.getInputStream();
  // read data from InputStreams and compare
}
```

Introducing TSK4J

Using new Java binding to the Sleuthkit C library, walk the volume system of a virtual machine disk at times T_1, T_2 and compare content.

```
VolSystem vsT1=new VolSystem("mount/vmName/0/sda");
VolSystem vsT2=new VolSystem("mount/vmName/sda");
List<Partition> psT1 = vsT1.getPartitions();
List<Partition> psT2 = vsT2.getPartitions();
for( int i = 0; i < psT1.size(); i++ ) {
  Partition pT1 = psT1.get(i);
  if( pT1.isAllocated() )       // has a file system
    continue;
  Partition pT2 = psT2.get(i);
  InputStream isT1 = pT1.getInputStream();
  InputStream isT2 = pT2.getInputStream();
  // read data from InputStreams and compare
}
```

Introducing TSK4J

Using new Java binding to the Sleuthkit C library, walk the volume system of a virtual machine disk at times T_1, T_2 and compare content.

```
VolSystem vsT1=new VolSystem("mount/vmName/0/sda");
VolSystem vsT2=new VolSystem("mount/vmName/sda");
List<Partition> psT1 = vsT1.getPartitions();
List<Partition> psT2 = vsT2.getPartitions();
for( int i = 0; i < psT1.size(); i++ ) {
  Partition pT1 = psT1.get(i);
  if( pT1.isAllocated() )       // has a file system
    continue;
  Partition pT2 = psT2.get(i);
  InputStream isT1 = pT1.getInputStream();
  InputStream isT2 = pT2.getInputStream();
  // read data from InputStreams and compare
}
```

Introducing TSK4J

Using new Java binding to the Sleuthkit C library, walk the volume system of a virtual machine disk at times T_1, T_2 and compare content.

```
VolSystem vsT1=new VolSystem("mount/vmName/0/sda");
VolSystem vsT2=new VolSystem("mount/vmName/sda");
List<Partition> psT1 = vsT1.getPartitions();
List<Partition> psT2 = vsT2.getPartitions();
for( int i = 0; i < psT1.size(); i++ ) {
  Partition pT1 = psT1.get(i);
  if( pT1.isAllocated() )       // has a file system
    continue;
  Partition pT2 = psT2.get(i);
  InputStream isT1 = pT1.getInputStream();
  InputStream isT2 = pT2.getInputStream();
  // read data from InputStreams and compare
}
```

Introducing TSK4J

Using new Java binding to the Sleuthkit C library, walk the volume system of a virtual machine disk at times T_1, T_2 and compare content.

```
VolSystem vsT1=new VolSystem("mount/vmName/0/sda");
VolSystem vsT2=new VolSystem("mount/vmName/sda");
List<Partition> psT1 = vsT1.getPartitions();
List<Partition> psT2 = vsT2.getPartitions();
for( int i = 0; i < psT1.size(); i++ ) {
  Partition pT1 = psT1.get(i);
  if( pT1.isAllocated() )       // has a file system
    continue;
  Partition pT2 = psT2.get(i);
  InputStream isT1 = pT1.getInputStream();
  InputStream isT2 = pT2.getInputStream();
  // read data from InputStreams and compare
}
```

Virtual Disk File System Differencing

Sleuthkit's *BodyFile* structure provides a convenient unit of manipulation. A single *BodyFile Record* represents a single file within a file system. Fields include

- file name
- inode (MFT entry)
- size
- owner, group
- hash of content (optional)
- create time, access time, modified time

So file system changes can be posed as BodyFile element differences.

Introducing Armour

What Is It?

- A shell-like tool for comparing Sleuthkit BodyFiles and thus file systems.
- Defines unary and binary operators for what is essentially a set membership problem.

What Can It Do?

- Enables the user to identify new files, deleted files, changed files, accessed files, files with create time of `calc.exe`, and so on.

How Is It Built?

- Java, with some Swing UI components.
- Uses TSK4J and Sleuthkit for the heavy-lifting.

Example Armour Binary Operators

Operators requiring two bodyfiles A, B, perhaps from same disk at times T_1, T_2. $a \in A, b \in B$:

Member Equality Definition	Set Operation	Result(Files)
a.inode == b.inode and a.path == b.path	$B - A$	New
ditto	$A - B$	Deleted
a.f == b.f for all fields f	$A \cap B$	Unchanged
a.inode == b.inode and a.f ! = b.f for some other f	$A \cap B$	Any Change
a.inode == b.inode and a.modT == b.modT and a.hash != b.hash	$A \cap B$	Disguised Modified

Result is always another bodyfile (closure).

Example Armour Unary Operators

Operators requiring a single bodyfile:

- Name satisfies pattern, e.g. `/WINDOWS/System32/*`.
- Has same creation time as `calc.exe`.
- Is executable (inspects content, so requires volume be available)

Again, result is always another bodyfile.

Armour In Action — The Assets

```
$ armour mount/winxp/0/sda,63 mount/winxp/sda,63
armour> ls
1 mount/winxp/0/sda,63 (11091)
2 mount/winxp/sda,63   (11102)
armour> bops
1 New Files
2 Changed Files
3 Disguised Changed Files
4 Unchanged Files
5 Accessed Files
armour> uops
1 path matches /WINDOWS/.*
2 isDirectory
3 isExecutable
```

Armour In Action — The Assets

```
$ armour mount/winxp/0/sda,63 mount/winxp/sda,63
armour> ls
1 mount/winxp/0/sda,63 (11091)
2 mount/winxp/sda,63   (11102)
armour> bops
1 New Files
2 Changed Files
3 Disguised Changed Files
4 Unchanged Files
5 Accessed Files
armour> uops
1 path matches /WINDOWS/.*
2 isDirectory
3 isExecutable
```

Armour In Action — The Assets

```
$ armour mount/winxp/0/sda,63 mount/winxp/sda,63
armour> ls
1 mount/winxp/0/sda,63 (11091)
2 mount/winxp/sda,63   (11102)
armour> bops
1 New Files
2 Changed Files
3 Disguised Changed Files
4 Unchanged Files
5 Accessed Files
armour> uops
1 path matches /WINDOWS/.*
2 isDirectory
3 isExecutable
```

Armour In Action — The Assets

```
$ armour mount/winxp/0/sda,63 mount/winxp/sda,63
armour> ls
1 mount/winxp/0/sda,63 (11091)
2 mount/winxp/sda,63   (11102)
armour> bops
1 New Files
2 Changed Files
3 Disguised Changed Files
4 Unchanged Files
5 Accessed Files
armour> uops
1 path matches /WINDOWS/.*
2 isDirectory
3 isExecutable
```

Armour In Action — Posing Questions

```
armour>bop 1 2 1  // new files
[3]
armour>bop 1 1 2  // deleted files
[4]
armour>bop 2 2 1  // changed files
[5]
armour>bop 4 2 1  // unchanged files
[6]
armour>uop 3 3    // executable new files
[7]
```

Armour In Action — Posing Questions

```
armour>bop 1 2 1  // new files
[3]
armour>bop 1 1 2  // deleted files
[4]
armour>bop 2 2 1  // changed files
[5]
armour>bop 4 2 1  // unchanged files
[6]
armour>uop 3 3    // executable new files
[7]
```

Armour In Action — Posing Questions

```
armour>bop 1 2 1  // new files
[3]
armour>bop 1 1 2  // deleted files
[4]
armour>bop 2 2 1  // changed files
[5]
armour>bop 4 2 1  // unchanged files
[6]
armour>uop 3 3    // executable new files
[7]
```

Armour In Action — Posing Questions

```
armour>bop 1 2 1  // new files
[3]
armour>bop 1 1 2  // deleted files
[4]
armour>bop 2 2 1  // changed files
[5]
armour>bop 4 2 1  // unchanged files
[6]
armour>uop 3 3    // executable new files
[7]
```

Armour In Action — Posing Questions

```
armour>bop 1 2 1  // new files
[3]
armour>bop 1 1 2  // deleted files
[4]
armour>bop 2 2 1  // changed files
[5]
armour>bop 4 2 1  // unchanged files
[6]
armour>uop 3 3    // executable new files
[7]
```

Armour In Action — Viewing Results

```
armour> ls
1 mount/winxp/0/sda,63 (11091)
2 mount/winxp/sda,63 (11102)
3 New Files | winxp/sda,63 | winxp/0/sda,63 (11)
4 New Files | winxp/0/sda,63 | winxp/sda,63 (0)
5 Any Change| winxp/sda,63 | winxp/0/sda,63 (677)
6 Unchanged | winxp/sda,63 | winxp/0/sda,63 (10414)
7 Executable|New Files|winxp/sda,63|winxp/0/sda,63 (4)

armour> cat 7
print bodyfile records for new, executable files

armour> table 3; table 5; table 7
opens Java Swing tables showing BodyFile contents
```

Armour In Action — Viewing Results

```
armour> ls
1 mount/winxp/0/sda,63 (11091)
2 mount/winxp/sda,63 (11102)
3 New Files | winxp/sda,63 | winxp/0/sda,63 (11)
4 New Files | winxp/0/sda,63 | winxp/sda,63 (0)
5 Any Change| winxp/sda,63 | winxp/0/sda,63 (677)
6 Unchanged | winxp/sda,63 | winxp/0/sda,63 (10414)
7 Executable|New Files|winxp/sda,63|winxp/0/sda,63 (4)

armour> cat 7
print bodyfile records for new, executable files

armour> table 3; table 5; table 7
opens Java Swing tables showing BodyFile contents
```

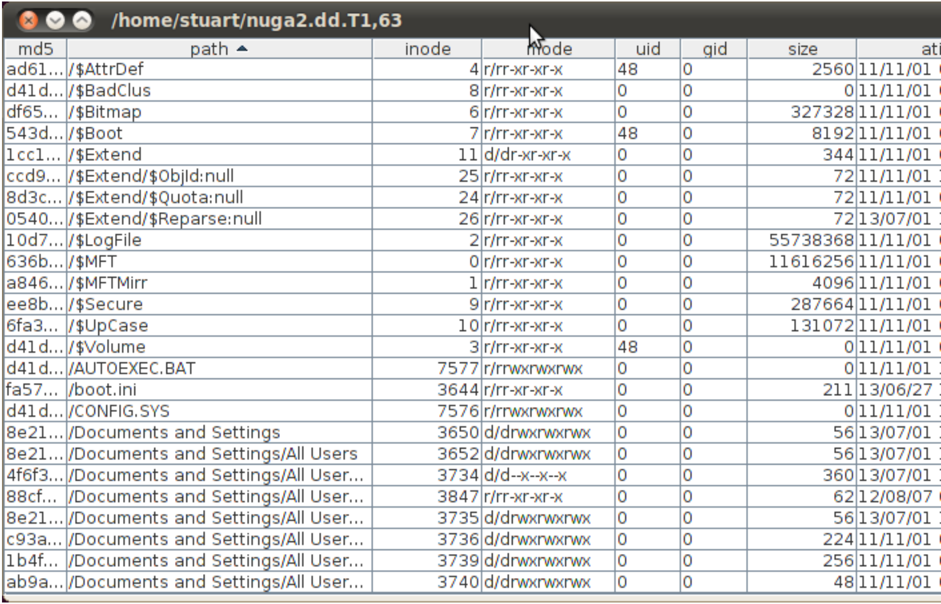
Armour In Action — Viewing Results

```
armour> ls
1 mount/winxp/0/sda,63 (11091)
2 mount/winxp/sda,63 (11102)
3 New Files | winxp/sda,63 | winxp/0/sda,63 (11)
4 New Files | winxp/0/sda,63 | winxp/sda,63 (0)
5 Any Change| winxp/sda,63 | winxp/0/sda,63 (677)
6 Unchanged | winxp/sda,63 | winxp/0/sda,63 (10414)
7 Executable|New Files|winxp/sda,63|winxp/0/sda,63 (4)

armour> cat 7
print bodyfile records for new, executable files

armour> table 3; table 5; table 7
opens Java Swing tables showing BodyFile contents
```

BodyFile Display As A Table


/home/stuart/nuga2.dd.T1,63

md5	path ▲	inode	mode	uid	gid	size	ati
ad61...	/$AttrDef	4	r/rr-xr-xr-x	48	0	2560	11/11/01
d41d...	/$BadClus	8	r/rr-xr-xr-x	0	0	0	11/11/01
df65...	/$Bitmap	6	r/rr-xr-xr-x	0	0	327328	11/11/01
543d...	/$Boot	7	r/rr-xr-xr-x	48	0	8192	11/11/01
1cc1...	/$Extend	11	d/dr-xr-xr-x	0	0	344	11/11/01
ccd9...	/$Extend/$ObjId:null	25	r/rr-xr-xr-x	0	0	72	11/11/01
8d3c...	/$Extend/$Quota:null	24	r/rr-xr-xr-x	0	0	72	11/11/01
0540...	/$Extend/$Reparse:null	26	r/rr-xr-xr-x	0	0	72	13/07/01
10d7...	/$LogFile	2	r/rr-xr-xr-x	0	0	55738368	11/11/01
636b...	/$MFT	0	r/rr-xr-xr-x	0	0	11616256	11/11/01
a846...	/$MFTMirr	1	r/rr-xr-xr-x	0	0	4096	11/11/01
ee8b...	/$Secure	9	r/rr-xr-xr-x	0	0	287664	11/11/01
6fa3...	/$UpCase	10	r/rr-xr-xr-x	0	0	131072	11/11/01
d41d...	/$Volume	3	r/rr-xr-xr-x	48	0	0	11/11/01
d41d...	/AUTOEXEC.BAT	7577	r/rrwxrwxrwx	0	0	0	11/11/01
fa57...	/boot.ini	3644	r/rr-xr-xr-x	0	0	211	13/06/27
d41d...	/CONFIG.SYS	7576	r/rrwxrwxrwx	0	0	0	11/11/01
8e21...	/Documents and Settings	3650	d/drwxrwxrwx	0	0	56	13/07/01
8e21...	/Documents and Settings/All Users	3652	d/drwxrwxrwx	0	0	56	13/07/01
4f6f3...	/Documents and Settings/All User...	3734	d/d--x--x--x	0	0	360	13/07/01
88cf...	/Documents and Settings/All User...	3847	r/rr-xr-xr-x	0	0	62	12/08/07
8e21...	/Documents and Settings/All User...	3735	d/drwxrwxrwx	0	0	56	13/07/01
c93a...	/Documents and Settings/All User...	3736	d/drwxrwxrwx	0	0	224	11/11/01
1b4f...	/Documents and Settings/All User...	3739	d/drwxrwxrwx	0	0	256	11/11/01
ab9a...	/Documents and Settings/All User...	3740	d/drwxrwxrwx	0	0	48	11/11/01

Armour The Report Writer

Armour mimics bash, so is scriptable. A malware analysis workflow:

```
// Record the disk state ahead of the run...
$ VBoxManage snapshot VM take "Clean"

// Run the malware sample in e.g. Cuckoo Sandbox...
$ submit.py sampleN.exe

// VMMount, and have Armour report all new files...
$ armour  -c "bop 1 2 1; cat 3" \
  mount/VM/0/sda,N mount/VM/sda,N > sampleN.NewFiles

// Wind back time and start again...
$ VBoxManage snapshot VM restorecurrent
```

Armour In The Real World

Enough of this virtual machine stuff! What about my real PC?

- Armour is just a BodyFile manipulation tool.
- Armour uses Sleuthkit for the heavy-lifting file system traversal.
- Neither know anything about virtual machines.

So, with a bootable Linux CD and a cheap external drive,
can do physical machine disk differencing too.

Armour In The Real World

- Time T_1. Boot from a trusted CD, with an external drive to hand:

```
$ dd if=/dev/sda of=/media/externalDrive
```

- From T_1 to T_2, regular computer use.
- Time T_2. Boot from an Armour-enabled CD, with the same external drive to hand:

```
$ mmls   /dev/sda   /media/externalDrive
$ armour /dev/sda,N /media/externalDrive,N
```

Armour/Sleuthkit analysis will discover all the malicious file system changes. There is nowhere to hide.

Nested Disk Differencing

For the paranoid malware sandboxer, snapshot the *host* before running malware in the local virtual environment. Then

- Apply file system, volume system differencing to the virtual disk.
- Boot the host from Armour CD, access earlier snapshot and do same difference investigation on physical disks.
- Will highlight the success or otherwise of `vmbreakout.exe`.

Conclusions, Future Work

- Precise disk differencing possible with open source tools.
- These tools can find every artifact, no matter how evasive.
- In the virtual world of malware sandboxing, disk differencing verifies local instrumentation.
- In the real world, a cheap external drive and a bootable CD enhance system security.

Plan to release to github. Looking for testers!