Fresh Produce

How We Can Integrate.. blah, blah...

by @williballenthin at FireEye

I’m here to talk about great workflows

- but I'm a UNIX-y person
- so I like command line tools

a bit about me

- I do forensics, incident response, and R&D
- I use my tools in different ways

tool use profiles

1. deep dive **forensics**
   - usually at zsh shell
   - iterative, exploratory command invocation

example: deep dive with interactive shell

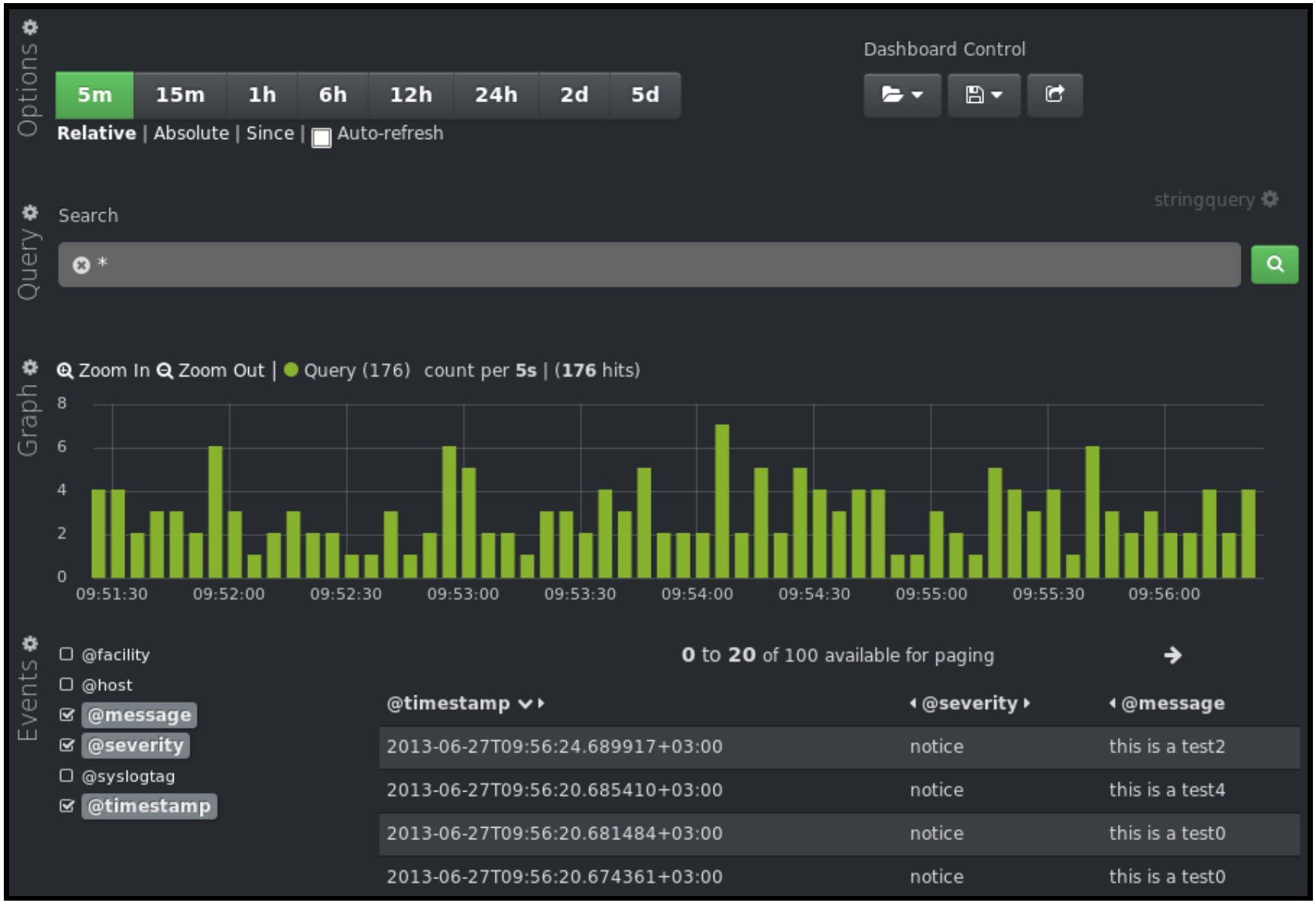
```
sudo fsstat /dev/sda1 | less
sudo fls -r /dev/sda1 | less
pushd $TOOLS/INDXParse
env/bin/python list_mft.py ./MFT.extracted | less

unzip pFPpHVqOjicpiWc.zip
7z e pFPpHVqOjicpiWc.zip
file pFPpHVqOjicpiWc.000
strings -a pFPpHVqOjicpiWc.000
strings -e l -a pFPpHVqOjicpiWc.000 | less
./XORSearch /media/host/data/tmp/tmp/bin/d3664n2qs.bin
./XORSearch /media/host/data/tmp/tmp/bin/d3664n2qs.bin DOS
./XORSearch /media/host/data/tmp/tmp/bin/psdbhja22.bin DOS
```

tool use profiles

1. deep dive
2. batched, en mass, **incident response**
    - scripted bash shell commands
    - fire and forget
    - scale
    - slicing and dicing

example: Kibana



tool use profiles

1. deep dive
2. incident response
3. research and **development**
    - compiled or interpeted languages
    - depend on libraries, else unmaintainable

I want my tools to work well in all these cases:

1. deep dive
2. incident response
3. research and development

aside: recent split of AD & Resolution1

tool use profiles, alternate

1. interactive
2. batched
3. programmatic

tool use profiles, alternate

1. **interactive**
   - human-digestable output
   - reasonable and hierarchical structure
   - formatted text, HTML, etc
2. batched
3. programmatic

example: useful interactive tool output

```
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext3
Volume Name:
Volume ID: cc9989aecd5a7c989040c9cda52a62ab

Last Written at: Tue Nov  4 03:39:52 2014
Last Checked at: Tue Aug 19 13:42:17 2014

Last Mounted at: Tue Nov  4 03:39:52 2014
Unmounted properly
Last mounted on: /

Source OS: Linux
Dynamic Structure
Compat Features: Journal, Ext Attributes, Resize Inode, Dir Index
InCompat Features: Filetype, Needs Recovery,
Read Only Compat Features: Sparse Super, Has Large Files
```

example: RegRipper tool output

```
perl -I $(pwd)/lib/lib/perl5 -Mlocal::lib rip.pl -r samples/XP/system -p appco
Launching appcompatcache v.20130425
appcompatcache v.20130425
(System) Parse files from System hive Shim Cache

Signature: 0xdeadbeef
WinXP, 32-bit
C:\WINDOWS\system32\regini.exe
ModTime: Mon Jul  7 11:59:59 2003 Z
UpdTime: Fri Jun 18 19:19:18 2004 Z
Size   : 33792 bytes

C:\WINDOWS\system32\ping.exe
ModTime: Mon Jul  7 11:59:59 2003 Z
UpdTime: Fri Jun 18 23:49:49 2004 Z
Size   : 16384 bytes

C:\WINDOWS\system32\dfrgntfs.exe
```

example: RegRipper tool output, 2

```
perl -I $(pwd)/lib/lib/perl5 -Mlocal::lib rip.pl -r samples/XP/software -p cls
Launching clsid v.20100227
clsid v.20100227
(Software) Get list of CLSID/registered classes

Classes\CLSID

Fri Jan 18 00:38:12 2008 Z
  {45FFAAA0-6E1B-11D0-BCF2-444553540000}  KsProxy DirectShow Audio Interface H
  {CBE3FAA0-CC75-11D0-B465-00001A1818E6}  KsProxy DirectShow Audio Interface H
Fri Jan 18 00:32:27 2008 Z
  {2C6594DB-04AD-490F-A447-DC8E2772E9CB}
Fri Jun 18 19:19:19 2004 Z
  {50E5E3D1-C07E-11D0-B9FD-00A0249F6B00}  RegWizCtrl
Fri Jun 18 19:19:17 2004 Z
  {C55A1680-CD5A-11CF-8D29-444553540000}  Registry Object
Fri Jun 18 19:00:18 2004 Z
  {E6ED0A00-43E0-11D1-BE58-00A0C90A4335}  EnStructureModification Class
```

tool use profiles, alternate

1. interactive
2. **batched**
   - machine parsable output
   - complete information, at expense of complexity
   - CSV, XMI, JSON, etc
3. programmatic

example: useful batchable tool output

```
0|\\\$MFT (filename)|0|0|1|0|78643200|1381184604|1381184604|1381184604|138118
0|\\\$MFTMirr|1|0|0|0|4096|1381184604|1381184604|1381184604|1381184604
0|\\\$MFTMirr (filename)|1|0|9|0|4096|1381184604|1381184604|1381184604|138118
0|\\\$LogFile|2|0|4|0|67108864|1381184604|1381184604|1381184604|1381184604
0|\\\$LogFile (filename)|2|0|5|0|67108864|1381184604|1381184604|1381184604|138
0|\\\$Volume|3|0|257|0|0|1381184604|1381184604|1381184604|1381184604
0|\\\$Volume (filename)|3|0|257|0|0|1381184604|1381184604|1381184604|13811846
0|\\\$AttrDef|4|0|0|0|50|1381184604|1381184604|1381184604|1381184604
0|\\\$AttrDef (filename)|4|0|0|0|40|1381184604|1381184604|1381184604|13811846
```

example: RegRipper tool output

```
perl -I $(pwd)/lib/lib/perl5 -Mlocal::lib rip.pl -r samples/XP/system -p appco
Launching appcompatcache v.20130425
appcompatcache v.20130425
(System) Parse files from System hive Shim Cache

Signature: 0xdeadbeef
WinXP, 32-bit
C:\WINDOWS\system32\regini.exe
ModTime: Mon Jul  7 11:59:59 2003 Z
UpdTime: Fri Jun 18 19:19:18 2004 Z
Size   : 33792 bytes

C:\WINDOWS\system32\ping.exe
ModTime: Mon Jul  7 11:59:59 2003 Z
UpdTime: Fri Jun 18 23:49:49 2004 Z
Size   : 16384 bytes

C:\WINDOWS\system32\dfrgntfs.exe
```

example: RegRipper tool output, 2

```
perl -I $(pwd)/lib/lib/perl5 -Mlocal::lib rip.pl -r samples/XP/software -p cls
Launching clsid v.20100227
clsid v.20100227
(Software) Get list of CLSID/registered classes

Classes\CLSID

Fri Jan 18 00:38:12 2008 Z
  {45FFAAA0-6E1B-11D0-BCF2-444553540000}  KsProxy DirectShow Audio Interface H
  {CBE3FAA0-CC75-11D0-B465-00001A1818E6}  KsProxy DirectShow Audio Interface H
Fri Jan 18 00:32:27 2008 Z
  {2C6594DB-04AD-490F-A447-DC8E2772E9CB}
Fri Jun 18 19:19:19 2004 Z
  {50E5E3D1-C07E-11D0-B9FD-00A0249F6B00}  RegWizCtrl
Fri Jun 18 19:19:17 2004 Z
  {C55A1680-CD5A-11CF-8D29-444553540000}  Registry Object
Fri Jun 18 19:00:18 2004 Z
  {E6FD0A00-43E0-11D1-BE58-00A0C90A4335}  EnStructureModification Class
```

tool use profiles, alternate

1. interactive
2. batched
3. **programmatic**
   - re-use expert knowledge!
   - custom code mixed with library code
   - interacting directly with objects
   - no stdin/stdout pipes or formatting

example: useful parsing library imports

```
from BinaryParser import Mmap
from MFT import Cache
from MFT import MFTEnumerator
from MFT import ATTR_TYPE
from MFT import MREF
from MFT import IndexRootHeader
```

example: not using RegRipper as a library

```
:-(
```

observation

these profiles are are often at odds with each other.

is there any hope?

flagship tools by profile

1. interactive
   - Autopsy
   - EnCase
   - FTK
2. batched
   - GRR
   - MIR
   - EnCase Enterprise
   - etc
3. programmatic
   - TSK
   - libmetz
   - etc

interesting: TSK seems to span them all

1. interactive: icat, fsstat, etc.
2. batched: bodyfile
3. programmatic: libtsk

observation

to support all three profiles, good architecture:

- frosting: scripting support
- middle: program for analyst
- base: robust, open library

observation, cont'd

some may start as libraries, and have tools tacked on top.
some tools may start as PoCs, then be split into library and tools.
with a solid base, achieving the frosting (workflow integration) can be a simple matter.

note: TSK only gets a ”pass” on the batch processing because it **invented** the format!

- this doesn't scale
- everyone can't do this
- actually, they can, because that's what we do today :'-(

I should not have to write a parser to interpret artifact parser results!

obvious solution

- build tools on robust, open libraries
- **unify output formats**
  - probably, use DFXML
  - or maybe, bodyfile, cause everyone already does
  - and JSON, cause, like Web2.0
- assuming we can all agree on one

example: DFXML object

```
<volume offset='32'>
    <byte_runs>
      <run offset='0' len='24900705' img_offset='32'>
    </byte_runs>
    <ftype>1</ftype>
    <ftype_str>ntfs</ftype_str>
    <block_size>4096</block_size>
    <block_count>4980142</block_count>
    <first_block>0</first_block>
    <last_block>4980141</last_block>
    <fileobject>
    ...
    </fileobject>
  </volume>
```

alternate solution

- (still) build tools on robust, open libraries
- de-standardize all output formats
- **accept output formatting specifications from the user**
- can still provide sane defaults for light usage

example: user defined CSV format spec

```
python list_mft.py /evidence/case001/CMFT --prefix "C:" \
  --format "{{ record.inode }}, {{ prefix }}{{ record.path }}, \
            {{ record.is_active }}, \
            {{ record.standard_information.accessed }}, \
            {{ record.filename_information.created }}, \
            {{ record.size }}" | head
```

example: CSV output

```
0, C:\$MFT, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 1818951
1, C:\$MFTMirr, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 409
2, C:\$LogFile, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 671
3, C:\$Volume, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 0
4, C:\$AttrDef, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 0
5, C:, 1, 2012-03-19 13:18:46.741314, 2005-04-30 21:04:47.484373, 0
6, C:\$Bitmap, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 2442
7, C:\$Boot, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 8192
8, C:\$BadClus, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 0
9, C:\$Secure, 1, 2005-04-30 21:04:47.484373, 2005-04-30 21:04:47.484373, 0
```

example: user defined XML format spec

```
<MFT_RECORD>
  <INODE>{{ record.inode }}</INODE>
  <PATH>{{ prefix }}{{ record.path }}</PATH>
  <CREATED>{{ record.standard_information.created }}</CREATED>
  <ACCESSED>{{ record.standard_information.accessed }}</ACCESSED>
  <MODIFIED>{{ record.standard_information.modified }}</MODIFIED>
  <CHANGED>{{ record.standard_information.changed }}</CHANGED>
  <SIZE>{{ record.size }}</SIZE>
</MFT_RECORD>
```

example: XML output

```
<MFT_RECORD>
  <INODE>0</INODE>
  <PATH>C:\$MFT</PATH>
  <CREATED>2005-04-30 21:04:47.484373</CREATED>
  <ACCESSED>2005-04-30 21:04:47.484373</ACCESSED>
  <MODIFIED>2005-04-30 21:04:47.484373</MODIFIED>
  <CHANGED>2005-04-30 21:04:47.484373</CHANGED>
  <SIZE>181895168</SIZE>
</MFT_RECORD>
<MFT_RECORD>
  <INODE>1</INODE>
  <PATH>C:\$MFTMirr</PATH>
  <CREATED>2005-04-30 21:04:47.484373</CREATED>
  <ACCESSED>2005-04-30 21:04:47.484373</ACCESSED>
  <MODIFIED>2005-04-30 21:04:47.484373</MODIFIED>
  <CHANGED>2005-04-30 21:04:47.484373</CHANGED>
  <SIZE>4096</SIZE>
</MFT_RECORD>
```

how do devs make this work?

as example, assume artifact parser

1. **stop print()ing as you go**
   - doesn't work with libraries to begin with
   - is a maintenance nightmare (how2add new format?)
2. construct in-memory data structures (lists, trees, etc) as you parse
3. traverse data structures and print output as the last step
4. ideally, use a declarative spec to define the formatting

aside: this is Model-View-Controller

MVC separates the concerns of presentation from logic and data.

- model: the parsed artifact
- view: a specific output format
- controller: (script interpreting paramters and flags)

different output formats are simply different views into the same model (parsed artifact)

how do devs make this work?

1. stop print()ing as you go
2. **use a templating engine to format output text**

text templating engines

templating engines generate formatted text by combining predefined text patterns with values from data structures

example: text templating engines (Jinja2)

```
User = namedtuple("User", ["name", "username"])
users = [User(url="https://www.icanhazcheezburger.org"', username="cats"),
         User(url="https://www.snoopy.net"', username="dogs")]
```

```
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

```
<ul>
  <li><a href="https://www.icanhazcheezburger.org">cats</a></li>
  <li><a href="https://www.snoopy.org">dogs</a></li>
</ul>
```

text templating engines by language

- Python - Jinja2
- Perl - Template Toolkit
- .NET - T4
- C - libctemplate
- Javascript - Handlebars.js
- Go - text/template

implementation example

- I patched RegRipper 'rip.pl' and 'appcompatcache.pl' to support user defined output formatting
- find it here: williballenthin/regripper branch 'user-defined-output-formatting'
- appcompatcache.pl: 113 additions, 43 deletions

remove inline printing

```
-# this is where we print out the files
- foreach my $f (keys %files) {
- ::rptMsg($f);
- push(@temps,$f) if (grep(/[Tt]emp/,$f));
- ::rptMsg("ModTime: ".gmtime($files{$f}{modtime})." Z");
- ::rptMsg("UpdTime: ".gmtime($files{$f}{updtime})." Z") if (exists $files{$f}
- ::rptMsg("Size : ".$files{$f}{size}." bytes") if (exists $files{$f}{size});
- ::rptMsg("Executed") if (exists $files{$f}{executed});
- ::rptMsg("");
- }
```

add additional initialization

```
+ my $ret = shift;
+ $ret->{"OS"} = "WinXP, 32-bit";
# header is 400 bytes; each structure is 552 bytes in size
my $num_entries = unpack("V",substr($data,4,4));
+ $ret->{"num_entries"} = $num_entries;
+ $ret->{"entries"} = ();
```

add objects as we go along

```
+ push(@{$ret->{"entries"}}, {
+ filename => $file,
+ size => $sz,
+ modtime => $modtime,
+ updtime => $updtime,
+ });
```

provide templates for formatting objects

```
sub getTemplates{
  # @return: hashref of name => hashref{name=>string, template=>string, versio

 return {
  "legacy" => {
    "name" => "legacy",
    "author" => "Willi Ballenthin",
    "version" => 20140405,
    "description" => "The original appcompatcache output format developed by H
    "template" => "[% FOREACH entry IN results.entries -%]
[% IF entry.filename -%]
[% entry.filename %]
UpdTime: [% helpers.gmtime(entry.updtime) -%]
[% IF entry.modtime %]\nModTime: [% helpers.gmtime(entry.modtime) %][% END -%]
[% IF entry.executed %]\nExecuted: [% entry.extime %][% END -%]
[% IF entry.size %]\nSize : [% entry.size %][% END -%]
\n\n[% END %][% END %]",
 }
```

sample usage: list templates

```
» perl rip.pl -r samples/XP/system -p appcompatcache -list_templates
  - "legacy" version 20140405 by Willi Ballenthin <willi.ballenthin@mandiant.c
  - "just_paths" version 20140405 by Willi Ballenthin <willi.ballenthin@mandia
                      </willi.ballenthin@mandiant.com></willi.ballenthin@mandian
```

sample usage: use legacy template

```
» perl rip.pl -r samples/XP/system -p appcompatcache -use_template legacy
Launching appcompatcache v.20130425
appcompatcache v.20130425
(System) Parse files from System hive Shim Cache

WinXP, 32-bit
C:\WINDOWS\system32\services.exe
UpdTime: Fri Jun 18 23:51:51 2004 Z
ModTime: Mon Jul  7 11:59:59 2003 Z
Size   : 101376

C:\WINDOWS\system32\lsass.exe
UpdTime: Fri Jun 18 23:51:51 2004 Z
ModTime: Mon Jul  7 11:59:59 2003 Z
Size   : 11776

...
```

sample usage: use new template

```
» perl rip.pl -r samples/XP/system -p appcompatcache -use_template just_paths
Launching appcompatcache v.20130425
appcompatcache v.20130425
(System) Parse files from System hive Shim Cache

WinXP, 32-bit
C:\WINDOWS\system32\services.exe
C:\WINDOWS\system32\lsass.exe
C:\WINDOWS\system32\oobe\msoobe.exe
C:\WINDOWS\system32\svchost.exe
C:\WINDOWS\msagent\agentsvr.exe
...
```

sample usage: use template from CLI

```
» perl rip.pl -r samples/XP/system -p appcompatcache -template_text "[% FOREAC
  - \"[% entry.filename %]\" ([% entry.size %])[% END %]"
Launching appcompatcache v.20130425
appcompatcache v.20130425
(System) Parse files from System hive Shim Cache

WinXP, 32-bit

  - "C:\WINDOWS\system32\services.exe" (101376)
  - "C:\WINDOWS\system32\lsass.exe" (11776)
  - "C:\WINDOWS\system32\oobe\msoobe.exe" (28160)
  - "C:\WINDOWS\system32\svchost.exe" (12800)
  - "C:\WINDOWS\msagent\agentsvr.exe" (235008)
  - "C:\WINDOWS\system32\spoolsv.exe" (51200)
  - "C:\WINDOWS\system32\wbem\wmiprvse.exe" (203776)
  - "C:\WINDOWS\system32\logonui.exe" (504320)
  - "C:\WINDOWS\system32\userinit.exe" (22016)
  - "C:\WINDOWS\explorer.exe" (1004032)
```

sample usage: use template from file

```
» cat /tmp/t.template
[% FOREACH entry IN results.entries %]
  - "[% entry.filename %]" ([% helpers.gmtime(entry.size) %])[% END %]


 » perl rip.pl -r samples/XP/system -p appcompatcache -template_file /tmp/t.te
Launching appcompatcache v.20130425
appcompatcache v.20130425
(System) Parse files from System hive Shim Cache

WinXP, 32-bit

  - "C:\WINDOWS\system32\services.exe" (Fri Jan  2 04:09:36 1970 Z)
  - "C:\WINDOWS\system32\lsass.exe" (Thu Jan  1 03:16:16 1970 Z)
  - "C:\WINDOWS\system32\oobe\msoobe.exe" (Thu Jan  1 07:49:20 1970 Z)
  - "C:\WINDOWS\system32\svchost.exe" (Thu Jan  1 03:33:20 1970 Z)
  - "C:\WINDOWS\msagent\agentsvr.exe" (Sat Jan  3 17:16:48 1970 Z)
  - "C:\WINDOWS\system32\spoolsv.exe" (Thu Jan  1 14:13:20 1970 Z)
```